Urika®-GX System Administration Guide

(2.1.UP00)

S-3016

Contents

1 About the Urika®-GX System Administration Guide

This publication contains administrative information about using the Cray® Urika®-GX system.

Typographic Conventions

`Monospace`	Indicates program code, reserved words, library functions, command-line prompts, screen output, file/path names, key strokes (e.g., `Enter` and `Alt-Ctrl-F`), and other software constructs.
`Monospaced Bold`	Indicates commands that must be entered on a command line or in response to an interactive prompt.
`Oblique` or *`Italics`*	Indicates user-supplied values in commands or syntax definitions.
Proportional Bold	Indicates a graphical user interface window or element.
`\` (backslash)	At the end of a command line, indicates the Linux® shell line continuation character (lines joined by a backslash are parsed as a single line). Do not type anything after the backslash or the continuation feature will not work correctly.

Scope and Audience

The audience of this publication is system administrators of the Urika®-GX system. This publication is not intended to provide detailed information about open source products used in the system. References to online documentation are included where applicable.

Record of Revision

Date	Addressed Release
May, 2018	`2.1UP00`
December, 2017	`2.0UP00`
April, 2017	`1.2UP00`
December, 2016	`1.1UP00`
August, 2016	`1.0UP00`
March, 2016	`0.5UP00`

Record of Revision

Major updates included in this revision are listed below:

- **New information:**

  - *Image Management with Docker and Kubernetes*
  - *Tenant Virtual Machine States*
  - *Multi-Tenancy*
  - *Execution of Lustre Sub-Commands Inside Tenant VMs*
  - *Execution of Lustre Sub-Commands Inside Tenant VMs*
  - *Multi-tenant HDFS*
  - *LDAP Server Start-up Issues*
  - *Reset an Administrator LDAP Password on Systems Using Urika-GX 1.2UP01 and Earlier Releases* on page 221
  - *Reset an Administrator LDAP Password when the OLC Scheme Password is Known* on page 223
  - *Reset an Administrator LDAP Password when the OLC Schema Password is Unknown* on page 222
  - *Save and Restore Tenant Information*
- **Updated information:**
  - *Configure SSL/TLS for Nagios Core*
  - *Enable SSL*
  - *Restrictions on Use*
  - *Urika-GX Service Modes*
  - *Security Related Troubleshooting Information*
  - *Get Started with Tenant Management*
  - *Tenant Management*
  - *Urika-GX CLI Commands for Managing Services*
  - *System Management Log File Locations*
  - *Updates to the 'Apache™ Spark Support' section to reflect information about using Spark on Kubernetes.*

Trademarks

2 The Urika-GX System

The Urika-GX system is a big data analytics platform optimized for analytic workflows. It combines a highly advanced hardware platform with a comprehensive analytic software stack to help derive optimal business value from data. The Urika-GX platform provides the tools required for capturing and organizing a wide variety of data types from different sources and enables analyzing big data and discovering hidden relationships.

The Urika-GX system also features a number of workload management tools as well as an optimized system administration tool for performing monitoring and management tasks.

For a list of features of the Urika-GX system, see S-3017, "*Urika®-GX System Overview*".

2.1 Administrative Components of Urika-GX

Urika-GX platforms have been developed by tightly integrating commodity hardware components, open-source software, and Cray proprietary hardware, to provide users a high performance, scalable and open compute platform.

Major administrative components of Urika-GX include:

- **System Management Workstation (SMW)** - The SMW is a server that acts as a single-point interface to a system administrator's environment. It provides an interface for performing administrative and monitoring capabilities.
  - **Hardware Supervisory System (HSS)** - HSS is an integrated system of hardware and software components that are used for managing and monitoring the system.
  - **Cobbler** - Cobbler is used on Urika-GX for provisioning and deployment.
- **Rack Controller (RC)** - The RC monitors the environmental sensors within the rack and manages communication between the SMW and other physical system components, including the rack, sub-rack and dANC (Dual Aries Network Card).
- **Intelligent Subrack Control Board (iSCB)** - The iSCB status command can be used to monitor the physical attributes of the sub-rack, such as the power supply, amperage, fan status, and temperature.
- **Aries Network Card Controller (ANCC)** - Each sub-rack chassis of the Urika-GX system contains two dANCs (dual Aries Network Cards). Each dANC contains 2 Aries chips, an Advanced RISC Machines (ARM) processor, and a number of environmental sensors to help monitor the system.
- **Integrated Dell Remote Access Controller (iDRAC)** - The iDRAC is a hardware that provides advanced agentless system management functionality for the SMW. It operates independently of the SMW's CPU and operating system. The version of iDRAC used on the Urika-GX system is iDRAC8.
- **System Monitoring and Performance Analysis Tools** - Urika-GX ships with Grafana and Nagios. These tools enable monitoring system resources and viewing performance statistics of various system components. For more information, see S-3015, "*Urika®-GX Analytic Applications Guide*".

- **Data Analytic Components** - Urika-GX features a number of data analytic tools that help perform analytic tasks, including managing and monitoring clusters, executing Hadoop and SPARK jobs, performing graph analytics, etc. For more information, see S-3015, "*Urika®-GX Analytic Applications Guide*" and S-3010, "*Cray™ Graph Engine User Guide*".
- **Security and Tenant Management Tools -** Secret files used on the system are managed by the Urika-GX Secret Manager. Tenancy is implemented through the use of a tenant VM that runs on physical nodes and provides controlled access to services on the physical nodes through a command proxy mechanism. For more information, refer to *Urika-GX Service Modes* on page 173 and *Tenancy* on page 179.

  NOTE: Only Spark and HDFS commands can be executed within a tenant VM in this release. All the commands for flexing the cluster, `mrun` and Cray Graph Engine (CGE) CLI commands cannot be executed within a tenant VM.

In addition, Urika-GX features a number of CLI scripts that facilitate system management and monitoring the system.

2.2 Network Components

There are 3 networks deployed on the Urika®-GX platform:

- **Aries High Speed Network (HSN)** - The Aries HSN provides high speed application and data network connectivity between nodes. This network provides node interconnect via high bandwidth, low latency DMA access. The hardware to support this network consists of an Aries Interface Board (AIB) connected to an available PCIe slot on each Urika-GX node and integrated into the node chassis assembly. The AIB is connected to the dANC integrated in the Urika-GX sub-rack. Copper cables provide an all-to-all connection of all dANCs in the system.
- **Operational Ethernet network**- The operational Ethernet network is used for ingesting user data. This network is comprised of a single unit 48-port GigE switch that provides dual 1GigE and/or dual 10GigE interfaces to the site network. Urika-GX's login nodes do not route through this switch and need to be directly connected to the site network. The operational network allows node connectivity externally from Urika-GX to the site network. The Urika-GX compute and I/O nodes are connected to a single managed Brocade ICX 6450-48, 48 port switch with a single power supply. Connectivity of this network to the site network is made possible by two available Gigabit Ethernet ports and/or two 10 Gigabit Ethernet ports on the ICX 6450-48 switch.

  The operational network can also be used to access data streaming applications and services directly from compute nodes.
- **Management Ethernet network** - The management Ethernet network is primarily used for system management, and not for user data. The management Ethernet network is comprised of two stacked 1U 48-port switches, which are located at the top of the Urika-GX rack, and can optionally contain redundant switch power supplies. These switches provide GigE management Ethernet connectivity to every node, System Management Workstation (SMW), Rack Controller (RC), Intelligent Subrack Control Board (iSCB), Power Distribution Units (PDUs), Dual Aries Network Cards (dANCs) and to the operational network that connects to the nodes.

  The Urika-GX system also contains the following subnets:

  - SMW subnet, which provides connectivity to the SMW and the RC.
  - Rack subnet, which provides connectivity to the dANCs and iSCB module.

  This network is supported by two managed Brocade ICX 6450-48, 48 port switches stacked together with two 10gigE optical interconnects. Each switch contains a single power supply, and can optionally contain redundant switch power supplies. The following VLANs are defined for this network to support management network traffic:

- ○ VLAN 102 - Uses ports 1-5 on each ICX 6450-48 switch. This is a dual-mode (tagged dual-mode for VLAN 102 and tagged for VLAN 103) VLAN. Untagged traffic on these ports belongs to VLAN 102. Traffic can be tagged for VLAN 103. The SMW HSS interface, the RC for a given rack, and the PDUs for a given rack are connected to these ports.
- ○ VLAN 103 Ports 6-12 on each ICX 6450-48 switch. Untagged traffic on these ports belongs to VLAN 103. The iSCBs and dANC cards are connected to these ports.
- ○ VLAN 104 Ports 13-48 on each ICX 6450-48 switch.

  NOTE: Traffic on this VLAN may be reduced if VLAN 105 is needed for storage as long as each compute node is connected to VLAN 104

  Untagged traffic on these ports belongs to VLAN 104. The compute nodes and the SMW node-side network are connected to these ports.
- ○ VLAN 105 Some number of Ports 13-48 on each ICX 6450-48 switch, as needed for storage management. Untagged traffic on these ports belongs to VLAN 105. The Storage Management Ports are connected to these ports.
- ○ VLAN 1 (default) is unused.

Traffic from the SMW to the subcomponents in the rack subnet, and vice versa, is routed through the corresponding RC.

For additional information, see the *Urika®-GX Hardware Guide*.

2.3 File Systems

Supported file system types on Urika-GX include:

- **Internal file systems**
  - ○ Hadoop Distributed File System (HDFS) - Hadoop uses HDFS for storing data. HDFS is highly fault-tolerant, provides high throughput access to application data, and is suitable for applications that have large data sets. Urika-GX also features tiered HDFS storage. HDFS data is transferred over the Aries network.
  - ○ Network File System (NFS) - The Urika-GX SMW hosts NFS, which is made available to every node via the management network.
  - ○ `/mnt/lustre` - This is a directory that hosts Lustre file system data if DAL/Sonexion is used.

⚠ **CAUTION:** Avoid using NFS for high data transfers and/or large writes as this will cause the network to operate much slower or timeout. NFS, as configured for Urika-GX home directories, is not capable of handling large parallel writes from multiple nodes without data loss. Though It is possible to configure NFS to handle parallel writes, it would require a hard mount, which would have undesired consequences.

File Locations

- Home directories are mounted on (internal) NFS, with limited space
- Distributed file system (Lustre), if provisioned, is mounted at `/mnt/lustre` and is suitable for larger files.

  Lustre mounts are isolated, with individual tenants having their own mount point.

2.4 System Nodes

Each Urika-GX node is a logical grouping of a processor, memory, and a data routing resource. Nodes can be categorized as compute, I/O, service and login nodes.

Table 1. Node Types and Descriptions

Node Type	Description
Compute nodes	Compute nodes run application programs.
I/O nodes	I/O nodes facilitate connecting to the supported external storage system.
Login nodes	Users log in to the Urika-GX system via login nodes and virtual machines (VMs). Login nodes store users' local files and facilitate launching jobs from the command line. They also offer the environment for users to build, compile, and monitor analytics applications.
Service nodes	Service nodes handle support functions such as user login and I/O.

All Urika-GX nodes run the CentOS operating system (version 7.3) as well as portions of the Cray Linux Environment (CLE).

2.5 Restrictions on Use

Hardware Considerations

The following items should be kept under consideration when using Urika-GX hardware:

- High speed network/management network switches must not be modified as this network is internal to Urika-GX.
- Moving the system from the rack Cray supplies to customer provided racks is not supported.
- Sub-rack and SMW hardware configuration must not be changed.
- PCIe devices should not be modified.
- Hardware and drivers installed on the SMW and nodes should not be modified.
- PDUs installed on the system should not be replaced.

Contact Cray Support if it is required to swap nodes between slots. The following options are supported:

- Connecting to the internal PDU power switches.
- Changing the hosts names of login nodes and the SMW.
- The single top of rack switch used for the operational network may be modified to meet site-specific needs. This switch is expected to be used to enable a direct connection from the site network to the compute and I/O nodes to support data ingestion and streaming analytics. This network may be modified to reflect site-specific IP addresses and node names that would be directly exposed to the site network. For information on how to configure the operational network, contact Cray support.

- The available space in the rack can be used for additional hardware, however proper power and cooling for that gear needs to be ensured.

Contact Cray Support for information related to:

- Optionally switching to higher bandwidth NICs on the login nodes or SMW connections to the site network.
- Changing the internal range of Cray's IP addresses in case there is a conflict.

Software Considerations

The following items should be kept under consideration when using Urika-GX software:

- Spark Shells using Kubernetes (i.e., those launched under the secure service mode) will be limited to 16 cores and 60 GiB memory and this cannot be overridden at the command line. This is due to a limitation of the lack of native Spark Shell support in the Spark on Kubernetes project that Cray has provided a workaround for in this release.
- Modifying the iSCB firmware is not supported.
- Modifying switch firmware (both Ethernet/InfiniBand) is not supported.
- Modifying node BIOS settings is not supported.
- Modifying the kernel and/or kernel modules is not supported.
- Deleting any factory installed software is not supported.
- Changing the default configurations of Mesos, Marathon, `mrun`, and Grafana is not supported.
- Launching of Docker containers through Docker commands is not supported. Users must use the Marathon interface for launching containers. For more information, refer to S-3015, "*Urika®-GX Analytic Applications Guide*".
- Building and managing new Docker images is currently not supported on Urika-GX. For more information, contact Cray Support.

Before installing any additional software on the Urika-GX system, a ticket should be opened with Cray Support to verify that the software will have no impact on the system. The following options are supported:

- Adding CentOS 7 packages that do not cause dependency issues with the Cray installed software. Only Cray-provided Linux updates and YUM repositories should be used.
- Installing additional HDP 2.6.1.0-129 compliant packages and modifying these packages for integrating into the existing software stack. This applies in the default service mode for HDP related items, except Spark. For more information, refer to *Urika-GX Service Modes* on page 173
- Tuning Hadoop and Spark configuration parameters listed in section "*Tunable Hadoop and Spark Configuration Parameters*" of S-3015, "*Urika®-GX Analytic Applications Guide*".

NOTE: Contact Cray Support if it is required to modify additional software configurations.

Security Considerations

If the Urika GX system is running in the secure mode in production, Cray does not recommend toggling back to the default mode while in production because, in the default mode, the security assurances provided by secure mode are not in place, and the security of data that was protected by secure mode may be compromised while running in the default mode. Cray cannot extend the secure mode security assurances to any system that has run in a production state in the default mode until that system has been fully re-deployed.

The following actions are not supported:

- It is recommended not to make any changes to the default set of Kubernetes and Kerberos configurations without consulting Cray Support, as doing so can adversely affect the functionality of the system in the secure service mode.
- Enabling of the `PermitUserEnvironment` option in `sshd_config(5)` or the passing of environment variables beyond those listed on the `ssh(1)` manual page.
- Changing any settings listed in `/etc/environment` into login sessions on Urika-GX physical nodes outside of the system through the login mechanism.
- Modifying the list of whitelist commands. For a list of commands that are part of the whitelist, see *Tenancy* on page 179
- Tenant NameNode configuration is managed automatically by the Urika-GX tenant management scripts. Manually altering the configurations of the tenant NameNode is not supported.
- Tenant names may only contain:
  - the letters a-z
  - the numbers 0-9
  - the characters '`-`' and '`.`'

    NOTE: The name '`default`' is reserved for the sample tenant configuration and cannot be used as a tenant name.
- The following items need to be kept under consideration while using the `ux-tenant-alter-vm` command:
  - Number of CPUs: At least 2 CPUs need to remain available when the number of CPUs is changed by this script. That is, if there are *N* number of CPUs, a maximum of *N*-2 CPUs can be assigned to a VM.
  - Amount of memory: At least 50% of the memory must remain available after a VM has been assigned memory using this command.

3 System Management

3.1 Check the Current Service Mode

Prerequisites

This procedure requires root privileges on the SMW.

About this task

Urika-GX supports two service modes, which dictate the list of services available. These modes include:

- Default
- Secure

Use the following instructions to determine the service mode the system is currently running in.

Procedure

1. Log on to the SMW as root.

   ```
   # ssh root@hostname-smw
   ```

2. Display the current service mode by using one of the following options:

   - Execute the `urika-state` command. This displays the current service mode as well as the status of all the services that are supported in that mode.
   - Execute the `urika-service-mode` command.

     ```
     # urika-service-mode
     Current mode is: default
     ```

   For more information, refer to the `urika-service-mode` and `urika-state` man pages.

3.2 Urika-GX Component Naming Conventions

The following table contains the component naming format for Urika®-GX systems:

Table 2. Urika-GX Component Naming Conventions

Component/Subject	Naming Pattern	Range
SMW	`s0`	N/A
Wild card, similar to `s0`.	`all`	N/A
Wild card, which refers to all the compute nodes	`all_comp`	N/A
Wild card, which refers to all the service nodes	`all_serv`	N/A
Machine partition	`p`*`P`*	`p0`
Rack	`r`*`R`*	*`R`*`:0 to 161`
Sub-rack. There are up to 4 sub-racks per rack. Each sub-rack contains up to 2 dual Aries Network Card (dANC) cards, up to 16 compute nodes, and up to 2 Intelligent Subrack Control Boards (iSCBs)	`r`*`R`*`s`*`S`*	*`S`*`:0 to 3`
Intelligent Subrack Control Board (iSCB)	`r`*`R`*`s`*`S`*`i`*`I`*	*`I`*`:0 to 1`
Dual Aries Network Card (dANC). There are up to 2 dANCs per sub-rack, accommodating up to 16 nodes	`r`*`R`*`s`*`S`*`c`*`C`*	*`C`*`:0 to 1`
High Speed Network (HSN) cable. The "`j`" name is visible on the cable connector face plate	`r`*`R`*`s`*`S`*`c`*`C`*`j`*`J`*	*`J`*`:0-15`
Aries ASIC within a dANC card	`r`*`R`*`s`*`S`*`c`*`C`*`a`*`A`*	*`A`*`:0 to 1`

Aries link control block within an Aries ASIC.	`r`*`R`*`s`*`S`*`c`*`C`*`a`*`A`*`l`*`RC`*	*`R`*`:0 to 5`
		`C``:0 to 7`

Network Interface Controller (NIC) within an Aries ASIC	`r`*`R`*`s`*`S`*`c`*`C`*`a`*`A`*`n`*`N`*	*`N`*`:0 to 3`
Node within a dANC card	`r`*`R`*`s`*`S`*`c`*`C`*`n`*`N`*	*`N`*`:0 to 7`
Accelerator within a node	`r`*`R`*`s`*`S`*`c`*`C`*`n`*`N`*`a`*`A`*	*`A`*`:0 to 7`
Board Management Control (BMC) within a node	`r`*`R`*`s`*`S`*`c`*`C`*`n`*`N`*`b`*`B`*	*`B`*`:0`

Tenant Naming Conventions

Tenant names may only contain:

- the letters a-z

- the numbers 0-9
- the characters '-' and '.

NOTE: The name `default` is reserved for the sample tenant configuration and cannot be used as a tenant name.

3.3 System Management Workstation (SMW)

The System Management Workstation (SMW) is the system administrator's console for managing a Cray system. The SMW is a server that runs the CentOS (version 7.3) operating system, Cray developed software, and third-party software. The SMW is also a point of control for the Hardware Supervisory System (HSS). The HSS data is stored on an internal hard drive of the SMW.

The SMW provides shell and web access to authorized users to perform administrative and monitoring tasks. The Nagios Core service also runs on the SMW.

Most system logs are collected and stored on the SMW. The SMW plays no role in computation after the system is booted. From the SMW an administrator can initiate the boot process, access the database that keeps track of system hardware, analyze log messages, and perform standard administrative tasks.

CAUTION: The SMW is a critical system component, which facilitates the operation of other hardware and software components. Therefore, it is important that all instructions in this publication be followed before making any changes/reconfigurations to the SWM, as well as before restarting the SMW.

3.3.1 Power On the System Management Workstation (SMW)

The SMW can be turned on by:

- Physically turning the SMW on via the power button.
- Using the iDRAC.

CAUTION:

The SMW is a critical system component, which facilitates the operation of other hardware and software components. Therefore, it is important that all instructions in this publication be followed before making any changes/reconfigurations to the SWM, as well as before restarting the SMW.

3.3.2 About the Integrated Dell Remote Access Controller (iDRAC)

The iDRAC is a systems management hardware and software solution that provides remote management capabilities, crashed system recovery, and power control functions for the System Management Workstation (SMW). The iDRAC alerts administrators to server issues, helps them perform remote server management, and reduces the need for physical access to the server. The iDRAC also facilitates inventory management and monitoring, deployment and troubleshooting. To help diagnose the probable cause of a system crash, the iDRAC can log event data and capture an image of the screen when it detects that the system has crashed.

For more information about the iDRAC, refer to online documentation at *http://www.dell.com*.

3.3.3 Control System Management Workstation (SMW) Power with the iDRAC8 Web Console

Prerequisites

Ensure that the SMW is up and running.

About this task

Use the iDRAC's web console to start up and shut down the System Management Workstation (SMW).

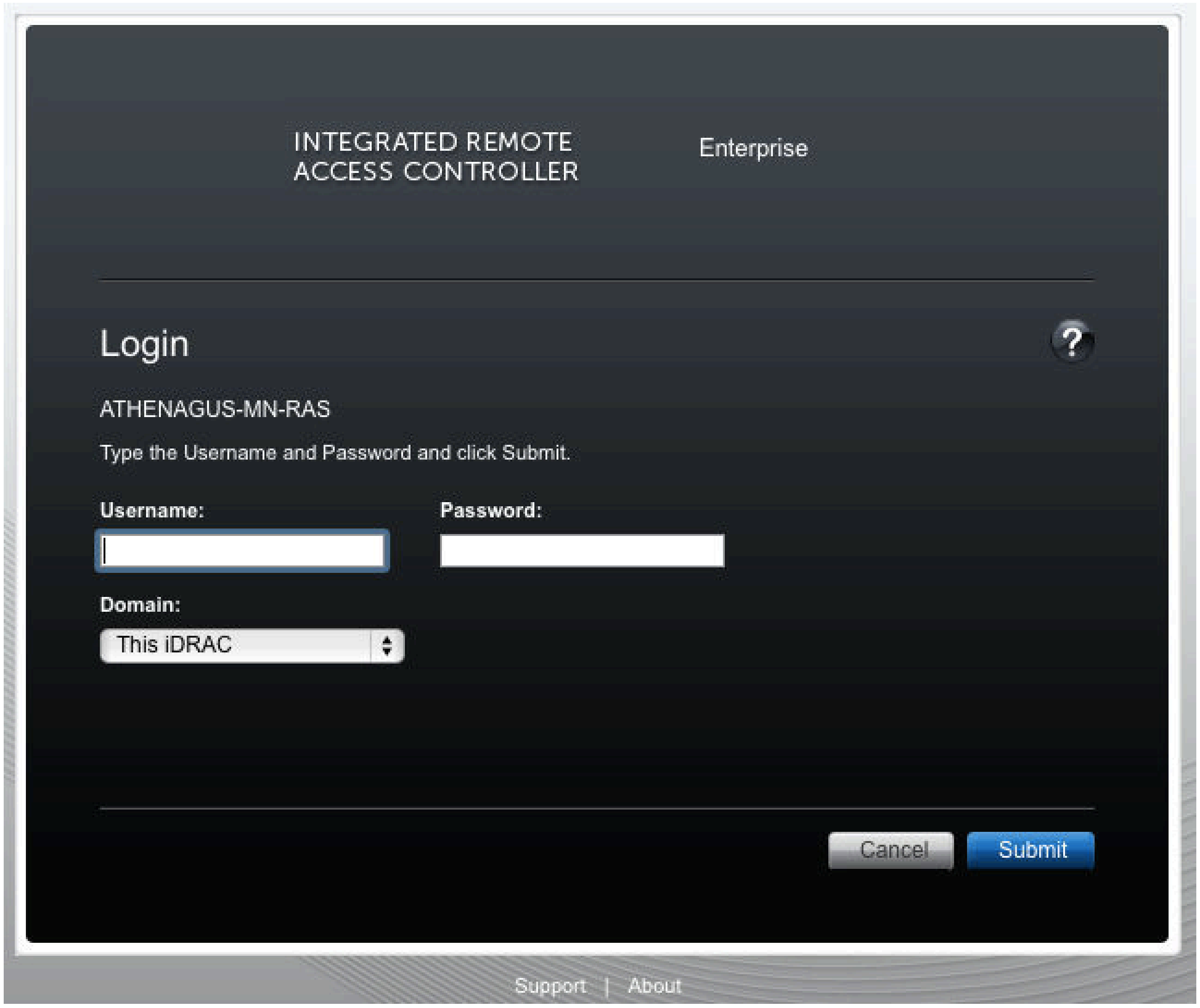
Procedure

1. Point a browser to the site-specific iDRAC IP address, such as https://*system-smw-ras*

   The iDRAC console's login screen appears.

2. Enter `root` and `initial0` in the **Username** and **Password** fields respectively. These are the default credentials that should only be used if the default credentials have not been changed.

Figure 1. iDRAC Login Screen

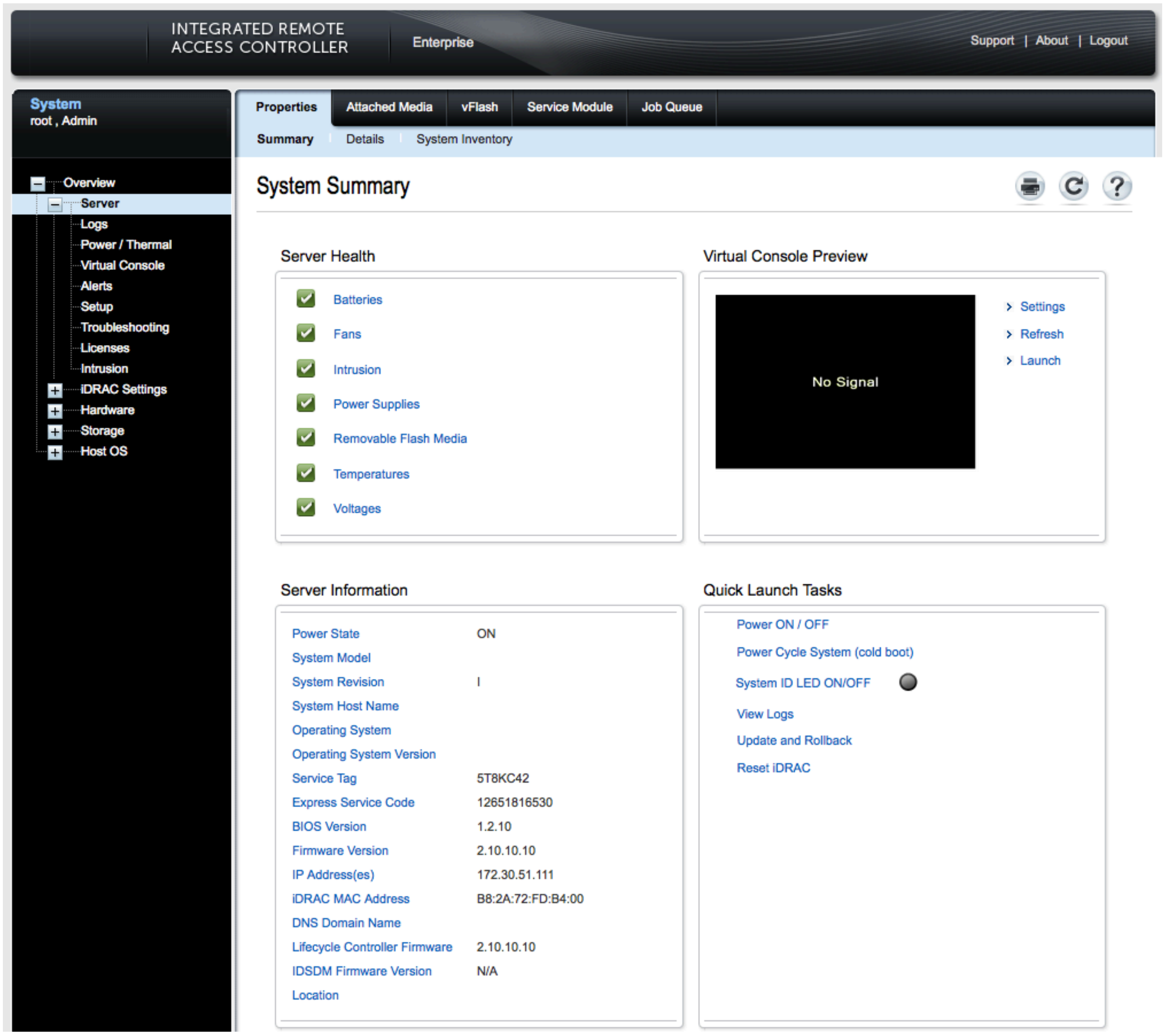


ATHENAGUS-MN-RAS

Type the Username and Password and click Submit.

3. On the **Quick Launch Tasks** section of the iDRAC UI, click on **Power ON/ OFF** link to control the SMW's power.

Figure 2. iDRAC Console



System Summary

Server Health

- ✓ Batteries
- ✓ Fans
- ✓ Intrusion
- ✓ Power Supplies
- ✓ Removable Flash Media
- ✓ Temperatures
- ✓ Voltages

Server Information

Power State	ON
System Model	
System Revision	I
System Host Name	
Operating System	
Operating System Version	
Service Tag	5T8KC42
Express Service Code	12651816530
BIOS Version	1.2.10
Firmware Version	2.10.10.10
IP Address(es)	172.30.51.111
iDRAC MAC Address	B8:2A:72:FD:B4:00
DNS Domain Name	
Lifecycle Controller Firmware	2.10.10.10
IDSDM Firmware Version	N/A
Location	

For more information about the iDRAC, visit *http://www.dell.com*

3.3.4 Synchronize the System Management Workstation (SMW) to the Site NTP Server

Prerequisites

This procedure requires root privileges.

About this task

The components of the Cray system synchronize time with the System Management Workstation (SMW) through Network Time Protocol (NTP). By default, the NTP configuration of the SMW is configured to stand alone; however, the SMW can optionally be configured to synchronize with a site NTP server. Follow this procedure to configure the SMW to synchronize to a site NTP server.

Procedure

1. Log on to the SMW as root.
2. Stop the NTP server by issuing the `systemctl stop ntpd` command.

   ```
   smw:~ # systemctl stop ntpd
   ```

3. Edit the `/etc/ntp.conf` file on the SMW to point to the new server.
4. Update the clocks

   ```
   smw:~ # ntpdate timeserver
   ```

5. Restart the NTP server by issuing the `systemctl start ntpd` command:

   ```
   smw:~ # systemctl start ntpd
   ```

   The SMW can continue to update the rest of the system by proxy. By default, the SMW qualifies as a stratum 3 (local) NTP server. For more information about NTP, refer to the Linux documentation.

6. Sync the hardware clock

   ```
   smw:~ # hwclock --systohc
   ```

7. Verify that the SMW has jitter from the NTP server

   ```
   smw:~ # ntpq -p
   ```

3.3.5 Synchronize Time of Day on System Nodes

Prerequisites

This procedure needs to be carried out as root.

About this task

Follow this procedure to configure Urika-GX compute nodes to synchronize to a site NTP server. This procedure is specific to a 48 node system.

Procedure

1. Stop the NTP server by issuing the `systemctl stop ntpd` command.

   ```
   # pdsh -w nid000[00-47] " systemctl stop ntpd"
   ```

2. Edit the `/etc/ntp.conf` file on the SMW to point to the new server.
3. Update the clocks

   ```
   # pdsh -w nid000[00-47] " ntpdate -s smw"
   ```

4. Sync the hardware clock

```
# pdsh -w nid000[00-47] "hwclock --systohc "
```

5. Restart the NTP server by issuing the `systemctl start ntpd` command:

```
# pdsh -w nid000[00-47] " systemctl start ntpd"
```

3.3.6 Reboot a Stopped System Management Workstation (SMW)

About this task

The SMW is an integral player in monitoring and maintaining optimal High Speed Network (HSN) traffic. If the SMW is down or being rebooted (i.e., not fully working), the Aries Network Card Controllers (ANCCs) will automatically throttle the high-speed network because the ANCCs are no longer hearing SMW heartbeats. This is done in order to prevent possible network congestion, which normally requires the SMW to be up in order to respond to such congestion. Once the SMW is up again, the ANCCs will unthrottle the network. (No attempt is made to prevent loss of data or to carry out operations that occur when the SMW is offline). The consequences of throttling are that the network will perform much more slowly than normal.

When the SMW comes up, it restarts, establishes communications with all external interfaces, restores the proper state in the state manager, and continues normal operation without user intervention.

For a scheduled or unscheduled shutdown and reboot of the SMW, it is necessary to have a backup of configuration files so that if one or more of the files in use becomes corrupted, a clean set of files is available with which to reboot.

Procedure

1. Ensure that there are no resiliency actions taking place (by executing `tail -f /var/opt/cray/log/p0-default/nlrd-YYYYMMDD`) at the time of a graceful SMW shutdown, otherwise wait until the action is finished.

```
# tail -f /var/opt/cray/log/p0-default/nlrd-YYYYMMDD
```

2. Boot the SMW.

3.4 Hardware Supervisory System (HSS)

HSS is an integrated system of hardware and software that monitors the hardware components of the system and proactively manages the health of the system. HSS communicates with nodes and management processors over an internal (private) Ethernet network that operates independently of the Cray Aries High Speed Network (HSN).

HSS includes the following components:

- HSS network
- HSS Command Line Interface (CLI)
- Aries Network Card Controllers (ANCCs)

- Rack controllers
- HSS daemons
- HSS database
- Various logs

HSS performs a number of administrative tasks, such as:

- Monitoring certain hardware system components
- Managing hardware and software failures
- Starting up and shutting down nodes
- Managing the High Speed Network (HSN)
- Maintaining system component states
- Managing the hardware inventory

HSS Command Line Interface

HSS has a command-line interface to manage and view the system from the SMW. For complete usage information, see the `xtcli(8)` man page.

Dual Aries Network Card (dANC) Controllers and Rack Controllers

A dANC control processor is hierarchically the lowest component of the monitoring system. The dANC Cray network card contains two Aries ASICs and an ANCC. There are 2 dANC cards per sub-rack, and hence 4 Aries ASICs, which support 16 nodes. The dANC monitors the general health of components, including items such as voltages, temperature, and various failure indicators. A version of Linux optimized for embedded controllers runs on each dANC controller.

Each rack has a rack control processor (rack controller) that monitors and controls the rack power and communicates with all dANC controllers in the rack. It sends a periodic heartbeat to the SMW to indicate rack health.

The rack controller connects to the dANC controllers via the Ethernet switch on each blade by an Ethernet cable and routes HSS data to and from the SMW. RC runs the same version of embedded Linux as the dANCs. The SMW, rack controllers, iSCBs, and ANCCs are all interconnected via Ethernet

The monitoring system uses periodic heartbeats. Processes send heartbeats within a time interval. If the interval is exceeded, the system monitor generates a fault event that is sent to the state manager. The fault is recorded in the event log, and the state manager sets an alert flag for the component (dANC controller or rack controller) that spawned it.

The rack and dANC controllers use NTP to keep accurate time with the SMW.

HSS Daemons

HSS daemons on the SMW and the controllers act to monitor and control the state of the system, and to respond to incidents such as hardware failures. The data path between the HSS CLI and the various daemons is via HSS events.

Cray System Network Routing Utility

The `rtr` command performs a variety of routing-related tasks for the High Speed Network (HSN). Tasks include:

- Generating and applying HSN routes
- Verifying that routes can be generated for the current configuration
- Verifying that generated routes are free of cyclic dependencies
- Dumping out a variety of routing-related and link-related information

HSS Database

The HSS database is a MariaDB relational database that contains the state of all the physical system components, including the System Management Workstation (SMW), Rack Controller (RC), Intelligent Subrack Control Board (iSCB), nodes, and the Aries Network Card Controller (ANCC). The state manager reads and writes the system state to the HSS database. The state manager keeps the database up-to-date with the current state of components and retrieves component information from the database when needed.

Log Files

Event Logs The event router records events to the event log in the `/var/opt/cray/log/event-`*`yyyymmdd`* file.

Log rotation takes place at specific time intervals. By default, one file is generated per day.

Dump Logs The `xtdumpsys` writes logs into the`/var/opt/cray/dump` directory by default.

SMW Logs SMW logs are stored in `/var/opt/cray/log/p0-`*`default`* on the SMW, and include logs for `xtconsole`, `xtconsumer`, `xtnlrd`, etc.

3.4.1 Hardware Supervisory System (HSS) Architecture Overview

HSS hardware on the Urika-GX system consists of a System Management Workstation (SMW), which is a rack-mounted Intel-based server running CentOS along with an Ethernet network that connects the SMW to a rack controller (RC) via a switch. The RC connects to one Aries Network Card Controller (ANCC) on each dual Aries Network Card (dANC) and consists of a mini PC running Linux. The ANCC has a 32-bit processor. Each hardware component in the HSS system runs a version of Linux with the relevant HSS software installed. RC is used to route data downstream from the SMW to the ANCCs and Intelligent Subrack Control Boards (iSCBs), and upstream from the ANCCs and iSCBs to the SMW.

HSS control and monitoring is performed by the SMW over the HSS Ethernet via a stacked managed switch, which uses VLANs to connect the SMW to the ANCCs, RC, and iSCBs.

The Urika-GX system can consist of 1, 2 or 3 sub-racks per rack, and 2 dANCs per sub-rack, resulting in a maximum of 6 dANCs per rack. Each dANC has 2 Aries ASICs, each of which has 4 NICs to support a single node per NIC connected by PCIe Gen 3.

HSS infrastructure software stack executes on the RC, SMW, and the ANCC to control and monitor the Aries ASIC.

Resiliency Communication Agent (RCA)

RCA is a messaging service that connects compute nodes to the HSS event and messaging system, and allows compute nodes to subscribe to and inject HSS events and messages.

Inventory Management

HSS keeps track of hardware inventory to identify which hardware components are up and running. It uses the `xthwinv` command to retrieve hardware component information.

Hardware Discovery

HSS plays an integral role in system hardware discovery. HSS components that play a role in this process include the HSS database and the `xtdiscover` command. For more information, see the `xtdiscover(8)` man page.

Hardware Supervisory System (HSS) Ethernet/Management Network

The HSS network provides interconnectivity between the System Management Workstation (SMW), Rack Controllers (RCs), and dual Aries Network Cards (dANCs) in a hierarchical fashion.

3.4.2 The xtdiscover Command

The `xtdiscover` command automatically discovers the hardware components on a Cray system and creates entries in the system database to reflect the current hardware configuration. The `xtdiscover` command identifies missing or non-responsive cabinets and empty or non-functioning Dual Aries Network Cards (dANCs). The `xtdiscover` command and the state manager ensure that the system status represents the real state of the hardware. When `xtdiscover` has finished, a system administrator can use the `xtcli` command to display the current configuration. No previous configuration of the system is required; the hardware is discovered and made available. Modifications can be made to components after `xtdiscover` has finished creating entries in the system database.

The `xtdiscover` interface steps a system administrator through the discovery process.

Prior to performing component discovery, the `xtdiscover` command will need to make sure that the Hardware Supervisory System (HSS) networking is set up properly, using a user-provided block of IP address space. This information is used to create the `/etc/hosts` file and DHCP entries for the HSS network. This setup typically only needs to be done once unless the address block is moved, or a new rack is added.

> **TIP:** Simply adding an additional rack within an existing address block will not affect the address assignments for the existing racks. If it is intended to add additional racks in the future, it is better to configure networking for all of them all at once. The `xtdiscover` command will automatically detect whether each rack is presently in the system and will set the system state accordingly.

If there are changes to the system hardware, such as populating a previously empty dANC, or adding an additional rack, then `xtdiscover` must be executed again, and it will perform an incremental discovery of the hardware changes. A full-system `xtdiscover` is not intended to be run while the High Speed Network (HSN) is actively routing traffic. When new blades are added during system operation with `xtwarmswap`, however, a mini-`xtdiscover` is automatically run to make the required updates to the database.

For more information, see the `xtdiscover(8)` man page.

3.4.3 Hardware Supervisory System (HSS) Component Location Discovery

Each Urika®-GX system rack is numbered starting at `0`. Each sub-rack within a rack has a dip switch that can set the rack and sub rack number. The iSCB conveys the rack and sub-rack numbers to the Aries Network Cards (ANCs) via an I^2C bus. The Dual Aries Network Card (dANC) blade has a slot-sense bit which tells it which dual dANC number it is within the sub-rack (`0` or `1`). The dANC uses the rack, sub-rack, and dANC number to construct its hostname. The Rack Controller (RC) determines its rack number from the location of the Intelligent Subrack Control Board (iSCB), encoded in a DHCP request sent by the iSCB and seen by RC.

3.4.4 Hardware Supervisory System (HSS) Daemons

HSS daemons and applications exchange information with the event router. They are located at: `/opt/cray/hss/default/bin` and are started when the System Management Workstation (SMW) boots. They can be managed via `systemd` and can be stopped and started via `systemctl stop hss` and `systemctl start hss` respectively. HSS daemons are configured dynamically by executing the `xtdaemonconfig` command.

Key HSS daemons include:

- State manager daemon (`state_manager`) - Performs HSS system hardware state management.
- Event router daemon (`erd`) and (`erdh`) - Performs HSS message routing.
- Node ID manager daemon (`nid_mgr`) - Manages node IDs and NIC addresses for every node in the system.

State Manager

HSS maintains the state of all components that it manages. The state manager, `state_manager`, runs on the SMW and uses a relational database (also referred to as the HSS database) to maintain/store the system state. The state manager keeps the database up-to-date with the current state of components and retrieves component information from the database when needed. Thus, the dynamic system state persists between SMW boots. The state manager uses the Lightweight Log Manager (LLM). The log data from state manager is written to: `/var/opt/cray/log/sm-`*`yyyymmdd`*. The default setting for state manager is to enable LLM logging. The state manager performs the following functions:

- Updates and maintains component state information
- Monitors events to update component states
- Detects and handles state notification upon failure
- Provides state and configuration information to HSS applications.

The state manager performs the aforementioned tasks on behalf of:

- System nodes
- Aries chips
- Aries HSN Links
- dual Aries Network Card (dANC)
- Rack controller (RC)
- Intelligent Subrack Control Board (iSCB)

In summary, the state manager subscribes to and listens for HSS events, records changes of states, and shares those states with other daemons.

The Event Router (`erd`)

HSS functions are event-driven. The event router daemon, `erd` runs on the SMW, rack controllers, and dANC controllers. HSS commands and daemons subscribe to events and inject events into the HSS system by using the services of the `erd`. The event router starts as each of the devices (SMW, rack controller, dANC controller) are started.

When the event router on the SMW receives an event from either a connected agent or from another event router in the hierarchy, the event is logged and then processed. HSS CLI commands use events to query and control HSS daemons.

Node ID (NID) Manager

The `nid_mgr` generates a list of mapping between node logical IDs and physical Network Interface Controller (NIC) IDs and distributes this information to the blade controllers. Along with the ability to assign NIDs automatically, the `nid_mgr` supports a mechanism that allows an administrator to control the NID assignment; this is useful for handling unique configurations. Administrator-controlled NID assignment is accomplished through the NID assignment file, `nids.ini`.

CAUTION: The `nids.ini` file can have a major impact on the functionality of a Cray system and should only be used or modified at the recommendation of Cray support personnel. Setting up this file incorrectly can make the Cray system unroutable.

Typically after a NID mapping is defined for a system, this mapping is used until some major event occurs, such as a hardware configuration change. This may require the NID mapping to change, depending on the nature of the configuration change. Adding additional racks does not typically result in a new mapping.

Since the operating system always uses NIDs, HSS converts these to NIC IDs when sending them on to the HSS network and converts them to NIDs when forwarding events from HSS network to a node.

3.4.5 Hardware Supervisory System (HSS) Administration and Diagnostic Commands Supported on Urika-GX

The following HSS commands are supported on Urika®-GX and need to be invoked from the System Management Workstation (SMW) to control HSS operations. Usage information for all of these commands can be viewed using the `-h` option, man pages are available where noted.

Table 3. HSS Administration Commands

Command	Description
`capmc`	Cray advanced power monitoring and control utility. See the `capmc(8)` man page for more information.
`hss_make_default_initrd.athena.rc`	Creates the Rack Controller (RC) image.
`hss_make_default_initrd.athena.danc`	Creates the dual Aries Network Card (dANC) image.
`hssbootlink`	Links a Linux kernel `bzImage` file, an `initramfs` file, and a `parameters` file so that they can be booted on a Controller by using PXE boot on an SMW.
`hssclone`	Clones the master image directory.
`hssds_init`	Creates the Hardware Supervisory System (HSS) data store; ensures the proper HSS data store user credentials are created and that the data store is ready for operation.
`hsspackage`	Facilitates creation of controller boot images.
`make_node_inventory`	Generates an inventory of Urika-GX nodes.

Command	Description
`nid2nic`	Prints all *`nid`*-to-*`nic_address`* mappings. See the `nid2nic(8)` man page for more information.
`rtr`	Performs system routing. See the `rtr(8)` man page for more information.
`xtagent`	Generic agent for accessing the event router.
`xtalive`	Checks for life of HSS daemons. See the `xtalive(8)` man page for more information.
`xtbounce`	Initializes Aries and HSN links. Powers down nodes as needed. See the `xtbounce(8)` man page for more information.
`xtcablecheck`	Compares the link endpoint pairs as known to the routing software with the link ID values set in MMRs in each LCB in the Aries ASIC to insure the HSN is cabled correctly.
`xtccpasswd`	Changes the password for all Rack Controllers (RCs), Dual Aries Network Card Controllers (dANCCs), and Intelligent Subrack Control Boards (iSCBs) on Urika-GX. It must be run as root.
`xtcc-ssh-keys`	Creates and modifies the `authorized_keys` file for Rack Controllers (RCs) and dANCCs.
`xtchecklink`	Checks HSN and PCIe link health.
`xtclass`	Displays the network topology class for this system.
`xtclear`	Clears component flags in the State Manager. For more information, see the `xtclear(8)` man page.
`xtcli`	Controls dANC and node power, displays status and sets administrative component status. See the `xtcli(8)` man page for more information.
`xtcon`	Provides a two-way connection to the console of a node. For more information, see the `xtcon(8)` man page.
`xtconsole`	Displays console text from one or more nodes.
`xtconsumer`	Displays HSS events.
`xtdaemonconfig`	Configures HSS daemons dynamically.
`xtdiscover`	Discovers and configures the Cray system hardware. This command is also used to populate the HSS database and set up HSS IP networking. For more information, see the `xtdiscover(8)` man page.
`xtdumpsys`	Gathers information when a system node stops responding or fails.
`xterrorcode`	Displays event error codes.
`xtfileio`	Performs file transfer to/from HSS controllers.

Command	Description
`xtfile2mem/xtmem2file`	Reads CPU or Aries memory and saves it in a file. Performs binary file-to-MMR/node memory and vice versa.
`xtgenid`	Generates HSS physical IDs.
`xtgenevent`	Injects arbitrary HSS events into the event stream.
`xthwerrlog`	Displays hardware errors retrieved from `xthwerrlogd` in user-friendly format.
`xthwerrlogd`	Monitors HW error log messages sent by the ANCC and by nodes.
`xtlogfilter`	Filters information from event router log files.
`xtmemio`	Performs reads or writes to Aries MMRs or node memory.
`xtnetwatch`	Watches the Cray system interconnection network for link control block (LCB) and router errors.
`xtnid2str`	Converts node identification numbers to physical names. See the `xtnid2str(8)` man page for more information.
`xtnlrd`	Responds to fatal Aries and HSN errors by rerouting the system.
`xtnmi`	Sends a non-maskable interrupt to target nodes. See the `xtnmi(8)` man page for more information.
`xtpcimon`	Monitors health of PCIe channels for Urika-GX systems.
`rackfw`	Flashes all devices in the Urika-GX system via out-of-band (OOB). See the `rackfw(8)` man page for more information.
`xtshow`	Shows components with selected characteristics. See the `xtshow(8)` man page for more information.
`xtsignal`	Sends a signal number or software interrupt to a remote process.
`xtwarmswap`	Allows Cray dANC cards or high-speed network cables to be warm swapped. See the `xtwarmswap(8)` man page for more information.
`xthwinv`	Retrieves hardware component information for selected modules.
`xtls`	Generates a summary of HSN link errors.
`xtpe`	Generates a summary of PCIe link errors.

There are a number of HSS diagnostics commands supported on Urika-GX. These commands need to be run from a compute node.

WARNING: All HSS diagnostics commands are intended for use by Cray Service Personnel only. Improper use of these restricted commands can cause serious damage to the system.

Table 4. HSS Diagnostic Commands

Command	Description
`xtbte_ata`	Perform system stress test to ensure that all logical endpoints go to all other end points using BTE put and/or get transactions.
`xtbte_ato`	Runs a number of applications individually or collectively as a suite to ensure that a system is ready for executing jobs after hardware or software upgrades or after power cycles.
`xtfma_ata`	Ensures that all logical endpoints go to all other end points using FMA put and/or get transactions.
`xtfma_ato`	Ensures that all logical endpoints target one end point using FMA Put and/or Get transactions. A round robin approach is used to step through each end point in the configuration.
`xtfma_amo`	Checks all AMO operations using an All-to-All algorithm. It tests AMOs with PUT (non-fetching) and GET (fetching) attributes.
`xtfbc`	Ensures that both the FMA and BTE logic blocks are tested. The test relies on the Generic Network Interface (GNI) API to directly communicate with the Cray network application-specific integrated circuit (ASIC).
`xtbte_perf`	Determines the one hop connections between nodes. It then performs BTE transfers over these one hop connections and determines the time taken to do so. The time duration is checked against an expected value for the link type. The actual data transferred is also verified.

3.4.6 Hardware Supervisory System (HSS) Environments

The HSS infrastructure environment is composed of a number of daemons, processes, and commands that help control and monitor physical system components.

HSS daemons supported on Urika-GX are listed in the following tables:

Table 5. ANCC HSS Daemons

Daemon	Description
Aries Network Card Controller (ANCC) System Daemon (`anccsysd`)	Controls power and state of the dual Aries Network Card (dANC) components, including Aries initialization and health monitoring.
ANCC Router Daemon (`anccrtrd`)	Handles requests from `rtr` on the System Management Workstation (SMW) to stage or install Aries Routes.
ANCC Bandwidth Throttle Daemon (`anccbwtd`)	Monitors High Speed Network (HSN) traffic and reports congestion indicators, assists in controlling congestion.

Daemon	Description
ANCC Network Daemon (`anccnwd`)	Monitors the Aries HSN link status, and reports soft and fatal errors.
ANCC PCIe Monitor Daemon (`anccpcimond`)	Monitors the Aries PCIe errors and status.
ANCC User-space Driver Daemon	Acts as the ANCC JTAG/MMR/Node Memory access driver.
ANCC Environment Monitor Daemon (`anccmond`)	Monitors various environmental sensors.
ERD File System Client (`erfsc`)	Acts as the ERFS dANC-level client.
Event Router Daemon (`erd`)	Performs HSS message routing
Controller Vitality Check Daemon (`cvcd`)	Monitors memory consumption, CPU utilization, file system usage, etc.

Table 6. Rack Controller HSS Daemon

Daemon	Description
Rack Controller System Daemon (`rcsysd`)	Monitors ANCC heartbeats, emits heartbeat for the State Manager, controls dANC power operations, and monitors the iSCB's health.
Controller Vitality Check Daemon (`cvcd`)	Monitors memory consumption, CPU utilization, file system usage, etc.
ERD File System Client (`erfsc`)	Acts as the ERFS RC-level client.
ERD File System Daemon (`erfsd`)	Facilitates the ERD file system
Event Router Daemon (`erd`)	Performs HSS message routing

Table 7. SMW HSS Daemons

Daemon	Description
State Manager (`state_manager`)	Performs HSS system hardware state management.
ERD File System daemon (`erfsd`)	Facilitates the ERD file system.
Event Router Daemon (`erd`)	Performs HSS message routing
NID Manager (`nid_mgr`)	Manages node IDs and NIC addresses for every node in the system.

NOTE: SMW HSS daemons are started and stopped via the `systemctl start hss` and `systemctl stop hss` commands respectively.

Additional Supporting HSS Daemons

The following supporting HSS daemons are started via the `systemctl start hss` command and log to `/var/opt/cray/log/p0-default` on the SMW.

Table 8. Supporting HSS Daemons

Daemon	Description
Network Link Resiliency Daemon (`xtnlrd`)	Assists in manual and automatic hardware swap-in/ swap-out.
Event Monitor (`xtconsumer`)	Monitors and logs a configurable set of HSS events.
HSN Error Monitor (`xtnetwatch`)	Monitors HSN hardware for errors and logs them.
PCIe Link Error Monitor (`xtpcimon`)	Monitors for PCIe errors on various devices.
Hardware Error Logger (`xthwerrlogd`)	Monitors HW error log messages sent by the ANCC and by nodes.
Node Console Logger (`xtconsole`)	Records node console output for every node in the system.

3.4.7 High Speed Network (HSN) Management

The Cray HSN is composed of a number of custom components developed by Cray that provide high-bandwidth, low-latency communication between all the compute processing elements of the system.

CAUTION: `xtbounce` should never be executed when nodes are up as this command will not gracefully shut nodes down, and will cause them to crash. For more information about using this command, contact Cray Support.

The HSN, which interconnects Aries chips, is configured by executing the `xtbounce` command to initialize the Aries and HSN links, followed by the `rtr` command to calculate Aries routes.

Aries chips are auto-initialized by the `anccsysd` daemon when the dANC powers on, or whenever `anccsysd` restarts.

3.4.8 Create Direct Connection between the System Management Workstation (SMW) and a Compute Node Console

The `xtcon` command is a console interface for system nodes. When it is executing, the `xtcon` command provides a two-way connection to the console of any running node.

Connect SMW to node `r0s0c1n0`

For this example, connect the SMW to the console of node `r0s0c1n0`:

```
smw:~> # xtcon r0s0c1n0
--- Console for component r0s0c1n0.  Use ^] to quit ---

CentOS release 6.5 (Final)
```

```
Kernel 2.6.32-431.el6_1.0000.8835-cray_ari_athena_c_cos on an x86_64

nid00008 login: root
Password:
Last login: Wed Sep 23 17:44:55 on ttyS0
[root@nid00008 ~]#
```

See the `xtcon(8)` man page for additional information.

3.4.9 Disable Hardware Components

If links, nodes, or Cray ASICs have hardware problems, the system administrator can direct the system to ignore the components with the `xtcli disable` command.

By default, when enabling a component, this command takes into consideration the hierarchy of components, performs the action upon the identified component(s), and cascades that action to any subcomponent of the identified component(s), unless the `-n` option is specified.

The `xtcli disable` command has the following form, where *`idlist`* is a comma-separated list of components (in cname format) that the system is to ignore. The system disregards these links or nodes.

```
xtcli disable [{-t type [-a] } | -n] [-f] idlist
```

IMPORTANT: The `-n` option with the `xtcli disable` command must be used carefully because this may create invalid system state configurations.

NOTE: The force option (`-f`) in some cases may cause issues with network resiliency operations.

Disabling of a rack, chassis, or dual Aries Network Card (dANC) will fail if any nodes under the component are in the `ready` state, unless the force option (`-f`) is used. An error message will indicate the reason for the failure. If the system is currently routed and the nodes are running, blades should not be disabled unless they are first removed from the high-speed network via the `xtwarmswap` with the `--remove` option. For this reason, the disabling will fail if any Aries LCBs are currently in the on state. The nodes must be shut down or halted prior to running `xtwarmswap` command with the `--remove` option.

For more information, see S-3018, "*Urika®-GX Network Resiliency Guide*".

Disabling of a node in the `ready` state will fail, unless the force option (`-f`) is used. An error message will indicate the reason for the failure.

The state of `empty` components will not change when using the `disable` command, unless the force option (`-f`) is used.

For detailed information about using the `xtcli disable` command, see the `xtcli(8)` man page.

3.4.10 Enable Hardware Components

If links, nodes, or Cray ASICs that have been disabled are later fixed, the system administrator can add them back to the system with the `xtcli enable` command.

The `xtcli enable` command has the following form, where *`idlist`* is a comma-separated list of components (in cname format) for the system to recognize.

```
xtcli enable [{-t type [-a] } | -n] [-f] idlist
```

IMPORTANT: The `-n` option with the `xtcli disable` command must be used carefully because this may create invalid system state configurations.

The state of `empty` components does not change when using the `xtcli enable` command, unless the force option (`-f`) is used.

The state of `off` means that a component is present on the system. If the component is a dANC controller, node, or ASIC, then this will also mean that the component is powered off. If the administrator disables a component, the state shown becomes `disabled`. When the `xtcli enable` command is used to enable that component for use once again, its state switches from `disabled` to `off`. In the same manner, enabling an empty component means that its state switches from `empty` to `off`.

On a running/routed system, dANCs that were not enabled when the network was brought up should not be enabled with the `xtcli enable` command, as they will not be routed into the Aries High Speed Network (HSN). Instead they should be added to the HSN with the `--add` option of the `xtwarmswap` command. This will automatically enable the blade when it is successfully routed into the system.

For more information, see the `xtcli(8)` man page.

3.4.11 Set Hardware Components to EMPTY

Use the `xtcli set_empty` command to set a selected component to the EMPTY state. HSS managers and the `xtcli` command ignore empty or disabled components.

Setting a selected component to the EMPTY state is typically done when a component, usually a blade, is physically removed. By setting it to EMPTY, the system ignores it and routes around it.

Only blades that are not currently routed into the HSN (i.e. removed with the `xtwarmswap` command) should be set to empty while the system is running, and only if the plan is to then physically remove the dANC from the system, otherwise the `xtcli disable` command should be used.

IMPORTANT: The `-n` option with the `xtcli disable` command must be used carefully because this may create invalid system state configurations.

For more information, see the `xtcli(8)` man page.

Set a dANC to the EMPTY state

```
crayadm@smw> xtcli set_empty -a r0s0c1n0
```

3.4.12 Stop Components Using the Hardware Supervisory System (HSS)

WARNING: Power down the rack(s) with software commands. Tripping the circuit breakers may result in damage to system components and to the file systems, and has an effect similar to that of abruptly powering-off any Unix system.

WARNING: Before powering down a Dual Aries Network Card (dANC) or a rack, ensure the operating system is not running on any nodes associated with dANCs.

The `xtcli power down` command powers down the specified rack and/or dANCs within the specified partition, chassis, or list of dANCs. Racks must be in the READY state to receive power commands. The `xtcli power` down command uses the following form, where *`physIDlist`* is a comma-separated list of racks or dANCs present on the system:

```
crayadm@smw:~> xtcli power down physIDlist
```

WARNING: Although a dANC is powered off, the Hardware Supervisory System (HSS) in the rack is alive and has power.

For information about powering down a component, see the `xtcli_power(8)` man page.

Power down a specified dANC

For this example, power down a dANC with the ID `r0s0c0`:

```
crayadm@smw:~> xtcli power down r0s0c0
```

3.4.13 Unlock Hardware Components

It may be required to unlock components when a daemon unexpectedly dies while holding a lock.

Use the HSS `xtcli lock` command to unlock components. This command is useful when a daemon unexpectedly dies while holding a lock.

The system administrator can manually check for locks with the `xtcli lock show` command and then unlock them. Unlocking a component does not print out the state manager session ID. The `-u` option must be used to unlock a component as follows:

```
crayadm@smw> xtcli lock -u lock_number
```

Where *`lock_number`* is the value given when initiating the lock; it is also indicated in the `xtcli lock show` query. Unlocking does nothing to the state of the component other than to release locks associated with it.

Unlock rack `r0`

```
crayadm@smw> xtcli -u r0
```

3.4.14 Capture and Analyze System-level and Node-level Dumps

The `xtdumpsys` command collects and analyzes information from a Cray system that is failing or has failed, has crashed, or is hung. Analysis is performed on, for example, event log data, active heartbeat probing, voltages, temperatures, health faults, in-memory console buffers, and high-speed interconnection network errors. When failed components are found, detailed information is gathered from them.

To collect similar information for components that have not failed, invoke the `xtdumpsys` command with the `--add` option and name the components from which to collect data. The HSS `xtdumpsys` command saves dump information in `/var/opt/cray/dump/`*`timestamp`* by default.

NOTE: When using the `--add` option to add multiple components, separate components with spaces, not commas.

The following example shows usage of the `xtdumpsys` command:

```
crayadm@smw> xtdumpsys --add r0s2c1
```

The `xtdumpsys` command is written in Python and supports plug-ins written in Python. A number of plug-in scripts are included in the software release. Call `xtdumpsys --list` to view a list of included plug-ins and their respective directories. The `xtdumpsys` command also now supports the use of configuration files to specify `xtdumpsys` presets, rather than entering them via the command line.

For more information, see the `xtdumpsys(8)` man page.

3.4.15 Collect Debug Information From Hung Nodes Using the xtnmi Command

CAUTION: This is not a harmless tool to use to repeatedly get information from a node at various times; only use this command when debugging data from nodes that are in trouble is needed. The `xtnmi` command output may be used to determine problems such as a core hang. `xtnmi` will stop a running node. It is best used when a node is not running correctly and debugging information is needed, or to stop a node that is running incorrectly.

The sole purpose of the `xtnmi` command is to collect debug information from unresponsive nodes. As soon as that debug information is displayed to the console, the node panics.

For additional information, see the `xtnmi(8)` man page.

An example of using the `xtnmi(8)` command is:

```
smw$ xtnmi r0s1c0n4,r0s0c1s2
The following ID(s) will be NMI'd: r0s1c0n4,r0s0c1s2.
All expected responses (1) were received.
```

3.4.16 Find Node Information

Translate Between Physical ID Names and Integer NIDs

To translate between physical ID names (rnames) and integer NIDs, generate a system map on the System Management Workstation (SMW) and filter the output, enter the following command:

```
crayadm@smw> rtr --system-map | grep rname | awk '{ print $1 }'
```

For more information, see the `rtr(8)` man page.

Find Node Information Using the `xtnid2str` Command

The `xtnid2str` command converts numeric node identification values to their physical names (cnames). This allows conversion of Node ID values, ASIC NIC address values, or ASIC ID values.

For additional information, see the `xtnid2str(8)` man page.

Find the physical ID for node 12

```
smw# xtnid2str 12
node id 0xc = 'r0s0c1n4'
```

Find the physical ID for nodes 0, 1, 2, and 3

```
smw# xtnid2str 0 1 2 3
node id 0x0 = 'r0s0c0n0'
node id 0x1 = 'r0s0c0n1'
node id 0x2 = 'r0s0c0n2'
node id 0x3 = 'r0s0c0n3'
```

Find Node Information Using the `nid2nic` Command

The `nid2nic` command prints the *`nid`*-to-*`nic`* address mappings, *`nic`*-to-*`nid`* address mappings, and a specific *`physical_location`*-to-*`nic`* address and *`nid`* mappings.

For information about using the `nid2nic` command, see the `nid2nic(8)` man page.

3.4.17 Request and Display System Routing

Use the HSS `rtr` command to request routing for the HSN, to verify current route configuration, or to display route information between nodes. Upon startup, `rtr` determines whether it is making a routing request or an information request.

For more information, see the `rtr(8)` man page.

Display routing information

The `--system-map` option to `rtr` writes the current routing information to `stdout` or to a specified file. This command can also be helpful for translating node IDs (NIDs) to physical ID names.

```
crayadm@smw> rtr --system-map
```

Route the entire system

The `rtr -R | --route-system` command sends a request to perform system routing. If no components are specified, the entire configuration is routed as a single routing domain based on the configuration information provided by the state manager. If a component list (*`idlist`*) is provided, routing is limited to the listed components. The state manager configuration further limits the routing domain to omit disabled blades, nodes, and links and empty blade slots.

```
crayadm@smw> rtr --route-system
```

CAUTION: The `rtr -R` command should not be executed if the system nodes have already been booted and are using the Aries network.

CAUTION: Be sure the Aries ASCIs have been initialized using the `xtbounce` command and that the HSN links have been initialized.

3.4.18 Initiate a Network Discovery Process

Use the HSS `rtr --discover` command to initiate a network discovery process.

```
crayadm@smw> rtr --discover
```

The discovery process must be done on the system as a whole—it cannot be applied to individual partitions. Therefore, discovery will immediately fail if the system does not have partition `p0` enabled.

The `rtr --discover` process should be used under the following circumstances:

- During an initial install, after successful execution of `xtdiscover`
- During the installation of additional cabinets in an existing installation, after the successful execution of `xtdiscover`
- During an upgrade of optical cabling in a system, after all recabling is complete

The `rtr --discover` process is NOT required under the following circumstances:

- On any single group system at any time, even those listed above
- During a warmswap operation

See the `rtr(8) man` page for additional information.

3.4.19 Power Up a Rack or Dual Aries Network Card (dANC)

IMPORTANT: Change the state of the hardware only when the operating system is not running or is shut down.

The `xtcli power up` command powers up the specified rack and/or dual Aries Network Card (dANC) within the specified partition, chassis, or list of dANCs. Racks must be in the `READY` state to receive power commands.

The `xtcli power up` command has the following form, where `physIDlist` is a comma-separated list of racks or dANCs present on the system.

```
xtcli power up physIDlist
```

For more information, see the `xtcli_power(8)` man page.

Power up dANC in `r1s2c0`

```
crayadm@smw:~> xtcli power up r1s2c0
```

3.4.20 Check the Status of System Components

Check the status of the system or a component with the `xtcli status` command on the System Management Workstation (SMW). By default, the `xtcli status` command returns the status of nodes.

The `xtcli status` command has the following form:

```
xtcli status [-n] [-m] [{-t type -a}] node_list
```

Where *type* may be: `cc`, `bc`, `cage`, `node`, `aries`, `aries_lcb`, `pdc`, or `qpdc`. The list must have component IDs only and contain no wild cards.

For more information, see the `xtcli(8)` man page.

```
crayadm@smw:~> xtcli status -t node r0s0c1n1
Network topology: class 0
Network type: Aries
           Nodeid: Service  Core Arch|  Comp state    [Flags]
------------------------------------------------------------------
         r0s0c1n1:       -       ATHE|       ready    [noflags|]
------------------------------------------------------------------
```

3.4.21 Check Compute Node High Speed Network (HSN) Connection

Use the Linux `ping` command to verify that a compute node is connected to the the HSN. The command must be run from a node, not the System Management Workstation (SMW).

For more information, see the Linux `ping(8)` man page.

Verify that a compute node is connected to the network

```
nid00007:~> ping nid00015
PING nid00015 (10.128.0.16) 56(84) bytes of data.
64 bytes from nid00015 (10.128.0.16): icmp_seq=1 ttl=64 time=0.032 ms
64 bytes from nid00015 (10.128.0.16): icmp_seq=2 ttl=64 time=0.010 ms
```

3.4.22 Monitor the Health of PCIe Channels

Processors are connected to the high-speed interconnect network (HSN) ASIC through PCIe channels.

The `xtpcimon` command runs on the SMW and is started when the system is initialized.

Any PCIe-related errors are reported to `stdout`, unless directed to a log file.

If the optional `/opt/cray/hss/default/etc/xtpcimon.ini` initialization file is present, the `xtpcimon` command uses the settings provided in the file.

3.4.23 Poll a Response from an HSS Daemon, Manager, or the Event Router

The `xtalive` command can be used to ensure that the HSS has connectivity to the ANCCs and that, by default, `anccsysd` is running and responding to events on the ANCCs.

For more information, see the `xtalive(8)` man page.

Check the state manager

```
crayadm@smw> xtalive -l smw -a sm s0
```

3.4.24 View Component Alert, Warning, and Location History

Use the `xtcli comp_hist` command to display component alert, warning, and location history. Either an error history, which displays alerts or warnings found on designated components, or a location history may be displayed.

Display the location history for component `r1s0c1n2`

```
crayadm@smw> xtcli comp_hist -o loc r1s0c1n2
```

For more information, see the `xtcli(8)` man page.

3.4.25 Display Alerts and Warnings

Use the `xtshow` command to display alerts and warnings. Type commands as **`xtshow`** `--option_name`, where *`option_name`* is `alert`, `warn`, or `noflags`.

Alerts are not propagated through the system hierarchy, only information for the component being examined is displayed. For example, invoking the `xtshow --alert` command for a cabinet does not display an alert for a node. Similarly, checking the status of a node does not detect an alert on a cabinet.

Show all alerts on the system

```
crayadm@smw:~> xtshow --alert
```

Alerts and warnings typically occur while the HSS `xtcli` command operates; these alerts and warnings are listed in the command output with an error message. After they are generated, alerts and warnings become part of the state for the component and remain set until manually cleared.

For additional information, see the `xtshow(8)` man page.

3.4.26 Display Error Codes

When an HSS event error occurs, the related message is displayed on the SMW. The `xterrorcode` command on the SMW displays a single error code or the entire list of error codes.

Display HSS error codes

```
crayadm@smw>  xterrorcode errorcode
```

A system error code entered in a log file is a bit mask; invoking the `xterrorcode` *`bitmask_code_number`* command on the SMW displays the associated error code.

Display an HSS error code using its bit mask number

```
crayadm@smw> xterrorcode 131279
Maximum error code (RS_NUM_ERR_CODE) is 447
code = 207, string = 'Node Voltage Fault'
```

3.4.27 Display Component State Information

Use the HSS `xtshow` command to identify the state of components. Commands are typed as xtshow --*`option_name`*.

Identify all nodes in the `disabled` state

```
crayadm@smw:~> xtshow --disabled
L1s that are disabled...
Cages that are disabled...
             r0s3:       -       ATHE|    disabled    [noflags|]
L0s that are disabled...
Nodes that are disabled...
Aries that are disabled...
AthenaISCBs that are disabled...
           r0s3i0:       -       ATHE|    disabled    [noflags|]
AriesLcbs that are disabled...
```

The --disabled option in the preceding example shows all the disabled hardware components. For more information, see the `xtshow(8)` man page.

3.4.28 Clear Component Flags

Use the `xtclear` command to clear system information for selected components. Type commands as `xtclear --option_name`, where *`option_name`* is `alert`, `reserve`, or `warn`.

Clear all warnings in specified cabinet

For this example, clear all warnings in cabinet `c13-2`:

```
smw:~> xtclear --warn r1
```

Alerts, reserves, and warnings must be cleared before a component can operate. Clearing an alert on a component frees its state.

For more information, see the `xtclear(8)` man page.

3.4.29 Flash Management on Urika-GX

HSS is responsible for flashing all devices in the Urika®-GX system via an out-of-band (OOB) mechanism. This can be achieved via the System Management Workstation (SMW) `rackfw` command, which is a flash tool. The SMW flash tool is used to flash firmware of the following components:

- Intelligent Subrack Control Board (iSCB) image

- Aries Network Card Controller (ANCC) image
- Dual Aries Network Card (dANC) FPGA

The `rackfw` command generates a summary output similar to the following on completion of an update or query.

Flash the dANC, iSCB image and ANCC image

```
crayadm@smw:~> rackfw
---ANCC STATUS---------------------
r0s0c0: OK
r0s0c1: OK
r0s1c0: MISMATCH
r0s1c1: OK


---ISCB STATUS---------------------
r0s0i0: OK       1.0rc4
r0s1i0: MISMATCH 1.0rc3


---SUMMARY------------------------
   1 ANCCs out of date
   3 ANCCs current
   1 ISCBs out of date
   1 ISCBs current
```

For more information, see the `rackfw(8)` man page.

3.4.30 Create and Modify the `authorized_keys` File Using the `xtcc-ssh-keys` Command

The `xtcc-ssh-keys` command creates and modifies the `authorized_keys` file for Rack Controllers (RCs) and Dual Aries Network Card Controllers (dANCCs) on Urika-GX systems. Due to constraints on volatile memory storage, it does not update the `authorized_keys` file for the Intelligent Subrack Control Boards (iSCBs). It must be run as root.

The `xtcc-ssh-keys` command takes no options. When run, it invokes the user-selected text editor (specified by the VISUAL or EDITOR environment variables and defaulting to vi) on a file maintained by Cray HSS. This file has the format described in `sshd(8)` under the heading "`AUTHORIZED_KEYS FILE FORMAT`". Adding a public key to this file permits a user authenticating using the corresponding private key to connect to the RCs and dANCCs without using a password. Once the file has been written from within the editor, the changes will take effect on booted controllers within one minute.

For more information, see the `xtcc-ssh-keys(8)` man page.

3.4.31 Change the Passwords of RC, dANCCs and iSCB using the `xtccpasswd` Command

The `xtccpasswd` command changes the password for all Rack Controllers (RCs), Dual Aries Network Card Controllers (dANCCs), and Intelligent Subrack Control Boards (iSCBs) on Urika-GX systems. It must be run as root. This command allows an administrator to grant access to the SMW without granting access to the controller network nodes.

If `xtccpasswd` is run without options, it prompts for a new password (which will not be echoed to the screen) and then confirms it. If the two entered passwords match exactly, a salted hash of the password is written to a file maintained by Cray HSS. Within one minute, all booted controllers in the Urika-GX system will be using the new password.

For more information, see the `xtccpasswd (8)` man page.

3.4.32 Gather Troubleshooting Information Using the xtdumpsys Command

The `xtdumpsys` command collects and analyzes information from a Cray system that is failing or has failed, has crashed, or is hung. Analysis is performed on, for example, event log data, active heartbeat probing, voltages, temperatures, health faults, in-memory console buffers, and high-speed interconnection network errors. When failed components are found, detailed information is gathered from them. To collect similar information for components that have not failed, invoke the `xtdumpsys` command with the `--add` option and name the components from which to collect data.

NOTE: Only the `crayadm` account can execute the `xtdumpsys` command.

For more information, see the `xtdumpsys(8)` man page.

3.5 Dual Aries Network Card (dANC) Management

There are a number of components that are used to perform dANC management on Urika-GX.

dANC Power

By default, the dANCs power up when the rack is powered up. They can however be turned off and on via the `xtcli power` command.

CAUTION: Powering off a dANC on a booted and routed system will route out that card out of the Aries High Speed Network (HSN) and make all 8 nodes (served by that dANC card) unusable.

dANC uBoot

The Linux `U-boot` utility supports booting a Linux image on the ARM processor (located on the dANC controller) via net boot or flash boot.

There are 3 valid boot modes for the ANCC and iSCB, as listed below:

1. Boot from flash. This is the default mode.
2. Netboot and write image to flash . This mode is used for software upgrades
3. Netboot and do not write to flash. This is a development option.

The Urika®-GX `rackfw` tool enables recovery mode for dANC booting. It will also allow for initializing/setting the boot mode in NVRAM as follows:

- Set 1 of 3 ANCC boot modes listed above. If the `nvRAM` is uninitialized or corrupted, it is required to initialize the `nvRAM`.
- Set the default ANCC image location/path, or the iSCB image
- Reboot specified device (`ANCC0`, `ANCC1`, `iSCB0` or `iSCB1`)

dANC Scrub Devices

The Urika-GX system supports scrubbing of non-volatile devices on the dANC. This includes the `DANFPGA` and dANC flash devices. The `DANFPGA FPGA` image and flash tool is included in the dANC ARM image and is flashed after the ARM has booted Linux.

dANC Monitoring

The dANC monitoring daemon, `anccmond` executes on the Dual Aries Network Card Controller (dANCC) and performs the following functions:

- Monitors Aries, AOC and board temperatures.
- Monitors the AOC and board power.
- Sends events to HSS ERD.
- Provides threshold warnings to iSCB via an I^2C bus.
- Sets up the temperature and voltages as a poll mechanism, so that the iSCB can poll for the data.

The HSS thresholds can be modified on the SMW using the HSS `xtdaemonconfig` command.

3.6 Analyze Node Memory Dump Using the kdump and crash Utilities on a Node

The `kdump` and `crash` utilities may be used to analyze the memory on any Urika®-GX compute node. The `kdump` command is used to dump node memory to a file. `kdump` is the Linux kernel's built-in crash dump mechanism. In the event of a kernel crash, `kdump` creates a memory image (also known as `vmcore`) that can be analyzed for the purposes of debugging and determining the cause of a crash. Dumped image of the main memory, exported as an Executable and Linkable Format (ELF) object, can be accessed either directly during the handling of a kernel crash (through `/proc/vmcore`), or it can be automatically saved to a locally accessible file system, to a raw device, or to a remote system accessible over the network. `kdump` is configured to automatically generate `vmcore` crash dumps on node crashes. These dumps can be found on the node in the crash partition, mounted to `nid000`*`XX`*`:/mnt/crash/var/crash/`*`datestamp`*`/*`, where *XX* ranges from `00-15` for a rack containing a single sub-rack, `00-31` for a rack containing 2 sub-racks, and `00-47` for a rack containing 3 sub-racks. After `kdump` completes, the `crash` utility can be used on the dump file generated by `kdump`. The `xtdumpsys` SMW utility can be used to extract vmcores from the cluster and store them on the SMW for crash analysis as well.

> **NOTE:** Cray recommends executing the `kdump` utility only if a node has panicked or is hung, or if a dump is requested by Cray.
>
> On the Urika-GX compute nodes, kdump's system facing configuration files are set to have a kdump file stored on a local hard drive partition that is mounted as `/mnt/crash` so the kernel crash dumps are store in `/mnt/crash/var/crash`. Urika-GX has two local HDDs. `kdump` stores the `vmcore` collections on one of these drives. It is advised not to modify the `/etc/kdump.conf` or `/etc/sysconfig/kdump` configuration files.

Using kdump

- Starting `kdump` - Log on to the desired node and use the following command to start the `kdump` utility:

  ```
  $ service kdump start
  ```

- Stopping `kdump` - Log on to the desired node and use the following command to stop the `kdump` utility:

```
$ service kdump stop
```

- Checking the status of `kdump` - Log on to the desired node and use the following command to check the status of the `kdump` utility:

```
$ service kdump status
```

For more information, see the `kdump(8)` and `crash(8)` man pages.

3.7 Cray Lightweight Log Management (LLM) System

The Cray Lightweight Log Management (LLM) system is the log infrastructure for Cray systems and must be enabled for systems to successfully log events. At a high level, a library is used to deliver messages to `rsyslog` utilizing the RFC 5424 protocol; `rsyslog` transports those messages to the SMW and places the messages into log files.

By default, LLM has a log trimming mechanism enabled called `xttrim`.

3.8 Urika-GX Node Power Management

A number of CLI scripts can be used to control node power. In addition, CLI scripts and the Nagios UI can be used to monitor node status on the Urika-GX system.

The command described in this section all need to be executed on the SMW as root.

Start Up a Single Node

Use the **`ux-nid-power-on`** command to start up a node. For example, use the following command to start up node 47:

```
$ ux-nid-power-on 47
```

Start Up All Nodes

Use the `ux-nid-power-on` script to start up all the nodes at once. For more information, see the `ux-nid-power-on` man page

```
# ux-nid-power-on
```

Stop a Single Node

Ensure that the analytics services are stopped on the node and Lustre is unmounted. Use the **`ux-nid-power-off`** command to stop a running node. For example, use the following command to stop node 47:

```
# ux-nid-power-off 47
```

Stop All Nodes

Ensure that the analytics services are stopped on the node and Lustre is unmounted. Use the `ux-nid-power-off` script to stop all the nodes at once.

```
# ux-nid-power-off
```

For more information, see the `ux-nid-power-off` man page.

Check the Status of Nodes

Node status can be viewed using the following mechanisms:

- By using the `xtcli status` command. For more information, see the `xtcli(8)` man page
- By using the `ux-nid-status` command. For more information, see the `ux-nid-status` and `ux-nid-cobbler-status` man pages.
- By using the Nagios UI at `http://`*`machine`*`-smw/nagios/`

3.9 Power Up the Urika-GX System

Prerequisites

This procedure requires root access. The examples of this procedure assumes that the instructions are being carried out on a 3 sub-rack (48 node) system.

About this task

The instructions documented in the procedure can be used for powering up the Urika®-GX system. For detailed information, or for troubleshooting issues, please contact Cray Support.

Procedure

1. Physically power on the Power Distribution Units (PDUs).
2. Turn on the System Management Workstation (SMW).

   It is recommended to wait a few minutes to allow the RC to boot and to verify that the RC has been turned on. This can be achieved by executing the following command:

   ```
   root@smw:~ xtalive -l l1
   The expected response was received.
   ```

   The statement: `"The expected response was received"` indicates that the RC has been turned on.
3. Ensure that `alert` or `reserved` is not displayed in the `Flags` column of results when the following command is executed:

   ```
   root@smw:~ xtcli status -t l0 s0
   ```

   If alerts are discovered and cleared, wait a few seconds and execute the `xtcli -t l0 s0` command again to ensure the alerts did not return. If they do return, contact Cray Support. If the errors do not return, the underlying transient problem can be ignored.

   If alerts are discovered and cleared, wait a few seconds and execute the `xtcli -t l0 s0` command again to ensure the alerts did not return. If they do return, contact Cray Support. If the errors do not return, the underlying transient problem can be ignored.

4. Power up the Aries Network Card Controllers (ANCCs) by executing the following command:

   ```
   root@smw:~ xtcli power up s0
   ```

5. Verify that the dANCs are in the 'ready' state. Also ensure that alert or reserved is not displayed in the Flags column of results when the following command is executed:

   ```
   root@smw:~ xtcli status -t l0 s0
   ```

   if alert or reserved is displayed in the Flags column of results when the preceding command is executed, use the xtclear_alert or xtclear_reserve command to clear those statuses.

6. Ensure that the ANCCs are powered on by executing the command:

   ```
   root@smw:~ xtalive
   All 6 expected responses were received.
   ```

   The above response may vary, depending on the number of sub-racks in the system. The previous example displays a response the system would return if there are 3 sub-racks in the rack.

7. Become the crayadm user.

   ```
   root@smw:~ su - crayadm
   ```

8. Initialize the High Speed Network (HSN) by executing the command:

   ```
   crayadm@smw:~ xtbounce --linktune s0
   ```

   This may step a few minutes to finish executing.

9. Use the HSS rtr command to request routing for the HSN

   ```
   crayadm@smw:~ rtr -R s0
   ```

10. Exit to become the root user again.

    ```
    crayadm@smw:~ exit
    ```

11.

12. Power on all the compute nodes of the system by executing the ux-nid-power-on script.

    For more information, see the ux-nid-power-on man page.

    ```
    root@smw:~ ux-nid-power-on all
    ```

    Once all the node IDs have reported that they are up, wait for 5 or so minutes. The login nodes take longer than the rest of the nodes.

13. Optional: Start the tenant VMs. For more information, see the ux-tenant-start man page.

14. Resolve any issues related to the Romana service.

    ```
    root@smw:~ main_romana_df_gateway.yml
    ```

15. Verify that the compute nodes have been turned on.

```
root@smw:~ xtcli status s0
```

All the nodes should send a response. If a response is not received from all the nodes, do not proceed.

16. Execute the following command:

```
root@smw:~ pall uptime |sort
```

Once all the node IDs have reported that they are up, wait for 5 or so minutes. The login nodes take longer than the rest of the nodes.

17. Mount Lustre

```
root@smw:~ pall mount /mnt/lustre
```

Here pall is an alias for `pdsh -w` and should be configured in `/etc/profile.d/smw_deploy_profile.sh`

18. Confirm that the required nodes have a mount point by executing the following command:

```
root@smw:~ pall df -k |grep lustre |sort
```

19. Start up the analytic applications using the `urika-start` script.

For more information, see the `urika-start` man page.

```
root@smw:~ urika-start
```

This starts up all the analytic applications installed on the system.

20. Pause for 5 minutes or so.

It is important to allow Hadoop to perform the required set up before proceeding. Hadoop will initially be in safe mode when it starts.

21. Verify that the analytic services have started.

```
root@smw:~ urika-inventory
```

22. Confirm services are up on all the expected nodes and that all processes are listed as online when the `urika-state` command is executed.

```
root@smw:~ urika-state
```

23. Optional: Check the general health of the system.

```
root@smw:~ urika-check-platform
```

3.10 Power Down the Urika-GX System

Prerequisites

Instructions in this procedure require root access. This procedure assumes that the instructions are being carried out on a 3 sub-rack system.

About this task

The instructions documented in the procedure should be following in the order listed to power off the Urika®-GX system. For detailed information, or for troubleshooting issues, please contact Cray Support. It is assumed that the instructions in procedure are being carried out on a 48 node system. Please use node IDs according to the actual system configuration.

Procedure

1. Log on to the System Management Workstation (SMW) as root.
2. Power down the analytic applications using the `urika-stop` script.

   ```
   root@smw:~ urika-stop
   ```

   For more information, see the `urika-stop` man page.
3. Execute the `urika-state` command to ensure that analytic applications have shut down.

   The HA Proxy service may still be running.

   ```
   root@smw:~ urika-state
   ```

   For more information, see the `urika-state` man page.
4. Unmount all external Lustre mounts.

   This may not be required if `/mnt/lustre` or a Sonexion storage system are not used. For more information, contact support.

   ```
   root@smw:~ pall umount /mnt/lustre
   ```
5. Optional: Log on to a storage node (`nid00031` on a 3 sub-rack system), switch to the `lustre-utils` directory and stop Lustre if using a Direct Attached Lustre (DAL) storage system.

   This step is needed only if a Direct Attached Lustre (DAL) is used. Do not perform this step if using the Sonexion storage system.

   ```
   root@nid00031:~ cd /etc/opt/cray/lustre-utils
   root@nid00031:~ ./lustre_control stop -a
   ```
6. Log on to the SMW.
7. Stop all the tenant VMs.

   ```
   root@smw:~ ux-tenant-status -R -F name,state|grep -v =notfound|awk \
   '{print $1}'|awk -F= '{print $2}'|xarg
   ```
8. Execute the `ux-nid-power-off` command to power off all the nodes.

   ```
   root@smw:~ ux-nid-power-off all
   ```

   For more information about the `ux-nid-power-off` command, see the `ux-nid-power-off` man page.
9. Log on to the SMW and execute the `capmc node_status` command.

It is important to wait until all the nodes are in the `off` state before proceeding. This process takes around 5-10 minutes. The `capmc` command should be executed on a different shell environment than that used to execute the `ux-nid-power-off` command. if a new shell is used for executing the `ux-nid-power-off` command, the `/root/admin.openrc` file would need to be sourced again, as shown in the previous step.

```
root@smw:~ capmc node_status
```

10. Ensure that `[noflags|]` is displayed in the `Flags` column of the results when the following commands are executed:

```
root@smw:~ xtcli status -t node s0
Network topology: class 0
Network type: Aries
           Nodeid: Service  Core Arch|  Comp state      [Flags]
--------------------------------------------------------------------------------
---
         r0s0c0n0:       -       ATHE|       off        [noflags|]
         r0s0c0n1:       -       ATHE|       off        [noflags|]
         r0s0c0n2:       -       ATHE|       off        [noflags|]
         r0s0c0n3:       -       ATHE|       off        [noflags|]
         r0s0c0n4:       -       ATHE|       off        [noflags|]
         r0s0c0n5:       -       ATHE|       off        [noflags|]
         r0s0c0n6:       -       ATHE|       off        [noflags|]
         r0s0c0n7:       -       ATHE|       off        [noflags|]
         r0s0c1n0:       -       ATHE|       off        [noflags|]
         ...
```

11. Power down the dual Aries Network Cards (dANCs) by executing the following command:

```
root@smw:~ xtcli power down s0
```

12. Ensure that `[noflags|]` is displayed in the `Flags` column of the results when the following command is executed

```
root@smw:~ xtcli status -t aries s0
Network topology: class 0
Network type: Aries
           Nodeid: Service  Core Arch|  Comp state      [Flags]
--------------------------------------------------------------------------------
---
         r0s0c0n0:       -       ATHE|       off        [noflags|]
         r0s0c0n1:       -       ATHE|       off        [noflags|]
         r0s0c0n2:       -       ATHE|       off        [noflags|]
         r0s0c0n3:       -       ATHE|       off        [noflags|]
         r0s0c0n4:       -       ATHE|       off        [noflags|]
         r0s0c0n5:       -       ATHE|       off        [noflags|]
         r0s0c0n6:       -       ATHE|       off        [noflags|]
         r0s0c0n7:       -       ATHE|       off        [noflags|]
         r0s0c1n0:       -       ATHE|       off        [noflags|]
         ...
```

13. Turn off the SMW.

```
root@smw:~ shutdown -h
```

14. Ensure that Sonexion is unmounted before shutting down breakers on rack, if Lustre is installed on the same rack.

15. Physically power down the rack by turning off the Power Distribution Units (PDUs).

3.11 Urika-GX CLI Commands for Managing Services

Urika-GX features a number of scripts for controlling and monitoring installed analytic applications. The commands need to be run as root from the SMW.

`urika-start`

The `urika-start` command is used to start analytic services on the Urika-GX system. This command can be used to start services on one node, a number of nodes, or on all the system nodes. To specify more than one node, specify the node IDs separated by commas. The list of services that are started using this command depends on the service mode (default or secure) that the system is running in. This command starts all the running services (supported in the current service mode) if used without any options. To start a specific service, use the `-s` or `--service` option to specify the name of that service.

Using the `urika-start` command without the `-s` option will not start the Docker or Spark Thrift Server services. Use the `-s` option to start these services, as shown below:

```
urika-start -s spark_thrift
```

```
urika-start -s docker
```

CAUTION: Before starting the Spark History server, it is required to start the following services in the order listed:

1. HDFS NameNode
2. HDFS DataNode

For more information, see the `urika-start(8)` man page.

Start all the services on all the nodes

```
# urika-start
```

Start all the services on a particular node

```
# urika-start --limit nid00014
```

Start all the services on a few nodes.

```
# urika-start --limit nid00014, nid00015
```

Start a particular service (HUE in this example) on all the nodes:

```
# urika-start --service hue
```

Start a particular service (HUE in this example) on a particular node:

```
# urika-start --service hue
```

urika-stop

The `urika-stop` command is used to stop analytic services running on one node, a number of nodes or on all the Urika-GX system nodes. To specify more than one node, specify the node IDs separated by commas. To stop a specific service, use the `-s` or `--service` option to specify the name of that service. This command stops all the running analytic services if used without any options.

For more information, see the `urika-stop` man page.

Stop all the running services on all the nodes.

```
# urika-stop
```

Stop all the running services on a particular node.

```
# urika-stop --limit nid00014
```

Stop all the services on a few nodes.

```
# urika-stop --limit nid00014, nid00015
```

Stop a particular service (Spark in this example) on all the nodes:

```
# urika-stop --service spark
```

Stop a particular service (Spark in this example) on a particular node.

```
# urika-stop --service spark
```

urika-inventory

The `urika-inventory` command displays the list of analytic services running on each node, as well as the IP address of each node. For more information, see the `urika-inventory(8)` man page.

Display the list of services running on each Urika-GX node:

```
# urika-inventory
```

urika-rev

The `urika-rev` command displays the version of deployed analytic components on Urika-GX nodes and checks if a different version of a software component is installed on individual nodes.

For more information, see the `urika-rev(8)` man page.

Display the version components deployed on Urika-GX:

```
# urika-rev
```

urika-state

The `urika-state` command displays the status of Urika-GX analytic services and the service mode the system is operating in.

For more information, see the `urika-state(8)` man page.

Display status of services:

```
# urika-state
```

Sequence of Execution of Scripts

The urika-state command can be used to view the state services running on Urika-GX. Use this command to ensure that the following dependencies are met:

- Before executing the `start -s mesos_master` command, ensure that the ZooKeeper service is up and running.
- Before executing the `stop -s mesos_master` command, ensure that the Marathon and Mesos slave services have already been stopped
- Before executing the `start -s mesos_slave` command, ensure that the ZooKeeper and Mesos Master services are up and running.

In addition, it is recommended to stop the Mesos slave service before stopping Marathon via the `urika-stop -s marathon` command. The `urika-stop -s mesos_slave` command does not have any dependencies, though any running tasks would be lost. Therefore, it is recommended to ensure that all the existing jobs finish running before executing the `stop -s mesos_slave` command.

Additional Scripts

- `ux-nid-deploy` - Deploys a new image to one or more nodes. For more information, see the `ux-nid-deploy` man page.
- `ux-nid-undeploy` - Wipes out all data from the node. For more information, see the `ux-nid-undeploy` man page.
- `ux-nid-partition-add` - Partitions a node. For more information, see the `ux-nid-partition-add` man page.
- `ux-nid-partiition-view` - Displays node partitions. For more information, see the `ux-nid-partiition-view` man page.

3.12 Remote HDFS Remote Access and Multihoming on Urika-GX

- **Multihoming** - Urika-GX compute nodes can be multihomed between the Aries HSN and the operational network.
- **Remote HDFS Access** - HDFS can be set up on Urika-GX to enable external access.

For assistance in enabling the aforementioned features, please contact Cray Support.

3.13 Update the InfluxDB Data Retention Policy

Prerequisites

This procedure requires root privileges. Before performing this procedure, use the `urika-state` command to ensure that the system is operating in the service mode that supports using InfluxDB. For more information, see the `urika-state` man page and refer to *Urika-GX Service Modes* on page 173.

About this task

The data retention policy for InfluxDB defaults to infinite, i.e. data is never deleted. As a result, Spark and Hadoop Grafana dashboards may take longer to load and InfluxDB may take up more space than necessary. To reduce the amount of space being used by InfluxDB, the data retention policy for each database needs to be reduced, as described in this procedure. Reducing the data retention policy for Spark and Hadoop databases can reduce the load time of the Spark and Hadoop Grafana dashboards.

Procedure

1. Log on to login2 and become root.

2. Switch to the `/var/lib/influxdb/data` directory.

   ```
   # cd /var/lib/influxdb/data
   ```

3. Use the du command to show how much space being used.

   The sizes below are shown as examples. Actual sizes on the system may vary.

   ```
   $ du -sh *
   14G    Cray Urika GX
   1.5G      CrayUrikaGXHadoop
   906M      CrayUrikaGXSpark
   21M    _internal
   #
   ```

4. Connect to InfluxDB to view the current data retention policy.

   ```
   # /bin/influx
   Visit https://enterprise.influxdata.com to register for updates, InfluxDB server
   management, and monitoring.
   ```

```
Connected to http://localhost:8086 version 0.12.2
InfluxDB shell 0.12.2
> show retention policies on "Cray Urika GX"
name     duration    shardGroupDuration    replicaN    default
default    0               168h0m0s        1         true
```

5. Update the data retention policy according to requirements.

   In this example the data retention duration is changed from 0 (forever) to 2 weeks (504 hours).

   ```
   > alter retention policy default on "Cray Urika GX" Duration 2w
   > show retention policies on "Cray Urika GX"
   name     duration    shardGroupDuration    replicaN    default
   default    504h0m0s    24h0m0s             1         true
   > exit
   ```

   The change will take a while to be applied. The default is 30 minutes.

6. Verify that the data retention change has taken effect

   ```
   # du -sh *
   3G    Cray Urika GX
   1.5G    CrayUrikaGXHadoop
   906M    CrayUrikaGXSpark
   21M    _internal
   ```

3.14 Service to Node Mapping

The list of services available for use depends on the security mode the system is running under. For more information, refer to *Urika-GX Service Modes* on page 173.

Table 9. Urika-GX Service to Node Mapping (2 Sub-rack System)

Node ID(s)	Service(s) Running on Node /Role of Node
`nid00000`	• ZooKeeper • ncmd • Mesos Master • Marathon • Primary HDFS NameNode • Hadoop Application Timeline Server • Collectl • nrpe • kubelet
`nid000[01-07, 09-13, 17-29]`	• Collectl • Mesos Slave • Data Node

Node ID(s)	Service(s) Running on Node /Role of Node
	● YARN Node Manager (if running) ● nrpe ● kubelet
`nid00008`	● ZooKeeper ● Secondary HDFS NameNode ● Mesos Master ● Oozie ● Hive Server2 ● Spark Thrift Server ● Hive Metastore ● WebHCat ● Postgres database ● Marathon ● YARN Resource Manager ● Collectl ● nrpe ● kubelet
`nid00014` (Login node 1)	● HUE ● HAProxy ● Collectl ● Urika-GX Applications Interface UI ● Cray Application Management UI ● Jupyter Notebook ● Service for flexing a YARN cluster ● Documentation and Learning Resources UI ● nrpe ● kubelet ● Kubernetes Controller
`nid00015`, `nid00031` (I/O nodes)	These are nodes that run Lustre clients and nrpe
`nid00016`	● ZooKeeper ● Mesos Master ● Marathon ● Hadoop Job History Server ● Spark History Server

Node ID(s)	Service(s) Running on Node /Role of Node
	• Collectl • nrpe • kubelet
`nid00030` (Login node 2)	• HUE • HA Proxy • Collectl • Service for flexing a YARN cluster • Grafana • InfluxDB • nrpe • kubelet

Table 10. Urika-GX Service to Node Mapping (3 Sub-rack System)

Node ID(s)	Service(s) Running on Node /Role of Node
`nid00000`	• ZooKeeper • ncmd • Mesos Master • Marathon • Primary HDFS NameNode • Hadoop Application Timeline Server • Collectl • nrpe • kubelet
`nid00001-nid00015`, `nid00017-nid00029`, `nid00033-nid00045`	• Collectl • Mesos Slave • Data Node • YARN Node Manager (if running) • nrpe • kubelet
`nid00016`	• ZooKeeper • Mesos Master • Marathon • Hadoop Job History Server

Node ID(s)	Service(s) Running on Node /Role of Node
	● Spark History Server ● Collectl ● nrpe ● kubelet
`nid00030` (Login node 1)	● HUE ● HA Proxy ● Collectl ● Urika-GX Applications Interface UI ● Jupyter Notebook ● Service for flexing a YARN cluster ● Documentation and Learning Resources UI ● nrpe ● kubelet ● Kubernetes Controller
`nid00031`, `nid00047` (I/O nodes)	These are nodes that run Lustre clients ● kubelet
`nid00032`	● ZooKeeper ● Secondary NameNode ● Mesos Master ● Oozie ● Hive Server2 ● Hive Metastore ● WebHcat ● Postgres database ● Marathon ● YARN Resource Manager ● Collectl ● Spark Thrift Server ● nrpe ● kubelet
`nid00046` (Login node 2)	● HUE ● HA Proxy ● Collectl ● Grafana

Node ID(s)	Service(s) Running on Node /Role of Node
	● InfluxDB ● Service for flexing a YARN cluster ● nrpe ● kubelet

For additional information, use the `urika-inventory` command as root on the SMW to view the list of services running on node, as shown in the following example:

```
# urika-inventory
```

For more information, see the `urika-inventory` man page.

3.15 Image Management with Docker and Kubernetes

About Docker

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security enables running many containers simultaneously on a given host. Because of the lightweight nature of containers, users can run more containers on a given hardware combination than if using virtual machines.

Images shipped with the system are managed by Docker when Urika-GX is operating under the default service mode. The SMW hosts Urika-GX's container repository, which is a container itself and can only host Cray developed containers currently.

NOTE: Building and managing new Docker images is currently not supported on Urika-GX.

For more information, visit *https://www.docker.com*.

About Kubernetes

Kubernetes is used for automating deployment, scaling and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery. It performs container management tasks, such as, running containers across many different machines, scaling up or down by adding or removing containers when demand changes, keeping storage consistent with multiple instances of an application, distributing load between the containers and launching new containers on different machines if something fails.

On Urika-GX, Kubernetes is used to manage containers in the secure service mode.

NOTE: Currently, Kubernetes supports only Spark images on Urika-GX.

For more information, visit *https://kubernetes.io/*.

About the Cray Spark Image

In order to run Spark on Kubernetes, Urika-GX ships with customized Spark images, which are based on the Spark version used on the system.

3.15.1 Execute Spark Jobs on Kubernetes

Spark jobs run inside containers, which are managed via Kubernetes on the Urika-GX system. This section provides some examples for executing Spark jobs, retrieving output, and viewing logs etc.

The system needs to be running in the secure mode and the user needs to be logged on a login node to run the examples shown in this section.

Running a Spark Pi Example Job

The following examples shows how to run a simple Spark Pi job inside a container. It uses `spark.app.name` as the Spark job's name.

```
$ spark-submit --class org.apache.spark.examples.SparkPi \
--conf spark.app.name=spark-pi \
local:///opt/spark/examples/target/scala-2.11/jars/spark-
examples_2.11-2.2.0-k8s-0.5.0.jar
```

The path to the JAR file must be relative to the path inside the container, not the path that exists on the system. Inside the container, the Spark home directory is `/opt/spark` instead of `/opt/cray/spark2/default`.

The preceding command produces output similar to the following (only a portion of the output is shown below for brevity):

```
2018-02-26 16:16:47 INFO  HadoopStepsOrchestrator:54 - Hadoop Conf
directory: /etc/hadoop/conf
2018-02-26 16:16:47 INFO  HadoopConfBootstrapImpl:54 -
HADOOP_CONF_DIR defined. Mounting Hadoop specific files
2018-02-26 16:16:48 WARN  NativeCodeLoader:62 - Unable to load native-
hadoop library for your platform... using builtin-java classes where
applicable
2018-02-26 16:16:48 INFO  LoggingPodStatusWatcherImpl:54 - State
changed, new state:
     pod name: spark-pi-1519683406605-driver
     namespace: username
     labels: spark-app-selector ->
spark-027d506894bd4b2ca86692f03f9fab5a, spark-role -> driver
     pod uid: bceaf8b2-1b42-11e8-8b39-001e67d33475
     creation time: 2018-02-26T22:16:48Z
     service account name: spark
     volumes: spark-local-dir-0-tmp, hadoop-properties, spark-token-
pxz79
     node name: N/A
     start time: N/A
     container images: N/A
     phase: Pending
     status: []
.......
```

The output of the Spark Driver can be viewed by executing `kubectl logs` *pod_name* and looking at the pod's logs. The pod's name is displayed near the top of the console output, as shown in the preceding example. Execute the `kubectl logs` *pod_name* command and `grep` the output, as shown below:

```
$ kubectl logs spark-pi-1519683406605-driver | grep "is roughly"
Pi is roughly 3.1351356756783786
```

Spark Executor pods are cleaned up and deleted after they finish running. Therefore, their output is not accessible.

Running a Spark Pi Example Job

A Pyspark pi example job is very similar to a Scala Spark PI, but information is specified slightly differently. If there are any JAR files, they should be provided via the `--jars` flag.

```
$ bin/spark-submit --conf spark.app.name=pyspark-pi \
--jars local:///opt/spark/examples/target/scala-2.11/jars/spark-
examples_2.11-2.2.0-k8s-0.5.0.jar \
local:///opt/spark/examples/src/main/python/pi.py
```

Execute the `kubectl logs` *`pod_name`* command and `grep` the output, as shown below:

```
# kubectl logs pyspark-pi-1519684161476-driver | grep "is roughly"
Pi is roughly 3.141600
```

Using HDFS

The `HADOOP_CONF_DIR` parameter will automatically be set to the appropriate value for the current user during Spark start up.

How to Run Jobs and Use the Resource Staging Server

Simply provide the location of the Spark jar and files on the local file system and they will be loaded into the Spark Resource Staging Server so that resources will be available inside the Spark container.

```
$ bin/spark-submit --class TriangleCounts --conf spark.app.name=spark-
triangles \
/home/users/builder/nid00006/workspace/socrates-cactus-spark-tests/
target/scala-2.11/spark-tests_2.11-1.0.jar \
/user/builder/datasets/cactus-spark-triangles/small-triangles.txt
```

Check the results using the pod name.

```
$ kubectl logs spark-triangles-1520878896538-driver | grep "riangles:"
numTriangles: 10624230
Number of triangles: 3541410
```

Resource Configuration for Spark Jobs

The following Spark configuration settings may be used to control the amount of resources that Spark will request from Kubernetes for any job launched using `spark-submit` under the secure service mode, i.e., under Kubernetes:

Table 12. Spark Configuration Settings and their Default Values

Configuration Setting	Default Value	Notes
`spark.executor.instances`	5	Number of Spark executor containers that will be launched to provide job execution under Kubernetes
`spark.executor.cores`	1	Number of cores requested per executor
`spark.executor.memory`	96g	Amount of memory requested per executor
`spark.driver.cores`	1	Number of cores requested for the driver This should be increased if a job does a lot of work in the driver e.g. aggregations, result collection
`spark.driver.memory`	16g	Amount of memory requested per driver

CAUTION: Please be aware that due to Kubernetes's service scheduling mechanism, there are always some services running on all the nodes, using fractional CPU cycles. This may block any requests that attempt to use the maximum number of cores on the system because a small fraction of those cores is already allocated.

If a job is not showing any progress, find out the current status of the associated Kubernetes pod by running `kubectl describe pod` *`pod-name`*. If there are insufficient resources to launch a job, the system will return a message similar to the following at the end of the output:

```
Warning  FailedScheduling  55s (x8 over 1m)  default-scheduler  0/16 nodes are available: 1
PodToleratesNodeTaints, 16 Insufficient cpu.
```

4 System Monitoring

4.1 System Monitoring Tools

Urika-GX provides many tools for monitoring system components, resources and services.

Table 13. Monitoring Tools

Monitoring Tool	Monitored Component/Service/Resource
HSS	Helps monitor physical system components, such as PCIe channels and Dual Aries Network Card (dance).
iSCB	Helps monitor sub-rack level attributes, such as power supply, amperage, fan status, and temperature info are monitored via the iSCB. **CAUTION:** iSCB CLI commands other than the `status` command should NOT be executed on the Urika-GX system, unless advised by Cray Support. For more information, contact Cray support.
The `capmc`, `ux-nid-cobbler-status` and `ux-nid-status` commands	Helps monitor node status.
Nagios	Helps monitor: ● CPU - per node and aggregated ● Memory - per node and aggregated ● Storage ○ Lustre ○ SSD and HDD per node and aggregated ● Management, operational and Aries network bandwidth used ● Node status
Grafana	Provides information about utilization of system resources, such as CPU, memory, I/O, etc.
`urika-state` command	Retrieves the status of analytic applications.

4.2 Monitor Resource Utilization and Node Status Using Nagios

Prerequisites

- As an admin, run the following command to add user `nagios` to the tenant VM before using Nagios:

  ```
  # ux-tenant-add-user -u nagios -s /bin/bash tenant
  ```

- Ensure that the Nagios service is running.

About this task

Nagios is used on Urika-GX to monitor:

- Node status
- Usage of:
  - CPU (per node and aggregated)
  - Memory (per node and aggregated)
  - Bandwidth (per node and aggregated) of management, HSS, operational networks and the Aries High Speed Network (HSN) network. Also provides per node and aggregated bandwidth for the Aries High Speed network.
  - Lustre, HDD and SSD storage (per node and aggregated)

In addition, Nagios can be used on Urika-GX to create and send alert notifications.

For more information, refer to *Tenant Management* on page 185

Procedure

1. Access the Nagios UI by pointing a browser at `http://`*`machine`*`-smw/nagios/`
2. Enter `crayadm` as the user name and `initial0` as the password to log on to the UI.

   These are the default credentials for logging on to the Nagios UI on Urika-GX.

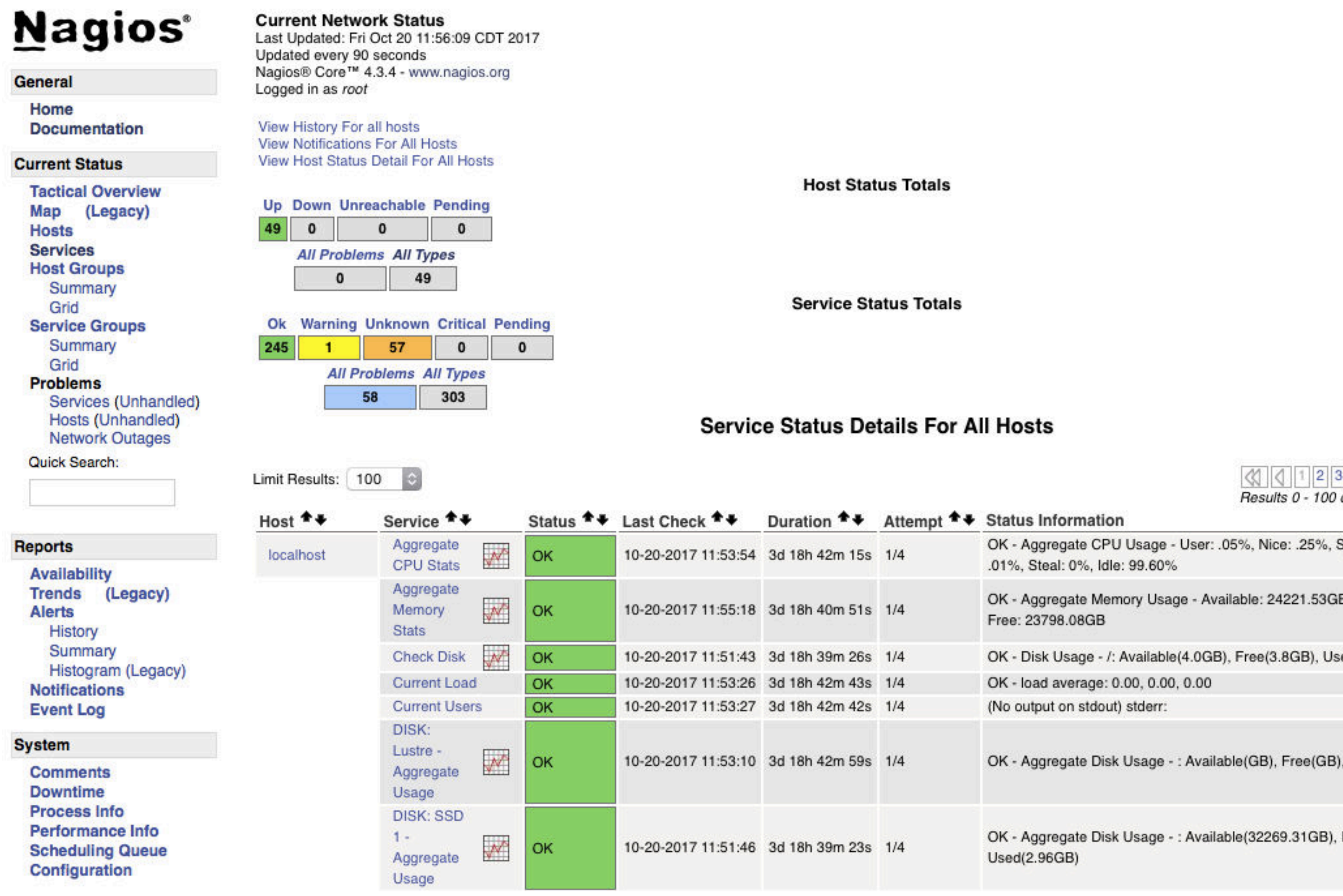
   If the system is switched to the secure service mode, the administrator will need to run the following command to add user `nagios` to the tenants:

   ```
   # ux-tenant-add-user -u nagios -s /bin/bash
   ```

3. Use the Nagios UI menu options to perform monitoring tasks, such as viewing reports and logs, as well as node, system, storage and network status.

   The following figure displays the **Services** page of the Nagios UI. The chart icon in the **Service** column can be used for viewing metrics/statistical data, exporting page content as PDFs, viewing reports etc.

Figure 3. Nagios Services Page



Nagios®

General
- Home
- Documentation

Current Status
- Tactical Overview
- Map (Legacy)
- Hosts
- Services
- Host Groups
  - Summary
  - Grid
- Service Groups
  - Summary
  - Grid
- Problems
  - Services (Unhandled)
  - Hosts (Unhandled)
  - Network Outages

Quick Search:

Reports
- Availability
- Trends (Legacy)
- Alerts
  - History
  - Summary
  - Histogram (Legacy)
- Notifications
- Event Log

System
- Comments
- Downtime
- Process Info
- Performance Info
- Scheduling Queue
- Configuration

Current Network Status
Last Updated: Fri Oct 20 11:56:09 CDT 2017
Updated every 90 seconds
Nagios® Core™ 4.3.4 - www.nagios.org
Logged in as *root*

View History For all hosts
View Notifications For All Hosts
View Host Status Detail For All Hosts

Host Status Totals

Up	Down	Unreachable	Pending
49	0	0	0

All Problems	*All Types*
0	49

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
245	1	57	0	0

All Problems	*All Types*
58	303

Service Status Details For All Hosts

Limit Results: 100

1 2 3
Results 0 - 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
localhost	Aggregate CPU Stats	OK	10-20-2017 11:53:54	3d 18h 42m 15s	1/4	OK - Aggregate CPU Usage - User: .05%, Nice: .25%, S .01%, Steal: 0%, Idle: 99.60%
	Aggregate Memory Stats	OK	10-20-2017 11:55:18	3d 18h 40m 51s	1/4	OK - Aggregate Memory Usage - Available: 24221.53GE Free: 23798.08GB
	Check Disk	OK	10-20-2017 11:51:43	3d 18h 39m 26s	1/4	OK - Disk Usage - /: Available(4.0GB), Free(3.8GB), Us
	Current Load	OK	10-20-2017 11:53:26	3d 18h 42m 43s	1/4	OK - load average: 0.00, 0.00, 0.00
	Current Users	OK	10-20-2017 11:53:27	3d 18h 42m 42s	1/4	(No output on stdout) stderr:
	DISK: Lustre - Aggregate Usage	OK	10-20-2017 11:53:10	3d 18h 42m 59s	1/4	OK - Aggregate Disk Usage - : Available(GB), Free(GB)
	DISK: SSD 1 - Aggregate Usage	OK	10-20-2017 11:51:46	3d 18h 39m 23s	1/4	OK - Aggregate Disk Usage - : Available(32269.31GB), Used(2.96GB)

Select the **Documentation** link on the Nagios UI and visit and *https://www.nagios.org* and *pnp4nagios.org* for more information.

4.2.1 Configure SSL/TLS for Nagios Core

Prerequisites

- This procedure requires root privileges.
- Ensure that the `mod_ssl` package is installed on the system. If it is not, install it by executing the following as root on the SMW:

```
# yum install -y mod_ssl
```

About this task

This procedure describes how to configure the Nagios Core server to use certificates for SSL/TLS. It can also to be used for troubleshooting SSL/TLS connection related issues.

Procedure

1. Log on to the SMW as root.

   All of the remaining steps will be performed from within the root user's home directory to ensure the created files are not accessible to anyone except the root user.

2. Switch to the user's home directory.

   ```
   # cd ~
   ```

3. Generate a private key file.

   ```
   # openssl genrsa -out keyfile.key 2048
   ```

4. Generate certificate request file.

   ```
   # openssl req -new -key keyfile.key -out certrequest.csr
   ```

   Press the **Enter** key for all the prompts, but enter `hostname-smw` for **Common Name**.

   In the example, notice that `core-013.domain.local` has been used, which means that this is the address to use when the Nagios Core server is accessed via the web browser, otherwise the system will return warnings in the web browser.

   ```
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [XX]:AU
   State or Province Name (full name) []:NSW
   Locality Name (eg, city) [Default City]:Sydney
   Organization Name (eg, company) [Default Company Ltd]:My Company Pty Ltd
   Organizational Unit Name (eg, section) []:
   Common Name (eg, your name or your server's hostname) []:core-013.domain.local
   Email Address []:

   Please enter the following 'extra' attributes
   to be sent with your certificate request
   A challenge password []:
   An optional company name []:
   ```

   As it can been see above, a password was not supplied as it is not necessary.

5. Sign the certificate request.

   At this point, the newly created certificate request needs to be signed by a CA.

   - **Use a trusted CA company**

     If the certificate is to be obtained from a trusted company like VeriSign, send them a copy of the certificate request, which can be viewed by executing the following command:

     ```
     cat certrequest.csr
     ```

Provide the CA with everything returned by the above, including the following lines:

```
-----BEGIN CERTIFICATE REQUEST----- and -----END CERTIFICATE REQUEST-----
```

After receiving the signed certificate, copy the certificate into a new file called `certfile.crt`. The certificate received will contain a lot of random text. Paste that into the new file, which can be modified using any editor, such as vi.

```
vi certfile.crt
```

Paste everything, including the following lines:

```
-----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----
```

Save the file and close the editor. Proceed to step 6.

- **Use a Microsoft Windows CA** - If Microsoft Windows CA is to be used to sign the certificate request, follow the steps in *https://support.nagios.com/kb/article.php?id=597* to obtain a`certfile.crt` file and then proceed to the step 6.
- **Self sign the certificate** - Self-sign the certificate by executing the following command:

```
# openssl x509 -req -days 365 -in certrequest.csr -signkey keyfile.key -out
certfile.crt
```

This should produce output saying that the signature was OK and that it retrieved the private key. Self-signing will result in '`connection not secure`' warnings, as expected. Add an exception for this certificate.

6. Copy the certificate files to the correct location and set permissions

```
# cp certfile.crt /etc/pki/tls/certs/
# cp keyfile.key /etc/pki/tls/private/
# chmod go-rwx /etc/pki/tls/certs/certfile.crt
# chmod go-rwx /etc/pki/tls/private/keyfile.key
```

The SSL file might already exist in `/etc/apache2/2.2/conf.d/ssl.conf`. If this is the case, ignore the following commands:

```
# cp /etc/apache2/2.2/samples-conf.d/ssl.conf /etc/apache2/2.2/conf.d/
# chmod +w /etc/apache2/2.2/conf.d/ssl.conf
```

7. Update the Apache configuration.

a. Open the `/etc/httpd/conf.d/ssl.conf` file.

```
# vi /etc/httpd/conf.d/ssl.conf
```

b. Locate these lines.

```
SSLCertificateFile /etc/pki/tls/certs/certfile.crt
```

c. Modify the preceding lines to:

```
SSLCertificateKeyFile /etc/pki/tls/private/keyfile.key
```

d. Save the file.

8. Restart the Apache web server for the new certificate key to be used.

```
===== CentOS 5.x / 6.x | RHEL 5.x / 6.x | Oracle Linux 5.x / 6.x =====
# service httpd restart
===== CentOS 7.x | RHEL 7.x | Oracle Linux 7.x =====
# systemctl restart httpd.service
```

9. Allow port 443 inbound traffic on the local firewall so that the Nagios Core web interface can be reached.

```
===== CentOS 5.x / 6.x | RHEL 5.x / 6.x | Oracle Linux 5.x / 6.x =====

# iptables -I INPUT -p tcp --destination-port 443 -j ACCEPT
# service iptables save
# p6tables -I INPUT -p tcp --destination-port 443 -j ACCEPT
# service ip6tables save

===== CentOS 7.x | RHEL 7.x | Oracle Linux 7.x =====

# firewall-cmd --zone=public --add-port=443/tcp
# firewall-cmd --zone=public --add-port=443/tcp --permanent
```

10. Test the connection to the server by directing a web browser to `https://serveNname/`.

    There is no `nagios/` extension in the URL. This is to test the connection to Apache to verify that the certificate works.

    If a self-signed certificate warning is returned, add a security exception. The Apache test web page will be displayed upon success. The Nagios Core server can now be accessed by directing a web browser at `https://serverName/nagios/`. More detailed information about this can be found in the following KB article: *https://support.nagios.com/kb/article.php?id=598*. If an error is returned, check the firewall and backtrack through this document, making sure all the steps have been performed.

    If a user points a browser at `http://serverName`, they will be directed to `https://serverName`, which can cause certificate warnings. If it is required to redirect them to `https://serverName.domain.com`, change the `RewriteRule` in the `/etc/httpd/conf/httpd.conf` file.

```
RewriteRule (.*) https://serverName.domain.com%{REQUEST_URI}
```

11. Restart the HTTPD service.

4.2.2 Configure the Nagios Server to Send Email Notifications

Prerequisites

This procedure requires administrative privileges.

About this task

Use the instructions in this procedure to create Email alerts for sending notifications to users when needed.

Procedure

1. Log on to the SMW as root.

2. Stop the Nagios service.

   ```
   # service nagios stop
   ```

3. Add the user's contact name, alias, and Email to the `/usr/local/nagios/etc/objects/contacts.cfg` file using the following format:

   ```
   define contact {
           contact_name                            Contact1
           alias                                   ContactNameAlias
           email                                   email-address
           service_notification_period             24x7
           service_notification_options            w,u,c,r,f,s
           service_notification_commands           notify-service-by-email
           host_notification_period                24x7
           host_notification_options               d,u,r,f,s
           host_notification_commands              notify-host-by-email
   }
   define contact {
           contact_name                            Contact2
           alias                                   ContactNameAlias2
           email                                   email-address
           service_notification_period             24x7
           service_notification_options            w,u,c,r,f,s
           service_notification_commands           notify-service-by-email
           host_notification_period                24x7
           host_notification_options               d,u,r,f,s
           host_notification_commands              notify-host-by-email
   }
   ```

   The `service_notification_options` directive is used to define the service states for which notifications can be sent out to this contact. Valid options are a combination of one or more of the following:

   - w: Notify on WARNING service states
   - u: Notify on UNKNOWN service states
   - c: Notify on CRITICAL service states
   - r: Notify on service RECOVERY (OK states)
   - f: Notify when the service starts and stops FLAPPING
   - n (none): Do not notify the contact on any type of service notifications

   The `host_notification_options` directive is used to define the host states for which notifications can be sent out to this contact. Valid options are a combination of one or more of the following:

   - d: Notify on `DOWN` host states
   - u: Notify on `UNREACHABLE` host states
   - r: Notify on host `RECOVERY` (UP states)
   - f: Notify when the host starts and stops `FLAPPING`
   - s: Send notifications when host or service scheduled downtime starts and ends
   - n (none): Do not notify the contact on any type of host notifications.

   By default, a contact and a contact group are defined for administrators in the `contacts.cfg` file and all the services and hosts will notify the administrators. Add suitable Email ID for the administrator in the `contacts.cfg` file.

4. Add the details (as shown in the following example) to add a group to which the mail needs to be sent:

```
define contactgroup{
        contactgroup_name                  Group1
        alias                              GroupAlias
        members                            Contact1,Contact2
}
```

5. Specify the contact name and contact group name for the services for which the notification needs to be sent in the `/usr/local/nagios/etc/objects/templates.cfg` file.

```
define host{
   name                            gluster-generic-host
   use                             linux-server
   notifications_enabled           1
   notification_period             24x7
   notification_interval           120
   notification_options            d,u,r,f,s
   register                        0
   contact_groups               Group1
   contacts                        Contact1,Contact2
   }

 define service {
   name                            gluster-service
   use                             generic-service
   notifications_enabled          1
   notification_period             24x7
   notification_options            w,u,c,r,f,s
   notification_interval           120
   register                        0
   _gluster_entity                 Service
   contact_groups                 Group1
   contacts                    Contact1,Contact2
}
```

Notifications for individual services can be configured by editing the corresponding node configuration file. For example, to configure notification for the brick service, edit the corresponding node configuration file as shown below:

```
define service {
     use                             brick-service
     _VOL_NAME                       VolumeName
     __GENERATED_BY_AUTOCONFIG       1
     notes                           Volume : VolumeName
     host_name                       RedHatStorageNodeName
     _BRICK_DIR                      brickpath
     service_description             Brick Utilization - brickpath
     contact_groups                 Group1
       contacts                    Contact1,Contact2
}
```

6. Edit the `/usr/local/nagios/etc/objects/commands.cfg` file to add `$NOTIFICATIONCOMMENT$\n` after the `$SERVICEOUTPUT$\n` option in `notify-service-by-email` and in the `notify-host-by-email` command definition.

```
# 'notify-service-by-email' command definition
define command{
        command_name    notify-service-by-email
        command_line    /usr/bin/printf "%b" "***** Nagios *****\n
\nNotification Type: $NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost:
$HOSTALIAS$\nAddress: $HOSTADDRESS$\nState: $SERVICESTATE$\n\nDate/Time:
$LONGDATETIME$\n\nAdditional Info:\n\n$SERVICEOUTPUT$\n $NOTIFICATIONCOMMENT$
\n" | /bin/mail -s "** $NOTIFICATIONTYPE$ Service Alert: $HOSTALIAS$/
$SERVICEDESC$ is $SERVICESTATE$ **" $CONTACTEMAIL$
        }
```

   This section can also be modified to change the service which sends mail. To do so, change `/bin/mail -s` to the preferred mail service, for example `/bin/sendmail`.

7. Restart the Nagios service.

```
# service nagios restart
```

   The Nagios server sends notifications during status changes to the mail addresses specified in the file. It is important to note that:

   - By default, the system ensures three occurrences of the event before sending mail notifications.
   - By default, Nagios mail notifications are sent using the `/bin/mail` command. To change this, modify the definition for the `notify-host-by-email` and `notify-service-by-email` commands in the `/usr/local/nagios/etc/objects/commands.cfg` file and configure the mail server accordingly.

4.2.3 Change the Default Log File Path and Rotation Interval

Prerequisites

This procedure requires root privileges.

About this task

Nagios log files are located at `/usr/local/nagios/var/nagios.log` by default. This procedure provides instructions for modifying this path. In addition, this procedure provides instructions for adjusting the default log level and the default location of archived log files.

Procedure

1. Log on to the SMW as root.

```
# ssh root@machine-smw
```

2. Stop the Nagios service if it is running.

```
# service nagios stop
```

3. Switch to the `/usr/local/nagios/etc` directory.

   ```
   # cd /usr/local/nagios/etc
   ```

4. Use an editor to modify the `nagios.cfg` configuration file as desired.

   - Locate the line `log_file=/usr/local/nagios/var/nagios.log` This line specifies the default path to the log file. Modify the path as desired.
   - Locate the line `log_rotation_method=d`. This line specifies the rotation interval. d stands for daily. Modify this line as desired. The other options are defined in the comments block above it.
   - Locate the line `log_archive_path=/usr/local/nagios/var/archives`. This is where rotated log files are archived. Modify the path as desired.

5. Save and quit the `nagios.cfg` file.

6. Restart the Nagios service.

   ```
   # service nagios restart
   ```

4.2.4 Configure Email Alerts

Prerequisites

This procedure requires root privileges.

About this task

This procedure provides instructions for updating certain Nagios configuration files for configuring alerts. Specifically, it describes how to modify the `contacts.cfg` file to update and define a list of contacts for sending Email alerts to. In addition, it describes how to modify the `localhost.cfg` and `node_services.cfg` files to configure alerts for services running on nodes.

Procedure

1. Log in to the SMW as root.

   ```
   # ssh root@machine-smw
   ```

2. Stop the Nagios service if it is running.

   ```
   # service nagios stop
   ```

3. Switch to the `/usr/local/nagios/etc/objects` directory.

   ```
   # cd /usr/local/nagios/etc
   ```

4. Use an editor to modify the `contacts.cfg` configuration file as desired.

   A contact group will be defined by default in the `contacts.cfg` file.

   ```
   define contactgroup{
           contactgroup_name       admins
   ```

```
        alias                   Nagios Administrators
        members                 admin1
        }
```

5. Add as many members as needed.

   Multiple contact groups can be defined if needed.

   A contact is defined in this file by default.

```
define contact{
        contact_name                    admin1                  ; Short name of user
        use                             generic-contact         ; Inherit default values from generic-contact
template (defined above)
        alias                           full name               ; Full name of user
        email                           admin1@company.com      ; <<***** Change this to the required email
address *****
        }
```

6. Updated the alias and Email as desired.

7. Save and quit the `contacts.cfg` file.

8. Open `localhost.cfg` and find the service that alerting needs to be set up for.

   The following example shows how to set up the service for aggregate CPU usage.

```
define service{
        use                             local-service
        host_name                       localhost
        service_description             Aggregate CPU Stats
        check_command                   check_cpu_aggr!0.5!0.8
        notifications_enabled           1
        notification_period             24x7
        notification_options            w,u,c,r,f
        notification_interval           60
        contact_groups                  admins
        contacts                        admin1
}
```

9. Define the notification period, interval, options, the contact groups and contacts as desired.

   For more information, refer to *https://access.redhat.com/documentation/en-US/Red_Hat_Storage/3/html/Console_Administration_Guide/Configuring_Nagios_to_Send_Mail_Notifications.html*

10. Save and quit the `localhost.cfg` file.

11. Restart the Nagios service.

```
# service nagios restart
```

12. Repeat this process for `node_services.cfg` file as well.

4.2.5 Modify Nagios Plug-in Threshold

Prerequisites

This procedure requires root privileges.

About this task

Each Cray plugin requires thresholds for the metrics they monitor. All plugins require a `warning` and `critical` level, set via parameters. Some require a third parameter. These plugins include `cray_check_disk`, `cray_check_disk_aggr`, and `cray_check_network`. Executing the plugin with the `-h` option will list the available options and the arguments to pass in for these options. For example, `./cray_check_disk -h`. A brief description of what each plugin measures will also be available via the `-h` option. The plugins are located in `/usr/local/nagios/libexec` on either the SMW or one of the nodes. The warning and critical levels are digits (up to 4 decimal points) between 0.0 and 1.0, which represent the percentage of the metric.

The following plugins are configurable on the SMW at the default path `/usr/local/nagios/libexec`:

- `cray_check_cpu_aggr`
- `cray_check_disk`
- `cray_check_disk_aggr`
- `cray_check_memory_aggr`
- `cray_check_network`
- `cray_check_node_state`.

The following plugins are configurable on each of the nodes at the default path (`/usr/local/nagios/libexec`):

- `cray_check_aries` (only available on compute nodes)
- `cray_check_cpu`
- `cray_check_disk`
- `cray_check_memory`

For details on configuring custom services and hosts please refer to Nagios documentation.

Procedure

1. Log on to the SMW as root.

   ```
   # ssh root@machine-smw
   ```

2. Stop the Nagios service if it is running.

   ```
   # service nagios stop
   ```

3. Switch to the `/usr/local/nagios/etc/objects` directory.

4. Modify the `localhost.cfg` file as needed.

There is a `define service` block as shown below for the aggregate CPU plug-in.

```
define service{
        use                             local-service
        host_name                       localhost
        service_description             Aggregate CPU Stats
        check_command                   check_cpu_aggr!0.5!0.8
}
```

Here, a custom command name is defined, the name being arbitrary. In this example it it called `check_cpu_aggr`, which is the last line shown above. It is followed by an exclamation mark, a number, another exclamation mark, and another number. The '!' separates the parameters required by the plugin (`-w` and `–c`), and the numbers are the arguments to these plugins (i.e. the `warning` and `critical` levels). So in this instance, the warning level is set to 0.5 and critical to 0.8

5. Adjust the `warning` and `critical` levels for this plugin
6. Save and quit the `localhost.cfg` file.
7. Restart the Nagios service.

   ```
   # service nagios restart
   ```

8. Switch to the `/usr/local/nagios/etc` directory.
9. Modify the `nrpe.conf` configuration file as needed.
   a. Define the command name and the path including the arguments to the plugins to be executed by this command.

      The command for the `cray_check_cpu` plugin is shown below:

      ```
      command[check_cpu]=/usr/local/nagios/libexec/cray_check_cpu -w 0.5 -c 0.8
      ```

      In the preceding example, `-w` is the warning level, whereas `-c` is the critical level.
   b. Update the thresholds as needed.
10. Save and quit the `nrpe.conf` file.
11. Restart the `nrpe` service.

    ```
    # service nrpe restart
    ```

12. Repeat this process for all the nodes and services in those nodes that need adjusting.

4.3 Get Started with Using Grafana

Grafana is a feature-rich metrics, dashboard, and graph editor. It provides information about utilization of system resources, such as CPU, memory, I/O, etc.

Major Grafana components and UI elements include:

- **Dashboard** - The **Dashboard** consolidates all the visualizations into a single interface. Urika-GX ships with pre-defined dashboards for resource utilization information. For information about these dashboards, see *Default Grafana Dashboards* on page 77.
- **Panels** - The **Panel** is the basic visualization building block of the GUI. Panels support a wide variety of formatting options and can be dragged, dropped, resized, and rearranged on the **Dashboard**.
- **Query Editor** - Each Panel provides a **Query Editor** for the data source, which is InfluxDB on the Urika-GX system. Information retrieved from the **Query Editor** is displayed on the **Panel**.
- **Data Source** - Grafana supports many different storage back-ends for retrieving time series data. Each **Data Source** has a specific **Query Editor** that is customized for the features and capabilities that the particular **Data Source** exposes. Urika-GX currently supports InfluxDB as its data source.
- **Organization** - Grafana supports multiple organizations in order to support a wide variety of deployment models. Each **Organization** can have one or more **Data Sources**. All Dashboards are owned by a particular Organization.
- **User** - A **User** is a named account in Grafana. A user can belong to one or more **Organizations**, and can be assigned different levels of privileges via roles.
- **Row** - A **Row** is a logical divider within a **Dashboard**, and is used to group **Panels** together.

Roles

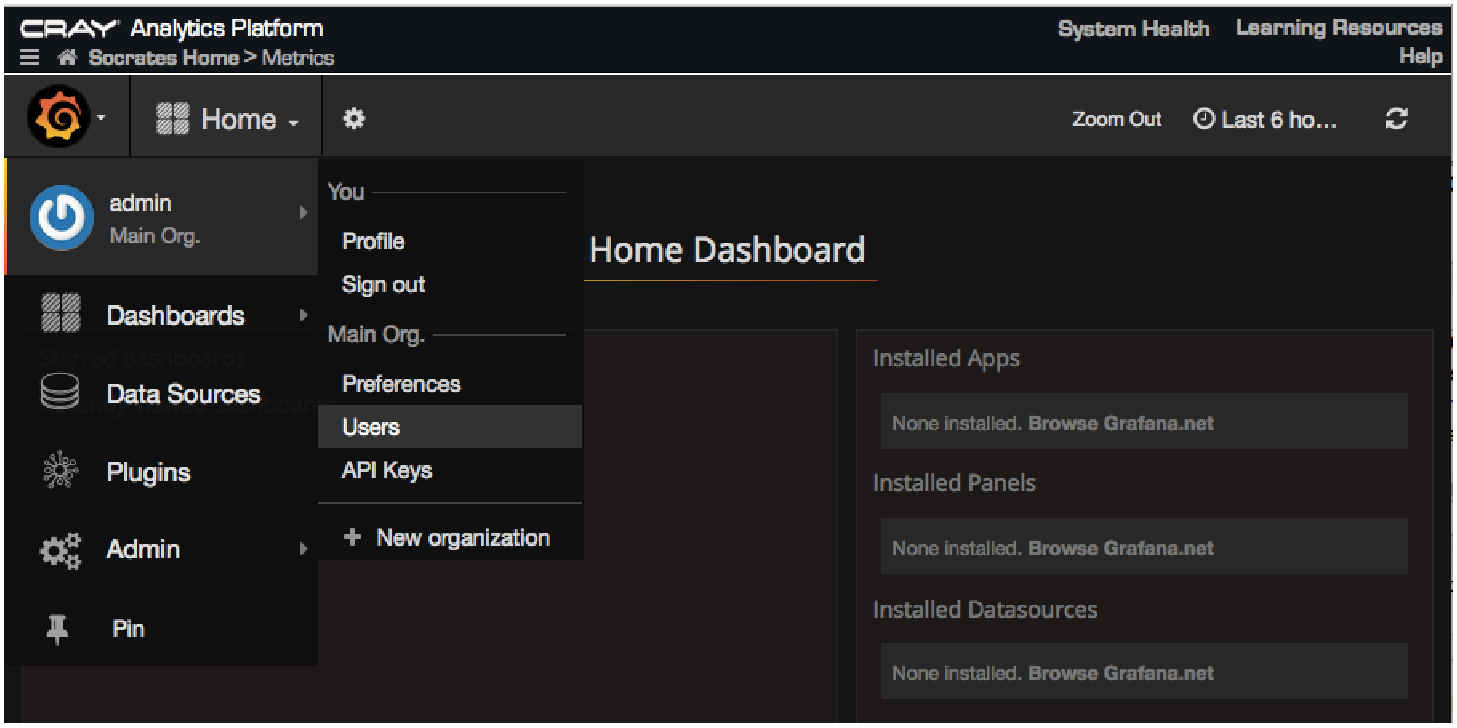
Users and **Organizations** can be assigned roles to specify permissions/privileges. These roles and their privileges are listed in the following table:

Table 14. Grafana Roles and Permissions

Role	Edit roles	View graphs	Edit/create copy of existing graphs	Add new graphs to existing dashboards	Create new/ import existing dashboards
Admin	Yes	Yes	Yes	Yes	Yes (These dashboards persist between sessions)
Viewer (default role)	No	Yes	No	No	No
Editor	No	Yes	Yes	Yes	Yes (These dashboards get deleted when the editor logs out)
Read only editor	No	Yes	Yes	Yes	No

Each role type can also export performance data via the dashboard. When a new user signs up for the first time, the default role assigned to the user is that of a Viewer. The admin can then change the new user's role if needed. All users have a default role of a **Viewer** when they first log on to Grafana. Administrators can change roles assigned to users as needed using the **Admin** menu.

Figure 4. Change User Roles Using the Admin Menu



Since the time displayed on the Grafana UI uses the browser's timezone and that displayed on the Spark History server's UI uses the Urika-GX system's timezone, the timezones displayed on the two UIs may not be the same.

By default, Grafana's refresh rate is turned off on the Urika-GX system. Sometimes the Hadoop and Spark Grafana dashboards take longer to load than expected. The usual cause of this issue is the time it takes InfluxDB to return all of the requested data. To reduce the time for the Hadoop and Spark dashboards to display data, refer to *Update the InfluxDB Data Retention Policy* on page 54. Reducing the amount of data retained makes Grafana dashboards display faster.

4.4 Default Grafana Dashboards

The default set of Grafana dashboards shipped with Urika-GX include:

- **Aggregate Compute Node Performance Statistics -** Displays graphs representing statistical data related to network, CPU, and I/O utilization for all Urika-GX nodes.

  This dashboard contains the following graphs:

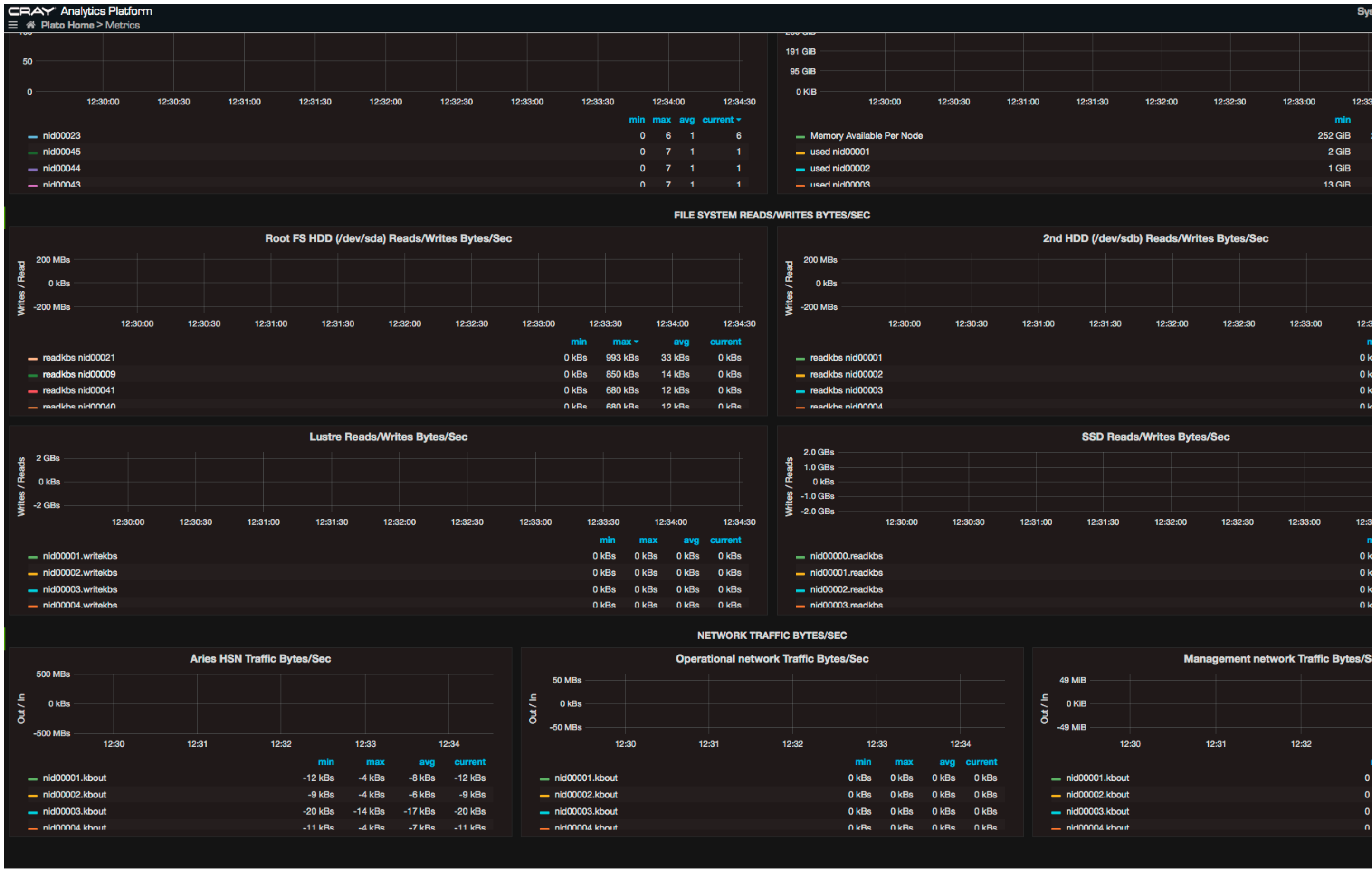
  - **CPU AND MEMORY**
    - **CPU utilization -** Displays the overall CPU utilization for all nodes.
    - **Memory utilization -** Displays the overall memory used by all nodes.
  - **FILE SYSTEM DATA RATES**
    - **SSD Reads/Writes Bytes/Second -** Displays SSD utilization for all nodes.

- **Second Hard Drive (/dev/sdb) Reads/Writes Bytes/Sec -** Displays utilization of secondary hard disk space for all nodes.
- **Root FS HDD (/dev/sda) Reads/Writes Bytes/Sec -** Displays utilization of root files system on the hard drive for all nodes.
- **Lustre Reads/Writes Bytes/Second -** Displays the aggregate Lustre I/O for all the nodes.

○ NETWORK READS AND WRITES

- **Aries HSN Bytes/Sec In/Out -** Displays the overall Aries network TCP traffic information for nodes. Note that non-TCP Aries traffic, including most traffic generated by CGE, is not shown here.
- **Operational network Bytes/sec In/Out -** Displays the overall operational network traffic information for all nodes.
- **Management network Bytes/sec In/Out -** Displays the overall management network traffic information for all nodes.

Figure 5. Aggregate Compute Node Performance Statistics


CRAY Analytics Platform
Plato Home > Metrics
FILE SYSTEM READS/WRITES BYTES/SEC
Root FS HDD (/dev/sda) Reads/Writes Bytes/Sec
2nd HDD (/dev/sdb) Reads/Writes Bytes/Sec
Lustre Reads/Writes Bytes/Sec
SSD Reads/Writes Bytes/Sec
NETWORK TRAFFIC BYTES/SEC
Aries HSN Traffic Bytes/Sec
Operational network Traffic Bytes/Sec
Management network Traffic Bytes/S
Writes / Read
Writes / Reads
Out / In
12:30:00
12:30:30
12:31:00
12:31:30
12:32:00
12:32:30
12:33:00
12:33:30
12:34:00
12:34:30

	min	max	avg	current
nid00023	0	6	1	6
nid00045	0	7	1	1
nid00044	0	7	1	1
nid00043	0	7	1	1

	min
Memory Available Per Node	252 GiB
used nid00001	2 GiB
used nid00002	1 GiB
used nid00003	13 GiB

Root FS HDD (/dev/sda) Reads/Writes Bytes/Sec

	min	max	avg	current
readkbs nid00021	0 kBs	993 kBs	33 kBs	0 kBs
readkbs nid00009	0 kBs	850 kBs	14 kBs	0 kBs
readkbs nid00041	0 kBs	680 kBs	12 kBs	0 kBs
readkbs nid00040	0 kBs	680 kBs	12 kBs	0 kBs

2nd HDD (/dev/sdb) Reads/Writes Bytes/Sec

readkbs nid00001	0 k
readkbs nid00002	0 k
readkbs nid00003	0 k
readkbs nid00004	0 k

Lustre Reads/Writes Bytes/Sec

	min	max	avg	current
nid00001.writekbs	0 kBs	0 kBs	0 kBs	0 kBs
nid00002.writekbs	0 kBs	0 kBs	0 kBs	0 kBs
nid00003.writekbs	0 kBs	0 kBs	0 kBs	0 kBs
nid00004.writekbs	0 kBs	0 kBs	0 kBs	0 kBs

SSD Reads/Writes Bytes/Sec

nid00000.readkbs	0 k
nid00001.readkbs	0 k
nid00002.readkbs	0 k
nid00003.readkbs	0 k

Aries HSN Traffic Bytes/Sec

	min	max	avg	current
nid00001.kbout	-12 kBs	-4 kBs	-8 kBs	-12 kBs
nid00002.kbout	-9 kBs	-4 kBs	-6 kBs	-9 kBs
nid00003.kbout	-20 kBs	-14 kBs	-17 kBs	-20 kBs
nid00004.kbout	-11 kBs	-4 kBs	-7 kBs	-11 kBs

Operational network Traffic Bytes/Sec

	min	max	avg	current
nid00001.kbout	0 kBs	0 kBs	0 kBs	0 kBs
nid00002.kbout	0 kBs	0 kBs	0 kBs	0 kBs
nid00003.kbout	0 kBs	0 kBs	0 kBs	0 kBs
nid00004.kbout	0 kBs	0 kBs	0 kBs	0 kBs

Management network Traffic Bytes/S

nid00001.kbout	0
nid00002.kbout	0
nid00003.kbout	0
nid00004.kbout	0

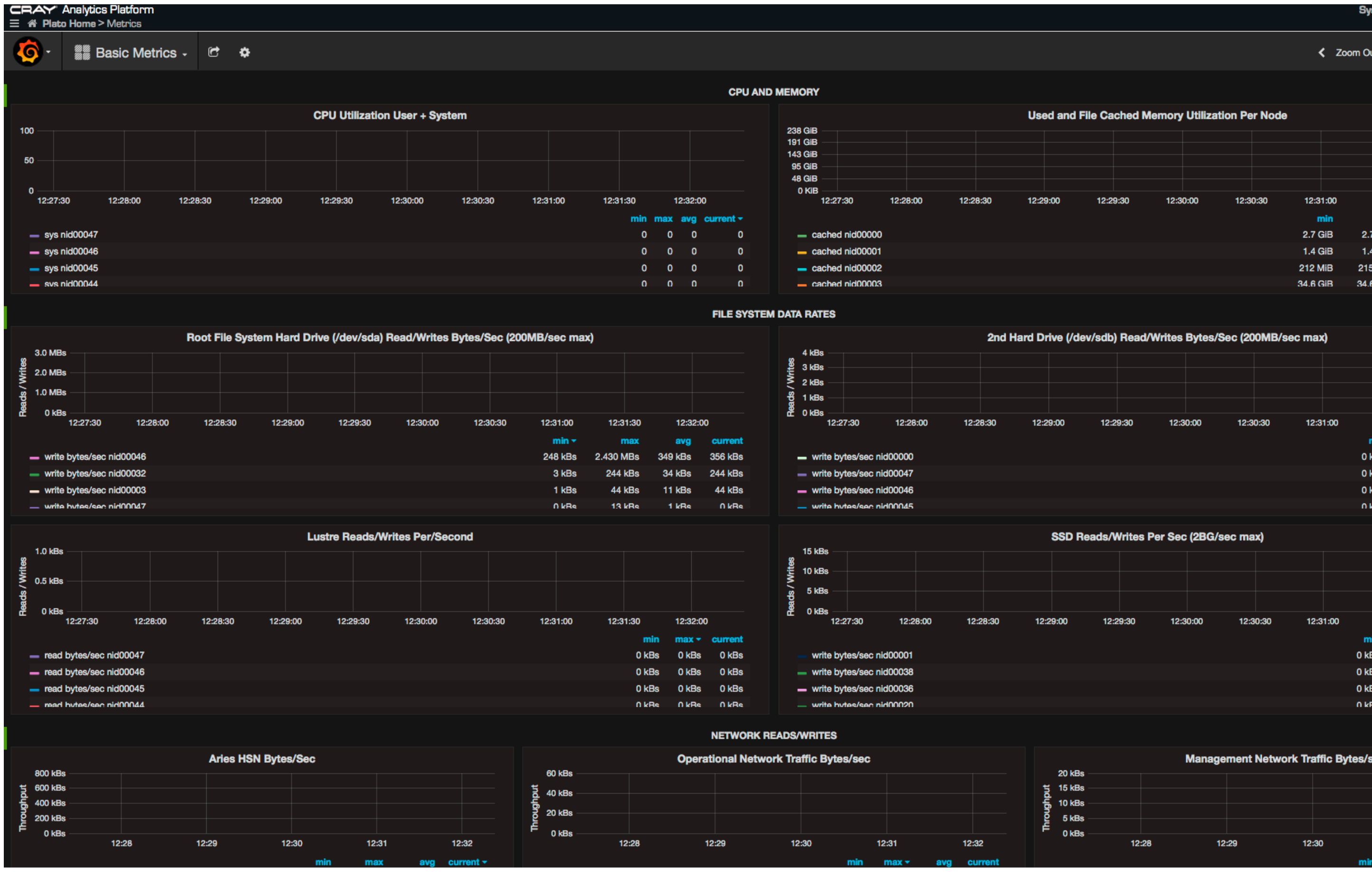
- **Basic Metrics -** Displays graphs representing statistical data related to network, CPU, and I/O utilization for the Urika-GX system, as a whole.

  TIP: It is recommended that administrators use the **Basic Metrics** dashboard before using other dashboards to retrieve a summary of the system's health.

This dashboard contains the following graphs:

- **CPU AND MEMORY**
  - **Used and File Cached Memory** - Displays the used and file cached memory for each node.
  - **CPU Utilization User + System** - Displays the CPU utilization by the user and system for each node
- **FILE SYSTEM DATA RATES**
  - **Root File System Hard Drive (/dev/sda) Reads/Writes Bytes/Sec (200MB/sec max)** - Displays information about the usage of memory on the root file system for each node.
  - **2nd Hard Drive (/dev/sdb) Read/Writes** - Displays information about the usage of memory on the secondary hard drive of each node.
  - **Lustre Read/Writes Bytes/Second** - Displays the number of Lustre reads/writes for each node.
  - **SSD Read/Writes Bytes/Second** - Displays the number of reads/writes of the SSDs installed on the system.
- **NETWORK READS/WRITES**
  - **Aries HSN Bytes/Sec In/Out** - Displays the Aries network TCP traffic information for each node. Note that non-TCP Aries traffic, including most traffic generated by CGE, is not shown here.
  - **Operational network Bytes/sec In/Out** - Displays the overall operational network traffic information for each node.
  - **Management network Bytes/sec In/Out** - Displays the overall management network traffic information for each node.
- **FILE SYSTEM UTILIZATION**
  - **Root File System Hard Drive (/dev/sda) Reads/Writes Bytes/Sec (200MB/sec max)** - Displays information about the usage of memory on the root file system for each node.
  - **2nd Hard Drive (/dev/sdb) Read/Writes** - Displays information about the usage of memory on the secondary hard drive of each node.
  - **Lustre Read/Writes Per/Second** - Displays the number of Lustre reads/writes for each node.
  - **SSD Read/Writes Per/Second** - Displays the number of reads/writes of each node's SSDs.
- **NETWORK ERRORS AND DROPED PACKETS**
  - **Aries HSN Dropped Packets and Errors Per Sec** - Displays the number of dropped packets/errors per second for the Aries network TCP traffic information for each node. Note that non-TCP Aries traffic, including most traffic generated by CGE, is not shown here.
  - **Operational network Dropped Packets and Errors Per Sec** - Displays the number of dropped packets/errors per second for the operational network TCP traffic information for each node.
  - **Management network Dropped Packets and Errors Per Sec** - Displays the number of dropped packets/errors per second for the management network TCP traffic information for each node.

Figure 6. Basic Metrics Dashboard


Analytics Platform
Plato Home > Metrics
Basic Metrics
Zoom Out
CPU AND MEMORY
CPU Utilization User + System
Used and File Cached Memory Utilization Per Node
FILE SYSTEM DATA RATES
Root File System Hard Drive (/dev/sda) Read/Writes Bytes/Sec (200MB/sec max)
2nd Hard Drive (/dev/sdb) Read/Writes Bytes/Sec (200MB/sec max)
Lustre Reads/Writes Per/Second
SSD Reads/Writes Per Sec (2BG/sec max)
NETWORK READS/WRITES
Aries HSN Bytes/Sec
Operational Network Traffic Bytes/sec
Management Network Traffic Bytes/s
Reads / Writes
Throughput

	min	max	avg	current
sys nid00047	0	0	0	0
sys nid00046	0	0	0	0
sys nid00045	0	0	0	0
sys nid00044	0	0	0	0

	min
cached nid00000	2.7 GiB
cached nid00001	1.4 GiB
cached nid00002	212 MiB
cached nid00003	34.6 GiB

	min	max	avg	current
write bytes/sec nid00046	248 kBs	2.430 MBs	349 kBs	356 kBs
write bytes/sec nid00032	3 kBs	244 kBs	34 kBs	244 kBs
write bytes/sec nid00003	1 kBs	44 kBs	11 kBs	44 kBs
write bytes/sec nid00047	0 kBs	13 kBs	1 kBs	0 kBs

write bytes/sec nid00000
write bytes/sec nid00047
write bytes/sec nid00046
write bytes/sec nid00045

	min	max	current
read bytes/sec nid00047	0 kBs	0 kBs	0 kBs
read bytes/sec nid00046	0 kBs	0 kBs	0 kBs
read bytes/sec nid00045	0 kBs	0 kBs	0 kBs
read bytes/sec nid00044	0 kBs	0 kBs	0 kBs

write bytes/sec nid00001
write bytes/sec nid00038
write bytes/sec nid00036
write bytes/sec nid00020

- **Compute Node Performance Statistics -** Displays graphs representing statistical data related to network, CPU, and I/O utilization for all Urika-GX compute nodes.

  This dashboard contains the following graphs:

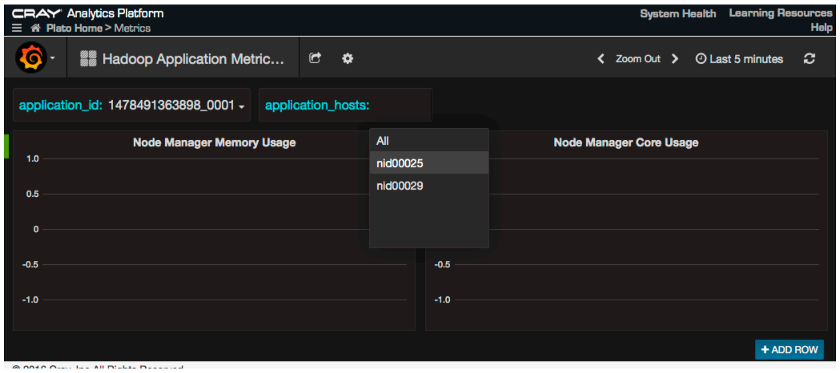
  - **CPU MEMORY UTILIZATION**
    - **CPU utilization -** Displays the overall CPU utilization for all the compute nodes.
    - **Memory utilization -** Displays the overall memory (in KB or GB) used by all the compute nodes.
  - **FILE SYSTEM READS/WRITE BYTES/SEC**
    - **Root File System Hard Drive (/dev/sda) Reads/Writes Bytes/Sec (200MB/sec max) -** Displays information about the usage of memory on the root file system by compute nodes..
    - **2nd Hard Drive (/dev/sdb) Read/Writes -** Displays information about the usage of memory by compute nodes on the secondary hard drive.
    - **Lustre Read/Writes Per/Second -** Displays the number of Lustre reads/writes by compute nodes.
    - **SSD Read/Writes Per/Second -** Displays the number of reads/writes of the compute node SSDs installed on the system.
  - **NETWORK READS/WRITES**

- **Aries HSN Bytes/Sec In/Out** - Displays the Aries network TCP traffic information for compute nodes. Note that non-TCP Aries traffic, including most traffic generated by CGE, is not shown here.
- **Operational network Bytes/sec In/Out** - Displays the overall operational network traffic information for compute nodes.
- **Management network Bytes/sec In/Out** - Displays the overall management network traffic information for compute nodes.

Figure 7. Compute Node Performance Statistics

- **Hadoop Application Metrics** - This section contains the following graphs:
  - **Node Manager Memory Usage** - Displays the average memory usage per application in mega bytes per second (MBPS). The Y-axis is in MBPS.
  - **Node Manager Core Usage** - Displays the average CPU usage per application in MilliVcores . The Y-axis refers to MilliVcores.

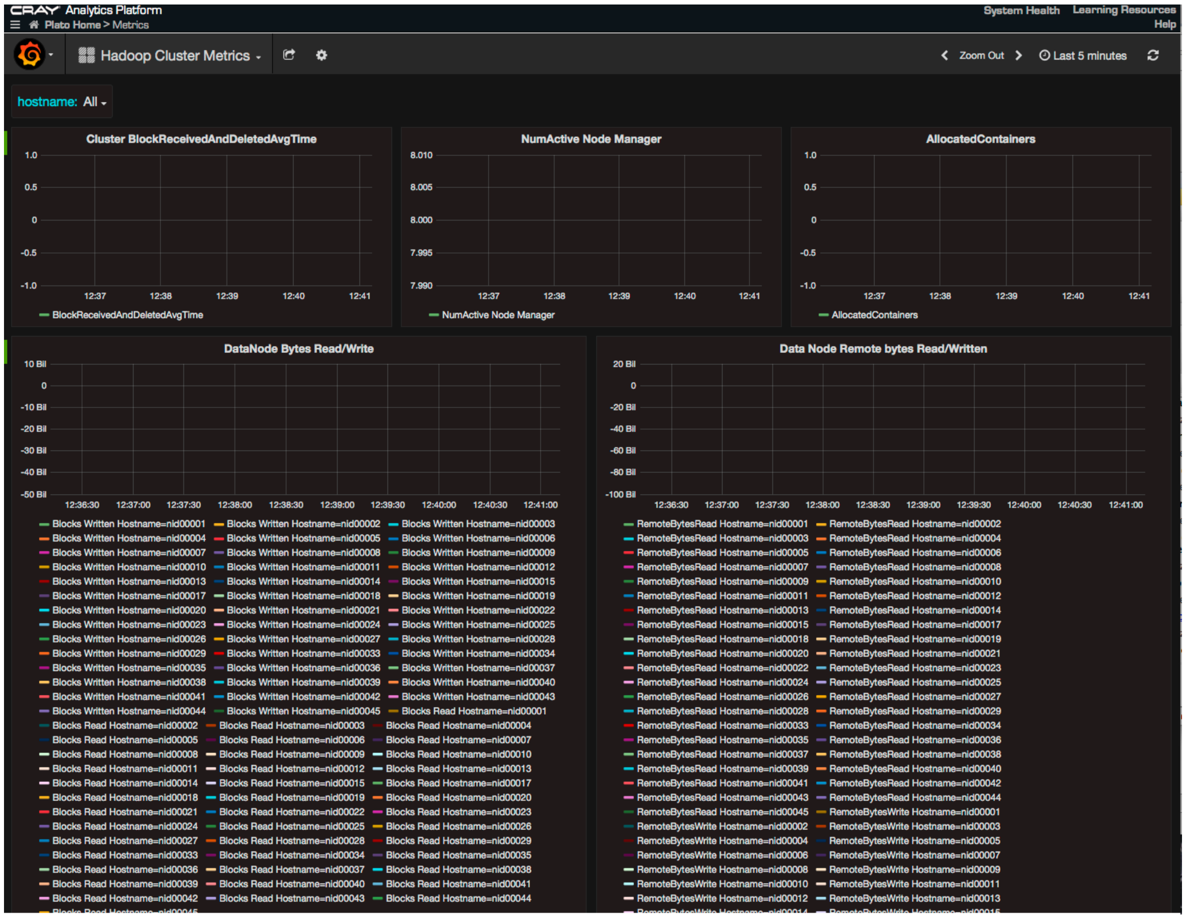
Figure 8. Hadoop Applications Metrics Dashboard



- **Hadoop Cluster Metrics** - Displays graphs representing statistical data related to Hadoop components, such as HDFS Data Nodes and HDFS Name Nodes.

  This dashboard contains the following sections:
  - **Cluster BlockReceivedAndDeletedAvgTime** - Displays the average time in milliseconds for the hdfs cluster to send and receive blocks. The Y-axis represents time in milliseconds.
  - **NumActive Node Manager** - Displays the number of Node Managers up and running in the Hadoop cluster at a given time. The Y-axis represents a linear number.
  - **AllocatedContainers** - Displays the number of allocated YARN containers by all the jobs in the hadoop cluster. The Y-axis represents a linear number.
  - **DataNode Bytes Read/Write** - Displays the number of bytes read or write per node from local client in the hadoop cluster. The Y-axis refers to a linear number. Read is shown on the positive scale. Write is shown on the negative scale.
  - **Data Node Remote bytes Read/Written** - Displays the number of bytes read or written per node from remote clients in the Hadoop cluster. The Y-axis represents a linear number. The number of reads are shown on the positive scale. The number of writes are shown on the negative scale.

- *Figure 9. Hadoop Cluster Metrics Dashboard*


CRAY Analytics Platform
Plato Home > Metrics
System Health
Learning Resources
Help
Hadoop Cluster Metrics
Zoom Out
Last 5 minutes
hostname: All
Cluster BlockReceivedAndDeletedAvgTime
BlockReceivedAndDeletedAvgTime
NumActive Node Manager
NumActive Node Manager
AllocatedContainers
AllocatedContainers
DataNode Bytes Read/Write
Blocks Written Hostname=nid00001
Blocks Read Hostname=nid00001
Data Node Remote bytes Read/Written
RemoteBytesRead Hostname=nid00001
RemoteBytesWrite Hostname=nid00001

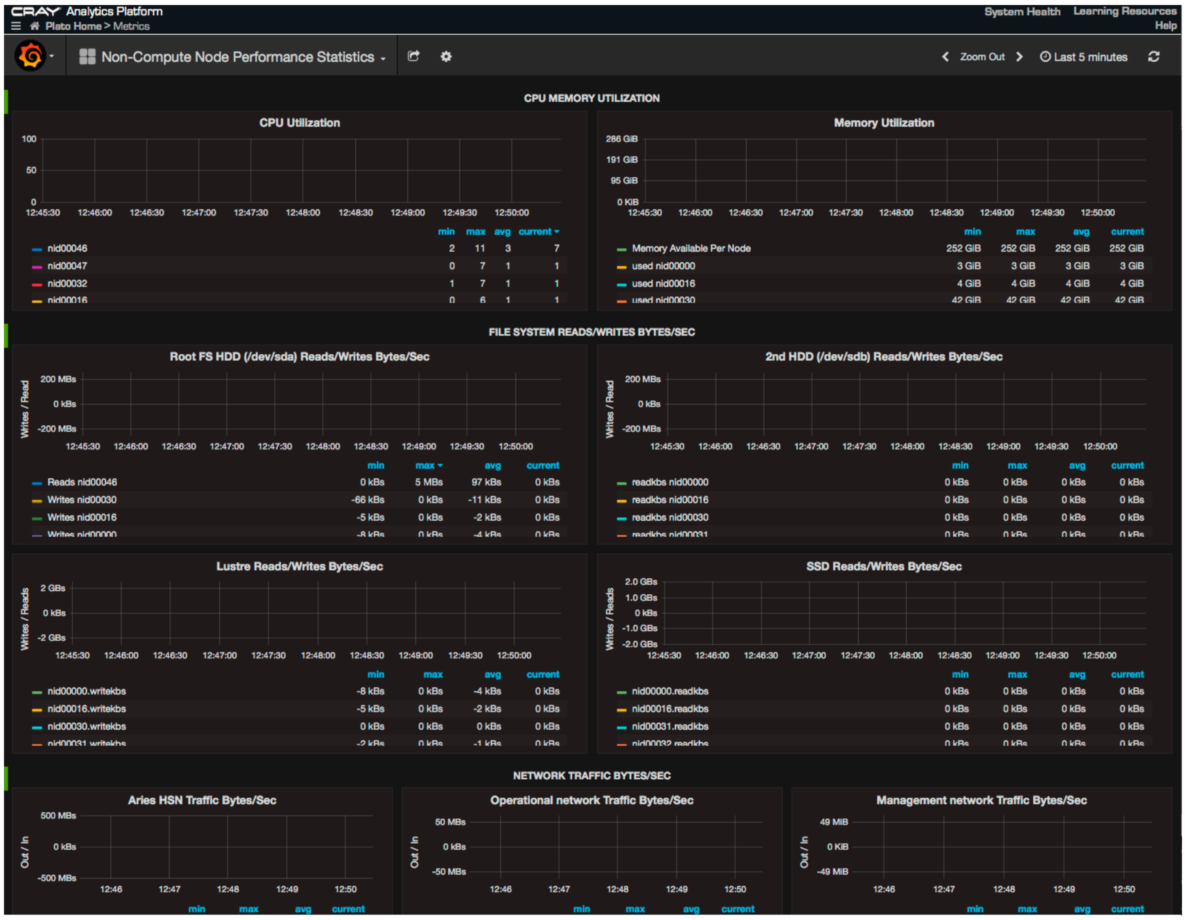
- **Non-compute Node Performance Statistics** - Displays graphs representing statistical data related to network, CPU, and I/O utilization for all the non-compute (I/O and login) nodes of Urika-GX.

  This dashboard contains the following graphs:

  - **CPU MEMORY UTILIZATION**
    - **CPU utilization** - Displays the overall CPU utilization for I/O and login (non-compute) nodes.
    - **Memory utilization** - Displays the overall memory (in KB or GB) used by all the I/O and login nodes.
  - **FILE SYSTEM READS/WRITE BYTES/SEC**
    - **Root File System Hard Drive (/dev/sda) Reads/Writes Bytes/Sec (200MB/sec max)** - Displays information about the usage of memory on the root file system by I/O and login nodes.
    - **2nd Hard Drive (/dev/sdb) Read/Writes** - For I/O and login nodes, displays information about the usage of memory by compute nodes on the secondary hard drive.
    - **Lustre Read/Writes Bytes/Second** - Displays the number of Lustre reads/writes by I/O and login nodes.
    - **SSD Read/Writes Bytes/Second** - Displays the number of SSD reads/writes of the I/O and login nodes.
  - **NETWORK READS/WRITES**

- **Aries HSN Bytes/Sec In/Out** - Displays the Aries network TCP traffic information for I/O and login nodes. Note that non-TCP Aries traffic, including most traffic generated by CGE, is not shown here.
- **Operational network Bytes/sec In/Out** - Displays the overall operational network traffic information for I/O and login nodes.
- **Management network Bytes/sec In/Out** - Displays the overall management network traffic information for I/O and login nodes.

Figure 10. Non-compute Node Performance Statistics



CRAY Analytics Platform
Plato Home > Metrics
System Health
Learning Resources
Help
Non-Compute Node Performance Statistics
Zoom Out
Last 5 minutes
CPU MEMORY UTILIZATION
FILE SYSTEM READS/WRITES BYTES/SEC
NETWORK TRAFFIC BYTES/SEC
Aries HSN Traffic Bytes/Sec
Operational network Traffic Bytes/Sec
Management network Traffic Bytes/Sec
Writes / Read
Writes / Reads
Out / In

CPU Utilization

	min	max	avg	current
nid00046	2	11	3	7
nid00047	0	7	1	1
nid00032	1	7	1	1
nid00016	0	6	1	1

Memory Utilization

	min	max	avg	current
Memory Available Per Node	252 GiB	252 GiB	252 GiB	252 GiB
used nid00000	3 GiB	3 GiB	3 GiB	3 GiB
used nid00016	4 GiB	4 GiB	4 GiB	4 GiB
used nid00030	42 GiB	42 GiB	42 GiB	42 GiB

Root FS HDD (/dev/sda) Reads/Writes Bytes/Sec

	min	max	avg	current
Reads nid00046	0 kBs	5 MBs	97 kBs	0 kBs
Writes nid00030	-66 kBs	0 kBs	-11 kBs	0 kBs
Writes nid00016	-5 kBs	0 kBs	-2 kBs	0 kBs
Writes nid00000	-8 kBs	0 kBs	-4 kBs	0 kBs

2nd HDD (/dev/sdb) Reads/Writes Bytes/Sec

	min	max	avg	current
readkbs nid00000	0 kBs	0 kBs	0 kBs	0 kBs
readkbs nid00016	0 kBs	0 kBs	0 kBs	0 kBs
readkbs nid00030	0 kBs	0 kBs	0 kBs	0 kBs
readkbs nid00031	0 kBs	0 kBs	0 kBs	0 kBs

Lustre Reads/Writes Bytes/Sec

	min	max	avg	current
nid00000.writekbs	-8 kBs	0 kBs	-4 kBs	0 kBs
nid00016.writekbs	-5 kBs	0 kBs	-2 kBs	0 kBs
nid00030.writekbs	0 kBs	0 kBs	0 kBs	0 kBs
nid00031.writekbs	-2 kBs	0 kBs	-1 kBs	0 kBs

SSD Reads/Writes Bytes/Sec

	min	max	avg	current
nid00000.readkbs	0 kBs	0 kBs	0 kBs	0 kBs
nid00016.readkbs	0 kBs	0 kBs	0 kBs	0 kBs
nid00031.readkbs	0 kBs	0 kBs	0 kBs	0 kBs
nid00032.readkbs	0 kBs	0 kBs	0 kBs	0 kBs

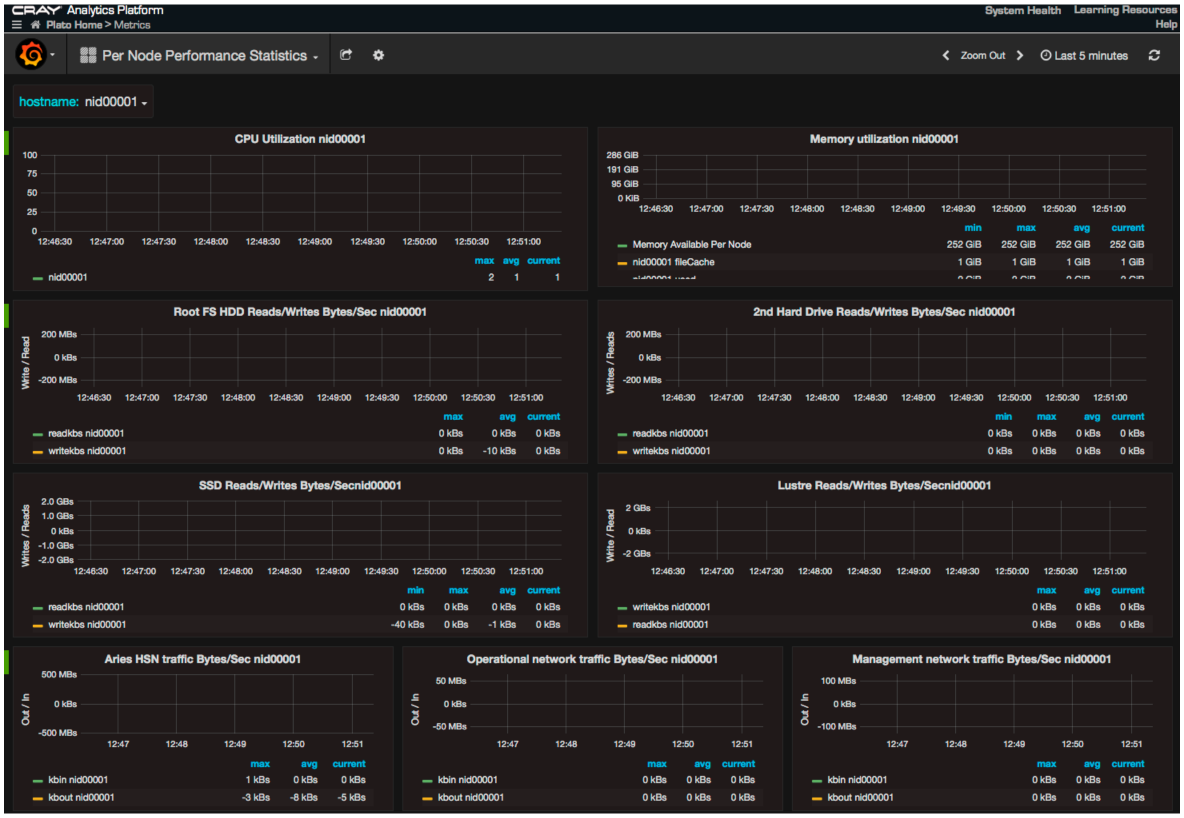
- **Per Node Performance Statistics** - Displays graphs representing statistical data related to network, CPU, and I/O utilization for individual Urika-GX nodes. The node's hostname can be selected using the **hostname** drop down provided on the UI.

  This dashboard contains the following graphs:

  - **CPU utilization** - Displays the CPU utilization for the selected node.
  - **Memory utilization** - Displays the memory (in KB or GB) used by the selected node.
  - **Root File System Hard Drive (/dev/sda) Reads/Writes Bytes/Sec**- Displays the HDD SDA utilization for the selected node.
  - **2nd Hard Drive (/dev/sdb) Read/Writes** - Displays the HDD SDB utilization for the selected node.
  - **SSD Read/Writes Bytes/Second** - Displays the number of SSD reads/writes per second for each node.

- ○ **Lustre Read/Writes Bytes/Second -** Displays the number of Lustre reads/writes by the selected node.
- ○ **Aries HSN Bytes/Sec In/Out -** Displays the Aries network TCP traffic information for the selected node. Note that non-TCP Aries traffic, including most traffic generated by CGE, is not shown here.
- ○ **Operational network Bytes/sec In/Out -** Displays the overall operational network traffic information for the selected node.
- ○ **Management network Bytes/sec In/Out -** Displays the overall management network traffic information for the selected node.
- ○ **NFS HDFS Lustre Percentage Used -** Displays the percentage of NFS, HDFS and Lustre used by the selected node.
- ○ **File System Percent Used -** Displays the percentage of file system used by the selected node.

Figure 11. Per Node Performance Statistics


CRAY Analytics Platform
Plato Home > Metrics
System Health
Learning Resources
Help
Per Node Performance Statistics
Zoom Out
Last 5 minutes
hostname: nid00001
CPU Utilization nid00001
Memory utilization nid00001
Root FS HDD Reads/Writes Bytes/Sec nid00001
2nd Hard Drive Reads/Writes Bytes/Sec nid00001
SSD Reads/Writes Bytes/Secnid00001
Lustre Reads/Writes Bytes/Secnid00001
Aries HSN traffic Bytes/Sec nid00001
Operational network traffic Bytes/Sec nid00001
Management network traffic Bytes/Sec nid00001
Write / Read
Writes / Reads
Out / In

CPU Utilization nid00001

	max	avg	current
nid00001	2	1	1

Memory utilization nid00001

	min	max	avg	current
Memory Available Per Node	252 GiB	252 GiB	252 GiB	252 GiB
nid00001 fileCache	1 GiB	1 GiB	1 GiB	1 GiB

Root FS HDD Reads/Writes Bytes/Sec nid00001

	max	avg	current
readkbs nid00001	0 kBs	0 kBs	0 kBs
writekbs nid00001	0 kBs	-10 kBs	0 kBs

2nd Hard Drive Reads/Writes Bytes/Sec nid00001

	min	max	avg	current
readkbs nid00001	0 kBs	0 kBs	0 kBs	0 kBs
writekbs nid00001	0 kBs	0 kBs	0 kBs	0 kBs

SSD Reads/Writes Bytes/Secnid00001

	min	max	avg	current
readkbs nid00001	0 kBs	0 kBs	0 kBs	0 kBs
writekbs nid00001	-40 kBs	0 kBs	-1 kBs	0 kBs

Lustre Reads/Writes Bytes/Secnid00001

	max	avg	current
writekbs nid00001	0 kBs	0 kBs	0 kBs
readkbs nid00001	0 kBs	0 kBs	0 kBs

Aries HSN traffic Bytes/Sec nid00001

	max	avg	current
kbin nid00001	1 kBs	0 kBs	0 kBs
kbout nid00001	-3 kBs	-8 kBs	-5 kBs

Operational network traffic Bytes/Sec nid00001

	max	avg	current
kbin nid00001	0 kBs	0 kBs	0 kBs
kbout nid00001	0 kBs	0 kBs	0 kBs

Management network traffic Bytes/Sec nid00001

	max	avg	current
kbin nid00001	0 kBs	0 kBs	0 kBs
kbout nid00001	0 kBs	0 kBs	0 kBs

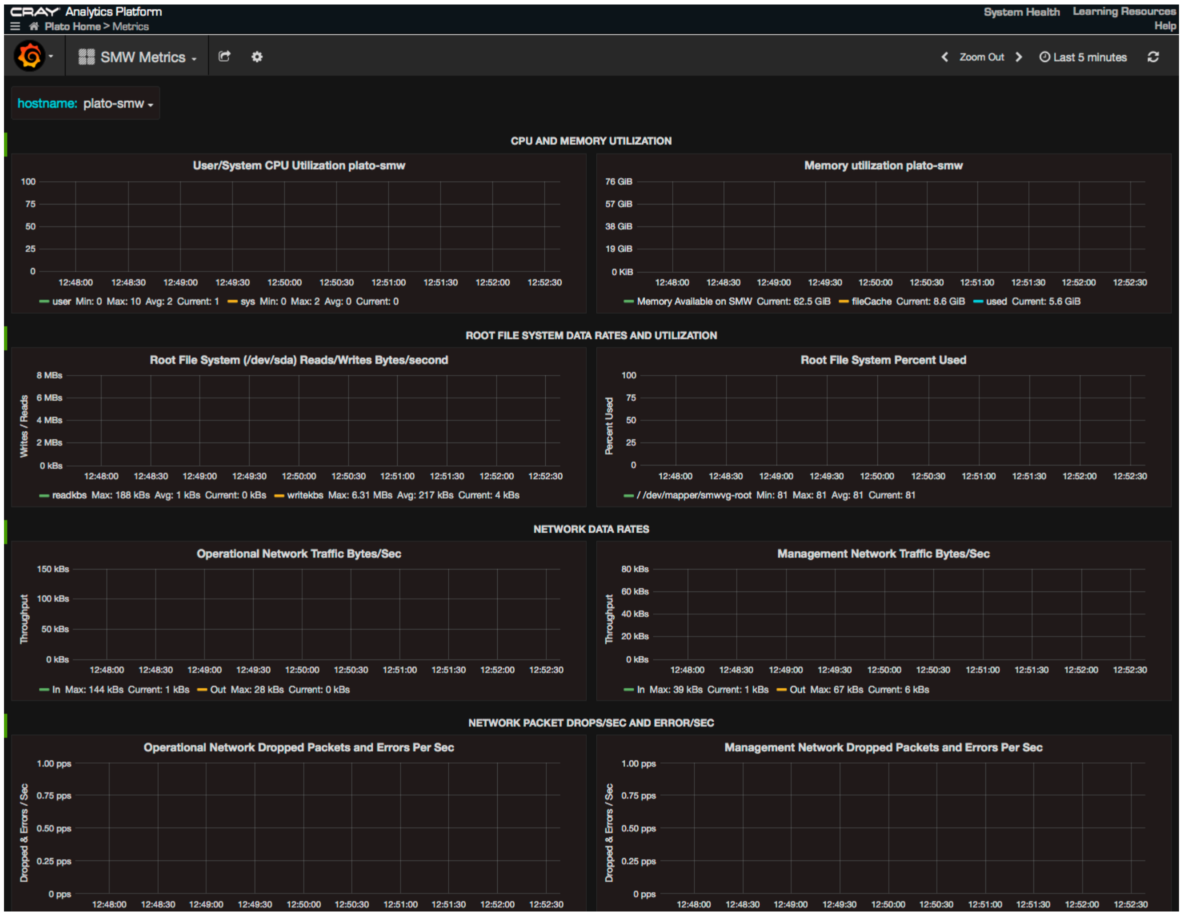
- **SMW Metrics/ -** Displays graphs representing statistical data related to SMW's resources, such as the CPU's memory utilization, root file system, etc.

  This dashboard contains the following sections:

  - ○ **CPU and MEMORY UTILIZATION -** Displays the memory consumption by the SMW's CPU.
    - ▪ User/System CPU Utilization `MACHINE_NAME`
    - ▪ Memory utilization
  - ○ **Root File System Data Rates and Utilization -** Displays memory usage and data rate of the SMW's root file system.

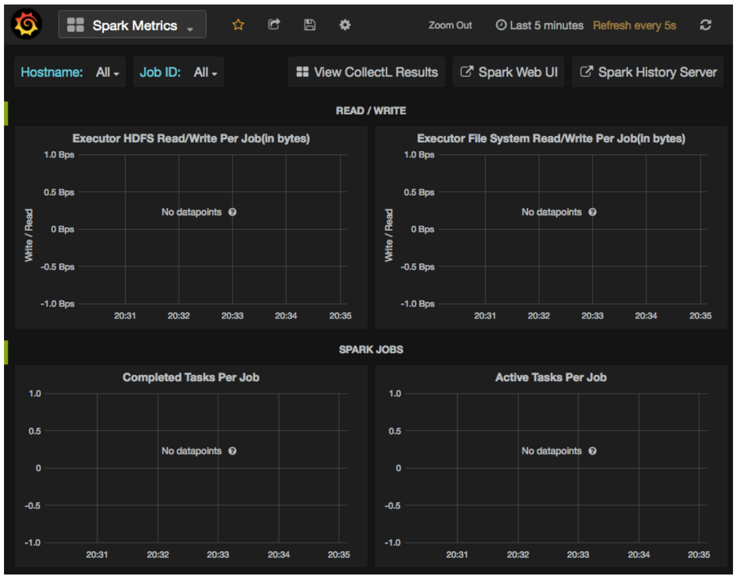
        - **Root File System Hard Drive (/dev/sda) Reads/Writes Bytes/Sec** - Displays information about the usage of memory on the root file system of the SMW
        - **Root File System Percent Used** - Displays the percentage for used SMW root file system space.
    - **NETWORK DATA RATES**
        - **Operational Network Traffic Bytes/sec** - Displays the operational network's data rate.
        - **Management Network Traffic Bytes/sec** - Displays the management network's data rate.
    - **NETWORK PACKET DROPS/SEC AND ERRORS/SEC**
        - **Operational Network Dropped and Errors Per Sec** - Displays the number of dropped packets and errors per second for the operational network.
        - **Management Network Dropped and Errors Per Sec** - Displays the number of dropped packets and errors per second for the management network.

Figure 12. SMW Metrics Dashboard


CRAY Analytics Platform
Plato Home > Metrics
System Health
Learning Resources
Help
SMW Metrics
Zoom Out
Last 5 minutes
hostname: plato-smw
CPU AND MEMORY UTILIZATION
User/System CPU Utilization plato-smw
user Min: 0 Max: 10 Avg: 2 Current: 1
sys Min: 0 Max: 2 Avg: 0 Current: 0
Memory utilization plato-smw
Memory Available on SMW Current: 62.5 GiB
fileCache Current: 8.6 GiB
used Current: 5.6 GiB
ROOT FILE SYSTEM DATA RATES AND UTILIZATION
Root File System (/dev/sda) Reads/Writes Bytes/second
Writes / Reads
readkbs Max: 188 kBs Avg: 1 kBs Current: 0 kBs
writekbs Max: 6.31 MBs Avg: 217 kBs Current: 4 kBs
Root File System Percent Used
Percent Used
/ /dev/mapper/smwvg-root Min: 81 Max: 81 Avg: 81 Current: 81
NETWORK DATA RATES
Operational Network Traffic Bytes/Sec
Throughput
In Max: 144 kBs Current: 1 kBs
Out Max: 28 kBs Current: 0 kBs
Management Network Traffic Bytes/Sec
Throughput
In Max: 39 kBs Current: 1 kBs
Out Max: 67 kBs Current: 6 kBs
NETWORK PACKET DROPS/SEC AND ERROR/SEC
Operational Network Dropped Packets and Errors Per Sec
Dropped & Errors / Sec
Management Network Dropped Packets and Errors Per Sec
Dropped & Errors / Sec

- **Spark Metrics** - Displays graphs representing statistical data related to Spark jobs. This dashboard also contains links for viewing the **Spark Web UI** and **Spark History Server**.

Figure 13. Grafana Spark Metrics Dashboard



Graphs displayed on this dashboard are grouped into the following sections:

- **READ/WRITE**: Displays statistics related to the file system statistics of a Spark executor. Results in the graphs of this section are displayed per node for a particular Spark Job. The Y-axis displays the number in bytes, whereas the X-axis displays the start/stop time of the task for a particular Spark Job.

  This section contains the following graphs:

  - **Executor HDFS Read/Write Per Job (in bytes)**: Reading and writing from HDFS.
  - **Executor File System Read/Write Per Job (in bytes)**: Reading and writing from a File System.

- **SPARK JOBS**: Displays statistics related to the list of executors per node for a particular Spark Job. The Y-axis displays the number of tasks and X-axis displays the start/stop time of the task for a particular Spark Job.

  This section contains the following graphs:

  - **Completed Tasks Per Job**: The approximate total number of tasks that have completed execution.
  - **Active Tasks Per Job**: The approximate number of threads that are actively executing tasks.
  - **Current Pool Size Per Job**: The current number of threads in the pool.
  - **Max Pool Size Per Job**: The maximum allowed number of threads that have ever simultaneously been in the pool.

- **DAG Scheduler -** Displays statistics related to Spark's Directed Acyclic Graphs.

  This section contains the following graphs:

  - **DAG Schedule Stages -** This graph displays the following types of DAG stages:
    - Waiting Stages: Stages with parents to be computed.
    - Running Stages: Stages currently being run.
    - Failed Stages: Stages that failed due to fetch failures (as reported by `CompletionEvents` for `FetchFailed` end reasons) and are going to be resubmitted.

    - **DAG Scheduler Jobs** - This graph displays the following types of DAG scheduler jobs:
        - All Jobs - The number of all jobs
        - Active Jobs - The number of active jobs
    - **DAG Scheduler Message Processing Time** - This graph displays the processing time of the DAG Scheduler.
- **JVM Memory Usage** - The memory usage dashboards represent a snapshot of used memory.
    - **JVM Memory Usage - init**: Represents the initial amount of memory (in bytes) that the Java virtual machine requests from the operating system for memory management during start up. The Java virtual machine may request additional memory from the operating system and may also release memory to the system over time. The value of `init` may be undefined.
    - **JVM Memory Usage - Used**: Represents the amount of memory currently used (in bytes).
    - **JVM Memory Usage - Committed**: Represents the amount of memory (in bytes) that is guaranteed to be available for use by the Java virtual machine. The amount of committed memory may change over time (increase or decrease). The Java virtual machine may release memory to the system and committed could be less than `init`. Committed will always be greater than or equal to used.
    - **JVM Memory Usage - Max**: Represents the maximum amount of memory (in bytes) that can be used for memory management. Its value may be undefined. The maximum amount of memory may change over time if defined. The amount of used and committed memory will always be less than or equal to max if max is defined. A memory allocation may fail if it attempts to increase the used memory such that `used > committed` even if `used <= max` is still true, for example, when the system is low on virtual memory.
    - **JVM Heap Usage**: Represents the maximum amount of memory used by the JVM heap space
    - **JVM Non-Heap Usage**: Represents the maximum amount of memory used by the JVM non-heap space

4.5 Update InfluxDB Security Settings

Prerequisites

- This procedure requires root privileges.
- Ensure that the system is running in the service mode that allows use of the InfluxDB. Execute the `urika-state` or `urika-service-mode` commands to check the service mode. For more information, refer to the `urika-state` or `urika-service-mode` man pages and see *Urika-GX Service Modes* on page 173.

About this task

The default configuration for InfluxDB as shipped with Urika-GX is:

1. InfluxDB UI on socket `login2:8083` is disabled
2. No authorization is enabled.

This means that any user on login node 2 can call the `/bin/influx` command and modify databases. To restrict this to only a privileged user, the admin needs to create an authorized user and enable basic authentication for InfluxDB.

CAUTION: After creating the privilege user the Grafana data sources **must** to be updated to provide the name and password for Grafana queries to InfluxDB to succeed.

To enable add an authorized user and enable basic authentication:

Procedure

1. Log on to login2 (login node 2) and become root.

2. Connect to InfluxDB and add a user named `admin` user and set its password.

```
# /bin/influx
Visit https://enterprise.influxdata.com to register for updates, InfluxDB server
management, and monitoring.
Connected to http://localhost:8086 version 0.12.2
InfluxDB shell 0.12.2
> create user admin with password 'admin' with all privileges
> show users
user    admin
admin    true
```

3. Edit the `/etc/influxdb/influxdb.conf` file as root.

```
# vi /etc/influxdb/influxdb.conf
< change auth to be true, save file, exit >
# systemctl restart influxdb
```

At this point Grafana dashboards will still work because Grafana is shipped with credentials for InfluxDB to be user `admin` with `admin` as the password. If the privileged user differs from these defaults Grafana queries will fail and graphs in dashboards will not work. To resolve this issue, log on to Grafana as the admin user and open each data source, edit the InfluxDB details, updating the user and password to match that of the privileged InfluxDB user.

For more information, visit *https://docs.influxdata.com/*.

4.6 Update the InfluxDB Data Retention Policy

Prerequisites

This procedure requires root privileges. Before performing this procedure, use the `urika-state` command to ensure that the system is operating in the service mode that supports using InfluxDB. For more information, see the `urika-state` man page and refer to *Urika-GX Service Modes* on page 173.

About this task

The data retention policy for InfluxDB defaults to infinite, i.e. data is never deleted. As a result, Spark and Hadoop Grafana dashboards may take longer to load and InfluxDB may take up more space than necessary. To reduce the amount of space being used by InfluxDB, the data retention policy for each database needs to be reduced, as described in this procedure. Reducing the data retention policy for Spark and Hadoop databases can reduce the load time of the Spark and Hadoop Grafana dashboards.

Procedure

1. Log on to login2 and become root.

2. Switch to the `/var/lib/influxdb/data` directory.

   ```
   # cd /var/lib/influxdb/data
   ```

3. Use the du command to show how much space being used.

   The sizes below are shown as examples. Actual sizes on the system may vary.

   ```
   $ du -sh *
   14G    Cray Urika GX
   1.5G      CrayUrikaGXHadoop
   906M      CrayUrikaGXSpark
   21M    _internal
   #
   ```

4. Connect to InfluxDB to view the current data retention policy.

   ```
   # /bin/influx
   Visit https://enterprise.influxdata.com to register for updates, InfluxDB server
   management, and monitoring.
   Connected to http://localhost:8086 version 0.12.2
   InfluxDB shell 0.12.2
   > show retention policies on "Cray Urika GX"
   name    duration    shardGroupDuration    replicaN    default
   default    0               168h0m0s        1        true
   ```

5. Update the data retention policy according to requirements.

   In this example the data retention duration is changed from 0 (forever) to 2 weeks (504 hours).

   ```
   > alter retention policy default on "Cray Urika GX" Duration 2w
   > show retention policies on "Cray Urika GX"
   name    duration    shardGroupDuration    replicaN    default
   default    504h0m0s    24h0m0s            1        true
   > exit
   ```

   The change will take a while to be applied. The default is 30 minutes.

6. Verify that the data retention change has taken effect

   ```
   # du -sh *
   3G    Cray Urika GX
   1.5G    CrayUrikaGXHadoop
   906M    CrayUrikaGXSpark
   21M    _internal
   ```

4.7 Configuration Settings of Grafana

Grafana's configuration options can be specified using environment variables or the `grafana.ini` configuration file, which is located at `/etc/grafana/`. This file contains default settings that Urika-GX ships with, and can be modified as needed. Options in this configuration file can be overridden using environment variables.

Grafana needs to be restarted for the changes in the `grafana.ini` file to take effect. To do so, log onto login2 (login node 2), become root or use `sudo`, and then execute the following command:

```
# systemctl restart grafana-server
```

In addition, the metrics that Grafana displays on the UI are retrieved from the `collectl.conf` file, which specifies the list of metrics that need to be collected from each node and stored in InfluxDB. By default, a subset of the list of collected metrics is displayed on the Grafana dashboard. Users with admin or editor privileges can create/modify dashboards to update the default list of displayed metrics if needed. The `collectl.conf` configuration file contains instructions for making such updates.

4.8 Change the Default Timezone Displayed on Grafana

Prerequisites

This procedure requires root privileges. Before performing this procedure, use the `urika-state` command to ensure that the system is operating in the service mode that supports using Grafana. For more information, see the `urika-state` man page and refer to *Urika-GX Service Modes* on page 173.

About this task

Urika-GX ships with UTC as the default timezone displayed on the Grafana UI. This procedure can be used to change the timezone for newly created dashboards or copies of the predefined dashboards shipped with the system.

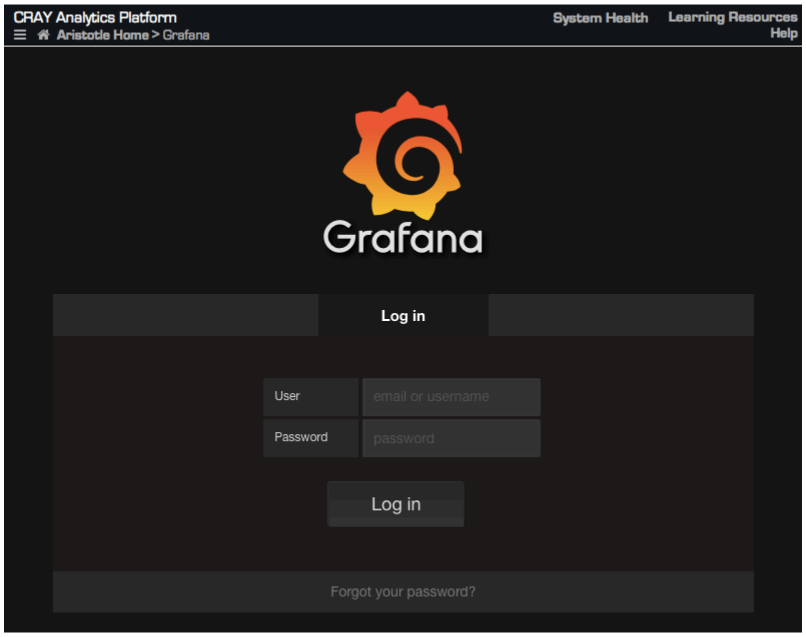
> **NOTE:** The timezone for the default set of Dashboards shipped with the system cannot be changed. Since the time displayed on the Grafana UI uses the browser's timezone and that displayed on the Spark History server's UI uses the Urika-GX system's timezone, the timezones displayed on the two UIs may not be the same.

Procedure

1. Access the Grafana UI using the **Urika-GX Applications Interface** by pointing a browser at `http://`*`hostname`*`-login1` (which is the recommended way of access) or at http://*`hostname`*-login2`:3000`.

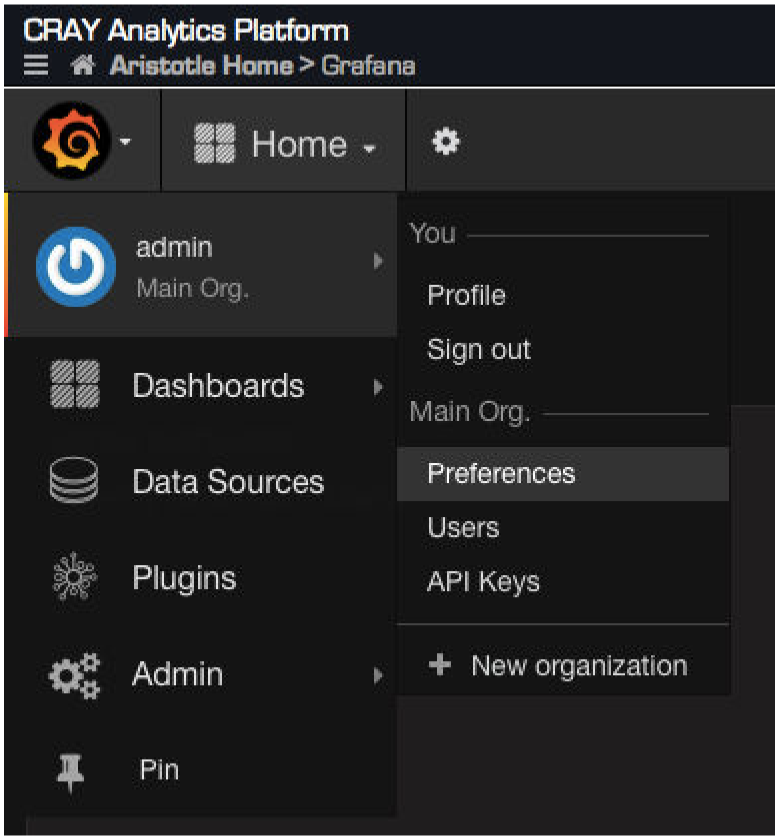
2. Log on to Grafana as an admin.

Figure 14. Grafana Login Screen



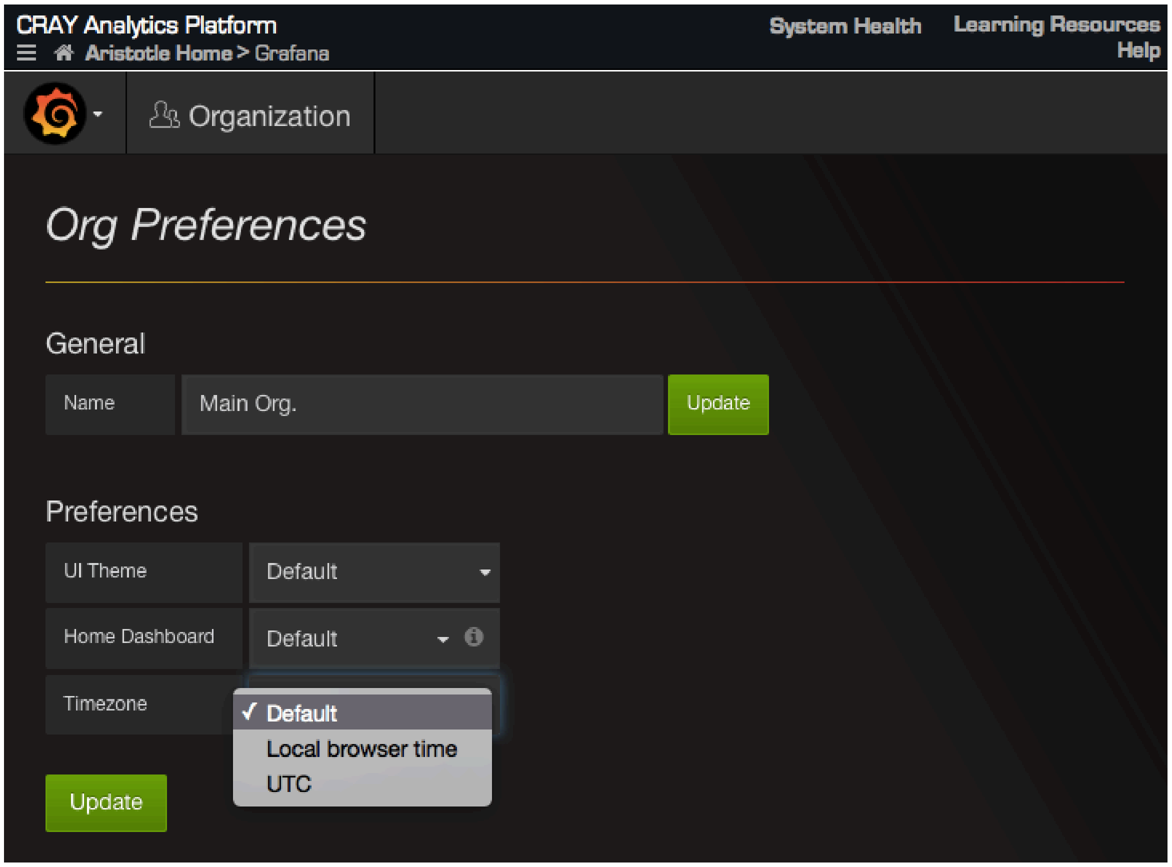
3. Select Home **>admin > Preferences**

Figure 15. Preferences Interface



4. Use the **Timezone** drop down on the **Org Preferences** page to select the timezone of choice.

Figure 16. Org Preferences Interface



Org Preferences

General

Name: Main Org.

Preferences

UI Theme: Default

Home Dashboard: Default

Timezone: ✓ Default / Local browser time / UTC

4.9 Create a New Grafana Dashboard

Prerequisites

This procedure requires administrative privileges. Before performing this procedure, use the `urika-state` command to ensure that the system is operating in the service mode that supports using Grafana. For more information, see the `urika-state` man page and refer to *Urika-GX Service Modes* on page 173.

About this task

In addition to the default set of dashboards, administrators can add additional dashboards to the Grafana UI, as described in the following instructions.

IMPORTANT: New graphs can be created/added by user of any role type. However, those added by a user with the role of a 'Viewer' persist only for the current session and will be deleted once the user logs out.

Procedure

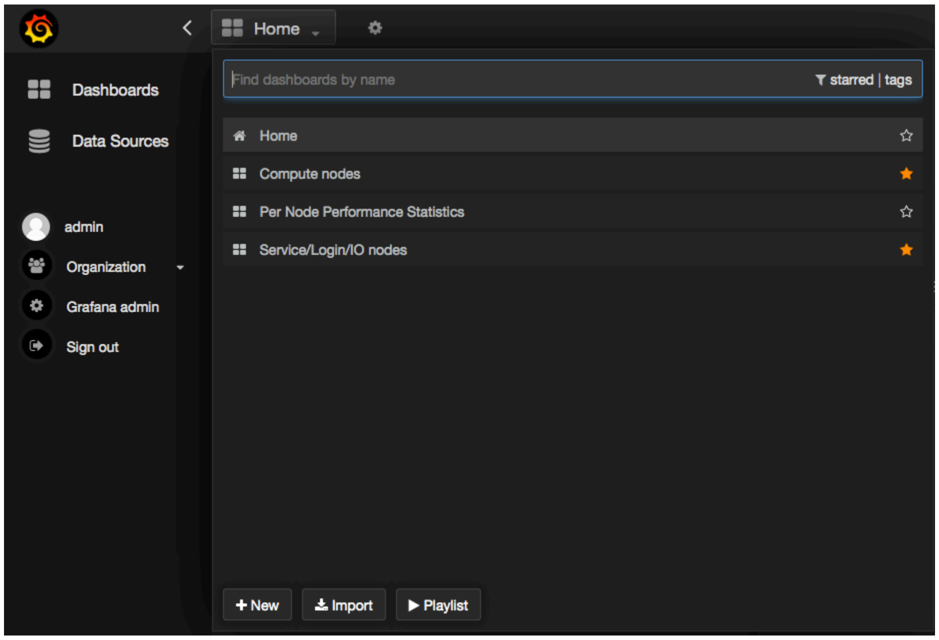
1. Access the Grafana UI using the **Urika-GX Applications Interface** by pointing a browser at `http://`*`hostname`*`-login1` (which is the recommended way of access) or at http://*`hostname`*-login2`:3000`.

2. Log on to Grafana by entering LDAP credentials, or credentials of the default Grafana account (username: `admin`, password: `admin`) to log on to Grafana.

   Figure 17. Grafana Login Screen

3. Select **Home** from the top of the interface and then select the **New** button from the displayed pop-up.

   Figure 18. New Dashboard Pop-up

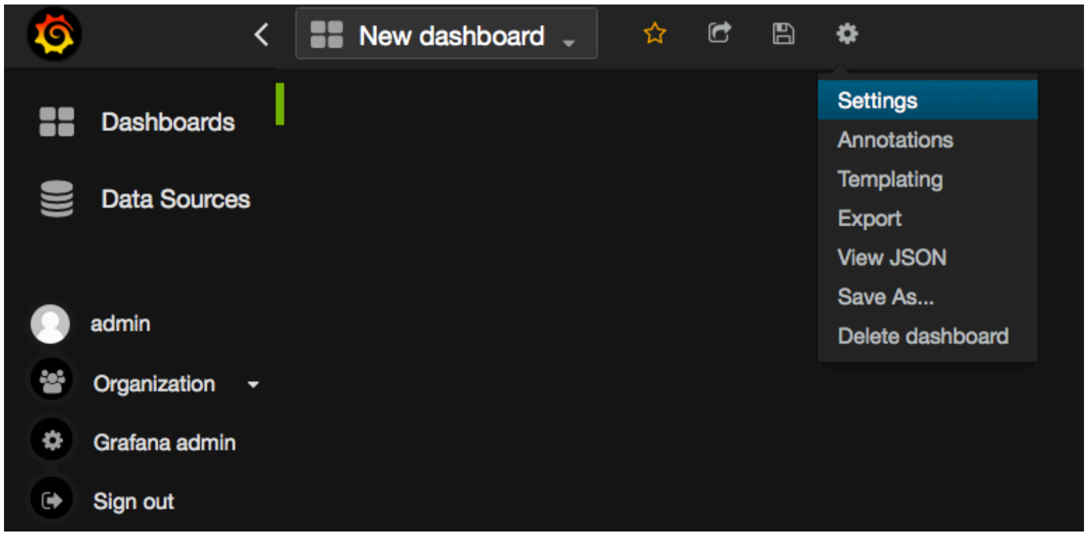


4. Select **Dashboard** from the left navigation menu

5. Add panels/graphs/rows to the new dashboard and customize as needed.

6. Optional: Update the dashboard's default settings using **Settings** from the configuration gear at the top of the interface.

Figure 19. Select Grafana Settings



4.10 Add a New Graph to the Grafana Dashboard

Prerequisites

This procedure requires administrative privileges. Before performing this procedure, use the `urika-state` command to ensure that the system is operating in the service mode that supports using Grafana. For more information, see the `urika-state` man page and refer to *Urika-GX Service Modes* on page 173.

About this task

In addition to the default set of panels, administrators can add additional panels to the **Dashboard** of the Grafana UI, as described in the following instructions.

Procedure

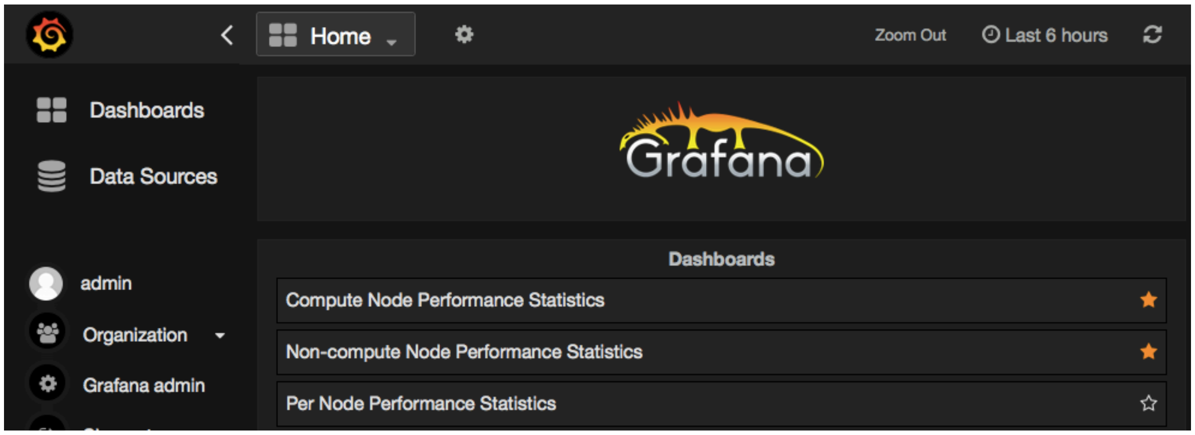
1. Access the Grafana UI using the **Urika-GX Applications Interface** by pointing a browser at `http://`*`hostname`*(which is the recommended method) or at http://*`hostname`*`:3000`.

2. Log on to Grafana by entering LDAP credentials, or credentials of the default Grafana account (username: `admin`, password: `admin`) to log on to Grafana.

   Figure 20. Grafana Login Screen

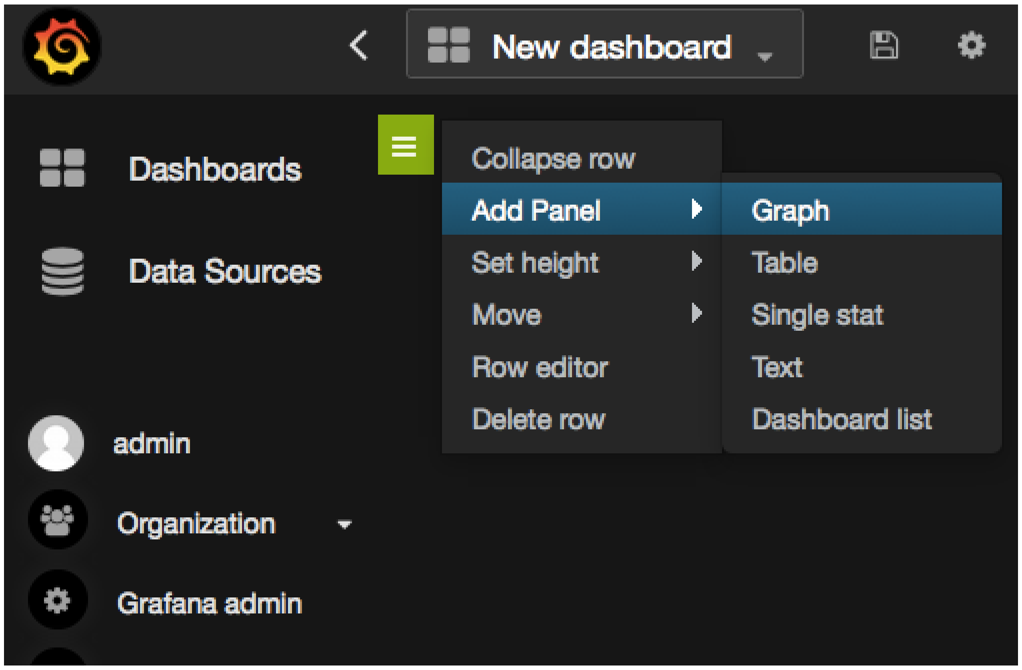
3. Select the dashboard that the new graph needs to be added to.

Figure 21. Default Grafana UI on Urika-GX


Home
Zoom Out
Last 6 hours
Dashboards
Data Sources
Grafana
admin
Organization
Grafana admin
Dashboards
Compute Node Performance Statistics
Non-compute Node Performance Statistics
Per Node Performance Statistics

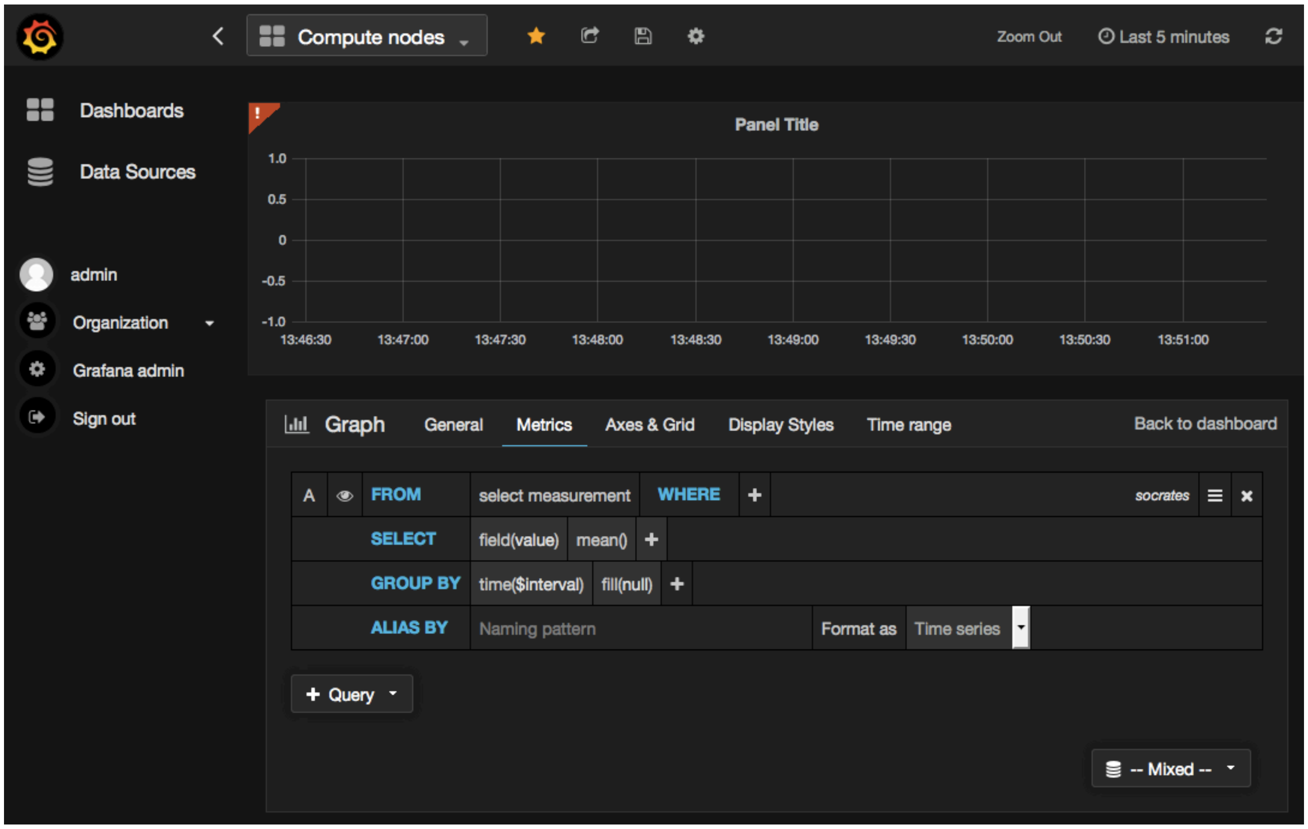
4. Select **Add Panel** > **graph** from the green menu on the left side of the Dashboard.

Figure 22. Add a New Graph to Grafana Dashboard



5. Click on the title of the new panel and select **edit** as shown in the following figure:

Figure 23. Edit a Grafana Panel



```
A  FROM      select measurement  WHERE  +                     socrates
   SELECT    field(value)  mean()  +
   GROUP BY  time($interval)  fill(null)  +
   ALIAS BY  Naming pattern            Format as  Time series
```

6. Enter a title for the graph in the **Title** field of the **General** tab and optionally specify a span and/or height for the graph.
7. Specify/select metrics using UI elements.
8. Optional: Click on the **Axes & Grid** and/or **Display Styles** tabs for specifying formatting options for the graph.

4.11 Start InfluxDB Before Hadoop Services

Prerequisites

This procedure requires root privileges. Before performing this procedure, use the `urika-state` command to ensure that the system is operating in the service mode that supports using InfluxDB. For more information, see the `urika-state` man page and refer to *Urika-GX Service Modes* on page 173.

About this task

If the **Hadoop Cluster Metrics** Grafana dashboard takes too long to populate, it may because the InfluxDB service was started after Hadoop services. Follow the instructions documented in this procedure to troubleshoot this issue.

Procedure

1. Log on to the SMW as root.
2. Stop YARN.

   ```
   # urika-stop --service yarn
   ```

3. Stop the HDFS services.

   ```
   # urika-stop --service hdfs
   ```

4. Verify that the InfluxDB service is running or start it if it is not already started.

   ```
   # urika-start --service influxdb
   ```

5. Start the HDFS services.

   ```
   # urika-start --service hdfs
   ```

6. Start YARN.

   ```
   # urika-start --service yarn
   ```

7. Refresh the browser to load the Hadoop Clusters Grafana dashboard again.

4.12 Monitor Subrack Attributes

Prerequisites

This procedure requires root access.

About this task

Access to the iSCB module is made through an SSH password-less login. Once logged on, the iSCB `status` command can be used to monitor the physical attributes of the sub-rack, such as the power supply, amperage, fan status, and temperature.

Procedure

1. Log on to the SMW as root.
2. SSH to the iSCB associated with the sub-rack under consideration.

   In the following, the iSCB associated with sub-rack `0`, having a component ID of `r0s0i0` is used as an example.

   ```
   # ssh r0s0i0
   ```

3. Execute the iSCB `status` command to retrieve the physical attributes of the sub-rack.

```
# iscb status
iSCB Status
   Node:  00  01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 Total
  Power:   *   *   *   *   *   *   *   *   *   *   *   *   *           *
 ID-LED:   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
Console:
    BMC:   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
Temp 'C: -65 -65 -64 -61 -63 -64 -61 -62 -63 -64 -65 -62 -64         -63
Watt  W:  94  86 102  98  78  83  97  87  96 107  92  86 107          87  1300

    PSU:        00        01        02        03        04        05     Total
  Power:        on        on        on        on        on        on
 Status:        ok        ok        ok        ok        ok        ok
   Temp:      22'C      24'C      23'C      23'C      23'C      23'C
    Fan: 6208,5280 6208,5280 6144,5280 6176,5248 6144,5248 6176,5312
    12V:       22A       21A       21A       21A       21A       21A      127A
  AC In:      211V      210V      210V      208V      207V      207V
   Watt:      300W      314W      308W      310W      310W      312W     1854W

    CFU:        00        01        02        03        04        05
 Status:        ok        ok        ok        ok        ok        ok
   Fan1:   3522rpm   3513rpm   3529rpm   3472rpm   3506rpm   3483rpm
   Fan2:   3540rpm   3513rpm   3538rpm   3486rpm   3529rpm   3492rpm
   Duty:       60%       60%       60%       60%       60%       60%
ok
```

4.13 Analyze Node Memory Dump Using the kdump and crash Utilities on a Node

The kdump and crash utilities may be used to analyze the memory on any Urika®-GX compute node. The kdump command is used to dump node memory to a file. kdump is the Linux kernel's built-in crash dump mechanism. In the event of a kernel crash, kdump creates a memory image (also known as vmcore) that can be analyzed for the purposes of debugging and determining the cause of a crash. Dumped image of the main memory, exported as an Executable and Linkable Format (ELF) object, can be accessed either directly during the handling of a kernel crash (through /proc/vmcore), or it can be automatically saved to a locally accessible file system, to a raw device, or to a remote system accessible over the network. kdump is configured to automatically generate vmcore crash dumps on node crashes. These dumps can be found on the node in the crash partition, mounted to nid000*XX*:/mnt/crash/var/crash/*datestamp*/*, where *XX* ranges from 00-15 for a rack containing a single sub-rack, 00-31 for a rack containing 2 sub-racks, and 00-47 for a rack containing 3 sub-racks. After kdump completes, the crash utility can be used on the dump file generated by kdump. The xtdumpsys SMW utility can be used to extract vmcores from the cluster and store them on the SMW for crash analysis as well.

NOTE: Cray recommends executing the kdump utility only if a node has panicked or is hung, or if a dump is requested by Cray.

On the Urika-GX compute nodes, kdump's system facing configuration files are set to have a kdump file stored on a local hard drive partition that is mounted as /mnt/crash so the kernel crash dumps are store in /mnt/crash/var/crash. Urika-GX has two local HDDs. kdump stores the vmcore collections on one of these drives. It is advised not to modify the /etc/kdump.conf or /etc/sysconfig/kdump configuration files.

Using kdump

- Starting kdump - Log on to the desired node and use the following command to start the kdump utility:

  ```
  $ service kdump start
  ```
- Stopping kdump - Log on to the desired node and use the following command to stop the kdump utility:

  ```
  $ service kdump stop
  ```
- Checking the status of kdump - Log on to the desired node and use the following command to check the status of the kdump utility:

  ```
  $ service kdump status
  ```

For more information, see the `kdump(8)` and `crash(8)` man pages.

4.14 Retrieve System Status Information Using the urika-check-platform Command

Prerequisites

This procedure requires root access to the System Management Workstation (SMW).

About this task

The `urika-check-platform` command can be used to retrieve various pieces of information.

Procedure

1. Log on to the SMW as root.

2. Execute the `urika-check-platform` command, as shown in the following examples.

   - Various HSS checks, Rack controller, dANCs, `xtcli` status, and `xtcablecheck` using the `--hss` option:

     ```
     # urika-check-platform --hss
     ```
   - Get revision info of Rack Controller, iSCBs, dANCs, and compute nodes and HSS using the `--rev` option:

     ```
     # urika-check-platform --rev
     ```

     If nodes are not listed then revision information is unavailable. Make sure `/global/cray/mfg/self_update_fw/intel_fw_chk.sh` is installed properly on each node.
   - Check if Aries is routed:

     ```
     # urika-check-platform --aries
     ```
   - Test node connectivity using the `--testnodes` option. First perform the ping test. If the nodes can be pinged, then ssh is attempted. If ssh succeeds then the Aries IP is resolved and displayed next to the hostname with the subnet mask:

```
# urika-check-platform --testnodes
```

- Display all of the aforementioned sections together by executing the command without any options:

```
# urika-check-platform
```

For more information, see the `urika-check-platform` man page.

4.15 iSCB Description

The Urika®-GX SR10216 subrack includes an Intelligent Subrack Control Board (iSCB) for remote system management. The iSCB module reports the functional status and provides power control over the system. An administrator can use the iSCB from a remote client to monitor the state of subrack components, such as: nodes, power supplies, and fan modules. The Hardware Supervisory System (HSS) provides a command-line interface using the `xtcli` command for the configuration and management of the SR10216 subrack system. Use only the HSS `xtcli` command to monitor and control the system.

CAUTION: Do not use the CLI commands other than the status command for the ISCB module (unless advised by Cray Support), as they may interfere with performance of the system management software. For more information, contact Cray support.

Each Urika-GX iSCB module has a single Gigabit Ethernet port and a dedicated logical serial port assigned for command and power management access. The iSCB module is typically managed over a secure private management network. Cray recommends the iSCB not be connected to a publicly accessible network.

Once logged into the HSS using the `xtcli` command, administrators can:

- Quickly access system and component information
- Quickly view hardware inventory, system event logs, and system configuration reports
- Dynamically control fan speeds or set static speeds
- Monitor sensors to dynamically change fan speeds and maintain optimal operating temperatures
- Power blades on and off
- Easily view current status for all hardware components (blades, cooling modules, and power modules)
- Configure alert notification destinations and policies

Refer to the *Urika®-GX System Administration Guide S-3016* for more information about using the `xtcli` command and other management utilities.

4.15.1 Log on to the iSCB

Access to the iSCB module is made through an SSH password-less login. Once logged on, iSCB CLI commands can be run to monitor and control components of the subrack.

```
[r0s0i0 root]# ssh r0s0i0
Last login: Wed Jun 29 20:55:06 2016 from r0s0i0
[r0s0i0 root]# iscb ver
iSCB Ver 1.1 (11:53:51 02/18/16 PST) active (0h)
ok
[r0s0i0 root]#
```

Log Off the iSCB

Log off the iSCB:

```
[r0s0i0 root]# exit
logout
Connection to r0s0i0 closed.
[r0s0i0 root]#
```

IP Address

When `iscb` starts up, the value of the `SCB IP Source` configured in NVRAM is used obtain the IP address: either `static` or `dhcp`. Switch IDSW3-8 on the SR10216 front panel defines the IP address type: static or DHCP.

Static IP. If the IDSW3-8 switch is set to On/1, the IP address used by `iscb` is based on this scheme:

```

192.168.XX.YZ
       XX: Cabinet ID
        Y: iSCB ID (1=iSCB0, 2=iSCB1)
        Z: Subrack ID
```

A non-functional or absent IDSW results in a default IP address setting of 10.11.1.10.

DHCP IP. If the IDSW3-8 switch is set to Off/0, the IP address used by `iscb` is based on addresses defined in the `dhcpd.conf/dhcpd.hss` and `/etc/hosts` files.

```
 /etc/hosts
192.168.2.10    r0s0i0 rx0y0s0i0 iscb_r0s0i0
192.168.2.11    r0s1i0 rx0y0s1i0 iscb_r0s1i0
192.168.2.12    r0s2i0 rx0y0s2i0 iscb_r0s2i0
192.168.2.13    r0s3i0 rx0y0s3i0 iscb_r0s3i0
```

iSCB Password

iSCB password. There is no default iSCB password. An administrator should provide one by running the `set passwd` command while logged in to the iSCB. The new password is NOT stored on the iSCB itself, but instead saved in NVRAM in the subrack's I/O module. The new password can be applied permanently by running the `nvram save` command.

To reset a forgotten iSCB password:

1. Set the iSCB module to Debug Mode by setting SW4-1 On (1000).
2. Log on to `iscb` without a password.
3. Run the `set passwd` command to define a password.
4. Run the `nvram save` command to save the password to NVRAM.
5. Reset the iSCB module to Normal Mode by setting SW4-1 Off (0000).
6. Run the `reboot` command to apply the new password.

4.15.2 iSCB Command Reference

Command List

bmc	Display the BMC status of specified node
bootmode	Display or set boot mode
cfu	Display information of all or specified CFUs
console	Display established cli connections or kill the specified connection
crash	Create a core dump?
danc	Display status and power on/off/cycle all or specified dANCs
fan	Display CFU status or set fan PWM duty value
fcb	Display power usage of CFU from Fan Control Board
help	Display available commands and usage
history	Display list of commands that were previously entered in the current CLI session
identify	Identify the node in the subrack by blinking the green/red LED and the node’s ID-LED
lasterr	Display recently occurred event log or clear the last log
led	Turn Off the green/red LED and the node’s ID-LED
log	Display event logs by specified log level: all, info, critical, and warning
node	Display status of all or specified nodes
nvram	Display or update the configuration parameters stored in NVRAM
pmled	Display powerman style ID-LED status
pmnode	Display powerman style power status
power	Power on/off/cycle/reset all nodes or specified node
profile	Display a profile of the specified device
psu	Display status, power consumption, and energy data of all or specified PSUs
psuclear	Reset the energy usage counter
psumodel	Display detailed model information and serial number of all PSUs
psumodelclear	Remove the serial numbers for the PSUs in the subrack
pump	Display pump sensor data from specified node
reboot	Reboot iSCB to apply modified parameter settings or to update firmware
restart	Restart the iSCB service
sendmail	Test sending an e-mail
sensor	Display temperature sensor data from all or specified nodes
set	Display or set iSCB parameters

shutdown	Shutdown all or specified nodes
status	Display Node, PSU and FAN status
stop	Stop the iSCB service
tty	Display status of all established remote console sessions or kill specified connection
updatefw	Update iSCB firmware
ver	Display the current firmware version and build date

Command Syntax

bmc

Display BMC data from all or specified nodes

Syntax: `bmc [node]`

Parameters:

`node`	Specific node, 1 through 16

Example:

```
[r0s0i0 /]# iscb bmc
No Name         Pwr PID  BMC    MACaddr           IPaddr-Ch1      IPaddr-Ch2
 0 r0s0n0       on  0070   1.08 00:1e:67:b5:2b:45 10.10.128.3     0.0.0.0
 1 r0s0n1       on  0070   1.08 00:1e:67:b5:2a:8c 10.10.128.4     0.0.0.0
 2 r0s0n2       on  0070   1.08 00:1e:67:b5:2a:5f 10.10.128.5     0.0.0.0
 3 r0s0n3       on  0070   1.08 00:1e:67:b5:2b:63 10.10.128.6     0.0.0.0
 4 r0s0n4       on  0070   1.08 00:1e:67:b5:29:2e 10.10.128.7     0.0.0.0
 5 r0s0n5       on  0070   1.08 00:1e:67:b5:2b:f4 10.10.128.8     0.0.0.0
 6 r0s0n6       on  0070   1.08 00:1e:67:b5:28:f7 10.10.128.9     0.0.0.0
 7 r0s0n7       on  0070   1.08 00:1e:67:b5:28:bb 10.10.128.10    0.0.0.0
 8 r0s0n8       on  0070   1.08 00:1e:67:b5:2a:46 10.10.128.11    0.0.0.0
 9 r0s0n9       on  0070   1.08 00:1e:67:b5:2a:41 10.10.128.12    0.0.0.0
10 r0s0n10      on  0070   1.08 00:1e:67:96:40:8b 10.10.128.13    0.0.0.0
11 r0s0n11      on  0070   1.08 00:1e:67:b5:2b:13 10.10.128.14    0.0.0.0
12 r0s0n12      on  0070   1.08 00:1e:67:b5:2b:72 10.10.128.15    0.0.0.0
13 r0s0n13      on  0070   1.08 00:1e:67:b5:28:02 10.10.128.16    0.0.0.0
14 r0s0n14      on  0070   1.08 00:1e:67:b5:2c:53 10.10.128.17    0.0.0.0
15 r0s0n15      on  0070   1.08 00:1e:67:b5:28:0c 10.10.128.18    0.0.0.0
ok
[r0s0i0 /]#
```

bootmode

Display or set boot mode

Syntax: `bootmode [device][mode]`

Parameters:

`device`	iSCB or dANC

Example:

```
[r0s0i0 /]#  iscb bootmode
iscb0:   normal (11H)
iscb1:   normal (11H)
danc0:  netboot (33H)
danc1:  netboot (33H)
ok
[r0s0i0 /]# iscb bootmode danc0
danc0: netboot (33H)
```

```
ok
[r0s0i0 /]#
```

cfu

Display information of all or specified CFUs

Syntax: `cfu [cfuno]`

Parameters:

`cfuno`	CFU number

Example:

```
[r0s0i0 /]# iscb cfu
No Stat  Fan0  Fan1 Duty Demand Margin
 0   ok  3432  3446   60     60    -46
 1   ok  3450  3417   60     60    -46
 2   ok  4090  4112   90     60    -45
 3   ok  3446  3428   60     60    -47
 4   ok  3443  3439   60     60    -47
 5   ok  3439  3450   60     60    -47
ok
ok
[r0s0i0 /]#
```

console

Display established CLI connections or kill specified connection. The asterisk (*) indicates the current connection.

Syntax: `console [kill slot_number]`

Parameters:

`kill slot_number`	Kill the specified CLI connection.

Example:

```
[r0s0i0 /]# iscb console
 No Auth Remote            Timeout         In        Out Last Command
*1  pass 10.10.0.1,52792         0        151       9961 ver power power -hhel..
 2  pass 10.10.0.1,35493         0         73       3905
ok
[r0s0i0 /]# iscb console kill 2
ok
[r0s0i0 /]# iscb console
 No Auth Remote            Timeout         In        Out Last Command
*1  pass 10.10.0.1,52792         0        177      10197 ver power power -hhel..
ok
[r0s0i0 /]#
```

crash

Create a core dump

Syntax: `crash`

Parameters:

Example:

```
[r0s0i0 /]# iscb crash
ok
[r0s0i0 /]#
```

danc

Display danc status or set specified danc power on/off/cycle.

Syntax: `danc [status|on|off|cycle] [0|1|all]`

Parameters:

`status`	Display status of specific dANC
`on`	Power on specific dANC
`off`	Power off specific dANC
`cycle`	Power cycle specific dANC

Example:

```
[r0s0i0 /]# iscb danc
# PMDev       Pwr +12V  Watt Ver Stat dANC Ars0 Ars1 AOC0 AOC1 AOC2 AOC3 AOC4
0 LM25066 AA  on  12.1  51.2   2 0000   30   38   39    0    0    0    0    0
1 LM25066 AA  on  12.1  51.8   2 0000   30   38   38    0    0    0    0    0
ok
[r0s0i0 /]# iscb danc status 0
 danc#: 0
PM Dev: LM25066 AA
 power: on
  +12V:  12.1 V
  Watt:   5.0 W
   Ver: 2
  stat: 0103h
htbeat: 59
  dANC: 22 'C (55)
Aries0: 29 'C (85)
Aries1: 30 'C (85)
  AOC0: 0 'C (-1)
  AOC1: 0 'C (-1)
  AOC2: 0 'C (-1)
  AOC3: 0 'C (-1)
  AOC4: 0 'C (-1)
ok
[r0s0i0 /]# iscb danc cycle all
ok
[r0s0i0 /]#
```

fan

Display or set CFU fan speed duty. If there is no access of iSCB from outside over 90 seconds, iSCB will change to full speed automatically

Syntax: `fan [cfuno|all][high|normal|low]`

Parameters:

`cfuno`	CFU number
`all`	All CFUs
`high`	Set to full speed. (100%)
`normal`	Set to normal speed. (85%)
`low`	Set to low speed. (60%)

Example:

```
[r0s0i0 /]# iscb fan
    CFU:        00        01        02        03        04        05
 Status:        ok        ok        ok        ok        ok        ok
   Fan1:   3468rpm   3470rpm   3501rpm   5405rpm   5510rpm   3486rpm
   Fan2:   3474rpm   3483rpm   3497rpm   6013rpm   6040rpm   3486rpm
   Duty:       60%       60%       60%      100%      100%       60%
ok
[r0s0i0 /]# iscb fan 05 normal
ok
[r0s0i0 /]# iscb fan
    CFU:        00        01        02        03        04        05
 Status:        ok        ok        ok        ok        ok        ok
   Fan1:   3504rpm   3504rpm   3513rpm   5394rpm   5454rpm   4365rpm
   Fan2:   3483rpm   3495rpm   3504rpm   6000rpm   6000rpm   4591rpm
   Duty:       60%       60%       60%      100%      100%       83%
ok
[r0s0i0 /]#
```

fcb

Display power usage of CFUs through Fan Control Board

Syntax: `fcb [fcb#]`

`fcb#`	The specified fan control board (fcb)

Example:

```
[r0s0i0 /]# iscb fcb
No Stat  +12V  Curr  Watt  Peak  Avrg Model
 0 ok    12.2   7.0  85.2 282.7  83.3 ADM1276-3
 1 ok    12.2   5.4  66.3 283.7  66.8 ADM1276-3
Power : 12.0 A 153 W  (Peak 558 W, Average 477 W)
Energy: 13520.6 Wh in last 101865secs(28h 17m 45s)
ok
[r0s0i0 /]#
```

help

Display available commands and usage

Syntax: `help [command]`

Parameters:

`command`	Displays usage of *command* name

Example:

```
[r0s0i0 /]# iscb help power
power on specific node(s) or all
  power on <STRING:nodes> { <INT:interval> }

power off specific node(s) or all
  power off <STRING:nodes> { <INT:interval> }

power cycle specific node(s) or all
  power cycle <STRING:nodes> { <INT:interval> }

reset specific node(s) or all
  power reset <STRING:nodes>

shutdown specific node(s) or all
  power shutdown <STRING:nodes>

  set powermask { <UINT:mask> }

  set powerlimit { <INT:node> <INT:shelf> }

ok
[r0s0i0 /]#
```

history

Display list of commands that were previously entered in the current CLI session

Syntax: `history`

Parameters: None

Example:

```
[r0s0i0 /]# history
tty
help
history
ver
ok
[r0s0i0 /]#
```

lasterr

Display recently occurred event log or clear the last log

Syntax: `lasterr [clear]`

Parameters:

`clear`	Clear the last log and reset the red LED on the iSCB front panel

Example:

```
[r0s0i0 /]# iscb lasterr
ALERT: CFU02 failed
ok
[r0s0i0 /]# iscb lasterr clear
ok
[r0s0i0 /]# iscb lasterr
ok
[r0s0i0 /]#
```

led

Turn Off the green/red LED and the node’s ID-LED

Syntax: `led <on|off> <node|all>`

Parameters:

`on`	Turn on the specific nodes idled. If node is already turned on, there will be no action.
`off`	Turn off the specific nodes idled. If node is already turned off, there will be no action.
`node|all`	Refer to ‘power’ command

Example:

```
[r0s0i0 /]# iscb led on 1
ok
[r0s0i0 /]# iscb led off 1
ok
[r0s0i0 /]# iscb led on all
ok
[r0s0i0 /]# iscb led off all
ok
```

log

Display event logs by one of three specified log levels: info, critical, and warning

Syntax: `log [info|critical|warning]`

Parameters:

`info`	Display notice, and info and higher level event logs
`critical`	Display err, crit, alert and emerg level event logs
`warning`	Display warning and higher level event logs

Example:

```
[r0s0i0 /]# iscb log critical
1: Jan  6 23:06:26 iscb:iscbd.sh: Detected location from IDSW: 0-0-0
2: Jan  6 23:06:26 iscb:iscbd.sh: 01/06/70 23:06:26: Starting iscb daemon. args:-d .
3: Jan  6 17:06:26 iscbd[1452]: >>>>> iscbd started
4: Jan  6 17:06:27 iscbd[1452]: NOTICE: Starting CLI service through the TCP port 7000
5: Jan  6 17:06:27 iscbd[1452]: SMTP mail service started. (email: server:)
6: Jan  6 17:06:27 iscbd[1452]: NOTICE: Starting remote console service through the TCP port 7001-7016
7: Jan  6 17:06:27 iscbd[1452]: NOTICE: iSCB0: running Stand-by mode.
8: Jan  6 17:06:32 iscbd[1452]: NOTICE: No HEARTBEAT from iSCB1
9: Jan  6 17:06:34 iscbd[1452]: NOTICE: iSCB0 is going to be Active..
ok
[r0s0i0 /]#
```

node

Display status of all or specified nodes

Syntax: `node [node|all]`

Parameters:

`node`	Display status for specified node, 1 through 16
`all`	Display status for all nodes

Example:

```
[r0s0i0 /]# iscb node
No Name         Pwr LED GPU BMC    PID  Sns GPUMap Watt Tmrg Console
 0 r0s0n0       on  off off   1.08 0070 4+0   0000  108  -63 idle
 1 r0s0n1       on  off off   1.08 0070 4+0   0000  101  -61 idle
 2 r0s0n2       on  off off   1.08 0070 4+0   0000  100  -66 idle
 3 r0s0n3       on  off off   1.08 0070 4+0   0000  107  -61 idle
 4 r0s0n4       on  off off   1.08 0070 4+0   0000  102  -63 idle
 5 r0s0n5       on  off off   1.08 0070 4+0   0000  101  -59 idle
 6 r0s0n6       on  off off   1.08 0070 4+0   0000  104  -64 idle
 7 r0s0n7       on  off off   1.08 0070 4+0   0000  110  -64 idle
 8 r0s0n8       on  off off   1.08 0070 4+0   0000  100  -69 idle
 9 r0s0n9       on  off off   1.08 0070 4+0   0000  112  -69 idle
10 r0s0n10      on  off off   1.08 0070 4+0   0000  109  -66 idle
11 r0s0n11      on  off off   1.08 0070 4+0   0000   99   ?? idle
12 r0s0n12      on  off off   1.08 0070 4+0   0000  101  -63 idle
13 r0s0n13      on  off off   1.08 0070 4+0   0000  102  -63 idle
14 r0s0n14      on  off off   1.08 0070 4+0   0000  157   ?? idle
15 r0s0n15      on  off off   1.08 0070 4+0   0000  112  -66 idle
Power : 88.0 A 1725 W  (Peak 1868 W, Average 1737 W)
Energy: 4987.5 Wh in last 10331secs(2h 52m 11s)
ok
[r0s0i0 /]# iscb node 3
     id: 3
   slot: 3
   name: r0s0n3
  power: on
 id-led: off
 status: ok
console: 7004,idle
prod-id: 0070
BMC ver: 1.08
 sensor: 4+0
    gpu: -
```

```
ok
[r0s0i0 /]#
```

nvram

Display or update the configuration parameters stored in NVRAM

Syntax: `nvram [save|clear]`

Parameters:

`save`	Store current configuration parameters into the NVRAM
`clear`	Remove current configuration parameters and load default parameters. Parameters are loaded but not saved until the `save` command is used.

Example:

```
[r0s0i0 /]# iscb nvram
Versions      1.1 (11:53:51 02/18/16 PST) NVRAM:1.0
Shelf         Type:SR10216A Location:0-0 Name:r0s0
Blades        Type:GB522Xaa BladeName:r0s0n0 PowerMask:0x0000
GPU/Co-Proc   Type:auto SlotMap:0000
TCP Ports     CLI:7000 RemoteConsole:7001-7016 Baud:115200
SCB IP Source dhcp
SCB0 ETH0 IP  192.168.2.10/255.255.255.192
SCB1 IP Addr  10.11.0.10/255.255.0.0 gw 10.11.0.1 ns 10.11.0.1
SCB2 IP Addr  10.11.0.20/255.255.0.0 gw 10.11.0.1
Time/Date     TZ:CST6CDT TimeServer:192.168.0.1
Event Log     Level:info
Remote Log    Level:info LogServer:192.168.0.1 Port:514
SNMP Trap     Level:off Community:public Receiver:0.0.0.0
E-Mail        Level:off ID: Server: User: Pass:
Blade Action  Shutdown:off ID-LED:warning
Fan Speeds(%) Default:100 Low:60 Normal:90 High:100
Margins ('C)  Low:30 High:45
Temperatures  Warning:90 Critical:95 CPUMax:95 GPUMax:95
Power limits  Node:0W Shelf:0W
FanControl    ActiveFan:on FastFan:off LiquidCooling:off
Connections   Telnet:on SSH:on ACE:on ACEloglevel:info ACEdebug=off
Options       Debug:off dANC:on
ok
[r0s0i0 /]#
```

pmled

Display powerman style ID-LED status

Syntax: `pmled [node|all]`

Parameters:

`node`	Display status for a specified node, 1 through 16
`all`	Display status for all nodes

Example:

```
[r0s0i0 /]# iscb pmled
node01: on
node02: off
node03: n/a
node04: n/a
node05: on
node06: on
node07: off
node08: off
node09: off
node10: off
node11: on
```

```
node12: on
node13: on
node14: on
node15: on
node16: on
ok
[r0s0i0 /]#
```

pmnode

Display powerman style power status

Syntax: `pmnode [node|all]`

Parameters:

`node`	Display status for a specified node, 1 through 16
`all`	Display status for all nodes

Example:

```
[r0s0i0 /]# iscb pmnode 7
node07: on
ok
[r0s0i0 /]#
```

power

Power on/off/cycle/reset the specific node(s). If the node is already turned on, there will be no action. When multiple nodes are given for power on/off/cycle operation, an interval value is required. <CR> key to abort the power operation.

Syntax: `power <on|off|cycle|shutdown|reset> <node|all> [interval]`

Parameters:

`on`	Turn on the specific nodes. If node is already turned on, there will be no action.
`off`	Turn off the specific nodes. If node is already turned off, there will be no action.
`cycle`	Turn off, then turn on, the specific nodes.
`reset`	Reset specific nodes. No interval time required for multiple nodes.
`shutdown`	OS shutdown of <all> or specified <nodes>. This command asserts the ATX power control signal for 0.4 second to gracefully shutdown the node if it is powered on. The specific node should be in proper ACPI mode w/ ATX power control. If the node is in abnormal mode such as hung-up or PANIC, this command may not work properly. Please check the node power status once the command completes. No interval time is required for multiple nodes.
`node|all`	Specific node 1 through 16 **or** `all`: all the nodes **or** 0xffff: Bit fields for multiple specific nodes 0x3: Node 1,2

	0x5: Node 1,3 0xff00: Node 9-16 **or** 1,2,..n: slot number separated by comma(,) 1,2,4: Node 1,2,4 **or** 1-n: slot number scope by dash(-) 1-4: Node 1,2,3,4 **or** cn000[1,2,..-n]: specified blade name by name with dash and comma cn000[1,4-6]: Node 1,4,5,6
`interval`	Interval time for power on operation of multiple nodes. Unit is 0.1 second. 0 means no interval.

Example:

```
[r0s0i0 /]# iscb power on 1
ok
[r0s0i0 /]# iscb power off 1
ok
[r0s0i0 /]# iscb power on all 1
ok
[r0s0i0 /]# iscb power off all 0
ok
```

profile

Display a profile of all devices/nodes

Syntax: `profile`

Example:

```
[r0s0i0 /]# iscb profile
Device   result dur  min  max  count err  last_access
blade0   0      5    3    5    10775 0    48781
    pm0  0      5    4    5    10774 0    48781
blade1   0      4    3    5    10775 0    48782
    pm0  0      5    4    5    10774 0    48782
blade2   0      4    4    5    10775 0    48783
    pm0  0      5    4    5    10774 0    48783
blade3   0      5    4    5    10774 0    48780
    pm0  0      4    4    5    10773 0    48779
blade4   0      5    4    6    10583 0    48782
    pm0  0      4    4    6    10582 0    48781
blade5   0      5    4    6    10583 0    48783
    pm0  0      4    4    6    10582 0    48782
blade6   0      5    4    6    10583 0    48784
    pm0  0      4    4    6    10582 0    48783
blade7   0      4    4    6    10582 0    48780
    pm0  0      5    4    6    10581 0    48780
blade8   0      4    3    6    10692 0    48781
    pm0  0      4    4    6    10691 0    48781
blade9   0      5    3    6    10692 0    48783
    pm0  0      4    4    6    10691 0    48782
blade10  0      5    4    5    10692 0    48784
    pm0  0      4    4    6    10691 0    48783
blade11  0      4    4    5    10691 0    48780
    pm0  0      5    4    6    10690 0    48780
blade12  0      4    4    6    10666 0    48780
    pm0  0      5    4    6    10665 0    48780
blade13  0      4    4    6    10666 0    48781
    pm0  0      5    4    6    10665 0    48781
blade14  0      4    4    6    10666 0    48782
    pm0  0      4    4    6    10665 0    48782
```

```
blade15  0      5    4    6    10666 0    48784
    pm0  0      4    4    5    10665 0    48783
PSU0     0      2    1    2    47337 0    48784
PSU1     0      2    1    2    47337 0    48784
PSU2     0      2    1    2    47337 0    48784
PSU3     0      1    1    2    47338 0    48784
PSU4     0      1    1    2    47338 0    48784
PSU5     0      1    1    2    47338 0    48784
FCB0     0      1    1    2    46589 0    48783
   pm    0      1    1    2    46589 0    48783
FCB1     0      1    1    2    46589 0    48783
   pm    0      1    1    2    46589 0    48783
dANC0    0      3    3    4    16156 0    48782
dANC0 pm 0      3    3    4    16167 0    48782
dANC1    0      3    3    4    16156 0    48782
dANC1 pm 0      3    3    4    16167 0    48782
ok
[r0s0i0 /]#
```

psu

Display status, power consumption, and energy data of all or specified PSUs.

Syntax: `psu <psuno>`

Parameters:

`psuno`	PSU number

Example:

```
[r0s0i0 /]# psu 4
psu#: 4
powr: on
modl: PSSH162202A PMBus 1.2/1.2
mode: 90h/08h
stat: ok (0000h)
temp: 24 'C
fan1: 6112 RPM
fan2: 5248 RPM
+12v: 12.0 V
curr: 22 A
 acv: 207 V
watt: 304 W
avrg: 0.0  W
ok
[r0s0i0 /]#
```

psuclear

Reset the energy usage counter

Syntax: `psuclear`

Parameters: None

Example:

```
[r0s0i0 /]# iscb psuclear
ok
[r0s0i0 /]#
```

psumodel

Display detailed model information and serial number of all PSUs.

Syntax: `psumodel [psuno]`

Parameters:

`psuno`	PSU number

Example:

```
[r0s0i0 /]# iscb psumodel
No Pwr  Stat Model                                    Serial Number
00 on   ok   PSSH162202A      H/W Rev.:1.0, F/W Rev.:10.5.0 01234567890123456789
01 on   ok   PSSH162202A      H/W Rev.:1.0, F/W Rev.:10.5.0 01234567890123456789
02 on   ok   PSSH162202A      H/W Rev.:1.0, F/W Rev.:10.5.0 01234567890123456789
03 on   ok   PSSH162202A      H/W Rev.:1.0, F/W Rev.:10.5.0 01234567890123456789
04 on   ok   PSSH162202A      H/W Rev.:1.0, F/W Rev.:10.5.0 01234567890123456789
05 on   ok   PSSH162202A      H/W Rev.:1.0, F/W Rev.:10.5.0 01234567890123456789
ok
[r0s0i0 /]#
```

pump

Display pump sensor data from specified node

Syntax: `pump [node]`

Parameters:

`node`	Specified node, 1 through 16

Example:

```
[r0s0i0 /]#  iscb pump 1
   slot: 1
   name: r0s0n1
present: on
  power: on
  pumps: no pump sensor
ok
[r0s0i0 /]#
```

reboot

Reboot iSCB to apply modified parameter settings or to update firmware

Syntax: `reboot`

Parameters: None

Example:

```
[r0s0i0 /]# iscb reboot
ok
[$ /]# Connection closed by foreign host.
```

sensor

Display temperature sensor data from all or specified nodes

Syntax: `sensor [node|all]`

Parameters:

`node`	Specified node, 1 through 16
`all`	Display status for all nodes

Example:

```
[r0s0i0 /]# iscb sensor
No Name         Pwr CPU1 CPU2  Inl Outl CNmg CNdt PCI1 PCI2 PCI3 PCI4 Gmrg Marg
 0 r0s0n0       on   -63  -68   24    -  -63   -1    -    -    -    -    1  -63
 1 r0s0n1       on   -45  -57   25    -  -45   -1    -    -    -    -    1  -45
 2 r0s0n2       on   -62  -67   25    -  -62   -1    -    -    -    -    1  -62
 3 r0s0n3       on   -58  -65   25    -  -58   -1    -    -    -    -    1  -58
 4 r0s0n4       on   -61  -67   25    -  -61   -1    -    -    -    -    1  -61
 5 r0s0n5       on   -61  -66   25    -  -61   -1    -    -    -    -    1  -61
 6 r0s0n6       on   -61  -66   26    -  -61   -1    -    -    -    -    1  -61
 7 r0s0n7       on   -51  -58   26    -  -51   -1    -    -    -    -    1  -51
 8 r0s0n8       on   -63  -65   24    -  -63   -1    -    -    -    -    1  -63
 9 r0s0n9       on   -62  -69   24    -  -62   -1    -    -    -    -    1  -62
10 r0s0n10      on   -60  -67   25    -  -60   -1    -    -    -    -    1  -60
11 r0s0n11      on   -59  -63   26    -  -60   -1    -    -    -    -    1  -60
12 r0s0n12      on   -62  -66   25    -  -62   -1    -    -    -    -    1  -62
13 r0s0n13      on   -61  -67   25    -  -61   -1    -    -    -    -    1  -61
14 r0s0n14      on   -59  -68   25    -  -59   -1    -    -    -    -    1  -59
15 r0s0n15      on   -58  -66   25    -  -58   -1    -    -    -    -    1  -58
ok
[r0s0i0 /]# sensor 4
     id: 4
   slot: 4
   name: r0s0n4
present: on
  power: on
sensors:  4
 status: ok
       p1_margin: ok  ( -60.0 'C)
       p2_margin: ok  ( -68.0 'C)
           inlet: ok  (  25.0 'C)
          outlet: disabled
           delta:     (    -1 'C)
     node_margin:     (   -60 'C)
ok
[r0s0i0 /]#
```

set

Display or set iSCB parameters

Syntax: `set <param> [value]`

Parameters:

`ace [on|off]`	Support ACE cluster. Not implemented for Urika-GX. Default is off.
`acelog [level]`	If ace is set, generate ace related log of defined level: info, warning, critical
`activefan [on|off]`	Dynamic Fan Control by node's health status. The each CFU will run at low, normal and high speed by where located node's temperature. If it is off, entire CFU will run at default fan speed. This setting is override by IDSW1-7. Default is on.
`authkey [auth_key]`	Authorized key into `/root/.ssh/authorized_key`.
`baud [speed]`	Remote console speed. Default is 115200.*
`blade [type [name [id]]]`	Blade type, name(max 14chars) and ID. Type is one of auto, GB5XX, GB6XX, GB8XX and GB828.*
`checkpsu [on|off]`	Check the PSUs. If it is off, the power and energy usage capability of subrack level will not work. Default is on.*
`cputemp [temp]`	The reference temperature of CPU. For Intel EPSD platform, this value is will be used for reference temperature for marginal bmc sensors. For example, the BMC reports CPU temperature to -60'C, the actual temperature is 90-60 = 30'C. Default is 90.

`danc [on|off]`	dANC module power on|off.
`debug [on|off]`	Debug option.*
`email [level [id [server [user [pass]]]]]`	Event level, e-mail address, mail server/user/passwd of E-Mail notification. Default community is public.*
`fanspeed [low normal high]`	PWM duty of low, normal and high fan speed. Default is 60%, 90%, and 100%.
`fastfan [on|off]`	Dynamic Fast Fan Control by node's health status. The entire CFU will run at low, normal and high speed by one of highest node's temperature. If it is off, entire CFU will run at normal active fan control mode. Default is off.
`fantemp [normal high]`	Temperature of normal and high FAN speed. Default is 50'C and 65'C. For example, if the blade exceeds 65'C, the CFU will run at high speed. temp [warning critical]. The temperature region of warning and critical. Default is 85'C and 90'C. For example, if the blade exceeds 85'C, the blade health status is set to warning.
`gpu [type [slotmap]]`	GPU/Co-proc type and PCI slot map. Type is one of auto, m20xx, k10, k20, k40, k80 and phi.*
`gputemp [temp]`	The critical temperature of GPU/Co-proc expansion. For nVidia M20XX and Kepler GPU, this value is will be used for reference temperature for warning and critical temperature region. For example, the nVidia GPU reports temperature to 90'C, the iSCB determines the GPU health status is 'CRITICAL'. But for Intel MIC, the health status will be determined by MIC SMC. Default is 90.
`gw [gw [unitid]]`	iSCB Gateway (10.10.1.254)*
`hashpasswd [pass]`	iSCB hashed password at `/etc/shadow`.
`ip [ip [unitid]]`	iSCB IP Address (10.10.1.11).*
`ipmb [on|off]`	The IPMB communication between iSCB and blade nodes. If it is off, entire health monitoring and MPM capability will not work. Default is on.*
`ipsrc [static|dhcp]`	IP source from dhcp or static. 'IDSW3-8 ON'overrides this setting.*
`loglevel [level]`	Event logging level: critical, warning and info.
`mac [mac address]`	MAC address.
`margin [low] [high]`	
`name [name]`	Subrack alias name, 14 chars maximum.*
`nm [nm [unitid]]`	iSCB Netmask (255.255.0.0).*
`ns [ns]`	iSCB NameServer IP address.*

`ntp [host]`	IP address of NTP server.*
`ntpserver [host]`	Hostname of NTP server. For example, time.nist.gov.*
`passwd [pass]`	set iSCB password.
`polling [seconds]`	polling interval.
`port [port]`	TCP port number of the CLI and Terminal Service. Default is 7000.*
`powerlimit [node shelf]`	The power usage limit of each blade and entire subrack. Default is 0(off). For example, if the blade power usage exceeds the power usage limit, iSCB will report a warning event for the blade.
`powermask [blade(s)]`	Define which blade(s) will not be affected by the power reset/off/cycle commands.
`reverse [on|off]`	The direction of slot numbering order. If it is on, the node’s slot number will be assigned from left to right. Default is off.
`sdrscan [on|off]`	?
`shelf [type [cid sid]]`	Subrack type and cabinet/subrack ID. Type is one of SR5110, SR6110, SR8104, SR8204, SR8202, SR10216. Cabinet-ID is 0..127. Subrack-ID is 0..7.*
`shelfname [name]`	Subrack alias name. 14 chars max.*
`shutdown [level]`	Event level of shutdown. Default is off.
`ssh [on|off]`	If it is off, the ssh service will be blocked. Default is on.*
`syslog [level [logserver [port]]`	Event level and logserver IP of remote syslog. Level is one of ‘critical’, ‘warning’ or ‘info’.*
`telnet [on|off]`	If it is off, the telnet login via TCP port 23 will be blocked. Default is on.*
`temp [warning critical]`	The temperature region of warning and critical. Default is 85’C and 90’C. For example, if the blade exceeds 85’C, the blade health status is set to warning.
`time [YYYYMMDDhhmmss]`	System time. Setting format is [[[[[YY]YY]MM]DD]hh]mm[.ss].
`trap [level [community [receiver]]]`	Event level, community and receiver IP of SNMP Trap notification. Default community is public.*
`tz [timezone]`	International time zone. Default is PST8PDT.*

* Run `nvram save` and `reboot` to apply changes to these parameters.

Example:

```
[r0s0i0 /]# iscb set ip
NVRAM  : 10.11.0.10
Current: 192.168.2.10
```

```
ok
[r0s0i0 /]# iscb set ip 10.10.1.11
ok
[r0s0i0 /]# iscb set ip
NVRAM  : 10.10.1.11
Current: 10.10.1.9
ok
[r0s0i0 /]# iscb nvram save
ok
[r0s0i0 /]# iscb reboot
ok
[$ /]# Connection closed by foreign host.
```

shutdown

Shutdown all or specified nodes

Syntax: `shutdown [node|all]`

Parameters:

`node`	Shutdown specified node, 1 through 16
`all`	Shutdown all nodes

status

Display node, PSU, and CFU or entire status

Syntax: `status [node|psu|fan]`

Parameters:

`node`	Display notice, info and higher logs
`psu`	Display err, crit, alert and emerg level event logs
`fan`	Display warning and higher level event logs

Example:

```
[r0s0i0 /]# iscb status
iSCB Status
   Node:  00  01  02  03  04  05  06  07  08  09  10  11  12  13  14  15 Total
  Power:   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
 ID-LED:
Console:
    BMC:   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
Temp 'C: -63 -61 -66 -61 -63 -60 -64 -62 -67 -68 -67   1 -63 -64   1 -65
Watt  W: 118 102 114 109 103 102 100 105  98  99 105 106 109 111 163 106  1750

    PSU:        00        01        02        03        04        05     Total
  Power:        on        on        on        on        on        on
 Status:        ok        ok        ok        ok        ok        ok
   Temp:      24'C      24'C      24'C      25'C      25'C      23'C
    Fan: 6176,5184 6176,5216 6144,5216 6176,5216 6112,5248 6144,5280
    12V:       21A       21A       21A       22A       21A       21A      127A
  AC In:      207V      208V      209V      207V      207V      207V
   Watt:      312W      312W      308W      306W      314W      304W     1856W

    CFU:        00        01        02        03        04        05
 Status:        ok        ok        ok        ok        ok        ok
   Fan1:   3465rpm   3497rpm   3356rpm   5389rpm   5471rpm   3501rpm
   Fan2:   3490rpm   3470rpm   3411rpm   6013rpm   6040rpm   3499rpm
   Duty:       60%       60%       60%      100%      100%       60%
ok
[r0s0i0 /]#
```

stop and start

Stop and start the iscb service

Syntax: `stop|start`

Parameters: None

Example:

```
[r0s0i0 /]# iscb stop
ok
[r0s0i0 /]# iscb ver
Can't connect iSCB service.
err: No such file or directory
[r0s0i0 /]# iscb start
ok
[r0s0i0 /]# iscb ver
iSCB Ver 1.1 (11:53:51 02/18/16 PST) passive (100h)
ok
[r0s0i0 /]#
```

tty

Display status of all established remote console sessions or kill specified connection

Syntax: `tty [kill channel_number]`

Parameters:

`kill channel_number`	Kill the specified Remote Console connection number.

Example:

```
[r0s0i0 /]# iscb tty
Port Mode   Device   Remote      Timeout     TCPin    TCPout     DEVin    DEVout
7001 telnet ttyUSB0  idle              0         0         0         0         0
7002 telnet ttyUSB1  idle              0         0         0         0         0
7003 telnet ttyUSB2  idle              0         0         0         0         0
7004 telnet ttyUSB3  idle              0         0         0         0         0
7005 telnet ttyUSB4  idle              0         0         0         0         0
7006 telnet ttyUSB5  idle              0         0         0         0         0
7007 telnet ttyUSB6  idle              0         0         0         0         0
7008 telnet ttyUSB7  idle              0         0         0         0         0
7009 telnet ttyUSB8  192.168.2.3       0         0       375       375         0
7010 telnet ttyUSB9  192.168.2.3       0         1     79688     79687         1
7011 telnet ttyUSB10 192.168.2.3       0         0       375       375         0
7012 telnet ttyUSB11 192.168.2.3       0         0     27391     27391         0
7013 telnet ttyUSB12 192.168.2.3       0         0       375       375         0
7014 telnet ttyUSB13 192.168.2.3       0         0       375       375         0
7015 telnet ttyUSB14 192.168.2.3       0         0       375       375         0
7016 telnet ttyUSB15 192.168.2.3       0         0       375       375         0
ok
[r0s0i0 /]#
```

updatefw

Update iSCB firmware. If a firmware image file is not provided, this command looks for firmware in the `/tmp` folder.

Syntax: `updatefw [firmware]`

Parameters:

`firmware`	Firmware image file

Example:

```
[r0s0i0 /]# scp athenaiscb-1.1.bin root@r0s0i0:/tmp
athenaiscb-1.0.bin                              100%   17MB   8.8MB/s
00:02
```

```
[r0s0i0 /]# ssh root@r0s0i0 iscb updatefw
ok
[r0s0i0 /]#
```

ver

Display firmware version and build date

Syntax: `ver`

Parameters: None

Example:

```
[r0s0i0 /]# iscb ver
iSCB Ver 1.1 (11:53:51 02/18/16 PST) active (0h)
ok
[r0s0i0 /]#
```

5 Resource Management

5.1 Manage Resources on Urika-GX

Mesos is used as the resource manager in the default service mode, whereas Kubernetes works as the resource manager in the secure services mode.

The resource management model of Mesos is different from traditional HPC schedulers. With traditional HPC schedulers, jobs request resources and the scheduler decides when to schedule and execute the job. Mesos on the other hand offers resources to frameworks that are registered with it. It is up to the framework scheduler to decide whether to accept or reject its resources. If framework rejects the offer, Mesos will continue to make new offers based on resource availability. Framework refers to implementation of different computing paradigms such as Spark, Hadoop, CGE etc.

For example, a user submits a spark job that requests 1000 cores to run. Spark registers as a framework with Mesos. Mesos offers its available resources to Spark. If Mesos offers 800 cores, Spark will either choose to accept the resources or reject it. If Spark accepts the resource offer, the job will be scheduled on the cluster. If it rejects the offer, Mesos will continue to make new offers.

Mesos Frameworks on Urika-GX

When users submit jobs to a Mesos cluster, frameworks register with Mesos. Mesos offers resources to registered frameworks. Frameworks can either choose to accept or reject the resource offer from Mesos. If the resource offer satisfies the resource requirements for a job, they accept the resources and schedule the jobs on the slaves. If the resource offer does not satisfy the resource requirements, frameworks can reject them. Frameworks will still be registered with Mesos. Mesos will update the resources it has at regular intervals (when an existing framework finishes running its job and releases the resources or when some other frameworks reject the resources) and continues to offer the resources to registered frameworks.

Each spark job is registered as a separate framework with Mesos. For each spark job that has been submitted, Mesos makes separate resource offers.

Marathon is registered as a single framework with Mesos. Marathon provides a mechanism to launch non-framework applications to run under Mesos. Marathon enables long-running services under Mesos such as databases, streaming engines etc. Cray has developed:

- the `mrun` command, which sets up resources for CGE and HPC jobs. For more information, see the `mrun` man page.
- scripts for setting up resources for YARN

These are submitted as applications to Marathon. Marathon negotiates for resources from Mesos and they get resources from Marathon.

Mesos tracks the frameworks that have registered with it. If jobs are submitted and there are no resources available, frameworks will not be dropped. Instead, frameworks will remain registered (unless manually killed by the user) and will continue to receive updated resource offers from Mesos at regular intervals. As an example, consider a scenario where there are four Spark jobs running and using all of the resources on the cluster. A user attempts to submit a job with framework Y and framework Y is waiting for resources. As each Spark job completes its execution, it releases the resources and Mesos updates its resource availability. Mesos will continue to give resource offers to the framework Y. Y can chose either to accept or reject resources. If Y decides to accept the resources, it will schedule its tasks. If Y rejects the resources, it will remain registered with Mesos and will continue to receive resource offers from Mesos.

Allocation of resources to Spark by Mesos

Let us say that the `spark-submit` command is executed with parameters `--total-executor-cores 100 --executor-memory 80G`. Each node has 32 cores. Mesos tries to use as few nodes as possible for the 100 cores requested. So in this case it will start Spark executors on 4 nodes (roundup(100 / 32)). Each executor has been requested to have 80G of memory. Default value for `spark.mesos.executor.memoryOverhead` is 10% so it allocates 88G to each executor. So in Mesos, it can been seen that `88 * 4 = 352 GB` allocated to the 4 Spark executors. For more information, see the latest Spark documentation at *http://spark.apache.org/docs*

Additional points to note:

- On Urika-GX, the Mesos cluster runs in High Availability mode, with 3 Mesos Masters and Marathon instances configured with Zookeeper.
- Unlike Marathon, Mesos does not offer any queue. Urika-GX scripts for flexing clusters and the `mrun` command do not submit their jobs unless they know the resource requirement is satisfied. Once the flex up request is successful, YARN uses its own queue for all the Hadoop workloads.
- Spark accepts resource offers with fewer resources than what it requested. For example, if a Spark job wants 1000 cores but only 800 cores are available, Mesos will offer those 800 to the Spark job. Spark will then choose to accept or reject the offer. If Spark accepts the offer, the job will be executed on the cluster. By default, Spark will accept any resource offer even if the number of resources in the offer is much less than the number of nodes the job requested. However, Spark users can control this behavior by specifying a minimum acceptable resource ratio; for example, they could require that Spark only accept offers of at least 90% of the requested cores. The configuration parameter that sets the ratio is `spark.scheduler.minRegisteredResourcesRatio`. It can be set on the command line with `--conf spark.scheduler.minRegisteredResourcesRatio=N` where `N` is between 0.0 and 1.0.
- `mrun` and flex scripts do not behave the way Spark behaves (as described in the previous bullet). `mrun` accepts two command-line options:
    - `--immediate=XXX` (default 30 seconds)
    - `--wait` (default `False`)

  When a user submits an `mrun` job, if more resources are needed than Mesos currently has available, the command will return immediately, showing system usage and how many nodes are available vs how many nodes are needed. If the user supplies the `--wait` option, this tells mrun to not return, but instead continue to poll Mesos until enough nodes are available. `mrun` will continue to poll Mesos for up to `--immediate` seconds before timing out. Finally, once Mesos informs `mrun` there are enough resources available; mrun will post the job to Marathon.

  When the requested resources are not available, flex scripts will display the current resources availability and exit.

- With `mrun`, the exact need must be met. If the user asks for 8 nodes, all CPU and memory on 8 nodes must be free for Marathon to accept the offer on behalf of `mrun`.
- The Marathon API does not offer a way to ask if the needs of a job can be fully satisfied before a request can be submitted. Therefore, Mesos is queried for its resource availability.
- Users request for resources from Mesos to give to YARN via Cray developed scripts for starting NodeManagers. The request is submitted to Marathon. This is called flex up. Once users get the requested resources, they can run their Hadoop jobs/ Hive queries / Oozie work-flows. Once they complete this, they release the resources back to Mesos via the Cray flex scripts. Flex scripts require the exact number of nodes to address requests and cannot run with fewer resources. When the number of resources requested in the flex up request does not match the current number of resources that are available with Mesos, an error message is displayed indicating that the number of resources available is less than the number of requested resources and that the user can submit a new flex up request.
- If the system is loaded and other frameworks (e.g. Spark) keep submitting smaller jobs, flex scripts may keep exiting if they do not receive the required number of nodes. This could lead to starvation of Hadoop jobs.

5.2 Use Apache Mesos on Urika-GX

Apache™ Mesos™ acts as the primary resource manager on the Urika-GX platform. It is a cluster manager that provides efficient resource isolation and sharing across distributed applications and/or frameworks. It lies between the application layer and the operating system and simplifies the process of managing applications in large-scale cluster environments, while optimizing resource utilization.

Architecture

Major components of a Mesos cluster include:

- **Mesos agents/slaves** - Agents/slaves are the worker instances of Mesos that denote resources of the cluster.
- **Mesos masters** - The master manages agent/slave daemons running on each cluster node and implements fine-grained sharing across frameworks using resource offers. Each resource offer is a list of free resources on multiple agents/slaves. The master decides how many resources to offer to each framework according to an organizational policy, such as fair sharing or priority.

  By default, Urika-GX ships with three Mesos masters with a quorum size of two. At least two Mesos masters must be running at any given time to ensure that the Mesos cluster is functioning properly. Administrators can use the `urika-state` and `urika-inventory` commands to check the status of Mesos masters and slaves. Administrators can also check the status of Mesos by pointing their browser at `http:`*`hostname-`*`login1:5050` and ensuring that it is up. In addition, executing the `ps -ef | grep mesos` command on the login nodes displays the running Mesos processes.

Components that Interact with Mesos

- **Frameworks** - Frameworks run tasks on agents/slaves. The Mesos Master offers resources to frameworks that are registered with it. Frameworks decide either to accept or reject the offer. If a framework accepts the offer, Mesos offers the resources and the framework scheduler then schedules the respective tasks on resources. Each framework running on Mesos consists of two components:

  - ○ A scheduler that is responsible for scheduling the tasks of a framework's job, within the accepted resources.
  - ○ An executor process that is launched on agent/slave nodes to run the framework's tasks.
- In addition to the aforementioned components, Urika-GX also supports Marathon and `mrun` (the Cray-developed application launcher) as ecosystem components of Mesos. `mrun` is built upon Marathon commands for ease of use and running data secure distributed applications. The `mrun` command sets up resources for CGE and HPC jobs.

  On Urika-GX, all tasks launched directly from Marathon need to be run as user `marathon`, and cannot be run as any other user ID. If a user tries to launch applications/tasks as non-Marathon user, the application will fail with error “`Not authorized to launch as` *`userID`*”. This behavior has no impact on Hadoop, Spark, `mrun` and/or CGE jobs.

Role of HAProxy

Requests received on the login nodes for the following services are proxied using HAProxy to the Urika-GX compute nodes:

- YARN Resource Manager
- HDFS NameNode
- Secondary HDFS NameNode
- Hadoop Application Timeline Server
- Hadoop Job History Server
- Spark History Server
- Oozie

For services like Mesos Masters and Marathon, while there are 3 instances running, one of them is the active leader. Requests received by the login node are proxied to the currently active leader. If a leader runs into issues, one of the backup leaders take over and the requests are proxied to the current leader.

HAProxy can be configured to provide SSL. Some possible solutions are documented in the security section of "*Urika®-GX System Administration Guide*".

Viewing Mesos Metrics from the CLI

Use the Cray-developed `urika-mesos_metrics` script to view Mesos related details. This script is located in the `/root/urika-tools/urika-cli` directory on the SMW and needs to be run as root.

Following is a sample output of the `urika-mesos_metrics` script:

```
# urika-mesos_metrics
 HTTP/1.1 200 OK
 Proceeding further...
******* MESOS CLUSTER METRICS **********
 Total cpus :  984
 Used cpus :  0
 Master elected :  1
******* MESOS FRAMEWORK METRICS **********
 Frameworks active :  1
 Frameworks connected :  1
 Frameworks disconnected:  0
 Frameworks inactive:  0
******* MESOS SLAVE METRICS **********
```

```
Slaves active :  41
Slaves connected :  41
Slaves disconnected:  0
Slaves inactive:  0
```

Troubleshooting information

- If the system indicates that the `mesos-slave` process is running, but it is not possible to schedule jobs, execute the following commands as root on each of the agent/slave node:

  ```
  # systemctl stop urika-mesos-slave
  # rm -vf /var/log/mesos/agent/meta/slaves/latest
  # systemctl start urika-mesos-slave
  ```
- If the Mesos UI displays orphaned Mesos tasks, refer to *Diagnose and Troubleshoot Orphaned Mesos Tasks* on page 253 to debug the issue.
- Mesos logs are located at `/var/log/mesos`, whereas log files of Mesos framework are located under `/var/log/mesos/agent/slaves/` on the node(s) the service runs on.

For more information about Mesos, visit *http://mesos.apache.org*.

5.2.1 Access the Apache Mesos Web UI

Prerequisites

- Before performing this procedure use the `urika-state` script to verify that the Mesos service is running.
- Check the service mode by executing `urika-service-mode` to ensure that the system is running in the service mode required for Mesos to run.

About this task

The Mesos web UI can be used to monitor components of the Mesos cluster, such as the Mesos slaves, aggregated resources and frameworks.

Do not launch applications through Mesos directly because all the frameworks are pre-configured on Urika-GX. Only a few frameworks (Spark and Marathon) are pre-configured to authenticate with Mesos.

Procedure

1. Access the Mesos UI using once of the following mechanisms:
   - Point a browse at the **Urika-GX Applications Interface** at: `http://`*`hostname`*`-login1` and then select the Mesos icon. This is the recommended approach.
   - Point a browser at `http://`*`hostname`*`-login1:5050` or `http://`*`hostname`*`-login2:5050`
2. Enter a username and the Mesos secret in the **Authentication Required** pop up's **UserName** and **Password** fields respectively.

   The Mesos secret can be retrieved from the `/security/secrets/`*`userName`*`.mesos` file, which is located on the SMW.

After logging on to the Mesos web UI, the users can view tasks in the summary page as well as resources reserved for that particular user. `crayadm` and `root` are global Mesos users that can view all the running frameworks and resource usage.

5.3 Use mrun to Retrieve Information About Marathon and Mesos Frameworks

Cray has developed the `mrun` command for launching applications. `mrun` enables running parallel jobs on Urika-GX using resources managed by Mesos/Marathon. In addition, this command enables viewing the currently active Mesos Frameworks and Marathon applications and enables specifying how `mrun` should redirect STDIN. It provides extensive details on running Marathon applications and also enables cancelling/stopping currently active Marathon applications.

The Cray Graph Engine (CGE) uses `mrun` to launch jobs under the Marathon framework on the Urika®-GX system.

CAUTION: The `mrun` command cannot be executed within a tenant VM or while the system is operating in the secure service mode. Both the `munge` and `ncmd` system services must be running for `mrun`/CGE to work. If either service is stopped or disabled, `mrun` will no longer be able to function

The `mrun` command needs to be executed from a login node. Some examples of using `mrun` are listed below:

Launch a job with `mrun`

```
$ mrun /bin/date
Wed Aug 10 13:31:51 CDT 2016
```

Display information about frameworks, applications and resources

Use the `--info` option of the `mrun` command to retrieve a quick snapshot view of Mesos frameworks, Marathon applications, and available compute resources.

```
$ mrun --info
Active Frameworks:
     IBM Spark Shell : Nodes[10] CPUs[ 240] : User[builder]
    Jupyter Notebook : Nodes[ 0] CPUs[   0] : User[urika-user]
            marathon : Nodes[20] CPUs[ 480] : User[root]
Active Marathon Jobs:
  /mrun/cge/user.dbport-2016-133-03-50-28.235572
                     : Nodes[20] CPUs[320/480] : user:user cmd:cge-
server
Available Resources:
                     : Nodes[14] CPUs[336] idle nid000[00-13]
                     : Nodes[30] CPUs[480] busy nid000[14-29,32-45]
                     : Nodes[ 2] CPUs[???] down nid000[30-31]
```

In the example output above, notice the `CPUs[320/480]` indicates that while the user only specified `mrun --ntasks-per-node=16 -N 20` (meaning the application is running on 320 CPUs), `mrun` intends **ALL** applications to have exclusive access to each node it is running on, and thus will ask Mesos for **ALL** the CPUs on the node, not just the number of CPUs per node the user requested to run on.

Retrieve a summary of running Marathon applications

Use the `--brief` option of the `mrun` command to obtain a more concise report on just the running Marathon applications and node availability.

```
$ mrun --brief
N:20 CPU:320/480  <user>  /mrun/cge/
user.dbport-2016-133-03-50-28.235572 cge-server -d /mn...
Status: Idle:14  Busy:30  Unavail:2
```

Retrieve information on a specific Marathon application

Use the `--detail` option of the `mrun` command to obtain additional information on a specific Marathon application. The application ID needs to be specified with this option.

```
$ mrun --detail /mrun/cge/user.dbport-2016-133-03-50-28.235572
Active Frameworks:
     IBM Spark Shell : Nodes[10] CPUs[ 240] : User[builder]
    Jupyter Notebook : Nodes[ 0] CPUs[   0] : User[urika-user]
            marathon : Nodes[20] CPUs[ 480] : User[root]
Active Marathon Jobs:
  /mrun/cge/user.dbport-2016-133-03-50-28.235572
                     : Nodes[20] CPUs[320/480] : user:<user> cmd:cge-
server
                     : [nid00032.urika.com]: startedAt:
2016-05-12T17:05:53.573Z
                     : [nid00028.urika.com]: startedAt:
2016-05-12T17:05:53.360Z
                     : [nid00010.urika.com]: startedAt:
2016-05-12T17:05:53.359Z
                     : [nid00007.urika.com]: startedAt:
2016-05-12T17:05:53.397Z
                     : [nid00001.urika.com]: startedAt:
2016-05-12T17:05:53.384Z
                     : [nid00019.urika.com]: startedAt:
2016-05-12T17:05:53.383Z
...
...
```

The additional information provided by the `--detail` option includes a list of all the node names the application is running on, and at what time the application was launched on those nodes.

Stop, cancel or abort a running Marathon application

Use the `--cancel` option of the `mrun` command to stop, cancel or abort a running Marathon application. The application ID needs to be specified with this option.

```
$ mrun --cancel /mrun/cge/user.3750-2016-133-20-01-07.394582
Thu May 12 2016 15:01:21.284876 CDT[][mrun]:INFO:App /mrun/cge/user.
3750-2016-133-20-01-07.394582 has been cancelled
```

If the application has already been cancelled, or completes, or does not exist, the following message is displayed:

```
$ mrun --cancel /mrun/cge/user.3750-2016-133-20-01-07.394582
App '/mrun/cge/user.3750-2016-133-20-01-07.394582' does not exist
```

CAUTION: The `root` user is allowed to use `mrun --cancel` to kill any Marathon-started job. All other users can only kill the Marathon jobs they launched using the mrun command. If a non-root user tries to use `mrun --cancel` to cancel any Marathon job that was not launched by that user using mrun, the system returns the following message:

```
mrun: error: Users may only cancel their own mrun jobs
```

Retrieve a list of nodes, CPU counts and available memory

- Use the `--resources` option of the `mrun` command to obtain a complete list of nodes, CPU counts, and available memory.

```
$ mrun --resources
 NID  HEX   NODENAME CPUs    MEM STAT
   0 0x00   nid00000   32 515758 idle
   1 0x01   nid00001   32 515758 busy
   2 0x02   nid00002   32 515758 idle
   3 0x03   nid00003   32 515758 busy
   4 0x04   nid00004   32 515758 busy
   5 0x05   nid00005   32 515758 idle
   6 0x06   nid00006   32 515758 idle
   7 0x07   nid00007   32 515758 busy
   8 0x08   nid00008   32 515758 busy
   9 0x09   nid00009   32 515758 busy
  10 0x0a   nid00010   32 515758 busy
  11 0x0b   nid00011   32 515758 busy
  12 0x0c   nid00012   32 515758 idle
  13 0x0d   nid00013   32 515758 idle
  14 0x0e   nid00014   32 515758 busy
  15 0x0f   nid00015   32 515758 busy
  16 0x10   nid00016   36 515756 idle
  17 0x11   nid00017   36 515756 idle
  18 0x12   nid00018   36 515756 busy
  19 0x13   nid00019   36 515756 busy
  20 0x14   nid00020   36 515756 idle
  21 0x15   nid00021   36 515756 idle
  22 0x16   nid00022   36 515756 busy
  23 0x17   nid00023   36 515756 idle
  24 0x18   nid00024   36 515756 idle
  25 0x19   nid00025   36 515756 busy
  26 0x1a   nid00026   36 515756 idle
  27 0x1b   nid00027   36 515756 busy
  28 0x1c   nid00028   36 515756 busy
  29 0x1d   nid00029   36 515756 idle
  30 0x1e   nid00030    0      0 unavail
  31 0x1f   nid00031    0      0 unavail
  32 0x20   nid00032   24 515758 busy
  33 0x21   nid00033   24 515758 idle
  34 0x22   nid00034   24 515758 idle
  35 0x23   nid00035   24 515758 idle
  36 0x24   nid00036   24 515758 idle
  37 0x25   nid00037   24 515758 idle
  38 0x26   nid00038   24 515758 busy
  39 0x27   nid00039   24 515758 idle
  40 0x28   nid00040   24 515758 idle
  41 0x29   nid00041   24 515758 idle
  42 0x2a   nid00042   24 515758 busy
  43 0x2b   nid00043   24 515758 busy
  44 0x2c   nid00044   24 515758 idle
  45 0x2d   nid00045   24 515758 idle
```

Node names that are marked as `unavail` are hidden from Mesos as available compute resources, such as the login node (`nid00030`). In the example above, some nodes have 24 CPUs/node, some have 32 CPUs/node and some have 36 CPUs/node. While not a typical situation, `mrun` does support this configuration, and a command such as `mrun -n 144 -N 4 ....` would in fact be allowed to proceed, and would be limited to using 4 nodes on `nid000[16-29]`, as they are the only ones with `(144/4 = 36)` CPUs per node.

Configuration Files

When `mrun` is invoked, it sets up some internal default values for required settings. `mrun` will then check if any system defaults have been configured in the `/etc/mrun/mrun.conf` file. An example `mrun.conf` file is shown below:

```
#
# (c) Copyright 2016 Cray Inc.  All Rights Reserved.
#
# Anything after an initial hashtag '#' is ignored
# Blank lines are ignored.
#
#NCMDServer=nid00000
#MesosServer=localhost      # same as --host
#MesosPort=5050
#MarathonServer=localhost
#MarathonPort=8080
#DebugFLAG=False            # same as --debug
#VerboseFLAG=False          # same as --verbose
#JobTimeout=0-0:10:0        # ten minutes, same as --time
#StartupTimeout=30          # 30 seconds,  same as --immediate
#HealthCheckEnabled=True    # Run with Marathon Health Checks enabled
#HCGracePeriodSeconds=5     # Seconds at startup to delay Health Checks
#HCIntervalSeconds=10       # Seconds between Health Check pings
#HCTimeoutSeconds=10        # Seconds to answer Health Check successfully
#HCMaxConsecutiveFailures=3 # How many missed health checks before app killed
```

If any of the lines above are not commented out, those values will become the new defaults for every `mrun` invocation.

Additionally, after the system `/etc/mrun/mrun.conf` file is loaded (if it exists), the user's private `$HOME/.mrun.conf` file will be loaded (if it exists). The following items should be noted:

- Any settings in the user's `$HOME/.mrun.conf` file will take precedence over any conflicting settings in the system `/etc/mrun/mrun.conf` file.
- Any command-line arguments that conflict with any pre-loaded configuration file settings will take precedence for the current invocation of `mrun`.

For more information, see the `mrun(1)` man page.

5.4 Launch an HPC Job Using mrun

Prerequisites

Use the `urika-state` command to ensure that both the Mesos and Marathon services are running and to check if the system is operating in the service mode that supports using `mrun`. For more information, see the `urika-state` man page and refer to *Urika-GX Service Modes* on page 173.

About this task

The `mrun` command can be used to launch HPC jobs on the Urika-GX system.

The `mrun` command cannot be executed within a tenant VM

Procedure

Launch the HPC job using the `mrun` command, specifying the number of nodes to allocate.

In the following example, *`my_hpc_app.exe`* is used as an example for the name of the application to run.

```
$ mrun -n 32 -N 8 my_hpc_app.exe arg1 arg2 arg3
```

Refer to the `mrun` man page for more information, further examples, environment variables, configuration files and command-line option descriptions of the `mrun` command.

5.5 Manage Long Running Services Using Marathon

Marathon is used by the Cray-developed command named `mrun` to allocate node resources from Mesos and launch application instances as needed. In addition, Cray-developed scripts for starting a cluster of YARN Node Managers are also launched through Marathon.

Before using Marathon, ensure that the system is running in the service mode that allows use of this service. Execute the `urika-state` or `urika-service-mode` commands to check the service mode. For more information, refer to the `urika-state` or `urika-service-mode` man pages and see *Urika-GX Service Modes* on page 173.

CAUTION: Unless it is required to shut down YARN nodes, analytic applications that use the Cray-developed scripts for flexing a cluster should not be deleted through the Marathon UI, as doing will lead to loss of YARN nodes.

On the Urika-GX system, there are always three Mesos Masters and three Marathon instances running, while one of them is the active leader. Requests received by the login node are proxied to the currently active leader. If a leader runs into issues, one of the backup leaders take over and the requests are proxied to the current leader.

Access the Marathon web UI by selecting **Marathon** on the **Urika-GX Applications Interface**, located at: `http://`*`hostname`*`-login1`. Though this is the recommended method of accessing Marathon, it can also be accessed at the port it runs on, i.e. at `http://`*`hostname`*`-login1:8080` or `http://`*`hostname`*`-login2:8080`

Figure 24. Urika-GX Applications Interface

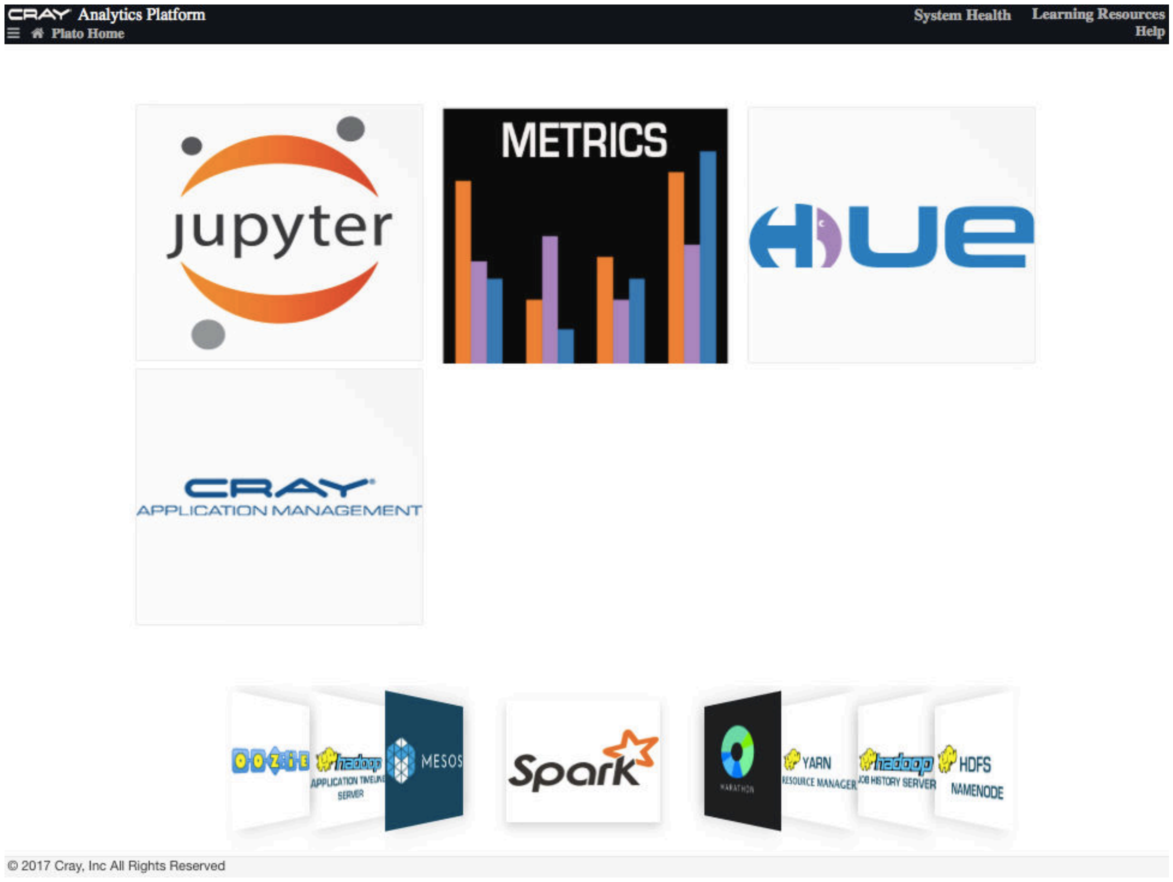
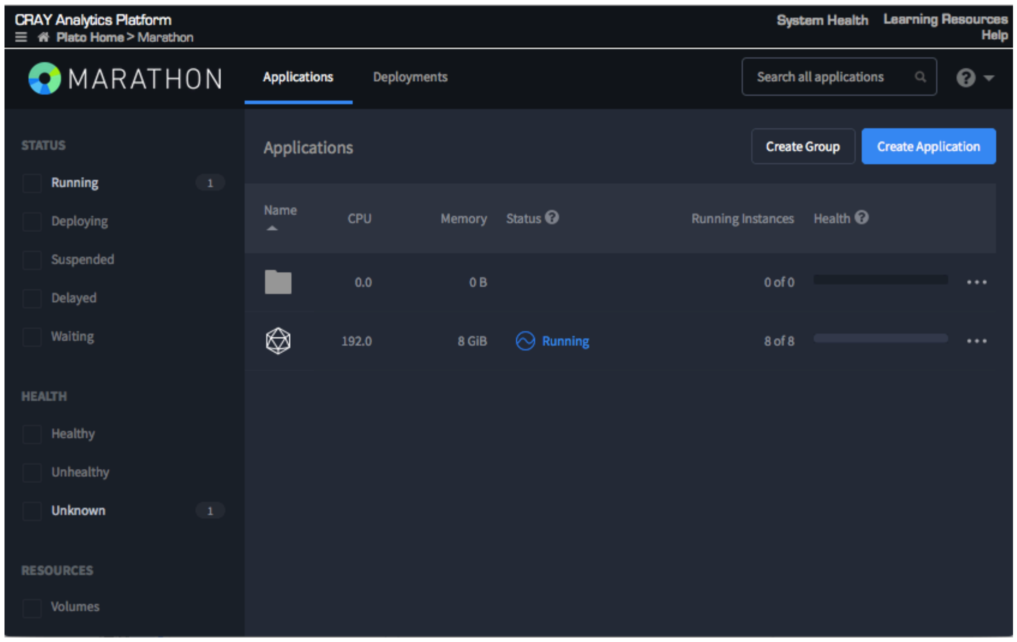


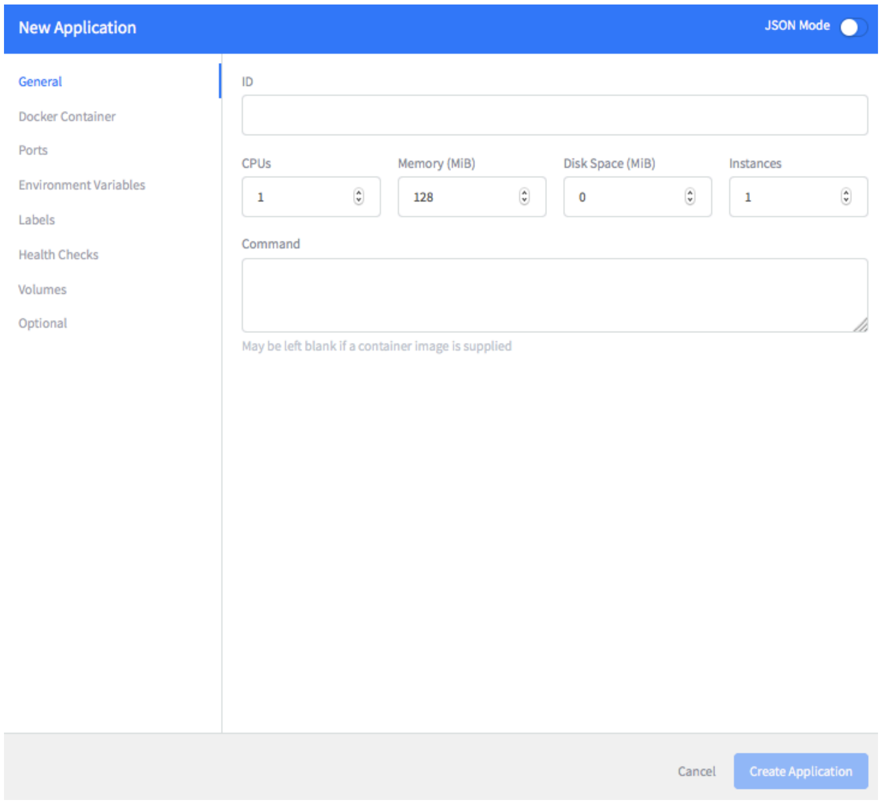
Figure 25. Marathon UI



Name	CPU	Memory	Status	Running Instances	Health
	0.0	0 B		0 of 0	
	192.0	8 GiB	Running	8 of 8	

Marathon also enables creating applications from the UI via the **Create Application** button, which displays the **New Application** pop up:

Figure 26. Create an Application Using the Marathon UI



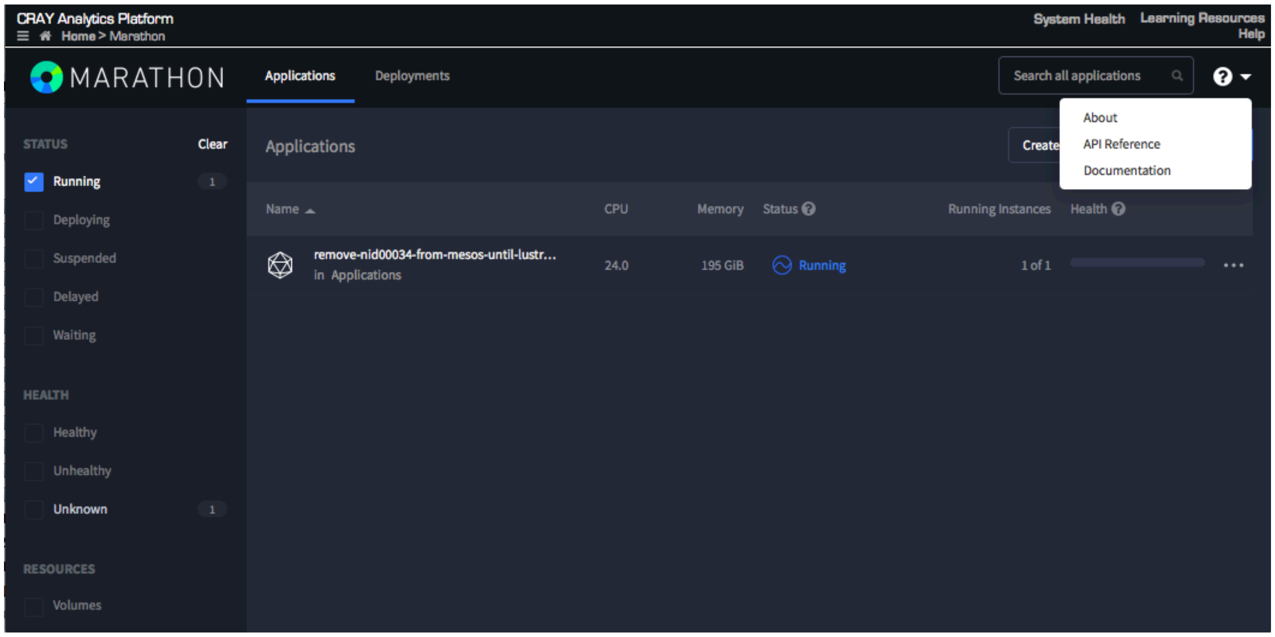
ID

CPUs	Memory (MiB)	Disk Space (MiB)	Instances
1	128	0	1

Command

May be left blank if a container image is supplied

Figure 27. Marathon UI



Name	CPU	Memory	Status	Running Instances	Health
remove-nid00034-from-mesos-until-lustr... in Applications	24.0	195 GiB	Running	1 of 1	

For additional information about using Marathon, select the help icon **(?)** and then select **Documentation**

5.6 Manage the Spark Thrift Server as a Non-Admin User

Prerequisites

- Ensure that the system is running in the service mode that allows use of the Spark Thrift Server. Execute the `urika-state` or `urika-service-mode` commands to check the service mode. For more information, refer to the `urika-state` or `urika-service-mode` man pages and see *Urika-GX Service Modes* on page 173.
- This procedure requires Tableau Desktop (version 10.0.1.0) and Simba Spark ODBC driver to be installed on the client machine:
- If using a MAC, the following procedure requires version 10.10 of the operating system.

About this task

Cray recommends to have the Spark Thrift to be started up by administrators, however, users can use the following instructions if they need to start up their own Spark Thrift server.

CAUTION: It is recommended for multiple users (admin and non-admin) to use the same Spark Thrift server (that has been started by an administrator) instead of spinning up their own individual servers, as doing so could result in resource lockdown. In addition, though it is possible for multiple users to connect to each other's Spark Thrift server, doing so can result in loss of connectivity if the server is brought down by the user who brings it up. If a user who starts up the Spark Thrift server brings it down, other users may experience loss of connection issues.

Procedure

1. Copy the `spark-env.sh`, `spark-defaults.conf` and `hive-site.xml` files to the local `$Home` directory.

   ```
   $ cp /opt/cray/spark2/default/conf/spark-env.sh $HOME
   $ cp /opt/cray/spark2/default/conf/spark-defaults.conf $HOME
   $ cp /opt/cray/spark2/default/conf/hive-site.xml $HOME
   ```

2. Modify the `hive-site.xml` configuration file to set `hive.server2.thrift.port` to a non-conflicting port.

3. Increase the number of compute nodes if needed by editing the `spark-defaults.conf` configuration file and changing the default value of `spark.cores.max` from 32 to the desired value.

4. Execute the `start-thriftserver.sh` script.

   ```
   $ /opt/cray/spark2/default/sbin/start-thriftserver.sh
   ```

5. Stop the Spark Thrift server when finished using it.

   ```
   $ /opt/cray/spark2/default/sbin/stop-thriftserver.sh
   ```

5.7 Manage Jobs Using the Cray Application Management UI

Prerequisites

Ensure that the system is running in the service mode that allows use of the Cray Application Management UI. Execute the `urika-state` or `urika-service-mode` commands to check the service mode. For more information, refer to the `urika-state` or `urika-service-mode` man pages and see *Urika-GX Service Modes* on page 173.

About this task

Cray® Application Management UI enables viewing, searching, filtering and performing actions on jobs submitted to the Urika-GX system.

Procedure

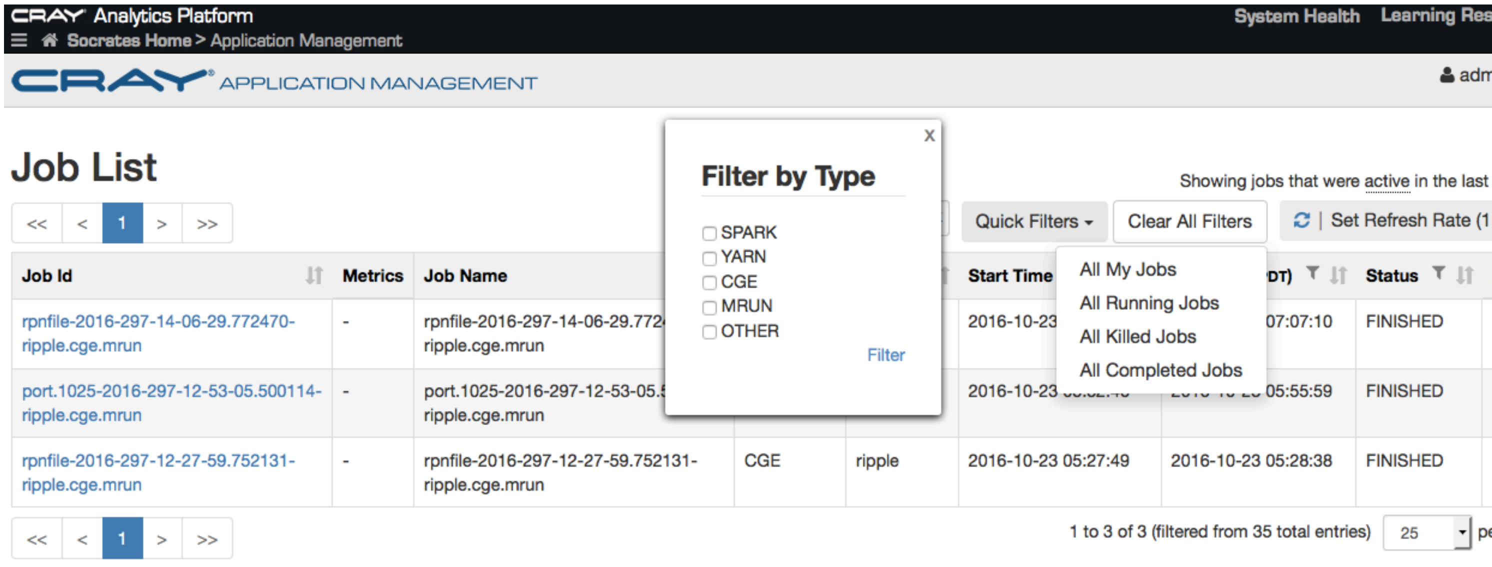
1. Point a browser at one of the following:
   - `http://`*`hostname`*`-login1` and then select the **Cray Application Management** icon.
   - `http://`*`hostname`*`-login1/applications`
2. Enter LDAP credentials on the login pop up. Alternatively, enter the default credentials (username: `admin`, password: `admin`) that the system ships with.

   Changing passwords from the UI is currently not supported for non-admin users. If an administrator has forgotten their password, they can change it via the `./manage` script. For more information, administrators should refer to S-3016, *Urika®-GX System Administration Guide*.

5.7.1 Overview of the Cray Application Management UI

The **Cray Application Management** UI is shown in the following figure:

Figure 28. ***Cray Application Management*** *UI*



Job Id	Metrics	Job Name			Start Time	DT)	Status
rpnfile-2016-297-14-06-29.772470-ripple.cge.mrun	-	rpnfile-2016-297-14-06-29.772 ripple.cge.mrun			2016-10-23	07:07:10	FINISHED
port.1025-2016-297-12-53-05.500114-ripple.cge.mrun	-	port.1025-2016-297-12-53-05.5 ripple.cge.mrun			2016-10-23	05:55:59	FINISHED
rpnfile-2016-297-12-27-59.752131-ripple.cge.mrun	-	rpnfile-2016-297-12-27-59.752131-ripple.cge.mrun	CGE	ripple	2016-10-23 05:27:49	2016-10-23 05:28:38	FINISHED

The **Search** field and **Quick Filters** drop down facilitate searching and filtering submitted jobs, based on the specified criteria. When these UI elements are used, the results are displayed in a table and the specified search/filter criteria is displayed at the top of these UI elements. A list of active jobs is displayed when no search/filtering criteria has been specified. By default, this shows jobs that were started in the last 24 hours, jobs that ended in the last 24 hours and jobs that are still running . Click on the text **active** to see a detailed description of functionality. If both the **Search** and **Quick Filters** UI elements are used at the same time, only jobs that match the selected quick filter will be displayed.

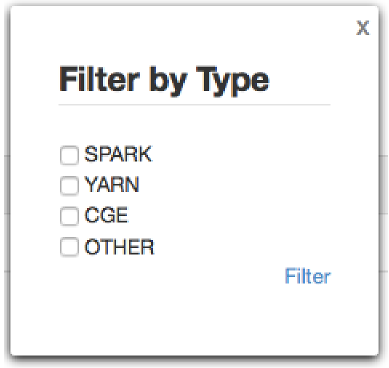
The table displayed on the UI contains information about submitted jobs and contains the following columns:

- **Job Id** - Displays the job ID for each submitted job. Selecting a job ID opens up another tab, which displays additional details of the job. For example, if a job ID for a Spark job is selected, a separate tab will be opened, displaying the Spark History Server.

  ATTENTION: Selecting a job ID for a job having "`OTHER`", "`MRUN`" or "`CGE`" as the job type will open up the Mesos UI in a separate tab.

- **Metrics** - Displays links that can be used for displaying the graphical representation of the job's metrics on the Grafana UI, which opens up in a separate browser tab.
- **Job Name -** Displays the name of the submitted job.
- **Type** - Displays types of all the submitted jobs. Jobs can be filtered based on type using the filter icon provided on the UI.

  Figure 29. Filtering by Type

- **User**- Displays the name of the user who submitted the job.
- **Start Time** - Displays the time the job started executing. Jobs can be filtered based on starting time using the filter icon provided on the UI.
- **End Time** - Displays the time the job finished executing. This column will be empty for jobs that have not finished executed yet.
- **Status** - Displays the current status of the job, which depends on the job's underlying framework.

  To filter a job based on its status, click the filter icon in this column's header.

  Make selections as needed and then select the **Filter** button on the pop-up. Click on the text **Failed** on the **Status** column to view logs for debugging failed jobs. For Spark jobs, this column contains a link titled **Finished**, which can be used to view and download logs that help identify whether or not the Spark job succeeded. Selecting this link will present a login screen, where users will need to enter their LDAP credentials.

  IMPORTANT: If the user logged in with the default user account (having `admin/admin` as the username and password), the system will require the user to log in again with their LDAP or system credentials to view Spark executor logs.

- **Action -** The kill button displayed in this column enables killing a running job. This column will be empty for jobs that have finished executing. Users can only kill jobs submitted by themselves. However, the system administrator can delete any job.

  NOTE: If the user logged in with the default user account (having `admin`/`admin` as the username and password), the system will require the user to log in again with their LDAP or system credentials to kill jobs of type "`CGE`" or "`MRUN`".

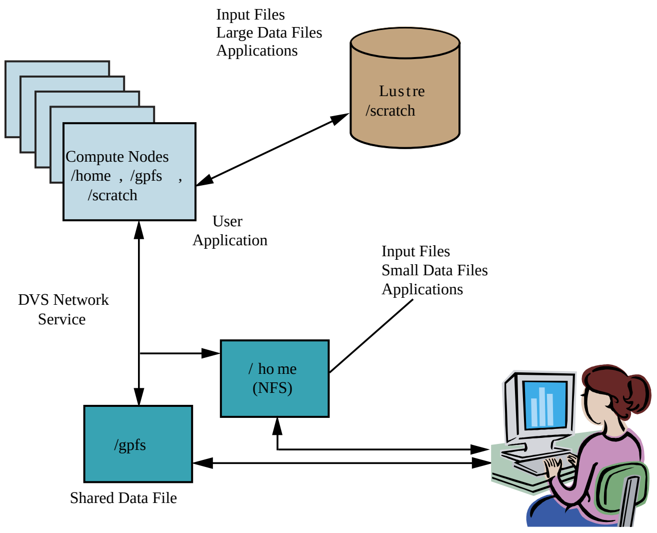
6 Cray DVS

6.1 Introduction to DVS

The Cray Data Virtualization Service (Cray DVS) is a distributed network service that provides transparent access to file systems residing on the service I/O nodes and remote servers in the data center. Cray DVS provides a service analogous to NFS™. It projects local file systems resident on I/O nodes or remote file servers to compute and service nodes within the Cray system. *Projecting* is simply the process of making a file system available on nodes where it does not physically reside. DVS-specific options to the `mount` command enable clients (compute nodes) to access a file system projected by DVS servers. Thus, Cray DVS, while not a file system, represents a software layer that provides scalable transport for file system services. See the `mount(8)` and `dvs(5)` man pages for more information.

Cray DVS uses the Linux-supplied VFS interface to process file system access operations. This allows DVS to project any POSIX-compliant file system.

Figure 30. Cray DVS Use Case

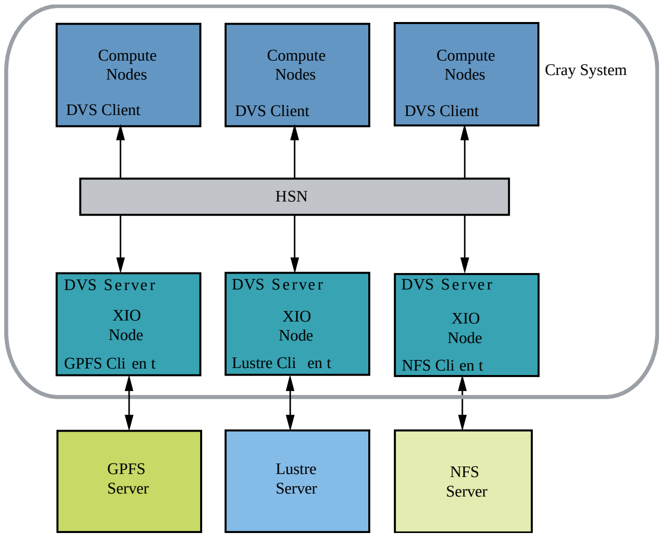


Cray DVS provides I/O performance and scalability to a large number of nodes, far beyond the typical number of clients supported by a single NFS server. Operating system noise and impact on compute node memory resources are both minimized in the Cray DVS configuration.

IMPORTANT: DVS servers use unlimited amounts of CPU and memory resources based directly on the I/O requests sent from DVS clients. For this reason, DVS servers should be dedicated and not share nodes with other services (Lustre nodes, login nodes, etc.).

Administration of Cray DVS is very similar to configuring and mounting any Linux file system. For more information, see the `dvs(5)` man page. Here is a system administrator's view of Cray DVS.

Figure 31. Cray DVS In a Cray System



6.1.1 Use Cray DVS on Urika-GX

Cray has successfully experimented with connecting Urika-GX to GPFS and NFS filesystems via Cray DVS. For more information and guidance, please contact Cray Support and visit *https://cray.my.salesforce.com/ka3330000008gb3*.

Major Features of DVS

- DVS supports failover and failback for parallel modes. The topic describes how it works and includes example console messages.
- DVS periodic sync promotes data and application resiliency and is more efficient than the DVS mount option `closesync`. The topic describes how it works and how it can be tuned.
- DVS statistics enable analysis of DVS performance on client and server nodes in CLE. There are many per-mount statistics available with this release.
- DVS can log requests sent to servers to aid in debugging. The topic shows an example log file and describes how to enable, disable, and reset request logging.
- DVS lists outstanding client requests, including the DVS server node and the amount of time the request has been waiting for a response.

6.1.2 DVS ioctl Interfaces

The following are provided for advanced users who require DVS `ioctl` interfaces. Most are correlates of environment variable and mount options with the same name.

Variable Name	Argument Type/Size	Purpose
`DVS_GET_FILE_ATOMIC` / `DVS_SET_FILE_ATOMIC`	signed 16-bit (must be 0 or 1 for SET)	Retrieves/sets the `atomic` option value for a file on a DVS mount.

Variable Name	**Argument Type/Size**	**Purpose**
`DVS_GET_FILE_BLK_SIZE` / `DVS_SET_FILE_BLK_SIZE`	signed 32-bit (must be > 0 for SET)	Retrieves/sets the DVS block size for a file on a DVS mount.
`DVS_GET_FILE_CACHE` / `DVS_SET_FILE_CACHE`	signed 16-bit (must be 0 or 1 for SET)	Retrieves/sets the cache option for a file on a DVS mount. To set the file cache, read-only option must be set.
`DVS_GET_FILE_CACHE_READ_SZ` / `DVS_SET_FILE_CACHE_READ_SZ`	signed 32-bit (must be > 0 for SET)	Retrieves/sets the `cache_read_sz` value for a file on a DVS mount.
`DVS_GET_FILE_CLOSESYNC` / `DVS_SET_FILE_CLOSESYNC`	signed 16-bit (must be 0 or 1 for SET)	Retrieves/sets the `closesync` option for a file on a DVS mount.
`DVS_GET_FILE_DATASYNC` / `DVS_SET_FILE_DATASYNC`	signed 16-bit (must be 0 or 1 for SET)	Retrieves/sets the current `datasync` value for a file on a DVS mount.
`DVS_GET_FILE_DEFEROPENS` / `DVS_SET_FILE_DEFEROPENS`	signed 16-bit (must be 0 or 1 for SET)	Retrieves/sets the `deferopens` value for a file on a DVS mount.
`DVS_GET_FILE_KILLPROCESS` / `DVS_SET_FILE_KILLPROCESS`	signed 16-bit (must be 0 or 1 for SET)	Retrieves/sets the `killprocess` option for a file on a DVS mount.
`DVS_GET_FILE_STRIPE_WIDTH` / `DVS_SET_FILE_STRIPE_WIDTH`	signed 32-bit (must be > 0 for SET)	Retrieves/sets the stripe width size for a file on a DVS mount. To set the stripe width, loadbalance option must be set.
`DVS_GET_NNODES`	signed 32-bit	Retrieves the number of nodes currently available for a mount point.
`DVS_GET_REMOTE_FS_MAGIC`	unsigned 64-bit	Gets the remote file system type for a file on a DVS mount.
`DVS_BCAST_IOCTL`	struct dvs_ioctl_tunnel	Used for DataWarp to allow specialized ioctl calls to be passed through DVS to a remote server.
`DVS_AUGMENTED_BCAST_IOCTL`	struct dvs_augmented_ioctl_tunnel	Used for DataWarp to allow specialized ioctl calls to be passed through DVS to a remote server.
`DVS_TUNNEL_IOCTL`	struct dvs_ioctl_tunnel	Used for DataWarp to allow specialized ioctl calls to be passed through DVS to a remote server.
`DVS_AUGMENTED_TUNNEL_IOCTL`	struct dvs_augmented_ioctl_tunnel	Used for DataWarp to allow specialized ioctl calls to be passed through DVS to a remote server.

6.1.3 DVS Client Mount Point Options

`atomic` / `noatomic`

`atomic` enables atomic stripe parallel mode. This ensures that stripe parallel requests adhere to POSIX read/write atomicity rules. DVS clients send each I/O request to a single DVS server to ensure that the bytes are not interleaved with other requests from DVS clients. The DVS server used to perform the read, write, or metadata operation is selected using an internal hash involving the underlying file or directory inode number and the offset of data into the file relative to the DVS block size.

`noatomic` disables atomic stripe parallel mode. If *`nodename`* or *`nodefile`* lists multiple DVS servers, and neither loadbalance nor cluster parallel mode is specified, DVS stripes I/O requests across multiple servers and does not necessarily adhere to POSIX read/write atomicity rules if file locking is not used.

- Default setting: `noatomic` or `0`
- Associated environment variable: *`DVS_ATOMIC`*
- Additional notes: none

`attrcache_timeout`

`attrcache_timeout` enables client-side attribute caching. File attributes and `dentries` for `getattr` requests, pathname lookups, etc. are read from DVS servers and cached on the DVS client for *n* seconds. Subsequent lookups or `getattr` requests use the cached attributes until the timeout expires, at which point they are read and cached again on first reference. Attribute caching can significantly increase performance, most notably in pathname lookup situations. When attribute caching is disabled, DVS clients must send a lookup request to a DVS server for every level of a pathname, and repeat this for every pathname operation. When it is enabled, it sends a lookup request to a DVS server for every level of a pathname once per *n* seconds.

- Default setting: `0`
- Associated environment variable: none
- Additional notes: Not recommended for read-write files systems.

`blksize=`*n*

`blksize=`*n* sets the DVS block size to *n* bytes.

- Default setting: `32768`
- Associated environment variable: *`DVS_BLOCKSIZE`*
- Additional notes: `blksize` is used in striping.

`cache` / `nocache`

`cache` enables client-side read caching. The client node performs caching of reads from the DVS server node and provides data to user applications from the page cache if possible, instead of performing a data transfer from the DVS server node.

`nocache` disables client-side read caching.

- Default setting: `nocache` or `0`

- Associated environment variable: *`DVS_CACHE`*
- Additional notes: Cray DVS is not a clustered file system; no coherency is maintained among multiple DVS client nodes reading and writing to the same file. If cache is enabled and data consistency is required, applications must take care to synchronize their accesses to the shared file.

`cache_read_sz`

`cache_read_sz` is a limit that can be specified to prevent reads over this size from being cached in the Linux page cache.

- Default setting: `0`
- Associated environment variable: *`DVS_CACHE_READ_SZ`*
- Additional notes: none

`closesync` / `noclosesync`

`closesync` enables data synchronization on last close of a file. When a process performs the final close of a file descriptor, in addition to forwarding the close to the DVS server, the DVS server node waits until data has been written to the underlying media before indicating that the close has completed. Because DVS does not cache data on client nodes and has no replay capabilities, this ensures that data is not lost if a server node crashes after an application has exited.

`noclosesync` is the default behavior of DVS. In the default case, DVS returns a `close()` request immediately.

- Default setting: `noclosesync` or `0`
- Associated environment variable: *`DVS_CLOSESYNC`*
- Additional notes: When DVS periodic sync is enabled, it is redundant to use `closesync`. Moreover, periodic sync is more efficient because it tracks which files are "dirty."

`datasync` / `nodatasync`

`datasync` enables data synchronization. The DVS server node waits until data has been written to the underlying media before indicating that the write has completed.

`nodatasync` disables data synchronization. The DVS server node returns from a write request as soon as the user's data has been written into the page cache on the server node.

- Default setting: `nodatasync` or `0`
- Associated environment variable: *`DVS_DATASYNC`*
- Additional notes: `datasync` can significantly impact performance.

`deferopens`

`deferopens` defers DVS client open requests to DVS servers for a given set of conditions. When a file is open in stripe parallel mode or atomic stripe parallel mode, DVS clients send the open request to a single DVS server only. Additional open requests are sent as necessary when the DVS client performs a read or write to a different server for the first time. The `deferopens` option deviates from POSIX specifications. For example, if a file was removed after the initial open succeeded but before deferred opens were initiated by a read or write operation to a new server, the read or write operation would fail with `errno` set to `ENOENT` (because the open was unable to open the file).

`nodeferopens` disables the deferral of DVS client open requests to DVS servers. When a file is open in stripe parallel mode or atomic stripe parallel mode, DVS clients send open requests to all DVS servers denoted by nodename or nodefile.

- Default setting: `nodeferopens` or `0`
- Associated environment variable: *`DVS_DEFEROPENS`*
- Additional notes: none

`distribute_create_ops` **/** **`nodistribute_create_ops`**

`distribute_create_ops` caused DVS to change its hashing algorithm so that create and lookup requests are distributed across all of the servers, as opposed to being distribute to a single server. This applies to creates, mkdirs, lookups, mknods, links, and symlinks.

`nodistribute_create_ops` is the default, and DVS uses its normal algorithm of using just one target server.

- Default setting: `nodistribute_create_ops` or `0`
- Associated environment variable: none
- Additional notes: none

`dwfs` **/** **`nodwfs`**

`dwfs` causes DVS to change its behavior to support a DataWarp file system.

`nodwfs` is the default, where DVS does not support a DataWarp file system.

- Default setting: `nodwfs` or `off`.
- Associated environment variable: none
- Additional notes: none

`failover` **/** **`nofailover`**

`failover` enables failover and failback of DVS servers. If multiple DVS servers are listed for a single DVS mount point and one or more of the servers fails, operations for that mount point continue by using the subset of servers still available. When the downed servers are rebooted and start DVS, any client mount points that had performed failover operations failback to once again include the servers as valid nodes for I/O forwarding operations.

`nofailover` disables failover and failback of DVS servers. If one or more servers for a given mount point fail, operations for that mount point behave as described by the corresponding `retry` or `noretry` option specified for the mount point.

- Default setting: `failover` or `1`
- Associated environment variable: none
- Additional notes: The failover option cannot be specified at the same time as the `noretry` option. If all servers fail, operations for the mount point behave as described by the `retry` option until at least one server is rebooted and has loaded DVS.

`hash`	The `hash` option has three possible values: **`fnv-1a`** `hash=fnv-1a` offers the best overall performance with very little variation due to differing numbers of servers. **`jenkins`** `hash=jenkins` is the hash that DVS previously used. It is included in the unlikely case of end-case pathological issues with the `fnv-1a` hash, but has worse overall performance. **`modulo`** `hash=modulo` does not do any hash at all, but rather takes the modulo of the seed that it is given. This option can potentially have high load balancing characteristics, but is extremely vulnerable to pathological cases such as file systems that only allocate even numbered inodes or a prime number of servers. ● Default setting: `fnv-1a` ● Associated environment variable: none ● Additional notes: Except in cases of extremely advanced administrators or specific advice from DVS developers, do not use the `hash` mount option. The best course of action is to let DVS use its default value.
`hash_on_nid`	With `hash_on_nid` set to `on`, DVS uses the nid of the client as the hash seed instead of using the file inode number. This effectively causes all request traffic for the compute node to go to a single server. This can help metadata operation performance by avoiding lock thrashing in the underlying file system when each process on a set of DVS clients is using a separate file. ● Default setting: `off` ● Associated environment variable: none ● Additional notes: Setting `hash_on_nid=on` also sets `hash=modulo` by default. `hash_on_nid` is forced `off` when the loadbalance option is used.
`killprocess` / **`nokillprocess`**	`killprocess` enables killing processes that have one or more file descriptors with data that has not yet been written to the backing store. DVS provides this option to attempt to ensure that a process is not affected by silent data loss, such as when data still resides in the kernel or file system page cache on the DVS server after a write has completed. `nokillprocess` disables the killing of processes that have written data to a DVS server when a server fails. When a server fails, processes that have written data to the server are not killed. If a process continues to perform operations with an open file descriptor that had been used to write data to the server, the operations fail (with `errno` set to `EHOSTDOWN`). A new open of the file is allowed, and subsequent operations with the corresponding file descriptor function normally. ● Default setting: `killprocess` or `1` ● Associated environment variable: *`DVS_KILLPROCESS`*

- Additional notes: If DVS periodic sync is enabled, DVS servers attempt to `fsync` dirty files to minimize the number of processes that are killed. DVS periodic sync will also `fsync` a file's data when the file is closed. While it is highly unlikely, if DVS periodic sync is not enabled, DVS cannot fully guarantee prevention of silent data loss with this option alone because a `close()` does not guarantee data has been transferred to the underlying media (see the `closesync` option).

`loadbalance` / `noloadbalance`

For a description of loadbalance mode, see *About DVS Loadbalance Mode* on page 148.

`noloadbalance` automatically sets the following mount options: `maxnodes = 1`, `cache = 1`, and `hash_on_nid = 0`.

- Default setting: `noloadbalance` or `0`
- Associated environment variable: none
- Additional notes: none

`magic`

`magic` defines what the expected file system magic value for the projected file system on the DVS servers should be. When a DVS client attempts to mount the file system from a server, it verifies that the underlying file system has a magic value that matches the specified value. If not, the DVS client excludes that DVS server from the list of servers it uses for the mount point and prints a message to the system console. Once the configuration issue on the DVS server has been addressed and the client mounts the correct file system, DVS can be restarted on the server. All clients subsequently verify that the server is configured correctly and include the server for that mount point. Many file system magic values are defined in the `/usr/include/linux/magic.h` file. Commonly used magic values on Cray systems are:

NFS	0x6969
GPFS	0x47504653
Lustre servers	0x0bd00bd1
Lustre clients	0x0bd00bd0

- Default setting: the underlying file system's magic value
- Associated environment variable: none
- Additional notes: none

`maxnodes`

`maxnodes` is used in configuring DVS modes. See *About DVS Modes* on page 145.

- Default setting: number of nodes available `(nnodes)`
- Associated environment variable: *`DVS_MAXNODES`*
- Additional notes: none

`mds`

`mds` specifies which DVS server meta-data operations are sent to.

- Default setting: required (e.g., `server1`)
- Associated environment variable: none
- Additional notes: Only used for DataWarp file systems.

`nodefile`	*`nodefile`* is equivalent to *`nodename`* but allows the administrator to specify a list of server nodes in a file instead of placing them on the mount line directly. This is more convenient for large sites that use many DVS server nodes. Node names are separated by a new line or a colon (:) character and no spaces. ● Default setting: required (unless *`nodename`* is used) ● Associated environment variable: none ● Additional notes: none
`nodename`	*`nodename`* is equivalent to *`nodefile`* but the administrator specifies a list of server nodes on the mount line directly. Node names are separated by a colon (:) character and no spaces. ● Default setting: required (unless *`nodefile`* is used) ● Associated environment variable: none ● Additional notes: none
`path`	`path` specifies the path of the directory on the DVS server that is to be projected. ● Default setting: required ● Associated environment variable: none ● Additional notes: none
`retry` / `noretry`	`retry` enables the retry option, which affects how a DVS client node behaves in the event of a DVS server node going down. If `retry` is specified, any user I/O request is retried until it succeeds, receives an error other than a "node down" indication, or receives a signal to interrupt the I/O operation. `noretry` disables retries of user I/O requests when the DVS server receiving the request is down. ● Default setting: `retry` or `1` ● Associated environment variable: none ● Additional notes: none
`ro_cache` / `no_ro_cache`	`ro_cache` enables read-only caching for files on writable mount points. Files opened with read-only permissions in `ro_cache` mode are treated as if they are on a DVS read-only cached mount point. If the file has any concurrent open that has write permissions, all instances of that file revert to the default `no_ro_cache` mode for the current and subsequent reads. `no_ro_cache` disables read-only caching for files on writable mount points. ● Default setting: `no_ro_cache` or `0`

- Associated environment variable: none
- Additional notes: none

`userenv` / `nouserenv`	`userenv` argument specifies that DVS must honor end user environment variable overrides for DVS mount options. `nouserenv` argument allows the administrator to block end user environment variable overrides for DVS mount options. ● Default setting: `userenv` or `1` ● Associated environment variable: none ● Additional notes: none

6.1.4 DVS Environment Variables

By default, user environment variables allow client override of options specified in the `/etc/fstab` entry and are evaluated a file is opened by DVS. However, if the `nouserenv` option is specified in the DVS entry, then user environment variables are disabled.

The following environment variables are for use in the default case:

Variable Name	Options	Purpose
`DVS_ATOMIC`	`on|off`	Overrides the `atomic` or `noatomic` mount options.
`DVS_BLOCKSIZE`	*n*	A nonzero number, *n* overrides the `blksize` mount option.
`DVS_CACHE`	`on|off`	Overrides the `cache` or `nocache` mount options. This option is available only for read-only mount points and is ignored for writeable mount points.
`DVS_CACHE_READ_SZ`	*n*	A positive integer, *n* overrides the `cache_read_sz` mount option.
`DVS_CLOSESYNC`	`on|off`	Overrides the `closesync` or `noclosesync` mount options.
`DVS_DATASYNC`	`on|off`	Overrides the `datasync` or `nodatasync` mount options. **NOTE:** Setting `DVS_DATASYNC` to on can slow down an application considerably. Consider using periodic sync.
`DVS_DEFEROPENS`	`on|off`	Overrides the `deferopens` or `nodeferopens` mount options.
`DVS_KILLPROCESS`	`on|off`	Overrides the `killprocess` or `nokillprocess` mount options.
`DVS_MAXNODES`	*n*	A nonzero number, *n* overrides the `maxnodes` mount option. The specified value of `maxnodes` must be greater than zero and less than or equal to the number of server nodes specified on the mount, otherwise the variable has no effect.

6.1.5 Modes

6.1.5.1 About DVS Modes

There are two primary ways to use Cray DVS: in serial mode or parallel mode, as indicated in the following table. In serial mode, one DVS server on a Cray service node projects a file system to multiple compute node clients. In parallel mode, multiple DVS servers—in configurations that vary in purpose, layout, and performance—project a file system to multiple compute node clients.

Table 15. Cray DVS Access Modes

Mode	Access Level	Pattern
Serial	Read/Write	Many clients, one server
Parallel	Read/Write	Many clients, many servers

DVS mode is not selected by a switch, but rather by configuration. It is determined by the system administrator's choice of DVS mount point options. A DVS mode is really just the name given to a collection of mount options chosen to achieve a particular goal. Users cannot choose among DVS modes unless the system administrator has configured the system to make more than one mode available. A system administrator can make several DVS modes available on the same compute node by mounting a file system with different mount point options on different mount points on that compute node. This table shows the rationale and an example mount entry for each DVS mode.

Mode	Rationale	Example Configuration Settings

Serial	Simplest implementation of DVS. Only option if no cluster/shared file system available.	`servers: r0s0c1n1`
		`options: maxnodes=1`

Cluster Parallel	Often used for a large file system, must be a shared file system such as GPFS (Spectrum Scale). Can distribute file I/O and metadata operations among several servers to avoid overloading any one server and to speed up operations. I/O for a single file goes only to the chosen server.	`servers: r0s1c1n1, r0s2c1n2, r0s2c0n1`
		`options: maxnodes=1`

Stripe Parallel	Used to distribute file I/O load at the granularity of a block of data within a file. Adds another level of parallelism to better distribute the load. I/O for a single file may go to multiple servers.	`servers: r0s1c1n1, r0s2c1n2, r0s2c0n1`
		`options: maxnodes=3`

Atomic Stripe Parallel	Used when stripe parallel makes sense and POSIX read/write atomicity required.	`servers: r0s1c1n1, r0s2c1n2, r0s2c0n1`
		`options: maxnodes=3,atomic`

Loadbalance	Used for near-optimal load distribution when a read-only file system is being used. By default, enables `readonly` and sets `cache=1`, `failover=1`, `maxnodes=1`, and `hash_on_nid=0`.	`servers: r0s1c1n1, r0s2c1n2, r0s2c0n1`
		`loadbalance: true`

Serial, cluster parallel, and atomic stripe parallel modes all adhere to POSIX read/write atomicity rules, but stripe parallel mode does not. POSIX read/write atomicity guarantees that all bytes associated with a read or write are not interleaved with bytes from other read or write operations.

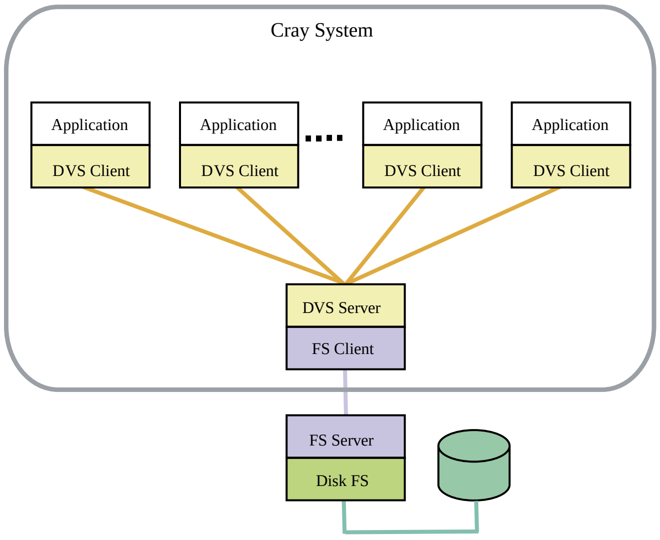
6.1.5.2 About DVS Serial Mode

Serial mode is the simplest implementation of DVS, where each file system is projected from a single DVS server to multiple clients (see figure). DVS can project multiple file systems in serial mode by assigning a new or existing DVS server to each additional file system in serial access mode and entering the appropriate mount point on the clients. The following example mount entry contains the mount options essential for serial mode: a single `nodename` and `maxnodes=1`.

```
mount -o nodename=server1, maxnodes=1
```

DVS serial mode adheres to POSIX read/write atomicity rules.

Figure 32. Cray DVS Serial Access Mode



6.1.5.3 About DVS Cluster Parallel Mode

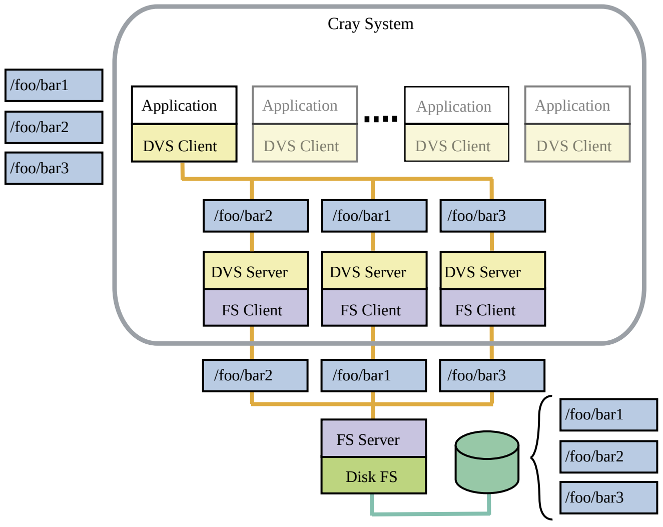
In cluster parallel access mode, each client interacts with multiple servers. For example, in the figure below, DVS is mounted to `/foo` on the DVS client, and three different files—`bar1`, `bar2`, and `bar3`—are handled by three different DVS servers, thus distributing the load. The server used to perform the read, write, or metadata operation is selected using an internal hash involving the underlying file or directory inode number. Once a server has been selected for a file, it looks similar to serial mode: all I/O from all clients involving that file routes to that server to prevent file system coherency thrash.

The following example mount entry contains the mount options essential for cluster parallel mode: more than one `nodename` and `maxnodes=1`.

```
mount -o nodename=server1:server2:server3, maxnodes=1
```

DVS cluster parallel mode adheres to POSIX read/write atomicity rules.

Figure 33. Cray DVS Cluster Parallel Access Mode



6.1.5.4 About DVS Loadbalance Mode

Loadbalance mode is a client access mode for DVS used to more evenly distribute loads across servers. The clients, Cray system compute nodes, automatically select the server based on a DVS-internal node ID (NID) from the list of available server nodes specified on the `/etc/fstab` line. Loadbalance mode is valid only for read-only mount points. Loadbalance mode automatically enables failover to another DVS server.

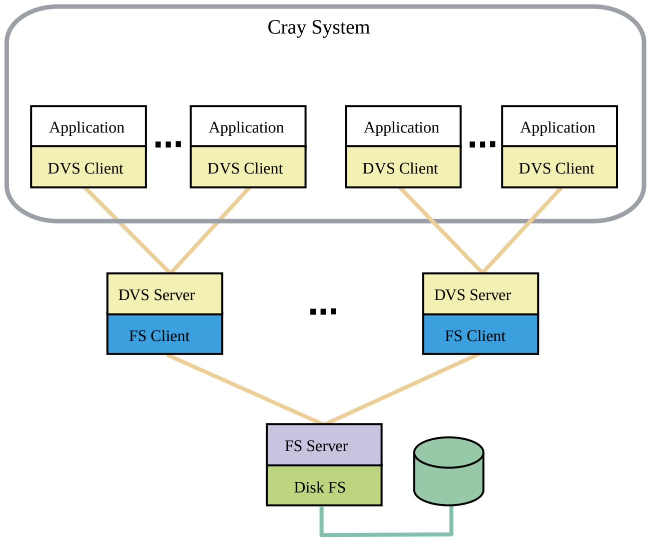
The following example mount entry contains the mount options essential for loadbalance mode: more than one `nodename`, `maxnodes=1`, and the `loadbalance` option.

```
mount -o nodename=server1:server2:server3, maxnodes=1, loadbalance
```

DVS automatically enables the `cache` mount option in loadbalance mode because it is a read-only mode. This means that a DVS client pulls data from the DVS server the first time it is referenced, but then the data is stored in the client's page cache. While the application is running, all future references to that data are local to the client's memory, and DVS will not be involved at all. However, if the node runs low on memory, the Linux kernel may remove these pages, and then the client must fetch the data from the DVS server on the next reference to repopulate the client's page cache.

To enable attribute caching as well, use the `attrcache_timeout` mount option for `loadbalance` mount points. This allows attribute-only file system operations to use local attribute data instead of sending the request to the DVS server. This is useful in loadbalance mode because with a read-only file system, attributes are not likely to change.

Figure 34. Cray DVS Loadbalance Mode



6.1.5.5 About DVS Stripe Parallel Mode

Stripe parallel mode builds upon cluster parallel mode to provide an extra level of parallelized I/O forwarding for clustered file systems. Each DVS server can serve all files, and DVS servers are automatically chosen based on the file inode and offsets of data within the file relative to the DVS block size value. For example, in the figure below, DVS is mounted to `/foo` on the DVS client, and the I/O for three different blocks (or segments) of data within file `bar`—`seg1`, `seg2`, and `seg3`—is handled by three different DVS servers, thus distributing the load at a more granular level than that achieved by cluster parallel mode. Note that while file I/O is distributed at the block level, file metadata operations are distributed as in cluster parallel mode: the metadata operations of a given file are always handled by the same DVS server. Stripe parallel mode provides the opportunity for greater aggregate I/O bandwidth when forwarding I/O from a coherent cluster file system. All I/O from all clients involving the same file routes each block of file data to the same server to prevent file system coherency thrash. GPFS has been tested extensively using this mode.

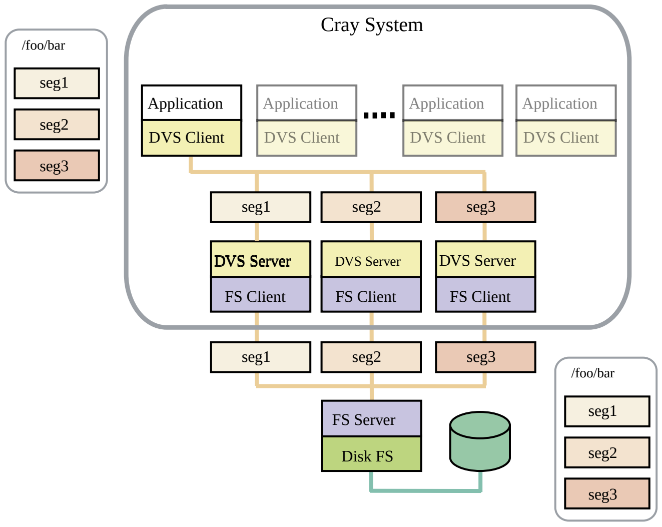
ATTENTION: NFS cannot be used in stripe parallel mode because NFS implements close-to-open cache consistency; therefore striping data across the NFS clients could compromise data integrity.

The following example mount entry contains the mount options essential for stripe parallel mode: more than one `nodename` and `maxnodes` > 1.

```
mount -o nodename=server1:server2:server3, maxnodes=3
```

DVS stripe parallel mode does not adhere to POSIX read/write atomicity rules. See *About DVS Atomic Stripe Parallel Mode* for information about how to achieve POSIX read/write atomicity with this mode.

Figure 35. Cray DVS Stripe Parallel Mode



6.1.5.6 About DVS Atomic Stripe Parallel Mode

Stripe parallel mode provides parallelism within a file at the granularity of the DVS block size. However, when applications do not use their own file locking, stripe parallel mode cannot guarantee POSIX read/write atomicity. In contrast, *atomic* stripe parallel mode adheres to POSIX read/write atomicity rules while still allowing for possible parallelism within a file. It is similar to stripe parallel mode in that the server used to perform the read, write, or metadata operation is selected using an internal hash involving the underlying file or directory inode number, and the offset of data into the file is relative to the DVS block size. However, once that server is selected, the entire read or write request is handled by that server only. This ensures that all I/O requests are atomic while allowing DVS clients to access different servers for subsequent I/O requests if they have different starting offsets within the file.

The following example mount entry contains the mount options essential for atomic stripe parallel mode: more than one `nodename`, `maxnodes` > 1, and the `atomic` option.

```
mount -o nodename=server1:server2:server3, maxnodes=3, atomic
```

Users can request atomic stripe parallel mode by setting the `DVS_ATOMIC` user environment variable to `on`.

6.1.6 Resiliency and Diagnostics

6.1.6.1 About DVS Failover

DVS clients use resiliency communication agent (RCA) events to determine when server nodes have failed or when DVS has been unloaded from a server node and when server nodes have been booted and DVS is reloaded. This ensures that all clients are informed of server failures and reboots in the same manner at the same time, which reduces the underlying file system coherency traffic associated with rerouting I/O operations away from downed servers and back to rebooted servers.

Cray DVS supports failover and failback for parallel modes:

- For cluster, stripe, and atomic stripe parallel modes, add the `failover` option to the mount line or `/etc/fstab` entry to specify failover and failback.
- For loadbalance mode, failover and failback are specified by default.

DVS failover and failback are done in an active-active manner. Multiple servers must be specified in the `/etc/fstab` entry for failover and failback to function. When a server fails, it is taken out of the list of servers to use for the mount point until it is rebooted. All open and new files use the remaining servers as described by the cluster, stripe, and atomic stripe parallel sections. Files not using the failed server are not affected.

When failover occurs:

- If all servers fail, I/O is retried as described by the `retry` option (see *DVS Client Mount Point Options* on page 138).
- Any mount point using loadbalance mode automatically recalibrates the existing client-to-server routes to ensure that the clients are evenly distributed across the remaining servers. When failback occurs, this process is repeated.
- Any mount point using cluster parallel mode automatically redirects I/O to one of the remaining DVS servers for any file that previously routed to the now-down server. When failback occurs, files are rerouted to their original server.
- Any mount point using stripe parallel mode or atomic stripe parallel mode automatically restripes I/O across the remaining DVS servers in an even manner. When failback occurs, files are restriped to their original pattern.

Client System Console Message: "DVS: file_node_down: removing from list of available servers for 2 mount points"

The following message indicates that a DVS server has failed.

```
DVS: file_node_down: removing r0s1c1n3 from list of available
servers for 2 mount points
```

In this example, `r0s1c1n3` is the DVS server that has failed and has been removed from the list of available mount points provided in the `/etc/fstab` entry for the DVS projection.

After the issue is resolved, the following message is printed to the console log of each client of the projection:

```
DVS: file_node_up: adding r0s1c1n3 back to list of available servers
for 2 mount points
```

6.1.6.2 About DVS Periodic Sync

DVS periodic sync improves data resiliency and facilitates a degree of application resiliency so that applications may continue executing in the event of a stalled file system or DVS server failure. Without periodic sync, such an event would result in DVS clients killing any processes with open files that were written through the failed server. Any data written through that server that was only in the server's page cache and not written to disk would be lost, and processes using the file would see data corruption.

Periodic sync works by periodically performing `fsync` on individual files with dirty pages on the DVS servers, to ensure those files are written to disk. For each file, the DVS client tracks when a DVS server performs a file sync and when processes on DVS clients write to it, and then notifies the DVS server when `fsync` is needed.

DVS periodic sync makes the DVS `closesync` mount option redundant. Periodic sync is more efficient than `closesync` because it is aware of which files may have dirty pages.

Use the following three `/proc` files to tune DVS periodic sync behavior (`echo` desired value into each file):

`/proc/fs/dvs/sync_num_threads`	Specifies the number of threads on the DVS server that perform sync operations. There must be at least 1 thread and the maximum number is 32. The default value is 8.
`/proc/fs/dvs/sync_dirty_timeout_secs`	On DVS *servers*, specifies the number of seconds that must have passed since the file was written before DVS syncs it. The default value is 300. The objective is to reduce unnecessary sync operations for files actively being updated. Decreasing this number increases the likelihood that the file is in use when it is synced. Increasing this number increases the likelihood that processes are killed during a server failure. On DVS *clients*, specifies the number of seconds that must have passed since the file was written before DVS asks the server for an updated sync time. The default value is 300. Decreasing this number increases the number of DVS requests being sent. Increasing this number increases the likelihood that processes are killed during a server failure.
`/proc/fs/dvs/sync_period_secs`	On DVS *servers*, specifies the number of seconds before the `sync_num_threads` syncs files on the DVS server (if necessary). The default value is 300. On DVS *clients*, specifies the number of seconds between checks for dirty files that need to request the last sync time from the server. The default value is 600.

A fourth `/proc` file, `/procsfs/dvs/sync_stats`, collects statistics of the syncing behavior. This setting is not tunable (read only).

6.1.6.3 About DVS Statistics

The `/proc/fs/dvs/stats` file contains statistics for system operations that cannot be correlated to a specific DVS mount point, and is thus most interesting on DVS servers.

The same type of information is also available on DVS clients at the mount point (`/proc/fs/dvs/mounts/0/stats`, `/proc/fs/dvs/mounts/1/stats`, etc.) Each of these files contains the statistics for that specific mount point only. More information about each of these mount points can be obtained by viewing the mount file that resides in the same directory (e.g., `/proc/fs/dvs/mounts/0/mount`).

- The first section of the file contains remote procedure call (RPC) message-passing statistics. Each line displays a file system operation, followed by counts of successful and failed send operations of that type, counts of successful and failed receive operations of that type, the time it took to process the most recent operation of that type, and the maximum time to process that operation in the history of the system.
- The second section of the file contains statistics on virtual file system (VFS) operations. Each line displays a callback name, followed by counts of successful and failed callbacks of that name, the time it took to process the most recent callback of that name, and the maximum time to process that callback in the history of the system.
- The third section contains fields for the smallest and largest request sizes (in bytes) for read and write operations, and the number of successful and failed interprocess communication (IPC) requests, IPC asynchronous requests, and IPC replies.

In addition, the `/proc/fs/dvs/ipc/stats` file displays DVS IPC statistics such as bytes transferred and received, NAK counts, and so forth. It also displays message counts by type and size.

DVS statistics are enabled and collected by default.

- To disable a DVS statistics file, write a zero into the file:

  `echo 0 > /proc/fs/dvs/stats`
- To re-enable a DVS statistics file, write a 1 into the file:

  `echo 1 > /proc/fs/dvs/stats`
- To reset the statistics values to zero, write a 2 into the file:

  `echo 2 > /proc/fs/dvs/stats`

6.1.6.4 About DVS Client Requests

DVS provides a list of outstanding requests on client nodes in `/proc/fs/dvs/ipc/requests`, which lists the DVS server node, the request, the DVS file system path, `uid`, time that the request has been waiting for a response, and the associated `apid`. If the request is from a process that was not spawned through `aprun`, the request `apid` is 0. An example output of the file looks like:

```
% cat /proc/fs/dvs/ipc/requests
server: r0s1c1n3 request: RQ_LOOKUP path: /dsl/ufs/home user: 12795 time: 0.000 sec
apid: 3871
```

The file appears on DVS servers but returns an error when a user tries to access it.

6.1.7 Caveats

6.1.7.1 Caveat: Client Consistency

DVS supports close-to-open consistency, which means that files on client and server are consistent at `open()` and `close()`. However, while a file is open, DVS does not guarantee that the file on the client and the file on the server are consistent.

6.1.7.2 Caveat: DVS blksize Must Match or be a Multiple of GPFS Block Size

When projecting a general parallel file system (GPFS), the client mount option `blksize` must match or be a multiple of the GPFS blocksize. When projecting multiple GPFS file systems that have different block sizes, DVS must have a different `/etc/fstab` entry for each file system.

In the case of two GPFS file systems, one with a 64 kilobyte (KB) block size, and another with a 1024KB block size, the `/etc/fstab` entries for DVS would look like the following:

```
/gpfs1 /dvs1 dvs path=/dvs1,nodefile=/etc/nidlist1,blksize=65536
/gpfs2 /dvs2 dvs path=/dvs2,nodefile=/etc/nidlist2,blksize=1048576
```

6.1.7.3 Caveat: Expanded File System Support

Setting up and mounting target file systems on Cray service nodes is the sole responsibility of the customer or an agent of the customer. Cray Custom Engineering is available to provide a tailored file system solution. Please contact a Cray service representative for more information.

6.1.7.4 Caveat: flock() Not Supported

DVS does not support `flock()` system calls and will return an error. DVS will set `errno` to `ENOTSUPP` when a `flock()` call is attempted for a DVS-projected file system.

DVS supports file locking with `fcntl()`.

For more information, see the `fcntl(2)` man page.

6.1.7.5 Caveat: mmap() Support

DVS currently supports only read-only access when the `MAP_SHARED` flag is specified to the memory map function `mmap()`. This is because writable shared mmaps are cached in the Linux kernel page cache, and DVS does not maintain consistency between DVS clients and DVS servers. Therefore the file data cached on the DVS client may not be kept coherent with other accesses to the file data from other nodes. This may cause DVS to set `errno` to `ENOSYS`. Write access is expected to be available in the next release.

Write access is available when the `MAP_PRIVATE` flag is specified to `mmap()` because the file data is private to the process that performed the `mmap()` and therefore coherency with other processes is not a problem.

6.1.8 Administrative Tasks

6.1.8.1 Disable DVS Fairness of Service

About this task

DVS creates user- or job-specific request queues for clients. Originally, DVS used one queue to handle requests in a FIFO (first-in, first out) fashion. This meant that since all clients shared one queue, a demanding job could tax the server disproportionately and the other clients would have to wait until the demanding client's request(s) completed. Fairness of Service creates a *list* of queues—one queue for each client and/or job. The list of queues is processed in a circular fashion. When a message thread is available, it fetches the first queue on the list, moves that queue to the end of the list, and processes the first message in that queue. This helps to distribute the workload and potentially helps contending applications perform better.

Fairness of Service is enabled by default. This procedure describes how to disable Fairness of Service.

Procedure

1. Stop DVS on all servers.

   NOTE: This would typically be done only during a maintenance interval, because this is a drastic action.

2. Edit `/etc/modprobe.d/dvs.conf` file on all the DVS server NIDs configured in the cluster to enter the following line :

```
options dvsipc_single_msg_queue=1
```

3. Restart DVS on all servers.

6.1.8.2 Force a Cache Revalidation on a DVS Mount Point

About this task

Mounting DVS with the attrcache_timeout=[time_in_seconds] option can improve performance because cached file attributes on a DVS client preclude the need to make some requests of the DVS server. Unfortunately, if the file attributes of the DVS mounted file system changed, the only way to revalidate the cache was to wait the entire timeout, which is often as long as four hours. Users can force a cache revalidation at any time, not just when the timeout has expired, by using the following procedure.

Procedure

1. Force a cache revalidation on a DVS client using one of the following methods:
   - To revalidate all cached attributes on a single DVS mount point, echo 1 into that mount point's `drop_caches proc` file. The following example uses the second mount point on the client and uses the `cat` command first to confirm that is the desired mount point. To specify a different mount point, replace the 2 with `0-`*n*.

     ```
     cat /proc/fs/dvs/mounts/2/mount
     echo 1 > /proc/fs/dvs/mounts/2/drop_caches
     ```
   - To revalidate all attributes across all DVS mounts, echo 1 into the universal DVS `drop_caches proc` file. For example:

     ```
     echo 1 > /proc/fs/dvs/drop_caches
     ```
2. (Optional) Clear out other caches on the DVS client to ensure that all data is revalidated.

   ```
   echo 3 > /proc/sys/vm/drop_caches
   ```
3. (Optional) Clear out other caches on the DVS *server* to ensure that all data is revalidated.

   If an NFS file system is the underlying file system, it is also likely that the same procedure will be required on the DVS servers to allow all of the changes on the NFS file system to properly propagate out to the DVS clients. This has to do with NFS caching behavior.

   ```
   echo 3 > /proc/sys/vm/drop_caches
   ```

6.1.8.3 Project XFS over DVS

About this task

The following procedure is used to project an XFS file system over DVS to compute and service node clients in Cray systems.

Procedure

1. Edit the `/etc/exports` file on the DVS server node and add the appropriate export:

```
/xfs/scratch *(rw,no_root_squash,no_subtree_check)
```

2. Find the attached XFS devices.

```
nid00006# ls -l /dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a000004*
lrwxrwxrwx 1 root root 10 May 3 10:29 /dev/disk/by-id/dm-uuid-
mpath-360080e50002ff41a0000040b517a2609 \
-> ../../dm-3
lrwxrwxrwx 1 root root 10 May 3 10:29 /dev/disk/by-id/dm-uuid-
mpath-360080e50002ff41a0000040e517a261e \
-> ../../dm-4
lrwxrwxrwx 1 root root 10 May 3 10:30 /dev/disk/by-id/dm-uuid-
mpath-360080e50002ff41a000004a851837c9b \
-> ../../dm-5
nid00006# fdisk -l /dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a000004*
Disk /dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a0000040b517a2609: 7999.4 GB,
7999443697664 bytes
255 heads, 63 sectors/track, 972543 cylinders, total 15623913472 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
Disk /dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a0000040b517a2609 doesn't
contain a valid \
partition table
Disk /dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a0000040e517a261e: 7999.4 GB,
7999443697664 bytes
255 heads, 63 sectors/track, 972543 cylinders, total 15623913472 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
Disk /dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a0000040e517a261e doesn't
contain a valid \
partition table
Disk /dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a000004a851837c9b: 1798.8 GB,
1798765019136 bytes
255 heads, 63 sectors/track, 218687 cylinders, total 3513212928 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
Disk /dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a000004a851837c9b doesn't
contain a valid partition table
```

3. Make a writable `/etc/lvm`.

```
nid00006# mkdir /tmp/lvm
nid00006# cp -rp /etc/lvm/* /tmp/lvm/
nid00006# cp /tmp/lvm/lvm.conf /tmp/lvm/lvm.tmp
nid00006# mv /tmp/lvm/lvm.tmp /tmp/lvm/lvm.conf
nid00006# mount -o bind /tmp/lvm /etc/lvm
```

4. Edit the `/etc/lvm/lvm.conf` file on the DVS server node.

a. Edit the `/etc/lvm/lvm.conf` file.

```
node/6# vi -n 6 /etc/lvm/lvm.conf
```

b. Add the following custom filter to the configuration file:

   filter = ["a|/dev/disk/by-id/dm-uuid-.*mpath-.*|", "r/.*/"]

c. Remove any previous filters from the configuration file.

5. Edit the `/etc/sysconfig/lvm` file to append the following string.

```
LVM_ACTIVATED_ON_DISCOVERED="enable" #LVM_VGS_ACTIVATED_ON_BOOT="`grep '/
dev/' /etc/fstab | sed 's/\t.*//g' | sed 's,/*[^/]\+/*$,,' | sed '$!N; /^\(.*\)\n
\1$/!P; D'`" LVM_VGS_ACTIVATED_ON_BOOT="fs1"
```

6. Initialize LVM physical disks and verify intitialization.

For this configuration, all the disks in `/dev/disk/by-id/` are for the LVM volumes.

a. Initialize LVM physical disks (while logged on to `nid00006`).

```
nid00006# pvcreate /dev/disk/by-id/dm-uuid-
mpath-360080e50002ff41a0000040b517a2609
Physical volume "/dev/disk/by-id/dm-uuid-
mpath-360080e50002ff41a0000040b517a2609" successfully created
nid00006# pvcreate /dev/disk/by-id/dm-uuid-
mpath-360080e50002ff41a0000040e517a261e
Physical volume "/dev/disk/by-id/dm-uuid-
mpath-360080e50002ff41a0000040e517a261e" successfully
created
```

b. Display the volumes to verify that the disks have been initialized.

```
nid00006# pvdisplay
"/dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a0000040b517a2609" is a new
physical \
volume of "7.28 TiB"
--- NEW Physical volume ---
PV Name /dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a0000040b517a2609
VG Name
PV Size 7.28 TiB
Allocatable NO
PE Size 0
Total PE 0
Free PE 0
Allocated PE 0
PV UUID Llb2m3-3AZC-LuWT-ttfH-2iB2-GH3J-oas5W1
"/dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a0000040e517a261e" is a new
physical \
volume of "7.28 TiB"
--- NEW Physical volume ---
PV Name /dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a0000040e517a261e
VG Name
PV Size 7.28 TiB
Allocatable NO
PE Size 0
Total PE 0
Free PE 0
Allocated PE 0
PV UUID PRCPkx-3Tan-IHgr-0ZLh-iTGt-XWzY-pBkUpM
```

7. Create the volume group. Keep the physical volumes in the same volume group.

```
nid00006# vgcreate fs1 /dev/disk/by-id/dm-uuid-
mpath-360080e50002ff41a0000040b517a2609
/dev/disk/by-id/dm-uuid-mpath-360080e50002ff41a0000040e517a261e
Volume group "fs1" successfully created
```

8. Create logical volumes.

```
nid00006# lvcreate --stripes 2 --stripesize 512k --extents 100%FREE --name fs1 fs1
Logical volume "fs1" created
```

9. Create the XFS file system using `mkfs`.

```
nid0006# mkfs -t xfs /dev/fs1/fs1
```

10. Mount the XFS file system and verify its availability.

    a. Mount the XFS file system to the service node.

    ```
    nid0006# mount -t xfs /dev/fs1/fs1 /mnt
    ```

    b. Display the mount to verify that the file system is available.

    ```
    nid00006# df -h --type xfs
    Filesystem Size Used Avail Use% Mounted on
    /dev/mapper/fs1-fs1 15T 34M 15T 1% /mnt
    ```

11. Edit the `/etc/fstab` file on the DVS server node to add the following:

```
/dev/mapper/fs1-fs1 /scratch xfs defaults 1 0
```

If there is a secondary XFS/DVS node used as a high-availability manual backup and failover device for the primary XFS/DVS node, do not add the mount definition entry to that node configuration. XFS is neither a clustered nor a shared file system and can be natively mounted on only one node at any given time. If the file system is mounted on more than one node, there is a great risk of data corruption.

6.1.8.4 Configure DVS Using `modeprobe.d`

About this task

Most DVS parameters are typically changed by adding lines to a `modprobe.d` configuration file or echoing values to a `/proc` file. Changes to a `modprobe.d` file are made prior to booting the affected nodes, and the changes take effect at boot.

The `dvs.conf` file is one of many files that are generated automatically and controlled by the Cray Configuration Management Framework (CMF). Such files can be identified by the warning statement in the file header.

In the following example, DVS request logging is enabled on a node with a node ID of `nid00023`.

Procedure

1. Create and change to the directory structure that is to be replicated on the target node.

```
smw# mkdir -p etc/modprobe.d
smw# cd etc/modprobe.d
```

2. Create and change to the directory structure that is to be replicated on the target node.

```
smw# ssh nid00023
smw# mkdir -p etc/modprobe.d/
smw# cd etc/modprobe.d/
```

3. Create or edit the DVS configuration file and add the options line(s) for the parameter to be set/changed.

For this example, the lines are from the `dvs_request_log_enabled` entry of the DVS kernel module parameters list below. Then comment/uncomment the appropriate lines, depending on which action is to be taken.

```
# Disable DVS request log
options dvsproc dvs_request_log_enabled=0

# Enable DVS request log
#options dvsproc dvs_request_log_enabled=1
```

6.1.8.5 Change Kernel Module Parameters Dynamically Using Proc Files

Some of the kernel module parameters in the following list can be changed dynamically by echoing values to `/proc` files on the appropriate nodes. Those that can be changed using that method are indicated in the list, including the name of the `/proc` file and the values to use. Note that such changes do not persist.

List of DVS Kernel Module Parameters

dvs_request_log_enabled Logs each DVS request sent to servers.

- Default value: 0 (disabled)
- To view read-only:
  cat `/sys/module/dvsproc/parameters/dvs_request_log_enabled`
- To change prior to boot, add these lines to
  `<`*`nid_of_choice`*`>/etc/modprobe.d/dvs-local.conf`:

  ```
  # Disable DVS request log
  options dvsproc dvs_request_log_enabled=0

  # Enable DVS request log
  #options dvsproc dvs_request_log_enabled=1
  ```

- To change dynamically:

  ```
  hostname# echo 0 > /proc/fs/dvs/request_log
  hostname# echo 1 > /proc/fs/dvs/request_log
  hostname# echo 2 > /proc/fs/dvs/request_log
  ```

  The value "2" resets the log.

dvs_request_log_size_kb Size (KB) of the request log buffer.

- Default value: 16384 KB (16384 * 1024 bytes)
- To view read-only:
  cat `/sys/module/dvsproc/parameters/dvs_request_log_size_kb`

- To change prior to boot, add these lines to `<nid_of_choice>/etc/modprobe.d/dvs-local.conf`:

  ```
  # Set size (in kb) of the request log buffer
  options dvsproc dvs_request_log_size_kb=17000
  ```

- To change dynamically:

  ```
  hostname# echo 17000 > /proc/fs/dvs/request_log_size_kb
  ```

dvs_instance_info

Contains the following fields, which are parameters for the DVS thread pool in the DVS IPC layer. Most of these fields can be changed through other module parameters (e.g., dvsipc_msg_thread_limit and dvsipc_single_msg_queue); however, this module parameter has priority over the individual ones and if set, will override them.

Table 16. dvs_instance_info Field Definitions

Field	Definition
thread_min	Number of threads created at startup.
thread_max	Maximum number of persistent threads.
thread_limit	Maximum number of valid threads that DVS IPC allows to exist at one time. DVS IPC will dynamically scale up the number of threads to this number as the load increases.
thread_concurrent_creates	Maximum number of IPC threads that can be in the process of forking a new thread.
thread_nice	Nice value for the threads.
single_msg_queue	Disables/enables fairness of service. Setting to 1 disables fairness of service by processing incoming requests with a FIFO method. Setting to 0 (default) groups requests into different queues (qhdrs) based on apid, and round-robbins the queues to maintain quality of service (QOS) among jobs.
init_free_qhdrs	Low water mark for the pool of unused qhdrs. When the pool falls below this number, more are allocated. Used only if single_msg_queue = 0.
max_free_qhdrs	Maximum number of unused qhdrs that can exist before the system starts freeing them. Used only if single_msg_queue = 0.
Interactions among fields: thread_min <= thread_max <= thread_limit thread_concurrent_creates <= thread_limit	

Field	Definition
init_free_qhdrs and max_free_qhdrs used only when single_msg_queue == 0	

IMPORTANT: For this parameter to be valid, values must be specified for all of the fields. To avoid unintentional changes, be careful when changing any field values.

- Default value: see modprobe.d examples
- To view read-only:
  cat `/sys/module/dvsipc/parameters/dvs_instance_info`
- To change prior to boot, add these lines to <*nid_of_choice*>`/etc/modprobe.d/dvs-local.conf` (comment out either the compute or service line, depending on the type of node(s) being configured):

```
# Set parameters for DVS thread pool in DVS IPC layer
# Defaults for compute nodes
options dvsipc dvs_instance_info=4,16,1000,0,0,0,1,1

# Defaults for service nodes
options dvsipc
dvs_instance_info=16,64,1000,0,0,0,64,2048
```

  For compute nodes, this translates to dvs_instance_info = thread_min = 4, thread_max = 16, thread_limit = 1000, thread_concurrent_creates = 0, thread_nice = 0, single_msg_queue = 0, init_free_qhdrs = 1, max_free_qhdrs = 1.

  For service nodes, this translates to dvs_instance_info = thread_min = 16, thread_max = 64, thread_limit = 1000, thread_concurrent_creates = 0, nice = 0, single_msg_queue = 0, init_free_qhdrs = 64, max_free_qhdrs = 2048.

  Note that if using the defaults for a compute node, ensure that dvsipc_config_type=0 is also set, and likewise, for a service node, ensure that dvsipc_config_type=1 for consistency.
- To change dynamically: N/A

dvsipc_config_type

Forces DVS to load in a mode optimized for DVS clients (0) or servers (1). This parameter can be used to make DVS load in a non-default manner. Frequently used for repurposed compute nodes.

- Default value: 0 for compute nodes, 1 for service nodes
- To view read-only:
  cat `/sys/module/dvsipc/parameters/dvsipc_config_type`
- To change prior to boot, add these lines to <*nid_of_choice*>`/etc/modprobe.d/dvs-local.conf` (comment out

either the client or server line, depending on the type of node(s) being configured):

```
# Load DVS as a client
options dvsipc dvsipc_config_type=0

# Load DVS as a server
options dvsipc dvsipc_config_type=1
```

- To change dynamically:

```
hostname# echo 0 > /proc/fs/dvs/ipc/config-type
hostname# echo 1 > /proc/fs/dvs/ipc/config-type
```

dvsipc_single_msg_queue Used to disable fairness of service, which is enabled by default. Setting this parameter to 1 disables fairness of service by forcing DVS to use a single message queue instead of a list of queues.

- Default value: 0 (fairness of service enabled)
- To view read-only: cat `/sys/module/dvsipc/parameters/dvsipc_single_msg_queue`
- To change prior to boot, use the single_msg_queue field of the dvs_instance_info parameter. If no other fields in dvs_instance_info need to be changed, it may be easier to change the dvsipc_single_msg_queue parameter directly by adding these lines to <*nid_of_choice*>`/etc/modprobe.d/dvs-local.conf`:

```
# Disable fairness of service
options dvsipc dvsipc_single_msg_queue=1

# Enable fairness of service
#options dvsipc dvsipc_single_msg_queue=0
```

- To change dynamically: N/A

dvsof_concurrent_reads Controls how many threads are allowed into the read path on the server. A value of -1 disables, 0 uses the number of cores on the CPU, and any other positive number sets the number of threads. Set to 0 for best DataWarp performance.

- Default value: -1
- To view read-only: cat `/sys/module/dvs/parameters/dvsof_concurrent_reads`
- To change prior to boot, add these lines to <*nid_of_choice*>`/etc/modprobe.d/dvs-local.conf`:

```
# Disable concurrent reads
#options dvs dvsof_concurrent_reads=-1

# Set number of threads able to do concurrent
# reads = number of cores on CPU
options dvs dvsof_concurrent_reads=0
```

```
# Set number of threads able to do concurrent
# reads = a positive number (e.g., 3)
#options dvs dvsof_concurrent_reads=3
```

- To change dynamically: N/A

dvsof_concurrent_writes

Controls how many threads are allowed into the write path on the server. A value of -1 disables, 0 uses the number of cores on the CPU, and any other positive number sets the number of threads. Set to 0 for best DataWarp performance.

- Default value: -1
- To view read-only:
  cat `/sys/module/dvs/parameters/dvsof_concurrent_writes`
- To change prior to boot, add these lines to
  `<`*`nid_of_choice`*`>/etc/modprobe.d/dvs-local.conf`:

```
# Disable concurrent writes
#options dvs dvsof_concurrent_writes=-1

# Set number of threads able to do concurrent
# writes = number of cores on CPU
options dvs dvsof_concurrent_writes=0

# Set number of threads able to do concurrent
# writes = a positive number (e.g., 3)
#options dvs dvsof_concurrent_writes=3
```

- To change dynamically: N/A

dvsproc_stat_control

(deprecated) Controls DVS statistics. This legacy parameter has been maintained for backward compatibility, but values are overridden by dvsproc_stat_defaults, if specified.

- Default value: 1 (enabled)
- To view:
  cat `/sys/module/dvsproc/parameters/dvsproc_stat_control`
- To change prior to boot, add these lines to
  `<`*`nid_of_choice`*`>/etc/modprobe.d/dvs-local.conf`:

```
# Disable DVS statistics
options dvsproc dvsproc_stat_control=0

# Enable DVS statistics
#options dvsproc dvsproc_stat_control=1
```

- To change dynamically:

  This is root writable at `/sys/module/dvsproc/parameters/dvsproc_stat_control`, but changes should be made only through the `/proc/fs/dvs/stats` interface, as shown in this example.

```
hostname# echo 0 > /proc/fs/dvs/stats
hostname# echo 1 > /proc/fs/dvs/stats
hostname# echo 2 > /proc/fs/dvs/stats
```

dvsproc_stat_defaults

Controls DVS statistics. Use this parameter to disable/enable and format DVS statistics.

- Default values: enable, legacy, brief, plain, notest
- To view:
  cat `/sys/module/dvsproc/parameters/dvsproc_stat_defaults`
- To change prior to boot, add these lines to
  `<`*`nid_of_choice`*`>/etc/modprobe.d/dvs-local.conf`:

```
# Disable/enable and format DVS statistics
options dvsproc
dvsproc_stat_defaults="enable,legacy,brief,plain,notest"
```

- To change dynamically:

  This is root writable at `/sys/module/dvsproc/parameters/dvsproc_stat_defaults`, but changes should be made only through the `/proc/fs/dvs/stats` interface, as shown in this example.

```
hostname# echo disable > /proc/fs/dvs/stats
hostname# echo enable > /proc/fs/dvs/stats
hostname# echo reset > /proc/fs/dvs/stats
hostname# echo json,pretty > /proc/fs/dvs/stats
```

estale_max_retry

Controls the number of times to retry an operation on the original server after it returns ESTALE.

- Default value: 36 iterations at a fixed 5 seconds per iteration (3 minutes)
- To view read-only:
  cat `/sys/module/dvsproc/parameters/estale_max_retry`
- To change prior to boot, add these lines to
  `<`*`nid_of_choice`*`>/etc/modprobe.d/dvs-local.conf`:

```
# Set the number of times to retry an operation
# on the original server after it returns ESTALE
options dvsproc estale_max_retry=
```

- To change dynamically (example changes estale_max_retry to 40 for illustration only):

```
hostname# echo 40 > /proc/fs/dvs/estale_timeout_secs
```

estale_timeout_secs

Controls the time to wait between retries of an operation after it returns ESTALE.

- Default value: 300 seconds
- To view read-only:
  cat `/sys/module/dvsproc/parameters/estale_timeout_secs`

- To change prior to boot, add these lines to `<nid_of_choice>/etc/modprobe.d/dvs-local.conf`:

  ```
  # Set the time to wait between retries of an
  # operation that returns ESTALE
  options dvsproc estale_timeout_secs=
  ```

- To change dynamically (example changes estale_timeout_secs to 400 for illustration only):

  ```
  hostname# echo 400 > /proc/fs/dvs/estale_timeout_secs
  ```

kdwfs_instance_info

Contains the following fields, which are parameters for the DataWarp thread pool in the DVS IPC layer.

Table 17. kdwfs_instance_info Field Definitions

Field	Definition
thread_min	Number of threads created at startup.
thread_max	Maximum number of persistent threads.
thread_limit	Maximum number of valid threads that DVS IPC allows to exist at one time. DVS IPC will dynamically scale up the number of threads to this number as the load increases.
thread_concurrent_creates	Maximum number of IPC threads that can be in the process of forking a new thread.
thread_nice	Nice value for the threads.
single_msg_queue	Disables/enables fairness of service. Setting to 1 disables fairness of service by processing incoming requests with a FIFO method. Setting to 0 (default) groups requests into different queues (qhdrs) based on apid, and round-robbins the queues to maintain quality of service (QOS) among jobs.
init_free_qhdrs	Low water mark for the pool of unused qhdrs. When the pool falls below this number, more are allocated. Used only if single_msg_queue = 0.
max_free_qhdrs	Maximum number of unused qhdrs that can exist before the system starts freeing them. Used only if single_msg_queue = 0.
Interactions among fields: thread_min <= thread_max <= thread_limit (set all three equal for best DataWarp performance) thread_concurrent_creates <= thread_limit	

Field	Definition
init_free_qhdrs and max_free_qhdrs used only when single_msg_queue == 0	

IMPORTANT: For this parameter to be valid, values must be specified for all of the fields. To avoid unintentional changes, be careful when changing any field values.

- Default value: 1,1,1024,4,-10,1,1,1
- To view read-only:
  cat `/sys/module/dvsipc/parameters/kdwfs_instance_info`
- To change prior to boot, add these lines to <*nid_of_choice*>`/etc/modprobe.d/dvs-dws.conf` (values shown are defaults):

  ```
  # Set parameters for DataWarp thread pool in DVS IPC
  layer
  options dvsipc
  kdwfs_instance_info=256,256,1024,4,-10,1,1,1
  ```

  This translates to kdwfs_instance_info thread_min = 256, thread_max = 256, thread_limit = 1024, thread_concurrent_creates = 4, nice = -10, single_msg_queue = 1, init_free_qhdrs = 1, max_free_qhdrs = 1.
- To change dynamically: N/A

lnd_name

`lnd_name` uniquely identifies the LNet network that DVS will use. DVS communicates it to the LNet service when DVS is being initialized. It must match the `cray_lnet.settings.local_lnet.data.lnet_name` value set in the `cray_lnet` service for DVS to boot properly.

- Default value: gni99
- To view read-only: not visible in `/sys/module`
- To change prior to boot, add these lines to <*nid_of_choice*>`/etc/modprobe.d/dvs-local.conf`, substituting for *`gnix`* the value found from the config set search:

  ```
  # Set identifier of LNet network DVS will use
  options dvsipc_lnet lnd_name=gnix
  ```
- To change dynamically: N/A

sync_dirty_timeout_secs

On DVS *servers*, specifies the number of seconds that must have passed since the file was written before DVS syncs it. The objective is to reduce unnecessary sync operations for files actively being updated. Decreasing this number increases the likelihood that the file is in use when it is synced. Increasing this number increases the likelihood that processes are killed during a server failure.

On DVS *clients*, specifies the number of seconds that must have passed since the file was written before DVS asks the server for an updated sync time. Decreasing this number increases the number of DVS requests being sent. Increasing this number increases the likelihood that processes are killed during a server failure.

This parameter is part of the periodic sync feature.

- Default value: 300 (servers and clients)
- To view read-only:
  cat `/sys/module/dvs/parameters/sync_dirty_timeout_secs`
- To change prior to boot, add these lines to `<`*`nid_of_choice`*`>/etc/modprobe.d/dvs-local.conf`:

  ```
  # Set the timeout (seconds) for syncing
  # dirty files on a DVS server or client
  options dvs sync_dirty_timeout_secs=300
  ```

- To change dynamically:

  ```
  hostname# echo 300 > /proc/fs/dvs/sync_dirty_timeout_secs
  ```

sync_num_threads

Specifies the number of threads on the DVS server that perform sync operations. The number of threads must be a minimum of 1 thread and a maximum of 32. Note that it can take up to `sync_period_secs` for changes to this value to take effect.

This parameter is part of the periodic sync feature.

- Default value: 8
- To view read-only:
  cat `/sys/module/dvs/parameters/sync_num_threads`
- To change prior to boot, add these lines to `<`*`nid_of_choice`*`>/etc/modprobe.d/dvs-local.conf`:

  ```
  # Set the number of threads that perform
  # sync operations on a DVS server
  options dvs sync_num_threads=8
  ```

- To change dynamically:

  ```
  hostname# echo 8 > /proc/fs/dvs/sync_num_threads
  ```

sync_period_secs

On DVS *servers*, specifies the number of seconds before the `sync_num_threads` syncs files on the DVS server (if necessary).

On DVS *clients*, specifies the number of seconds between checks for dirty files that need to request the last sync time from the server.

This parameter is part of the periodic sync feature.

- Default value: 300 (server), 600 (client)

- To view read-only:
  cat `/sys/module/dvs/parameters/sync_period_secs`
- To change prior to boot, add these lines to
  `<`*`nid_of_choice`*`>/etc/modprobe.d/dvs-local.conf`:

  ```
  # Set the sync period (seconds) on DVS server/client
  options dvs sync_period_secs=300
  ```

- To change dynamically:

  ```
  hostname# echo 300 > /proc/fs/dvs/sync_period_secs
  ```

6.1.8.6 About the Quiesce DVS-projected File System

Sites can use the DVS quiesce capability to temporarily suspend traffic to a DVS-projected file system on any number of DVS servers. When a directory is quiesced, the DVS server completes any outstanding requests but does not honor any new requests for that directory. Any outstanding requests for that directory are displayed in `/proc/fs/dvs/quiesce` file system interface. Administrators can read that proc file to know when it is safe to perform operations on the quiesced directory without any possibility of interference by a DVS client. DVS quiesce can be used when a file system needs to be repaired or to safely take a DVS server node out of service.

CAUTION: Because it may cause data corruption, do not use DVS quiesce to:

- quiesce a directory that is being used by DataWarp
- quiesce a directory on every DVS server

To use DVS quiesce, an administrator writes into and reads from `/proc/fs/dvs/quiesce`.

How Quiesce Works: the Userspace Application View

Userspace applications have no visibility into any specific quiesce information. A quiesced directory will present in one of two ways:

- Entirely normally, if the directory is quiesced only on servers that the application is not using.
- Useable but with degraded performance, if the application finds that its main server is quiesced and must query other servers.

How Quiesce Works: the Server View

To provide the quiesce capability, DVS servers keep a list of quiesced directories and the current outstanding requests on each quiesced directory. When an admin requests that DVS quiesce a directory on a server, DVS does the following:

- adds that directory to the server's list of quiesced directories
- iterates over all open files, closing any file that resides in the quiesced directory and setting a flag to indicate that the file was closed due to the directory being quiesced

When a DVS server receives a request from a client, DVS checks the request path against the list of quiesced directories. The comparison between the path name in the request and the quiesced directory is a simple string compare to avoid any access of the underlying file system that has been quiesced. If DVS finds that the request is for a quiesced file system, it sends a reply indicating that the request could not be completed due to a quiesce and noting which directory is quiesced. If the client request is for a file that has been closed due to quiesce, the server returns a reply to the client indicating that the request could not be completed due to a quiesce.

When an admin unquiesces a directory on a DVS server, DVS simply removes that directory from the server's list of quiesced directories and clears all quiesce-related flags for that directory.

How Quiesce Works: the Client View

When making a request of a server, a client may get back reply indicating that the request was for a file in a quiesced directory. The client then retries the operation on the next server in its server list. If it makes the request of every server in its server list and gets the same reply from each of them, then one of two things happens, depending on the type of request:

path name request	If the request is a path name request (lookup, stat, file open, etc.), then DVS reattempts the operation on a different server in a round-robin fashion until it finds a server that allows the operation to complete successfully.
open file	If the request is for an open file (read, write, lseek, etc.), then DVS attempts the operation on a different server. If the file is not open on any other servers, DVS attempts to open on the file on a server in a round robin fashion until it gets a successful open. DVS will then attempt to perform the operation.

If a client receives a reply indicating a quiesced directory, the client adds that directory to a list of quiesced directories held on the DVS superblock. This is intended to reduce network traffic by avoiding requests that target quiesced directories. The client's list of quiesced directories expires about every 60 seconds, thereby allowing clients to try those directories again in case one or more have been unquiesced during that time. This mechanism enables DVS to strike a balance between the timely unquiescing of a file system and a large reduction in network traffic and requests coming into the server. It also has the effect of naturally staggering clients when they start to use a server.

6.1.8.6.1 Use Case: Quiesce a Single Directory on a Single DVS Server

Prerequisites

This procedure requires administrative privileges.

About this task

The example provided in this procedure is for a scenario in which an admin wants to quiesce a directory `/gpfs/test/foo` on a DVS server. This is an unlikely use case, but an illustrative example.

Procedure

1. Quiesce the directory on the DVS server.

   ```
   dvs1# echo quiesce /gpfs/test/foo > /proc/fs/dvs/quiesce
   ```

2. Ensure that the directory was properly quiesced and see if there are any outstanding requests. Repeat this occasionally to know when all outstanding requests have been cleared.

   ```
   dvs1# cat /proc/fs/dvs/quiesce
   /gpfs/test/foo/:   Outstanding_Requests 3
   ```

3. Unquiesce the directory when finished with it.

   ```
   dvs1# echo unquiesce /gpfs/test/foo > /proc/fs/dvs/quiesce
   ```

4. Ensure that the directory is no longer on the quiesced list.

   ```
   dvs1# cat /proc/fs/dvs/quiesce
   ```

6.1.8.6.2 Use Case: Quiesce All Directories on a DVS Server

Prerequisites

This procedure requires administrative privileges.

About this task

The example provided in this procedure is for a scenario in which an admin wants to remove a DVS server from service but wants to let any outstanding request complete first.

Procedure

1. Quiesce all directories on that server on the DVS server.

   ```
   dvs1# echo quiesce / > /proc/fs/dvs/quiesce
   ```

2. Look for any outstanding requests and repeat this occasionally to know when all outstanding requests have been cleared.

   When no outstanding requests remain, the server can be removed from service.

   ```
   dvs1# cat /proc/fs/dvs/quiesce
   /:           Outstanding_Requests 3
   ```

3. Unquiesce all of its projected directories to allow traffic to this server to resume.

   ```
   dvs1# echo unquiesce / > /proc/fs/dvs/quiesce
   ```

7 Security

7.1 Authentication and Authorization

Urika-GX authentication and authorization are based on a number of mechanisms that are described in this section.

Standard Linux Login and Security Policies

Linux login mechanisms, such as SSH, provide the first layer of authentication that prevents user impersonation and providing the basic Linux user attributes used for standard access controls and so forth.

The standard Linux login mechanism relies on user configuration either within the Urika-GX local LDAP directory (provided by default at installation time) or within a site configured user directory. Setting up users in either of these places and configuring the Urika-GX to use a site configured directory is beyond the scope of this publication.

The standard Linux security policies ensure basic security on individual nodes of the cluster. These policies include file access control, process protection, memory protection, etc. Standard Linux security is also beyond the scope of this publication.

Urika-GX Authorized User List

The Urika-GX authorized user list further regulates access to different aspects of the Urika-GX system, such as deciding:

- whether a user who has been identified by the Linux login mechanism is actually permitted to use Urika-GX
- whether a user is restricted to tenant access only, or has access to Urika-GX physical nodes, and which tenant(s) the user is authorized to access

The authorized user list defines the various modes of user authorization within the Urika-GX cluster. A user must have an entry in this list to be granted access to either a tenant environment or a physical node within the Urika-GX. A user without an entry, even if that user can pass the standard Linux authentication checks, will not be permitted to log into a Urika-GX. One mode of access defined within the Urika-GX authorized user list is access to tenant VMs. Users authorized to log into a tenant VM are specifically granted access to that tenant within the Urika-GX authorized user list. In addition to logging into tenant VMs, users may also be permitted to log on to physical Urika-GX nodes, such as login node. This mode of access is also defined within the Urika-GX authorized user list.

How the `crayTenant` and `CrayLoginShell` Attributes Control User Access

In addition to Mesos secrets and Kerberos keytabs, the following authorization attributes are specific to Urika-GX:

- The `crayTenant` attribute is a multi-value attribute associated with a user that determines whether a given tenant VM will permit the user to log in. There may be zero or more of these attributes on any given user.
- The `crayLoginShell` attribute determines the shell that will be used for the user if the user attempts to log into a physical node, and can be set to the Urika-GX tenant proxy server (`utp-server`) or to any regular shell.

Urika-GX tenant management uses these attributes to manage users within tenants. The addition of a user to the authorized user list indicates that the new user is permitted to log into the Urika-GX cluster, and triggers the Secret Manager to set up secrets. The existence of one or more `crayTenant` attributes on the user triggers creation of tenant isolated secrets for that user as well. The addition or removal of `crayTenant` attributes on an existing user triggers the Secret Manager to adjust tenant isolated secrets accordingly, based on the user's current tenant membership. The removal of a user from the Urika-GX authorized user list triggers removal of all secrets for that user. Removal of tenant membership for a user within the Urika-GX authorized user list triggers removal of secrets for that user within the specific Tenant VM environment.

Relaxed and Restricted User Access

When the `crayLoginShell` attribute is set to `/opt/cray/urika-tenant-proxy/default/bin/utp-server`, the user is not permitted to establish an interactive login on any physical node. However, the user is permitted to make requests through the `utp-launch` command from a tenant VM that runs on the tenant's host node. This is called *restricted* user access. Restricted access users are only authorized to log into whatever tenant VMs they are authorized for (if any). From there, as tenant users, they may use the tenant proxy mechanism to gain access to selected services on physical nodes, but are not permitted to establish interactive logins on physical nodes. Restricted access users with no tenant membership have no authorization to log the Urika-GX in any way, even if the user can pass the authentication stage of logging into the system.

When the `crayLoginShell` attribute is set to a regular shell, such as `/bin/bash`, `/usr/bin/tcsh`, the user is permitted to establish an interactive login on a physical node. This is called *relaxed* user access. Relaxed access users are permitted to log into physical nodes and may establish interactive logins on physical nodes, such as the Urika-GX login nodes. If a relaxed access user is authorized as a member of a tenant, that user may also log into that tenant environment, where, as a tenant user, that user may also gain access to selected physical node services through the tenant proxy mechanism.

Hardened Mesos, Kerberos and the Urika-GX Secret Manager

Urika-GX's security implementation ensures trustworthy and transparent authentication of logged in users to the Mesos workload manager. Hardened Mesos ensures that user authentication continues into Mesos, where jobs submitted by users are established by the correct user and accounted to the correct user. This is done through the management of principal/secret pairs associated with each user found in the Urika-GX authorized user list, and access control lists (ACLs) associated with each principal within the Mesos configuration. Once a user is defined in the Urika-GX authorized user list, an administrator can, either automatically (via a crontab entry) or manually run the `usm-sync-users` command on the SMW to detect that user and install a Mesos secret/principal pair and ACL on behalf of the user. After that they will need to stop and start the Mesos cluster. Until this is done, the user will not be permitted to interact with Mesos to schedule or manage jobs, even though the user can log into the Urika-GX or one or more of its tenant VMs. For more information, see the `usm-sync-users` man page.

In addition to Hardened Mesos, in the secure service mode, Kerberos provides a foundation for the HDFS Kerberos authorization mode. This enables strong authentication between applications that use HDFS and the HDFS file system for trustworthy access control. In the default service mode, HDFS uses its simple authorization mode. In the simple authorization mode, the HDFS network protocol provides no authentication mechanism, so it is subject to spoofing by code that can emulate the HDFS client protocol on the internal Urika-GX network. Adding the Kerberos authentication layer to the HDFS protocol adds strong authentication, but limits use of HDFS to applications written to use Kerberos with HDFS. HDFS with Kerberos ensures that transactions between user jobs and HDFS do not permit a user to impersonate any other user while accessing HDFS resources across the internal networks in the Urika-GX cluster.

Urika-GX's authentication mechanism uses two separate authentication secrets, one for Kerberos and one for Mesos.

- **Mesos User Secrets and Principals**- For Mesos the authentication secret is the *secret* element of the *principal:secret* pair used to authenticate a user's job to Mesos.
- **Kerberos keytabs** - To keep the use of Kerberos as transparent as possible, Urika-GX uses special Kerberos authentication secrets called keytabs, which permit password free authentication of a user to Kerberos within the cluster. A keytab is a cryptographic token that can be used to obtain Kerberos tickets without the need for a user to enter a Kerberos password. Similarly to the way Mesos principal/secret pairs are managed by the Urika-GX Secret Manager, Kerberos keytabs are assigned by the Secret Manager as well when a user is detected in the authorized user list. Once a keytab has been generated, a user logging into the Urika-GX will automatically receive a Kerberos session key and all associated tickets for the duration of their session without needing to enter a Kerberos password. Until the user has a keytab, however, the user will not have access to HDFS. Kerberos is only used in the Urika-GX secure service mode.

Both of these secrets are user specific and both of them are stored in files for use by their respective mechanisms.

7.2 Urika-GX Service Modes

Urika-GX features application and data security for a number of applications. Many security mechanisms play a role in the overall security architecture to ensure the system is protected against unauthorized access.

- **Default Mode** - All the installed analytic applications are available under the default service mode. Applications are configured with basic security levels wherever possible, though certain applications may provide no security features. More specifically, HDFS runs using its simple authentication mode under the default service mode.
- **Secure Mode** - Urika-GX uses Kerberos authentication while running under the secure mode. Only a limited number of analytic applications are available under this mode and those applications are configured to run in a secure configuration, the exact details of which vary by application. Typically, this means that applications that interact with HDFS require valid Kerberos credentials. Moreover, user interfaces are either disabled or require authentication under this mode.

CAUTION: If the Urika GX system is running in the secure mode, Cray does not recommend toggling back to the default mode while in production. In the default service mode, the security assurances provided by secure service mode are not in place and the security of data that was protected by secure mode may be compromised while running in the default mode. Cray cannot extend the secure mode security assurances to any system that has run in a production state in the default mode until that system has been fully re-deployed.

Table 18. List of Services Available Under the Default and Secure Service Modes

Service	Available in Default Service Mode	Available in Secure Service Mode
Cray Programming Environment	Yes	No
SELinux	Yes	No
Analytic Applications and Resource Management Tools		
ZooKeeper	Yes	Yes
Spark	Yes	Yes
Mesos Master	Yes	No
Mesos Slave	Yes	No
Kubernetes	No	Yes
HDFS NameNode	Yes	Yes
HDFS Secondary NameNode	Yes	Yes
HDFS DataNode	Yes	Yes
CGE	Yes	No
Spark History Server	Yes	No
Spark Thrift Server	Yes	No
YARN	Yes	No
YARN Resource Manager	Yes	No
YARN Node Managers	Yes	No
Hadoop Job History Server	Yes	No
Hadoop Application Timeline Server	Yes	No
Hive MetaStore	Yes	No
HiveServer2	Yes	No
Hive WebHCat	Yes	No
HUE	Yes	No
Oozie Server	Yes	No
Marathon	Yes	No
Grafana	Yes	No
InfluxDB	Yes	No
JupyterHub	Yes	No
System Management Tools		
HSS	Yes	Yes
Nagios	Yes	Yes

Cobbler	Yes	Yes
Secret Manager	Yes	Yes
Tenant management and proxy tools	Yes	Yes
Analytic programming environment components ● R ● Numpy ● Scipy ● GIT ● Environment modules ● glibc-devel ● gcc ● Python 34 ● Python 27 ● Scala ● Apache Maven ● Anaconda Python	Yes	Yes
Miscellaneous Tools and UIs		
YAM	Yes	No
mrun	Yes	No
Urika Application Management (UAM)	Yes	No
Urika-GX Application Interface (UAI)	Yes	No
HAProxy	Yes	Yes
Connectivity to Tableau	Yes	No
Docker	No	Yes

Any additional services installed on the system will use their own security mechanisms and will not be affected by Urika-GX's default and secure modes.

Table 19. Relationship Between Access Levels and Service Modes

Mode	**Restricted Access**		**Relaxed Access**	
	As Member on Tenant	**On Physical Node**	**As Member on Tenant**	**On Physical Node**
Secure	Has proxied access to Spark and HDFS commands	No access	Has proxied access to Spark and HDFS commands	Has direct access to all the services supported in the secure mode

Default	Unsupported	Unsupported	Unsupported	Has direct access to all the services supported in the default mode

7.2.1 Modify the Service Mode

Prerequisites

This procedure requires root privileges on the SMW.

About this task

The `urika-service-mode` command can be used to change and view Urika-GX's service mode. This command accepts the following values, which correspond to the two service modes supported on the system:

- `secure`
- `default`

The system ships with the default service mode enabled. Switching to the secure mode enables Kerberos authentication on the Urika-GX system. For a list of services available for each mode, refer to *Urika-GX Service Modes* on page 173.

CAUTION: If the Urika GX system is running in the secure mode, Cray does not recommend toggling back to the default mode while in production. In the default service mode, the security assurances provided by secure service mode are not in place and the security of data that was protected by secure mode may be compromised while running in the default mode. Cray cannot extend the secure mode security assurances to any system that has run in a production state in the default mode until that system has been fully re-deployed.

Procedure

1. Log on to the SMW as root.
2. Execute the `urika-service-mode` command, specifying the desired security mode.

   If the two options `--debug` and `--nostop` of the `urika-service-mode` command are not used, the `urika-service-mode` command will automatically stop Urika-GX services before it changes the service mode. The admin will then need to start Urika-GX services using the `urika-start` command, unless the `--restart` option is provided. For more information, refer to the `urika-service-mode` man page.

   Some examples of using the `urika-service-mode` are shown below:

   To switch to the secure mode, execute:

   ```
   # # urika-service-mode --restart secure
   ```

   To switch to the default mode, execute:

   ```
   # # urika-service-mode --restart default
   ```

3. Verify that the desired service mode has been set using one of the following commands:

- `# urika-service-mode`
- `# urika-state`

7.2.2 User Interface Access in the Secure Service Mode

The Urika-GX system returns a number of error messages when users attempt to access application UIs in the secure mode.

Direct access to a UI

Each service is designated a specific port number that it runs on, while the system is operating in the default service mode. For example, Jupyter Notebook's UI runs at `http://`*`hostname`*`-login1:7800` in the default mode. Only services supported under secure mode will be accessible via their direct URLs under secure mode. Therefore, if it is attempted to access UIs running on the login node using a direct URL that includes the service's designated port number, the system will return an error stating that the server could not be contacted. The exact error message will depend on how the user attempts to access the service.

Accessing the Root URL

The root URL of the Urika-GX server is `http://`*`hostname`*`-login1`. If the user attempts to access this URL while the system is in the secure service mode, the system will return the following message:

```
Requested service unavailable under current security mode
```

Accessing an Application URL

If the user attempts to access an application specific URL, the behaviour will be different for individual services. For example, the system will return the following error if it is attempted to access `http://`*`hostname`*`-login1/marathon` in the secure mode:

```
Not Found
The requested URL /marathon was not found on this server.
```

UIs on the SMW

UIs running on the SMW can be accessed under both the default and secure service modes. For example, the Nagios UI can be accessed in both modes.

7.3 Security Architecture Overview

The Urika-GX security architecture has two key goals:

- **Authentication and authorization** - Making sure that no part of the system allows user impersonation. For more information, refer to *Authentication and Authorization* on page 171
- **Tenancy** - Providing a means to prevent unauthorized access to sensitive shared resources. For more information, refer to *Tenancy* on page 179

As part of the security architecture, two additional concepts work together to tailor the security level to the needs of the site and users. These include:

- **User access level -** Determines if a given user is constrained to working within a tenant Virtual Machine (restricted access) or permitted access to physical nodes on the Urika-GX (relaxed access)
- **System service mode -** Determines if the system is strictly enforcing policies (secure mode) or relaxing policy enforcement (default mode)

For more information, refer to *Urika-GX Service Modes* on page 173

7.4 Set up Passwordless SSH

About this task

Follow this procedure if the home directory does not contain a `.ssh` file with an `rsa_id.pub` in it.

Procedure

1. Log on to the tenant virtual machine.
2. Generate a public/private RSA key pair.

```
$ ssh-keygen
Enter file in which to save the key (/home/users/erl/.ssh/id_rsa):
Created directory '/home/users/erl/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/users/erl/.ssh/id_rsa.
Your public key has been saved in /home/users/erl/.ssh/id_rsa.pub.
The key fingerprint is:
ab:26:a2:a6:a2:ee:0a:fe:08:37:d4:a8:ff:f7:74:35 erl@erics_soc
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|                 |
|   o             |
|  o .   S   E    |
| o       . . .   |
|+ o     o .      |
|=+o.. oo .       |
|&*o+.+...        |
+-----------------+
$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys
$ ssh localhost "echo success"
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is fa:86:a7:9c:6c:d3:f2:2e:25:12:3f:27:2c:c2:f9:13.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
success
```

3. Execute the following if the home directory does have a `.ssh` directory with an `id_rsa.pub` in it but it is not possible to login to localhost without a password:

```
$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys
$ ssh localhost "echo success"
```

```
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is fa:86:a7:9c:6c:d3:f2:2e:25:12:3f:27:2c:c2:f9:13.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
success
```

If it is still not possible to login to localhost without a password, the user should make sure the private key file `.ssh/id_rsa` is not readable by anyone but themselves and that their `.ssh` and `.ssh/authorized_keys` directories are not writable by anyone but themselves.

Once it is possible to log into localhost without a password, execute proxied commands without a password. To test, execute the following:

```
$ utp-launch true; echo $?
The authenticity of host 'host-os (192.168.122.1)' can't be established.ECDSA
key fingerprint is 52:b4:f8:5a:9d:af:f6:ad:70:c4:a4:4b:df:44:e7:42.
Are you sure you want to continue connecting (yes/no)? yes
0
```

Note the '`0`' that is printed on the last line. If this is not '`0`' there may be an underlying issue. The second and subsequent times this is run, the user will not be prompted for host authenticity. This is shown below:

```
$ utp-launch true; echo $?
0
```

7.5 Tenancy

On Urika-GX, tenancy refers to the ability to host users inside of a virtualized environment that is isolated from physical cluster resources, while providing access to selected physical node services through a proxy mechanism. The intent of tenancy is to contain users who are leasing time or resources on Urika-GX and keep them separated from site users who are permitted access to physical nodes, such as a login node. On Urika-GX, tenancy is implemented through the use of tenant VMs that run on physical nodes and provides controlled access to services on the physical nodes through a command proxy mechanism.

Urika-GX Tenant Proxy

The Urika-GX tenant proxy mechanism provides access to a whitelist of commands that can be used from the tenant VM. The whitelist of commands is stored in `/etc/utp.d/command_whitelist`, and contains the following definitions:

- Kubernetes command - `kubectl logs`
- Spark commands:
  - `spark-submit`
  - `spark-shell`
  - `pyspark`
  - `run-example`
  - `sparkR`
  - `spark-sql`

- The `hdfs` command.
- Lustre commands - When Lustre directories are isolated by tenant, the whitelisted wrapper command protects against all attempts to specify a pathname or indirectly reach a pathname outside of tenant isolated space.
  - `setstripe`
  - `getstripe`
  - `setdirstripe`
  - `getdirstripe`
  - `mkdir`
  - `pool_list`
  - `osts`
  - `mdts`
  - `df`
  - `quota`
  - `data_version`
  - `hsm_state`
  - `hsm_set`
  - `hsm_clear`
  - `hsm_action`
  - `hsm_archive`
  - `hsm_restore`
  - `hsm_release`
  - `hsm_remove`
  - `hsm_cancel`
  - `swap_layouts`
  - `migrate`
  - `mv`
  - `help`
  - `version`
- Two convenient commands for diagnosing problems using Urika-GX's tenant proxy mechanism:
  - `env`
  - `true`
  - `false`

  IMPORTANT: The `env` command does not permit any command line arguments, so it does not support command execution, and only displays the environment. All the Cray Graph Engine (CGE) CLI commands, `mrun` and yam (commands for flexing the cluster) commands cannot be executed from within tenants in this release. Modifying the list of whitelist commands is currently not supported.

All of these commands are bound to specific file system paths by the command whitelist to ensure that they execute safely. The Spark and HDFS commands are all provided on the tenant VMs using a wrapper around the `utp-launch` command that is used to invoke commands through the Urika-GX tenant proxy, so users can use these commands as though they were logged onto a physical node. The two additional commands are not wrapped in this manner and need to be invoked using `utp-launch` as follows:

```
# utp-launch env
# utp-launch true
```

7.5.1 Configure a Bridge Port

Prerequisites

This procedure requires root privileges.

About this task

A bridge port needs to be configured before a tenant can be created

> **NOTE:** The instructions provided in this procedure should be used as guidance only. Site specific customizations/requirements are beyond the scope of this publication.

About Bridge Port 0 (Management Network)

The Urika-GX management network interface is `eth0` by default. The tenant management installation process dynamically creates the `br0` interface and modifies/creates the `/etc/sysconfig/network-scripts/`*`ifcfg-br0`*`|`*`ifcfg-eth0`* files on the login nodes.

Examples of management configuration file contents before and after Urika-GX tenant management installation are shown below:

- Before UTM installation.

  ```
  ifcfg-eth0
      DEVICE=eth0
      ONBOOT=yes
      HWADDR=xx:xx:xx:xx:xx:xx
      TYPE=Ethernet
      BOOTPROTO=none
      IPADDR=10.142.0.145
      NETMASK=255.255.0.0
  ```

  Note that `ifcfg-br0` does not exist in the preceding example.

- After UTM installation:

  ```
  ifcfg-eth0
      NAME="eth0"
      DEVICE="eth0"
      HWADDR=xx:xx:xx:xx:xx:xx
      ONBOOT=yes
      BOOTPROTO=none
      TYPE=Ethernet
      NM_CONTROLLED=no
      BRIDGE=br0
      USERCTL=yes
  ```

```
    PEERDNS=yes
    IPV6INIT=no

ifcfg-br0
    DEVICE=br0
    TYPE=Bridge
    ONBOOT=yes
    NM_CONTROLLED=no
    MTU=1500
    BOOTPROTO=none
    IPADDR=10.142.0.145
    NETMASK=255.255.0.0
```

About Bridge Port 1 (Operational Network)

`br1` does not exist by default when Urika-GX tenant management software is installed. The administrator can create a bare-bones network configuration for `br1` or produce a fully functioning bridge configuration associated with network port 1's configuration.

An example of a bare-bones `br1` configuration is shown below:

```
/etc/sysconfig/network-scripts/ifcfg-br1
    DEVICE=br1
    TYPE=Bridge
    ONBOOT=yes
    NM_CONTROLLED=no
    MTU=1500
    BOOTPROTO=none
```

About the `utm-host-net` Command

The `utm-host-net` command can be used to modify the `/etc/sysconfig/network-scripts/ifcfg-`*`br1`*`|enp`*`X`*`s0f1` files, which are required for setting up `br1`'s configuration. The syntax of the command is:

```
utm-host-net [--opns-enable-force] [--opns-enable-dryrun] [--opns-enable] ip_addr,hwaddr-not-used,\
netmask,default_gateway,dns1_server_ip,dns2_server_ip,domain
```

NOTE: Comma separated arguments should be passed to the `utm-host-net` command in exactly the same order as shown in the preceding command syntax.

Procedure

1. Log on to the node that the tenant VM needs to be created on.

2. Identify the Ethernet device.

   This can be `eth1` or a variation of `enp`*`X`*`s0f1`, where *X* equals {7, 8, ...}. `enp7s0f1` is used as an example in this procedure.

   ```
   # ip addr
   ...
   enp7s0f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
       link/ether 00:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
       inet xxx.xx.xx.xx/20 brd 172.30.63.255 scope global enp7s0f1T
   ```

   The values listed in `ifcfg-enp7s0f1` can be used to auto-generate new files required by the Urika-GX tenant management process.

3. Execute the `utm-host-net` command to set the configuration in place.

Following are some examples of using the `utm-host-net` command.

Help information of the `utm-host-net` is shown below:

```
# utm-host-net --help
Urika Tenant Management (UTM) network checker
Usage: utm-host-net [argument]

Where

STATUS CHECKING
  -c, --check
       Reports the UTM network configuration and network device status on host nodes.
       Returns success if devices needed by UTM are present and active.
  --mgmt-check
       Reports the management UTM network configuration and network device status on host nodes.
       Returns success if devices needed by UTM are present and active.
  --opns-check
       Reports the operational UTM network configuration and network device status on host nodes.
       Returns success if devices needed by UTM are present and active.
CONFIGURATION DETECTION
  -m, --mgmt-intf
       Prints the name of the management ethernet interface found on this system.
       Does not detect if the device is active.
  -o, --opns-intf
       Prints the name of the operations ethernet interface found on this system.
       Does not detect if the device is active.
  -b, --br-intf
       Prints the name(s) of all UTM bridge interface(s) found configured on this system.
       Does not detect if device(s) are active.
  --br0-intf
       Detects if the UTM br0 interface is configured on this system.
       Does not detect if the device is active.
  --br1-intf
       Detects if the UTM br1 interface is configured on this system.
       Does not detect if the device is active.
MAKING CHANGES TO THE NETWORK CONFIGURATION
  --mgmt-enable NETINFO
       Enables the management network for use with UTM.  Attempts to detect and only make
       changes where needed.
  --mgmt-enable-dryrun NETINFO
       Show changes that would be made by --mgmt-enable but don't actually make changes
  --opns-enable NETINFO
       Enables the operations network for use with UTM.  Attempts to detect and only make
       changes where needed.
  --opns-enable-force NETINFO
       Enables the operations network for use with UTM.  Will overwrite an existing UTM network configuration.
  --opns-enable-dryrun NETINFO
       Show changes that would be made by --opns-enable but don't actually make changes

    For the above options NETINFO is of the format: 'ip,hwaddr,mask,gw,dns1,dns2,domain'
      Where
        ip - the IP address of the network
        hwaddr - the device hardware MAC address
        mask - the netmask for the network
        gw - the network gateway
        dns1 - dns 1 host/ip address
        dns2 - dns 2 host/ip address
        domain - the network domain name
HELP
  -h, --help
       Prints this usage message
```

NOTE: Comma separated arguments should be passed to the `utm-host-net` command in exactly the same order as shown in the help information.

Execution of the `utm-host-net` command with the `--opns-enable-dryrun` option is shown below:

```
$ utm-host-net --opns-enable-dryrun \
172.30.51.152,xx,255.255.240.0,172.30.48.1,172.30.84.40,172.30.84.40,us.cray.com
Urika Tenant Management network configuration
dryrun mode: no changes will be made to the system
---- Status of Operations network ----
Missing bridge: br1
Found operation network ethernet device: enp7s0f1 state=[br1 NO_IP_LINK,enp7s0f1 UP(using bridge: NO)
(IP: 172.30.51.152/20)]
---- Action ----
The Operations network on this host will need to be be updated for UTM
```

```
DRY RUN:  The follwing network device files would be created under /etc/sysconfig/network-scripts
ifcfg-br1
------------------------
DEVICE=br1
TYPE=Bridge
BOOTPROTO=static
ONBOOT=yes
NM_CONTROLLED=no
IPADDR=172.30.51.152
NETMASK=255.255.240.0
GATEWAY=172.30.48.1
DNS1=172.30.84.40
DNS2=172.30.84.40
DOMAIN=us.cray.com

ifcfg-enp7s0f1
------------------------
NAME="enp7s0f1"
DEVICE="enp7s0f1"
ONBOOT=yes
BOOTPROTO=static
TYPE=Ethernet
NM_CONTROLLED=no
BRIDGE=br1
```

The `--opns-enable-dryrun` option is used in the preceding example only to indicate that the files requested by the user will be generated by the `utm-host-net` command. This option should be replaced with `--opns-enable` during the actual execution of the `utm-host-net` command.

Execution of the command with the `--opns-enable` option is shown below:

```
# utm-host-net --opns-enable \
172.30.51.152,xx,255.255.240.0,172.30.48.1,172.30.84.40,172.30.84.40,us.cray.com
Urika Tenant Management network configuration
---- Status of Operations network ----
Missing bridge: br1
Found operation network ethernet device: enp7s0f1 state=[br1 NO_IP_LINK,enp7s0f1
UP(using bridge: NO)(IP: 172.30.51.152/20)]
---- Action ----
The Operations network on this host will need to be be updated for UTM
Running: mv /etc/sysconfig/network-scripts/ifcfg-enp7s0f1 /etc/sysconfig/network-
scripts/pre-utm/ifcfg-enp7s0f1.111317_105827
Running: cp /tmp/tmp.6x4v6LPNQP/ifcfg-br1 /etc/sysconfig/network-scripts/ifcfg-
br1
Running: cp /tmp/tmp.6x4v6LPNQP/ifcfg-br1 /etc/sysconfig/network-scripts/post-
utm/ifcfg-br1
Running: cp /tmp/tmp.6x4v6LPNQP/ifcfg-enp7s0f1 /etc/sysconfig/network-scripts/
ifcfg-enp7s0f1
Running: cp /tmp/tmp.6x4v6LPNQP/ifcfg-enp7s0f1 /etc/sysconfig/network-scripts/
post-utm/ifcfg-enp7s0f1
Please reboot now.
```

Restarting the interfaces should be performed using the instructions provided in the next step.

4. Bring up the new interfaces.

   To reboot the `br1` and the `eht1/enp7s0f1/enp8s0f1` interfaces, use the `ifdown` and `ifup` commands.

   - For a bare bones `ifcfg-br1` configuration, execute:

     ```
     # ifup br1
     ```

   - For manually configured `ifcfg-br1` and `ifcfg-enpXs0f1` network commands or `utm-host-net` generated files, execute:

     ```
     # ifdown br1;ifdown enp7s0f1;ifup br1;ifup enp7s0f1
     ```

Sample contents of the generated `/etc/sysconfig/network-scripts/ifcfg-br1` file:

```
DEVICE=br1
TYPE=Bridge
BOOTPROTO=static
ONBOOT=yes
NM_CONTROLLED=no
IPADDR=172.30.51.152
NETMASK=255.255.240.0
GATEWAY=172.30.48.1
DNS1=172.30.84.40
DNS2=172.30.84.40
DOMAIN=us.cray.com
```

Sample contents of the generated `/etc/sysconfig/network-scripts/ifcfg-enp7s0f1` file:

```
NAME="enp7s0f1"
DEVICE="enp7s0f1"
ONBOOT=yes
BOOTPROTO=static
TYPE=Ethernet
NM_CONTROLLED=no
BRIDGE=br1
```

This creates a functional bridge required for the Urika-GX tenant management tools. For additional guidance, refer to the `README-install` file, located under the `/opt/cray/urika-tenant-management/default/cluster-installer` directory on the SMW.

7.5.2 Tenant Management

A number of processes are involved in tenant management, including creating, removing, modifying, viewing and checking the status of tenants. In addition, tenant management also involves managing users associated with tenants. These tasks are performed by a number of tenant management commands, as described in this section.

Tenant Creation

NOTE: Tenant creation can only be done when the system is operating in the secure service mode. Before a tenant can be created, it is important to set up the `br1` interface. For more information, refer to *Configure a Bridge Port* on page 181.

Creating a tenant on Urika-GX is a two part process. First, the tenant configuration must be established in the `/etc/sysconfig/uxtenant` directory, then the tenant VM needs to be created using the `ux-tenant-create` command. Urika-GX ships with a sample configuration for a tenant named `default`. While `default` can be deployed on the system as-is, it is not deployed on Urika-GX when the system is shipped. Use the `default` tenant's configuration as a template for setting up, configuring and naming a tenant as needed.

Tenant names may only contain:

- the letters a-z
- the numbers 0-9
- the characters '`-`' and '`.`'

NOTE: The name '`default`' is reserved for the sample tenant configuration and cannot be used as a tenant name.

Configuring a Tenant - There are several configuration files that come into play when setting up a new tenant. These configuration files drive the tenant creation process, and control the other aspects of tenant management. A tenant created using these configuration files takes on initial configuration, based on the values provided. The following options can be used if the configuration changes after a tenant has been created:

- manually configure the tenant VM and its associated resources using standard Linux administrative tools to match the new configuration
- remove and re-create the tenant VM using the new configuration

The chosen option depends on the nature of the change and the potential impact of removing and re-creating the tenant VM.

Configuration Files

- **The Tenant Configuration File** - To configure a new tenant, create a new configuration file in `/etc/sysconfig/uxtenant/tenants` with the name to be used for the tenant VM. There is a sample configuration provided called `default` in that directory that the be copied and edited. Here is an example:

  ```
  # Management network adapter
  UXTENANT_TENANT_MGMT_IP_ADDR=10.142.150.1
  UXTENANT_TENANT_MGMT_IP_ADDR_NETMASK=255.255.0.0
  UXTENANT_TENANT_MGMT_IP_ONBOOT=yes
  # Operational network adapter
  UXTENANT_TENANT_IP_ADDR=172.30.48.7
  UXTENANT_TENANT_IP_ADDR_NETMASK=255.255.240.0
  UXTENANT_TENANT_IP_ONBOOT=no
  UXTENANT_TENANT_IP_GATEWAY=172.30.48.1
  UXTENANT_TENANT_IP_DNS1=172.30.84.40
  UXTENANT_TENANT_IP_DNS2=172.31.84.40
  UXTENANT_TENANT_IP_DOMAIN=us.cray.com
  UXTENANT_TENANT_MOUNTS="lustre home"
  UXTENANT_TENANT_HOST=login1
  ```

  In the preceding code block:

  - `UXTENANT_TENANT_MGMT_IP_ADDR` is the address of the tenant VM on the Urika-GX management Ethernet. This should be an address in the range `10.142.150.1` to `10.142.255.254` and **must be unique** among all tenant configurations on a single Urika-GX.
  - `UXTENANT_TENANT_MGMT_IP_ADDR_NETMASK` is the netmask for the management Ethernet. Leave this as `255.255.0.0`.
  - `UXTENANT_TENANT_MGMT_IP_ONBOOT` is an indication of whether the VM should bring up the management Ethernet when it boots. Leave this as `yes`.
  - `UXTENANT_TENANT_IP_ADDR` is the IP address to be used for reaching the tenant VM from on the external network or VLAN. It is important that the IP address used here be in the same sub-net that the external IP address of the host physical node is in, since the external network of the tenant VM is bridged onto the physical node's external network interface.
  - `UXTENANT_TENANT_IP_ADDR_NETMASK` is the netmask for the IP address to be used for reaching the tenant VM from the external network or VLAN. It is important that the network mask be the same as that for the external network for the host physical node, since the external network of the tenant VM is bridged onto the physical node's external network interface.

- ○ UXTENANT_TENANT_IP_ONBOOT is a flag that indicates whether or not it is required to have the externally facing IP address to be active on boot of the tenant VM. For normal operations, this should be yes but an admin may want to set it to no during pre-production testing.
- ○ UXTENANT_TENANT_IP_GATEWAY is the gateway IP address for the public network for the tenant VM. This should be the same as the gateway IP address for the host physical node.
- ○ UXTENANT_TENANT_IP_DNS1 is the IP address of the primary name server for the tenant VM.
- ○ UXTENANT_TENANT_IP_DNS2 is the IP address of the secondary name server for the tenant VM
- ○ UXTENANT_TENANT_IP_DOMAIN is the DNS search domain for the tenant VM
- ○ UXTENANT_TENANT_MOUNTS is a list of named mount point configurations to use with the tenant VM, each one names a configuration file in the /etc/sysconfig/uxtenant/mounts directory.
- ○ UXTENANT_TENANT_HOST is the name of the host node configuration for the host node on which the admin wants the tenant VM to run. It names a configuration file in the /etc/sysconfig/uxtenant/hosts directory. There are two default host configurations, login1 and login2 that come with the Urika-GX tenant management package and are already installed on the system. In general, it will be required to pick one of these without needing to configure a new host.

- **Tenant Mount Point Configuration Files** - Notice the UXTENANT_TENANT_MOUNTS in this sample configuration. This sets up NFS mount points for use by the tenant. In the sample configuration, there are two sample mount point configurations, home and lustre, provided. Site specific ones can be added as needed. The sample mount points can be found by looking in the configuration files /etc/sysconfig/uxtenant/mounts/home and /etc/sysconfig/uxtenant/mounts/lustre respectively. The sample configuration assumes that the home directories being used are exported over the management Ethernet from the SMW as home directories. It further assumes that the users' home directories are all grouped together at the same level of the directory tree (e.g. /home/users/*userName*) and that this directory tree has already been created and exported globally, so there is nothing to be done to set up the NFS export specifically for this tenant:

```
UXTENANT_MOUNT_MOUNT_POINT=/home
UXTENANT_MOUNT_TENANT_ISOLATED=NO
UXTENANT_MOUNT_TYPE=nfs
UXTENANT_MOUNT_SERVER=smw
UXTENANT_MOUNT_OPTIONS="rw"
UXTENANT_MOUNT_EXPORT_OPTIONS="rw,no_root_squash,anonuid=12796,anongid=12796"
UXTENANT_MOUNT_EXPORT_CREATE=NO
```

  In the preceding code block:

  - ○ UXTENANT_MOUNT_MOUNT_POINT is the path on both the server and the client of the exported file system where the home directories live. In this case, it is /home.
  - ○ UXTENANT_MOUNT_TENANT_ISOLATED is a flag indicating whether the mount point is structured in such a way as to isolate users from different tenants from each other.
  - ○ UXTENANT_MOUNT_TYPE is the type of mount this is. The only supported type at this time is NFS. Do not change this value.
  - ○ UXTENANT_MOUNT_SERVER is the IP address (either named or numeric) of the NFS server exporting this mount point. If the server is not a local Urika-GX node, it is best to use a numeric address in case a DNS outage causes Urika-GX to lose the ability to resolve the server name, which can cause NFS outages. If the NFS server is internal to Urika-GX, it is statically configured for the tenant in /etc/hosts, therefore, a name can safely be used. The SMW is being used in this case, so the name smw is safe to use.

- `UXTENANT_MOUNT_OPTIONS` specifies mount options to be used when mounting this file system. In this case, `rw` is specified. Others may be specified as needed.
- `UXTENANT_MOUNT_EXPORT_OPTIONS` provides the export options to be used if exports are to be created for new tenant VMs. This will be ignored in this case because `UXTENANT_MOUNT_EXPORT_CREATE` is set to `NO`.
- `UXTENANT_MOUNT_EXPORT_CREATE` is a flag indicating whether or not to create a directory on the server (if one does not already exist) and set up a specific export to each tenant VM for this mount point on the server. If `root` on Urika-GX does not have privileges on the server, this must be set to `NO` or creation of tenants using this mount point will fail. If `root` on Urika-GX does have privileges on the server, and there is a reason why creating a new export for each new tenant is desirable, then it should be set to `YES`. This setting is most often set to `YES` in combination with the `UXTENANT_MOUNT_TENANT_ISOLATED` flag on a locally controlled mount because it allows tenant creation to fully manage tenant isolation within a directory. This will be examined more closely in the `lustre` example below. In this case, the flag is set to `NO`, meaning that the directory and export provided by the user we be used.

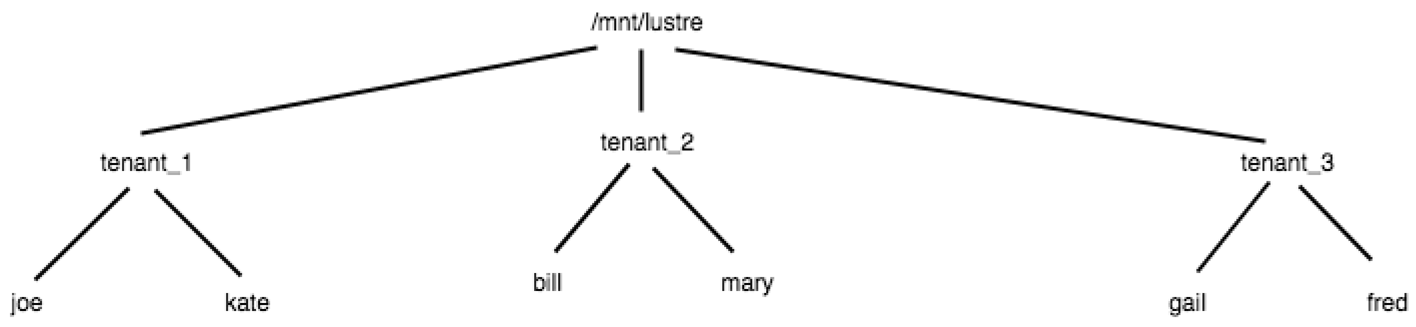
If a centrally managed home directory is being exported over NFS to systems in the data center, change the setting for `UXTENANT_MOUNT_SERVER` to specify the IP address of the central server, and the setting of `UXTENANT_MOUNT_MOUNT_POINT` to specify the directory containing home directories on the server. The remaining settings should be reviewed in light of any specific requirements.

Now, let's take a look at the sample Lustre configuration:

```
UXTENANT_MOUNT_MOUNT_POINT=/mnt/lustre
UXTENANT_MOUNT_TENANT_ISOLATED=YES
UXTENANT_MOUNT_TYPE=nfs
UXTENANT_MOUNT_SERVER=host-os
UXTENANT_MOUNT_OPTIONS="rw"
UXTENANT_MOUNT_EXPORT_OPTIONS="rw,no_root_squash,anonuid=12796,anongid=12796"
UXTENANT_MOUNT_EXPORT_CREATE=YES
```

In this case, the Lustre file system that is mounted globally on Urika-GX at the mount point `/mnt/lustre` is being exported over NFS to the tenant VM. This export uses a directory structure under `/mnt/lustre` that is designed to isolate the visibility of tenant files within each tenant. On the server the tree looks like the following:

Figure 36. /mnt/lustre Tree



In this diagram, each tenant has a sub-directory under `/mnt/lustre` that is named for the tenant. This sub-directory is specifically exported to the named tenant across the virtual host (VHOST) network on the node where the tenant VM is installed (`host-os`). When a new tenant VM is created, a new sub-directory and export is created for that tenant. The configuration settings of interest include:

- `UXTENANT_MOUNT_SERVER` is set to `host-os` which is a specially recognized name for the node on which the tenant is installed. Unlike other host-names or IP addresses, `host-os` is specifically interpreted by Urika-GX tenant management to refer to the host where the VM is defined so that operations can be done locally on that server to set up access to the NFS export across the virtual host (VHOST) network on that node.
- `UXTENANT_MOUNT_TENANT_ISOLATED` is set to YES in this case, indicating that the above directory structure will be used. This causes the mount point configuration in `/etc/fstab` on the tenant VM to be set up with `/mnt/lustre/`*`tenant_name`*, where `/mnt/lustre` is the value configured for the mount point in `UXTENANT_MOUNT_MOUNT_POINT` and *`tenant_name`* is the name of the tenant to which the tenant VM belongs.
- `UXTENANT_MOUNT_EXPORT_CREATE` is set to YES in this case, indicating that, if the path `/mnt/lustre/`*`tenant_name`* does not already exist on the server, Urika-GX tenant management will attempt to create it by logging into the server as root and creating the directory. By the same token, if there is no entry in `/etc/exports` specifically exporting `/mnt/lustre/`*`tenant_name`* to the tenant IP address, the Urika-GX tenant management infrastructure will attempt to create one by logging into the server as root and editing that file. This is useful for locally defined 'tenant isolated' mount points. If the mount point is not locally defined, but it is required to be tenant isolated, create the directory and export it to the tenant manually (or by some site defined procedure) prior to creating the tenant VM. In that case, set this setting to `NO` but leave the `UXTENANT_MOUNT_TENANT_ISOLATED` setting set to YES and leverage the naturally occurring tenant isolation on an external server.

- **Tenant Host Configuration Files -** Tenant host configuration files, located under `/etc/sysconfig/uxtenant/hosts`, can be used to configure nodes other than the `login1` or `login2` nodes to host tenant VMs. Following is an example for `login1`:

```
UXTENANT_HOST_VHOST_IP_HOST_ADDR=192.168.122.1
UXTENANT_HOST_HOSTNAME=login1
```

  The specifics of this configuration are as follows:

  - `UXTENANT_HOST_VHOST_IP_HOST_ADDR` is the IP address on the local virtual host (VHOST) network to be used for the `host-os` address to contact the host OS node for the VM. This address is used in setting up the VHOST network on the host node the first time a tenant is created there, and in configuring the DHCP host entries used to give each tenant VM an IP address on the VHOST network as well. There is no configuration of network mask, forwarding or DNS here, since the network mask is assumed to be `255.255.255.0` and no forwarding or DNS is done through the host OS node. The 24 bit prefix (in this case `192.168.122`) should always use the `192.168.` 32 bit prefix and should be unique both within all other `192.168.` networks on Urika-GX and within all host configuration files on Urika-GX.
  - `UXTENANT_HOST_HOSTNAME` is the name or IP address of the host node within the Urika-GX management network. This allows naming a host configuration file differently from the host name it actually refers to, in the unlikely event that this is needed. Generally, this should be the same as the name of the host configuration file.

  The information here is intended primarily to fill out the admin's understanding of the Urika-GX tenant management configuration. In most cases, `login1` or `login2` will be sufficient and should be used.

Creating a Tenant - Once a tenant is configured, create it using the `ux-tenant-create` command as root on the SMW. To create a tenant named *`myTenant`* in the configuration:

```
# ux-tenant-create myTenant
```

When this command completes, the tenant VM will be defined and running on the specified host node.

See the `ux-tenant-create` man page for more information on this command.

Stopping and Starting a Tenant

Use the `ux-tenant-stop` and `ux-tenant-start` commands as root on the SMW to start and stop tenant VMs on Urika-GX.

To stop a running tenant VM named *`myTenant`*, execute:

```
# ux-tenant-stop myTenant
```

To start that tenant VM execute:

```
# ux-tenant-start myTenant
```

See the `ux-tenant-stop` and `ux-tenant-start` man pages for more information on these commands.

Displaying Tenant Status

Use the `ux-tenant-status` command as root on the SMW to obtain the status of tenant VMs on Urika-GX. To list the status of all tenants on the system, execute:

```
# ux-tenant-status -a
TenantName          |PingStatus |SSHStatus |State     |LoadAverage    |Uptime  |Logged In |Management IP   |
default             |N/A        |N/A       |notfound  |N/A            |N/A     |None      |10.142.150.1    |
cb_tenant_00        |N/A        |N/A       |notfound  |N/A            |N/A     |None      |10.142.166.100  |
cb_tenant_01        |N/A        |N/A       |notfound  |N/A            |N/A     |None      |10.142.166.101  |
cb_tenant_02        |N/A        |N/A       |notfound  |N/A            |N/A     |None      |10.142.166.102  |
cb_tenant_03        |N/A        |N/A       |notfound  |N/A            |N/A     |None      |10.142.166.103  |
```

The `ux-tenant-status` command can also be used to list the status of a specific tenant or list of tenants. To list the status of the tenant named *`myTenant`* in the configuration, execute:

```
# ux-tenant-status myTenant
```

See the `ux-tenant-status` man page for more information on this command.

Removing A Tenant

A tenant VM that is no longer required can be removed by executing the `ux-tenant-remove` command as root on the SMW. This only removes the tenant VM and its related changes to Urika-GX, it does not remove the tenant configuration for the tenant VM. Use the `ux-tenant-create` command to recreate a deleted tenant. To remove a tenant named *`myTenant`*:

```
# ux-tenant-remove myTenant
```

If the tenant VM is running when the admin attempts to remove it, the removal will fail with an error indicating that the VM is currently running. In this case, either stop the VM using `ux-tenant-stop` or add the `--force` option to the `ux-tenant-remove` command as follows:

```
# ux-tenant-remove --force myTenant
```

See the `ux-tenant-remove` manual page for more information on this command.

Administering Existing Tenant VMs

Once created, a tenant VM can be administered like any other Linux system using standard Linux administrative tools. Changes to a tenant VM's configuration on the SMW do not propagate to the deployed tenant VM, so they must either be managed locally on the tenant VM administratively, or re-instantiated by destroying and re-deploying the tenant VM. It is worth noting that the tenant VM itself has limited storage resources, with the expectation that user data will be stored either within Urika-GX (Lustre or HDFS) or within user home directories. As long as user data are not stored on the tenant VM, and administrators keep careful track of local customization of existing tenant VMs, then it is fairly straightforward to re-deploy (using `ux-tenant-remove` and `ux-tenant-create`) a tenant that becomes damaged or badly out of date.

7.5.3 Tenant Virtual Machine States

A tenant VM's state can be retrieved via the `urika-tenant-status` command. The list of states that a tenant VM can be in at a given time include:

Table 20. Tenant States

State	Description
`notfound`	The tenant is configured but has not been created
`shutoff`	The tenant has been created but is currently not running
`running`	The tenant is currently running
`inshutdown`	The tenant is currently shutting down. This is a transient state
`crashed`	The tenant VM has failed due to a software error of some kind

State Transitions

A tenant VM can go from one state to another. The following tables depicts the list of possible state transitions:

Table 21. Tenant VM State Transitions

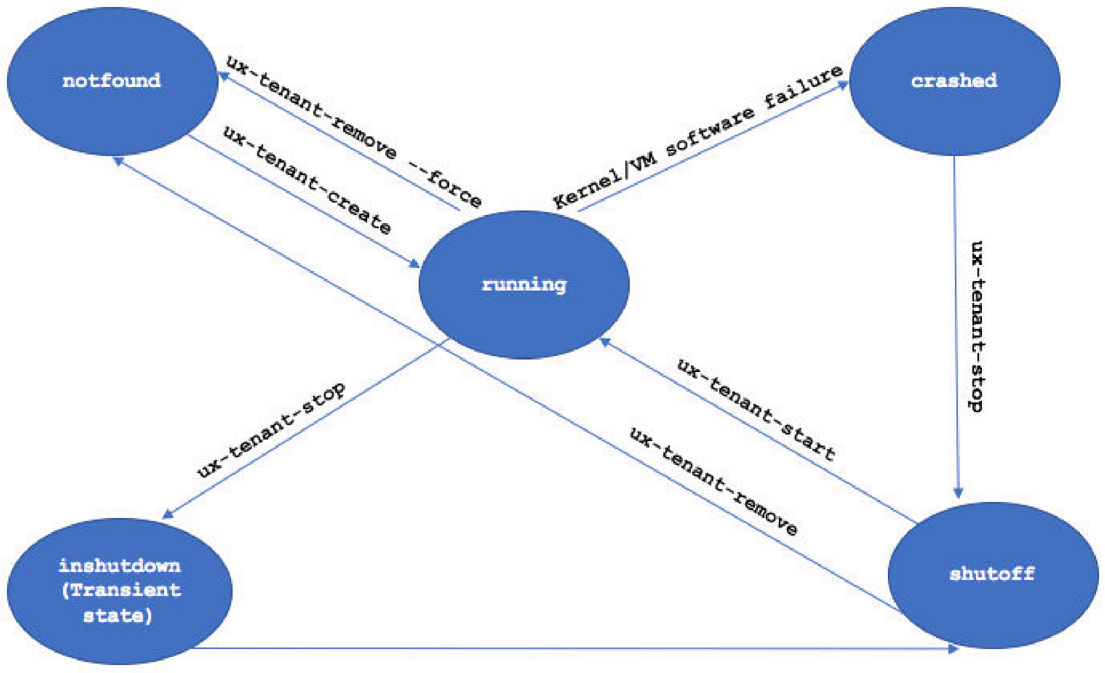
From	To	Command Used for performing the transition / Due to
`notfound`	`running`	`ux-tenant-create`. The `ux-tenant-create` command initially creates the VM and starts the HDFS Name nodes. The `ux-tenant-stop` stops the VM but does not stop the Name nodes because they do not use a significant amount of resources. It is safer not to stop and start the Name nodes.
`running`	`inshutdown`	`ux-tenant-stop`
`running`	`notfound`	`ux-tenant-remove --force`. The `--force` option enables removing a tenant that is currently running.
`running`	`crashed`	Kernel or VM software failure

From	To	Command Used for performing the transition / Due to
`crashed`	`shutoff`	`ux-tenant-stop`
`shutoff`	`running`	`ux-tenant-start`
`shutoff`	`notfound`	`ux-tenant-remove`
`inshutdown`	`shutoff`	This transition occurs automatically.

CAUTION: Tenant VMs are not robust to node redeployment or replacement. Therefore, it is importunate to note that tenant VMs will go into the `notfound` state if a node is redeployed or replaced.

These state transitions are depicted in the following diagram.

Figure 37. Tenant VM State Transitions



7.5.4 Tenant Management CLI Commands

The list of tenant management tools is shown in the following table. For a complete list of options to use with each command, refer to the corresponding man page.

Table 22. Tenant Management CLI Tools

Classification/Purpose	Command Name	Description
Tenant creation	`ux-tenant-create`	Creates a VM tenant based on a configuration template.
Tenant management	`ux-tenant-alter-vm`	Alters the tenant virtual machine (VM) resources, such as memory and CPU allocation size. Can also be used to grow the an existing VM's root disk image size. **NOTE:** The following items need to be kept under consideration while using this command: ● Number of CPUs: At least 2 CPUs need to remain available when the number of CPUs is changed by this script. That is, if there are *N* number of CPUs, a maximum of *N*-2 CPUs can be assigned to a VM. ● Amount of memory: At least 50% of the memory must remain available after a VM has been assigned memory using this command.
Tenant removal	`ux-tenant-remove`	Destroys the tenant instance or list of tenant instances.
Tenant control	`ux-tenant-start`	Starts a VM from stopped state.
	`ux-tenant-stop`	Stops a VM.

Classification/Purpose	Command Name	Description
Tenant Monitoring	`ux-tenant-status`	Provides a view of the tenant VM state
	`ux-tenant-validate`	Ensures that the physical VM host node is functional
User Management	`ux-tenant-add-user`	Adds a user to a VM tenant or group of tenants
	`ux-tenant-restrict`	Configures a user to use a restricted shell
	`ux-tenant-relax`	Configures a user to use a relaxed shell
	`ux-tenant-remove-user`	Removes a user from a tenant or group of tenants

Removing Users from a Tenant VM

Execute the `ux-tenant-remove-user` command to remove a user from a tenant VM. Executing the `usm-sync-users` command after executing `ux-tenant-remove-user` will remove any stale and inaccessible secret files for tenant user(s). Execute the `usm-sync-users` command if the deleted user accounts are added back again and `usm-sync-users` was not run since they were deleted.

If the `ux-tenant-remove-user` command is executed on a deleted user account, the system will return an error message indicating that the user is not authorized to execute this command.

CAUTION:

Running `ux-tenant-remove-user -u` *`user`* without specifying any tenants or the `-a/--all` argument will remove the user from the authorised users for the system. However it does not remove that user from per-tenant services e.g. HDFS.

To ensure that a user is removed from per-tenant services, run `ux-tenant-remove-user -u` *`user`* `-a` prior to removing them from the system.

7.5.5 Execution of Lustre Sub-Commands Inside Tenant VMs

A number of Lustre file system `lfs` sub-commands can be executed by tenant users. These include:

- `--version`
- `getdirstripe`
- `getstripe`
- `help`
- `migrate`
- `setstripe`
- `swap_layouts`

These commands are subject to the following set of rules to ensure the isolation of tenants within the Urika-GX multi tenant environment.

- The `lfs` command may not be used in its interactive mode, normally reached by invoking `lfs` without any sub-commands.
- Pathnames provided as arguments to `lfs` sub-commands:
  - are always treated as relative to the current working directory in which the `lfs` command is run.
  - may not contain elements that reference a parent directory, i.e., pathnames may not contain `/../`.
  - may not be absolute, i.e., pathnames may not begin with the `/` character.
  - may not contain elements that are symbolic links.

For more information on tenancy rules, execute:

```
# lfs help tenant-rules
```

Execute the following for additional help information, replacing *`sub-command-name`* with the name of the actual sub-command:

```
# help sub-command-name
```

The system will return the following error if a user attempts to view help information for an unsupported sub-command:

```
The sub-command command is either unknown or not supported for tenant users. For
more information on tenant user rules try 'lfs help tenant-rules'.
```

7.5.6 Get Started with Tenant Management

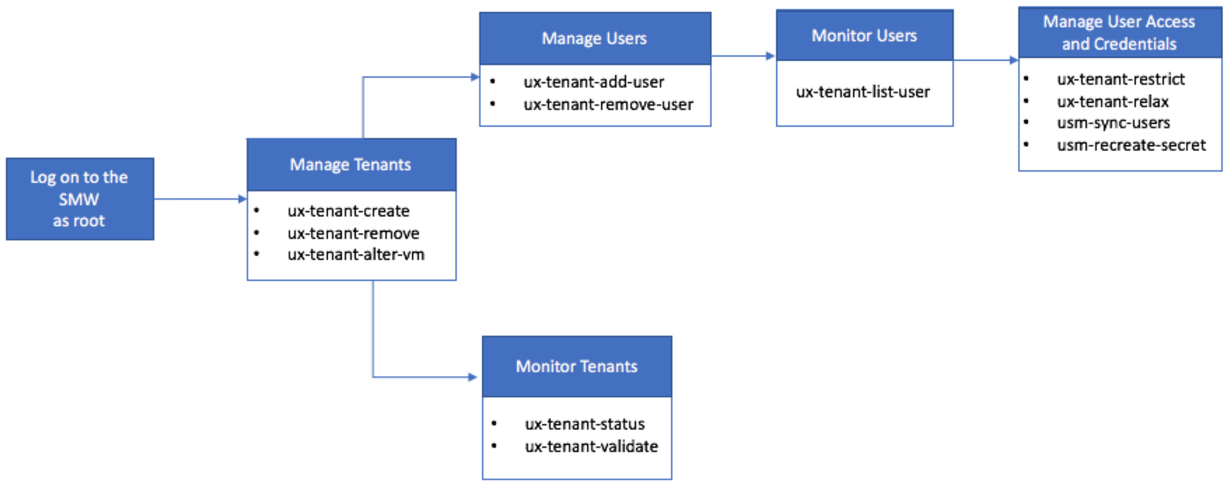
Prerequisites

- This procedure requires root privileges on the SMW.
- Before a tenant can be created, as described in this procedure, it is important to set up the `br1` interface. For more information, refer to *Configure a Bridge Port* on page 181.

About this task

Urika-GX features a number of CLI commands that can be used to manage tenant VMs and users. These commands need to be executed as root from the SMW and are depicted in the following figure:

Figure 38. Tenant Management Workflows



Each command serves a different purpose. For example, to create a new tenant VM, the administrator would log on the SMW as root, execute the `ux-tenant-create` command to create the new tenant and then execute the `ux-tenant-status` command to verify the tenant was created.

In the following instructions, *`tenantName`* and *`newUser`* are used as examples for the new tenant VM and user to be created, respectively.

Procedure

1. Log on to the SMW as root.

2. Configure a tenant VM.

   For information about tenant configuration steps and files, refer to *Tenant Management* on page 185

3. Ensure that the system is functioning in the secure service mode if creating a new tenant.

4. Create a tenant VM.

   The `ux-tenant-create` command creates and starts a tenant VM.

   Tenant names may only contain:

   - the letters a-z
   - the numbers 0-9
   - the characters '`-`' and '`.`'

     NOTE: The name '`default`' is reserved for the sample tenant configuration and cannot be used as a tenant name.

   ```
   # ux-tenant-create tenantName
   ```

5. Verify the tenant was created using the `ux-tenant-status` command

   ```
   # ux-tenant-status
   ```

6. Add/remove users (that have already been added to the site's LDAP directory via site specific procedures) to the tenant if required by performing the following steps:

   The steps to add users depend on the security mode the system is running under. Execute the `urika-service-mode` command to identify the system's service mode.

   - If the system is running in the default service mode, perform the following set of steps to add a user:
     1. Execute the `ux-tenant-add-user` command to add the user.
        ```
        # ux-tenant-add-user -u newUser  tenantName
        ```
     2. Verify that the user was added by executing the `ux-tenant-list-users` command
        ```
        # ux-tenant-list-users
        ```
     3. Execute the `usm-sync-users` command to sync the user, assign a Mesos principal and secret, and to assign a Kerberos keytab to the user by executing the `usm-sync-users` command.
        ```
        # usm-sync-users
        ```
     4. Stop the Mesos service.
        ```
        # urika-stop -s mesos_cluster
        ```
     5. Start the Mesos service.
        ```
        # urika-start -s mesos_cluster
        ```
   - If the system is running in the default service mode, perform the following set of steps to remove a user:
     1. Execute the `ux-tenant-remove-user` command to add the user.
        ```
        # ux-tenant-remove-user -u newUser  tenantName
        ```
     2. Verify that the user was removed by executing the `ux-tenant-list-users` command
        ```
        # ux-tenant-list-users
        ```
     3. Execute the `usm-sync-users` command to sync the user.
        ```
        # usm-sync-users
        ```
     4. Stop the Mesos service.
        ```
        # urika-stop -s mesos_cluster
        ```
     5. Start the Mesos service.
        ```
        # urika-start -s mesos_cluster
        ```
   - If the system is running the secure service mode, perform the following set of steps to add a user:
     1. Execute the `ux-tenant-add-user` command to add the user.
        ```
        # ux-tenant-add-user -u newUser  tenantName
        ```
     2. Verify that the user was added by executing the `ux-tenant-list-users` command
        ```
        # ux-tenant-list-users
        ```
     3. Execute the `usm-sync-users` command to sync the user, assign a Mesos principal and secret, and to assign a Kerberos keytab to the user by executing the `usm-sync-users` command.

```
# usm-sync-users
```

- If the system is running in the default service mode, perform the following set of steps to remove a user:
    1. Execute the `ux-tenant-remove-user` command to add the user.

       ```
       # ux-tenant-remove-user -u newUser  tenantName
       ```
    2. Verify that the user was removed by executing the `ux-tenant-list-users` command

       ```
       # ux-tenant-list-users
       ```
    3. Execute the `usm-sync-users` command to sync the user.

       ```
       # usm-sync-users
       ```

By default, users are added to a tenant VM with restricted user access, which prevents them from logging on to physical nodes, such as login nodes. If required, execute the `ux-tenant-relax` command to allow the user to log on to login nodes. For more information, refer to *Authentication and Authorization* on page 171, *Tenant Management* on page 185 and *Tenant Management CLI Commands* on page 192.

Troubleshooting

The default log file for the `usm-sync-users` and `usm-recreate-secret` commands is located at `/var/log/usm/urika-secret-manager.log`. `utp-server` logs are located on the host node under `/var/log/utp/`.

7.5.7 Multi-Tenancy

Urika-GX 2.1 provides secure multi-tenant operation in addition to default mode operation

Although Urika-GX enables users to use a single tenant infrastructure, users can also opt to use multi-tenancy, which features:

- **File system and data isolation** - All relevant file systems are available to tenants, including but not limited to NFS, Lustre and HDFS. However, a tenant can only access their own data and have no visibility into the existence of data belonging to other tenants. This includes local file systems used for temporary working space. In addition, users cannot see the top level directory or path of any other tenant due to separate mount points for NFS and Lustre. Lustre access is achieved by mounting Lustre directories from the node that is hosting the tenant VM using NFS across the virtual network shared by the tenant VM and the host node. There is no direct mount of Lustre available on the tenant VM.
- **Login isolation** - Each user logged into a tenant is isolated from other tenants and can only know about users belonging to the same tenant.
- **Job isolation** - Information belonging to Spark jobs running in a given tenant is visible only to the users belonging to the same tenant. Each job is executed within a container, which is orchestrated via Kubernetes.
- **Secret isolation** - Secrets relevant to a given tenant are visible only within that tenant virtual machine
- **Usage accounting** - Per tenant usage accounting data for tenant groups is available only to the tenant admin.
- **Network Isolation** - Network traffic is isolated for each tenant using the Romana overlay network on Kubernetes. It is configured automatically by tenant management commands. No separate actions are required from admins.
- **Installation of local packages** - Local packages can be installed on the tenant VM. However, this is only supported for applications that run entirely on the tenant VM, and not for applications that run on the cluster.

Securing Tenant Isolated Directories

To secure a tenant isolated directory it is necessary to secure the directory both against unauthorized detection and against unauthorized use. Securing tenant isolated directories against unauthorized observation involves securing the parent of the tenant isolated directory against reading by tenants while permitting the parent directory to be traversed in a pathname lookup. This can be achieved by removing group and world read permission on the directory, while retaining the group and world search (execute) permissions. The permission value 711 (`rwx----`) does this while allowing users to reach their authorized tenant sub-directories.

Securing the tenant sub-directory (or any directory containing tenant private data) against unauthorized use involves the following additional steps.

1. Assign a different group for each tenant and assign each target directory to its respective tenant group.
2. Assign all users who belong to the tenant to the group associated with the tenant so they will have access to the directory.
3. Secure each target directory against reading, writing or searching by anyone outside the tenant group. A mode of `1770 (trwxrwx----)` does this while only permitting tenant members to see the files in the directory.

If users within the tenant are expected to share write access to the target directory, creating their own files and directories in the target directory, then a mode of 1770 (trw---) may be helpful because it allows all members of the tenant group to read and write the directory but does not permit users to remove or rename files they do not own. If users within the tenant are not expected to share write access to the target directory the mode 750 (rwx-r-x---) may be helpful since this only allows the owner of the directory to create/remove/rename files and directories in it.

Automatic Assignment of Tenant Group Ownership and mode 0750 to an automatically Created Directory Defined in a 'mount' Configuration.

The mechanism of automatically assigning the tenant group ownership and mode 0750 to an automatically created directory defined in a 'mount' configuration is controlled by a setting in the tenant configuration file (`/etc/sysconfig/uxtenant/tenants/`*`tenantName`*) called `UXTENANT_TENANT_GROUP_NAME`. This is an optional setting that. If set, this setting causes any directory created as the result of setting up a mount (`/etc/sysconfig/uxtenant/mounts/`*`mountName`*) that has the `UXTENANT_MOUNT_EXPORT_CREATE` value set to '`yes`' with a permission of 0750 (rwxr-x---) and a group ownership equivalent to the named group. If the named group does not exist on the host node, the `ux-tenant-create` command will fail for that tenant. If `UXTENANT_TENANT_GROUP_NAME` is not set in the tenant configuration, newly created directories are created with mode 0755 (rwx-r-xr-x) and group `root`.

assigning the Tenant group ownership and 0750 mode to an automatically created

Securing Non-isolated Directories

Securing non-isolated directories containing tenant data (if such directories exist) is achieved using the approach described above for securing tenant isolated sub-directories against unauthorized use .

7.5.8 Multi-tenant HDFS

When the system is in the default service mode, tenants and non-restricted users use the same (default) NameNode. When the system is in the secure mode, tenants each have their own HDFS NameNode (name space) isolating them from non-restricted users and other restricted users in different tenants. Non-restricted users have access to the same NameNode that is used in default mode, with HDFS security enabled.

Under the Urika-GX multi-tenancy infrastructure, users can interact only with services specific to their tenant, and are unaware of the existence of other tenants. Each tenant can see their per-tenant data in HDFS and Lustre and are restricted from accessing data belonging to other tenants. An additional NameNode is created for each tenant upon tenant creation.

In order to maximize isolation and minimize overhead, the tenant NameNodes run inside Docker containers, which are orchestrated by Kubernetes, whereas the DataNodes run on the physical hosts. The Urika-GX tenant management scripts are responsible for managing the life-cycle of the NameNode container. Likewise, the Urika-GX tenant proxy is responsible for automatically directing and restricting tenant user `hdfs` commands to their respective NameNode.

In Urika-GX's multi-tenant set up, each tenant has a dedicated NameNode, which includes the HDFS configuration specific to that tenant NameNode. For a restricted user in secure mode, the Urika-GX tenant proxy is responsible for injecting the correct configuration parameters. Tenants can only interact with their own designated NameNode. Moreover, only the users that belong to a specific tenant can communicate with their respective NameNode. HDFS DataNode serves multiple NameNodes.

Each tenant is assigned its own HDFS configuration directory. Individual tenant members are restricted from overriding the Hadoop configuration directory and from specifying a specific NameNode on the CLI. As such, certain arguments passed to HDFS commands on the CLI are ignored to ensure security of tenant data. If these arguments are passed to the CLI, the system will return a warning indicating that it detected an argument that is not allowed for restricted users and that the argument is being removed.

When the cluster is switched to the secure service mode, Kerberos and HDFS Service Level Authorization (SLA) are used for authentication and authorization respectively. SLA restricts communication with a tenant NameNode to users that belong to that tenant. Kerberos is used to authenticate with the NameNode when the system is functioning in the secure service mode. HDFS commands from the VM do not require any additional arguments to specify the tenant NameNode IP addresses. Moreover, Kubernetes is used to start and monitor component containers. Other functionality, such as DNS and persistent volumes is also provided by Kubernetes.

NOTE: Tenant NameNode configuration is managed automatically by the Urika-GX tenant management scripts. Manually altering the configurations of the tenant NameNode is not supported.

Persistent Storage

By default, data inside of container is ephemeral. While much of the tenant's NameNode can be stateless and thus be unaffected by the ephemeral nature of containers, there are a few exceptions:

- **NameNode data directory** - Stores the name space directory tree and responsible for all of the file indexing. The loss of this data would result in a loss of all tenant data stored by the NameNode.
- **HDFS configuration files** - Store the specific configuration for a tenant.
- **Kerberos keytabs** - Contain Kerberos credentials for use of HDFS secure mode.

File System Permissions

The directory layout for persistent volumes is shown below:

- `/global/tenants/`*`tenant_name`*`/hdfs/conf mounted as /etc/hadoop/conf` in the tenant container.
- `/security/`*`tenant_name`*`/hdfs/service_keytabs/` mounted as `/etc/security/keytabs` in the tenant container.
- `/mnt/hdd-2/hdfs/nn-`*`tenant_name/`* mounted as `/mnt/hdd-2/hdfs/nn` in the tenant container.

Toggling

- When the system is toggled from the default to secure mode, tenants do not have access to any HDFS data they may have created when the system was in default mode.
- When the system is toggled from the secure to default mode, tenants no longer have access to any HDFS they created in secure mode. This is due to the fact they are using different NameNodes in the default or secure mode.
- Non-restricted users always communicate with the same NameNode, irrespective of the system mode, so their data is always available in either mode.
- The NameNodes are, however, persisted across toggling. Therefore, if a tenant creates data in secure mode and the system is subsequently toggled into default and again into secure the data is retained.

HDFS Tenancy and Tenant Management

Calling the `ux-tenant-create` command in the default service mode is not supported. When the system is operating in the secure service mode, the tenant NameNodes are started as part of the `urika-start` command's execution.

Users and Groups

Users and groups are extrinsic to HDFS, meaning that HDFS relies on some external system for user and group management. A user is identified by the Kerberos principal, which the user presents to the system. For example, a user with Kerberos principal `johnsmith@local` would be mapped to user `johnsmith` in HDFS.

HDFS provides a number of ways to map users to groups via group mapping providers. The most common implementations are Unix group mappings and LDAP. Unix group mappings rely on the underlying operating system to do a group lookup for a given username, whereas LDAP involves configuring HDFS to do an LDAP query to retrieve group membership for a given user.

Authorization

Authorization to HDFS is controlled via Service Level Authorization (SLA). SLA is controlled via an access control list (ACL) in the `hadoop-policy.xml` config file. The Urika-GX tenant management process automatically syncs the ACL with the tenant membership reported by Urika-GX tenant management.

7.6 Authorized User Management

Urika-GX Login Authorization

Once authenticated, a user must be present in the Urika-GX authorized user list to be permitted any mode of access to the Urika-GX.

NOTE:

Login authentication and basic Linux authorization are setup through normal Linux user management, either directly on the Urika-GX, in which case users are defined in the Urika-GX local LDAP directory, or in a site defined user directory. Both of these topics are outside of the scope of this publication.

Users are managed in Urika-GX's authorized user list using the following commands as root on the SMW:

- `ux-tenant-add-user` - add specified users to the authorized user list and to tenant membership.

- `ux-tenant-relax` - change user access to relaxed for specified users.
- `ux-tenant-restrict` - change user access to restricted for specified users.
- `ux-tenant-list-users` - produce a list of users and their attributes in the authorized user list.
- `ux-tenant-remove-user` - remove a user from tenant membership or from the authorized user list.
- `usm-sync-users` - enable Urika-GX to discover new users (that have been added via site-specific procedures) and create Mesos and Kerberos credentials for them. This command also removes secrets of user accounts that are no longer authorized.
- `usm-recreate-secret` - assign a new set of Mesos and Kerberos credentials to a user.

The `ux-tenant-add-user` command is used both to add users to the authorized user list and to assign users tenant membership within the authorized user list. For more details on command line options and arguments, see the `ux-tenant-add-user` man page.

The following examples show how to manage the authorized user list.

Add a User to the Authorized User List

Use the `ux-tenant-add-user` command to add a user to the authorized list of users.

```
# ux-tenant-add-user -u bob
```

In this example, `bob` is added to the authorized user list, but is not assigned tenant membership, which would permit him from logging on to a tenant VM. By default, `bob` is added to the list as a restricted access user, so he cannot log on to Urika-GX.

For more information, refer to the `ux-tenant-add-user` man page.

Add User to a Tenant VM

Add a user to a tenant using the `ux-tenant-add-user` command, specifying a tenant VM:

```
# ux-tenant-add-user -u bob mytenant
```

At this point, `bob` will be allowed to log into the tenant VM named *`mytenant`*.

For more information, refer to the `ux-tenant-add-user` man page.

Assign Relaxed Access to a User

Assign relaxed access to a user or list of users using the `ux-tenant-relax` command. The following shows an administrator assigning the relaxed access mode to `bob`:

```
# ux-tenant-relax -s /bin/bash bob
```

Note that relaxed or restricted access is based on the setting of the `crayLoginShell` attribute in the authorized user list. This attribute determines what shell will be provided for `bob` when he logs into a physical node (if he logs into a tenant VM he will use his normal Linux login shell). The above command explicitly sets the `crayLoginShell` attribute for `bob` to `/bin/bash`, which is also the default relaxed mode shell. The following command can also be used for the same purpose.

```
# ux-tenant-relax bob
```

For more information, refer to the `ux-tenant-relax` man page.

Assign Restricted User Access to a User

Execute the `ux-tenant-restrict` command to assign restricted access to a user.

```
# ux-tenant-restrict bob
```

For more information, refer to the `ux-tenant-restrict` man page.

Remove User's Tenant Membership

Use the `ux-tenant-remove-user` command, either to:

- remove `bob` from tenant membership or to remove `bob` entirely from the authorized users list:

  ```
  # ux-tenant-remove-user -u bob mytenant
  ```

- remove `bob` entirely from the authorized users list:

  ```
  # ux-tenant-remove-user -u bob
  ```

For more information, refer to the `ux-tenant-remove-user` man page.

View List of Authorized Users

Execute the `ux-tenant-list-users` to view the list of authorized users.

For more information, refer to the `ux-tenant-list-users` man page.

Urika-GX Mesos and Kerberos Authorization

Once a user is entered in the authorized user list and granted either tenant membership or relaxed access to Urika-GX, the user is able to log into Urika-GX , either in a tenant VM or on a physical login node. The user still will not have the ability to launch jobs under Mesos or (in the secure service mode) to access HDFS. In order to permit these forms of access, the user must be assigned a Mesos principal and secret and must (in the secure service mode) be assigned a kerberos keytab.

Discover New users

Use the `usm-sync-users` command to discover new users and create Mesos and Kerberos credentials for all of them.

```
# usm-sync-users
```

For more information, refer to the `usm-sync-users` man page.

Obtain a New Set of Secrets

If a user suspects that their secret or Kerberos keytab has been compromised, they should contact an administrator to get a new set of secrets assigned.

The following shows an administrator using the `usm-recreate-secret` to assign a new set of Mesos and Kerberos credentials to the user `bob`.

```
# usm-recreate-secret bob
```

Secrets are stored in the NFS shared directory `/security/secrets`. The `/security` directory is sub-divided into per-tenant directories as well, so there may be a directory called `/security/`*`mytenant`*`/secrets`. When users are members of tenants, their secrets are created initially in `/security/secrets` then linked to the tenant specific directories. On any given tenant, the tenant directory is mounted as `/security` so `/security/secrets/`*`secret-file`* has the same path both on physical nodes and on tenant VMs, but only the secrets belonging to users who are tenant members are visible in the tenant VM.

For more information, refer to the `usm-recreate-secret` man page.

Restarting Mesos After Syncing Users and Recreating New Secrets

The secrets for Mesos are registered with the Mesos configuration, requiring a restart of Mesos to pick up new secrets. This is an intrusive operation, because restarting Mesos will interrupt running Mesos jobs, so it is done as a manual step. After running `usm-sync-users` or `usm-recreate-secret`, wait for an appropriate maintenance window and run `urika-stop -s mesos_custer` then `urika-start -s mesos_custer` to restart Mesos.

TIP: Secrets of the '`marathon'`, '`spark'`, and '`haproxy`' accounts are not managed by the Urika-GX secret manager commands. Refer to *Modify the Secret of a Mesos Framework* on page 251 if the secrets of Urika-GX service-level Mesos secrets (such as `marathon`, `haproxy`, etc.) are compromised.

7.7 Guidance on LDAP Forwarding

Linux Authentication/Authorization

The Urika-GX cluster generally relies on LDAP to make user authentication and basic Linux authorization data available to physical nodes and to the tenant VM. Urika-GX ships with a local LDAP server running on `login1`, with a minimal set of users needed for sanity testing. If the local LDAP is examined or edited with an LDAP browser, it is important to note that the two organizational units that contain standard posix user and posix group information are the `ou=users` and `ou=groups` trees located under `dc=urika,dc=com`. In addition, the `ou=crayusers` organizational unit also lies in the same location. This unit should not be modified. It contains the Urika-GX authorized users list, and is manipulated using the `ux-tenant-add-user`, `ux-tenant-remove-user`, `ux-tenant-restrict`, and `ux-tenant-relax` commands. See *Authentication and Authorization* on page 171 for more information.

The use of the posix user and posix group configuration in the local LDAP is completely isolated from the Urika-GX authorized users list. As a result, users and posix groups can be added to the local LDAP or forwarded from the local LDAP to the site's central LDAP. It is also possible to completely replace the use of the local LDAP for authentication and posix user/group authorization without affecting the Urika-GX authorized user mechanism. There are no constraints placed on the site's authentication mechanism, except that it must provide enough information to permit Linux authentication and user login session creation to take place. Configuring a connection to the site LDAP or other authentication data sources is generally outside this publication's scope.

Sites are not required to use LDAP forwarding through the Urika-GX local LDAP to reach an external authentication data source, however, if sites do not, the admin will need to configure the site's authentication and manage data source themselves across all the physical nodes in the Urika-GX cluster and across the tenant VM. Use of LDAP forwarding through the local LDAP enables localizing this configuration in the forwarding configuration, allowing admins to manage it in a single central location.

Sites using a central LDAP server for standard Linux user authentication and authorization will likely want an efficient way to import users from that LDAP into their Urika-GX authorized user list. This can be scripted using the `ux-tenant-add-user`, `ux-tenant-remove-user`, `ux-tenant-restrict`, and `ux-tenant-relax` commands. The details of such a script are very site specific. This publication provides guidance in the form of a worked example of such a script.

WARNING: LDAP installations vary widely in type, local optimizations and policies. The material presented here is intended to be helpful for administrators but cannot account for that variation. The actual details of the site's implementation will vary according to their authentication data source and policies. There is no suggestion that the procedures outlined in this section will work as described at a site. In addition, the Urika-GX system relies on the existence of the `crayusr` and `crayadm` users as configured in the Urika-GX local LDAP for sanity checking on system deployment and re-deployment. If it is chosen to bypass the Urika-GX local LDAP prior to completion of such sanity checking, or if it is intended to use these sanity checking mechanisms on an on-going basis after bypassing the Urika-GX local LDAP, it will be required to provide authentication data for these two users through the site's own mechanism. Moreover, the script used as the basis of the worked example here is specific to Cray's LDAP servers. The information provided in this section is intended as guidance and to help structure a similar mechanism for importing users from the site LDAP or other mechanism. Site administrators are responsible for the correctness and completeness of the final site implementation. Cray does not assume responsibility for the actual content of these scripts.

Worked Example: The Urika-GX Authorized User List

Automatically updating the Urika-GX authorized user list is a matter of querying the authentication data and using the `ux-tenant-add-user`, `ux-tenant-remove-user`, `ux-tenant-restrict`,`ux-tenant-relax`, and `ux-tenant-list-users` commands to install and remove users in the Urika-GX authorized users list. Here is a simple script that queries an LDAP server for a list of users. It uses the `uid` field or a field name specified on the command line and the associated `loginShell` field or a similar field name specified on the command line. It also uses the `ux-tenant-add-user` command to add each user as a relaxed access user.

```
#! /bin/bash

# Read a list of users from LDAP and return the list as a space
# separated list of user names.
get_users() {
    host="-h ${1}"
    search_dn="-b ${2}"
    port=""
    query="-t uid"
    user_field="uid"
    if [ ! -z "${3}" ]; then
        port="-p ${3}"
    fi
    if [ ! -z "${4}" ]; then
        query="-t ${4}"
        user_field="${4}"
    fi
    search="ldapsearch ${host} ${port} -x ${search_dn} ${query}"
    grep="grep ^${user_field}:[[:space:]]"
    sed="sed -e s/^${user_field}:[[:space:]][[:space:]]*//"
    out="$(${search} | ${grep} | ${sed})"
    if [ $? -ne 0 ]; then
        return 1
    fi
    echo ${out}
    return 0
```

```
}

# Read the shell attribute for the specified user from LDAP and return
# the value as a string
get_user_shell() {
    user="${1}"
    host="-h ${2}"
    search_dn="-b ${3}"
    port=""
    query="-t uid"
    user_field="uid"
    shell_field="loginShell"
    if [ ! -z "${4}" ]; then
        port="-p ${4}"
    fi
    if [ ! -z "${5}" ]; then
        user_field="${5}"
    fi
    if [ ! -z "${6}" ]; then
        shell_field=${6}
    fi
    if [ ! -z "${5}" ]; then
        passwd="-w ${5} -x"
    fi
    if [ ! -z "${6}" ]; then
        port="-p ${6}"
    fi
    query="-t ${user_field}=${user} ${shell_field}"
    search="ldapsearch ${host} ${port} -x ${search_dn} ${query}"
    grep="grep ^${shell_field}:[[:space:]]"
    sed="sed -e s/${shell_field}:[[:space:]][[:space:]]*//"
    if ! out="$(${search} | ${grep} | ${sed})"; then
        return 1
    fi
    echo ${out}
    return 0
}

usage() {
    error="${1}"
    (
        if [ ! -z ${error} ]; then
            echo "${error}"
        fi
        echo "usage: import_users_from_openldap.sh -b search_dn -h host [-p port]"
        echo "                                     [-u field-name] [-s field-name] [-r] [-n]"
        echo ""
        echo "Where:"
        echo ""
        echo "    -h host"
        echo "        Specifies the host name / IP address of the LDAP server from which"
        echo "        to import users."
        echo "    -r"
        echo "        Instead of adding users from the source, remove them."
        echo "    -p port"
        echo "        Specifies the port number on which the LDAP server is listening for"
        echo "        queries.  Default is 389."
        echo "    -b search_dn"
        echo "        Specifies the DN in which to search for users.  The default is"
        echo "        'ou=users,<base_dn>' where <base_dn> is the argument to the -D"
        echo "        option above."
        echo "    -u field-name"
        echo "        Specifies the field used to search for user names in the LDAP"
        echo "        directory.  Default is 'uid'."
        echo "    -s field-name"
        echo "        Sepcifies the field used to lookup the user's login shell in the"
        echo "        LDAP server's user object.  Default is 'loginShell'."
        echo "    -n"
        echo "        Run this script as a dry run, listing the users and shells it wants"
        echo "        to add, but not actually adding them."
        echo ""
        echo "This command searches the specified LDAP directory for users and adds"
        echo "them to the Urika GX authorized users list, each as a relaxed mode"
        echo "user with a physical node login shell matching the login shell"
        echo "configured in the LDAP directory."
        echo ""
        echo "Example:"
        echo ""
        echo "  The following example command will import users (in dry-run mode) from"
        echo "  the LDAP server at cfdcg02.us.cray.com using the table with the DN"
        echo "  'ou=people,dc=datacenter,dc=cray,dc=com' assuming that the field name"
        echo "  for the user name is the default ('uid') and the field name for login"
        echo "  shell is the default ('loginShell'):"
        echo ""
        echo "    ./import_users_from_openldap.sh -h cfdcg02.us.cray.com -b 
"ou=people,dc=datacenter,dc=cray,dc=com" -n"

    ) >&2
    exit 1
}

# Option Processing
dry_run=""
host=""
base_dn=""
user_dn=""
passwd=""
port=""
search_dn=""
user_field=""
shell_field=""
remove_flag=""

OPTIONS=$(getopt -o h:p:b:u:s:nr -n 'import_users_from_openldap.sh' -- "$@")
if [ $? -ne 0 ]; then
    usage
fi
eval set -- "$OPTIONS"
while true ; do
    case "$1" in
        -h)
            host="${2}"
            shift;shift;;
        -r)
            remove_flag='yes'
            shift;;
        -p)
            port="${2}"
            shift;shift;;
        -b)
            search_dn="${2}"
            shift;shift;;
        -u)
            user_field="${2}"
            shift;shift;;
        -s)
            shell_field="${2}"
            shift;shift;;
        -n)
            dry_run="yes"
            shift;;
        --) shift; break;;
        *) usage; exit 1 ;;
    esac
done

if [ -z "${host}" ]; then
    usage "Host must be specified using the -H option"
fi

if [ -z "${search_dn}" ]; then
    usage "Search DN must be specified using the -b option"
fi

if ! users="$(get_users "${host}" "${search_dn}" "${port}" "${user_field}")"; then
    echo "ERROR: looking up user list from '${host}' at '${search_dn}' failed." >&2
    exit 1
fi

for u in ${users}; do
    if ! shell="$(get_user_shell "${u}" "${host}" "${search_dn}" "${port}" "${user_field}" "$
{shell_field}")"; then
```

```
        echo "WARNING: could not find shell for '${u}', skipping..."
        continue
    fi
    if [ -z "${remove_flag}" ]; then
        if [ -z "${dry_run}" ]; then
            echo "Adding user '${u}' with crayLoginShell '${shell}'"
            if ! ux-tenant-add-user -s "${shell}" -u "${u}" ; then
                echo "WARNING: adding user '${u}' with crayLoginShell '${shell}' failed, skipping this
user"
                continue
            fi
        else
            echo "Not really adding user '${u}' with crayLoginShell '${shell}' (dry-run)"
        fi
    else
        if [ -z "${dry_run}" ]; then
            echo "Removing user '${u}'"
            if ! ux-tenant-remove-user -u "${u}" ; then
                echo "WARNING: removing user '${u}' failed, skipping this user"
                continue
            fi
        else
            echo "Not really removing user '${u}' (dry-run)"
        fi
    fi
done
exit 0
```

This script mainly provides the means for getting a list of users, and for each user:

- getting the user's login shell setting
- adding the user using `uxtenant-add-user`

The following code snippet shows how the list of users is retrieved by this script:

```
# Read a list of users from LDAP and return the list as a space
# separated list of user names.
get_users() {
    host="-h ${1}"
    search_dn="-b ${2}"
    port=""
    query="-t uid"
    user_field="uid"
    if [ ! -z "${3}" ]; then
        port="-p ${3}"
    fi
    if [ ! -z "${4}" ]; then
        query="-t ${4}"
        user_field="${4}"
    fi
    search="ldapsearch ${host} ${port} -x ${search_dn} ${query}"
    grep="grep ^${user_field}:[[:space:]]"
    sed="sed -e s/^${user_field}:[[:space:]][[:space:]]*//"
    out="$(${search} | ${grep} | ${sed})"
    if [ $? -ne 0 ]; then
        return 1
    fi
    echo ${out}
    return 0
}
```

The arguments to the function are:

- The host where the site LDAP server is running ($1)
- The distinguished name of the table to search on that server for the user ($2)
- The TCP/IP port number on which the server is running (by default, not specified which lets `ldapsearch` choose a default of 389) ($3)
- The name of the field containing the user's name (by default `uid`) ($4)

This is used to compose an `ldapsearch` command that is then filtered and reduced to a list of user names by the `grep` and `sed` commands that follow it in the pipe line.

If the site LDAP has a way to distinguish users with access to Urika-GX using some user attribute, group membership, etc., querying might return all the users in a particular group or set of groups instead of simply taking the complete list of users.

Along similar lines, it may be required to gather the names of users based on tenant membership using, for example, group membership. In that case, go through users in a tenant, then possibly through users with relaxed access, and then install them differently:

- the tenant users with tenant membership and restricted access, i.e., not specifying a `crayLoginShell` value in the `ux-tenant-add-user` command
- the users with relaxed access (who might be a subset of users with tenant access) by searching for a different group membership, then:
    - getting the user's `loginShell` value to use as a `crayLoginShell` value when relaxing the user
    - adding each user using the `ux-tenant-add-user` command with no tenants specified. This will not alter existing user entries, but will add the new ones in restricted mode with no tenant membership.
    - relaxing each user (as appropriate) using the `ux-tenant-relax` command, while specifying the user's `crayLoginShell` value using the **-s** option

The code this script uses to look up the `loginShell` (or equivalent) parameter is:

```
# Read the shell attribute for the specified user from LDAP and return
# the value as a string
get_user_shell() {
    user="${1}"
    host="-h ${2}"
    search_dn="-b ${3}"
    port=""
    query="-t uid"
    user_field="uid"
    shell_field="loginShell"
    if [ ! -z "${4}" ]; then
        port="-p ${4}"
    fi
    if [ ! -z "${5}" ]; then
        user_field="${5}"
    fi
    if [ ! -z "${6}" ]; then
        shell_field=${6}
    fi
    if [ ! -z "${5}" ]; then
        passwd="-w ${5} -x"
    fi
    if [ ! -z "${6}" ]; then
        port="-p ${6}"
    fi
    query="-t ${user_field}=${user} ${shell_field}"
    search="ldapsearch ${host} ${port} -x ${search_dn} ${query}"
    grep="grep ^${shell_field}:[[:space:]]"
    sed="sed -e s/${shell_field}:[[:space:]][[:space:]]*//"
    if ! out="$(${search} | ${grep} | ${sed})"; then
        return 1
    fi
    echo ${out}
    return 0
}
```

Again, the parameters to the function specify the information needed to set up the `ldapsearch` command. The `ldapsearch` results are then filtered and processed by `grep` and `sed` to produce the `loginShell` value.

Putting all of this together in the main body of the script, here is the code that actually retrieves the data and builds the users:

```
for u in ${users}; do
    if ! shell="$(get_user_shell "${u}" "${host}" "${search_dn}" "${port}" "${user_field}" "$
{shell_field}")"; then
        echo "WARNING: could not find shell for '${u}', skipping..."
        continue
    fi
    if [ -z "${remove_flag}" ]; then
        if [ -z "${dry_run}" ]; then
            echo "Adding user '${u}' with crayLoginShell '${shell}'"
            if ! ux-tenant-add-user -s "${shell}" -u "${u}" ; then
                echo "WARNING: adding user '${u}' with crayLoginShell '${shell}' failed, skipping this
user"
                continue
            fi
        else
            echo "Not really adding user '${u}' with crayLoginShell '${shell}' (dry-run)"
        fi
    else
        if [ -z "${dry_run}" ]; then
            echo "Removing user '${u}'"
            if ! ux-tenant-remove-user -u "${u}" ; then
                echo "WARNING: removing user '${u}' failed, skipping this user"
                continue
            fi
        else
            echo "Not really removing user '${u}' (dry-run)"
        fi
    fi
done
```

It can be seen that the script adds a user using the `ux-tenant-add-user` command in the following detail:

```
if ! ux-tenant-add-user -s "${shell}" -u "${u}" ; then
    echo "WARNING: adding user '${u}' with crayLoginShell '${shell}' failed, skipping this user"
    continue
fi
```

Notice that this code uses the **-s** option to `ux-tenant-add-user` to set the `crayLoginShell` value to the `loginShell` value for the user. This is because this script was designed to import all users with relaxed user access, so it defines the users in one step.

Another way to achieve the same thing would be to do it in two steps:

```
if ! ux-tenant-add-user -s "${shell}" -u "${u}" ; then
    echo "WARNING: adding user '${u}' failed, skipping this user"
    continue
fi
if ! ux-tenant-relax -s "${shell}" ${u} ' ; then
    echo "WARNING: relaxing access for user '${u}' failed, this user may be restricted"
fi
```

This allows adding checks for whether or not the user should have relaxed user access before actually relaxing the user.

As a complement to adding a user, the following detail shows how to remove a user using the `ux-tenant-remove-user` command:

```
if ! ux-tenant-remove-user -u "${u}" ; then
    echo "WARNING: removing user '${u}' failed, skipping this user"
    continue
fi
```

It is also possible to simply remove a user from a tenant without removing the user altogether if tenant membership changes. This is shown below:

```
if ! ux-tenant-remove-user -u "${u}" "${tenant}" ; then
    echo "WARNING: removing user '${u}' failed, skipping this user"
    continue
fi
```

Here `${tenant}` is the name of the tenant from which to remove the user. This form of the command only removes tenant membership, it does not remove the user, even if the user is no longer a member of any tenant. The preceding form of the command is required to remove a user entirely from the Urika-GX authorized user list.

While not shown in the sample script, it may be required to obtain the list of users in the authorized user list. This would be useful, for example, in removing users who are removed from the site's authentication data. This can be achieved by using the `ux-tenant-list-users` command in the raw (`-R`) mode. The following shell function can help do this:

```
get_user_list() {
    ux-tenant-list-users -F name -R | while read expr; do
        eval "${expr}"
        echo "${name}"
    done
    return 1
}
```

A slight modification can help retrieve all the users in a given tenant, as shown below:

```
get_user_list() {
    tenant="${1}"
    ux-tenant-list-users -F name -R "${tenant}"| while read expr; do
        eval "${expr}"
        echo "${name}"
    done
    return 1
}
```

In this case the `${tenant}` argument filters the list of users for users who are authorized for the specified tenant. If multiple tenants are specified, the tenant list selects users who are present in at least one of the specified tenants.

User Secrets

Urika-GX uses 'secret' files to authenticate users to both Kerberos (keytabs) and Mesos (principal/secret pairs). Whenever the contents of the Urika-GX authorized users list change, these secrets need to be synchronized with the authorized user list using the `usm-sync-users` command. Run this command on the SMW after updating the Urika-GX authorized users list.

When Mesos secrets change, the Mesos cluster needs to be restarted. This is done using the following command sequence:

```
smw# urika-stop -s mesos_cluster
smw# urika-start -s mesos_cluster
```

WARNING: Stopping and starting the Mesos cluster will cause running jobs within the cluster to terminate. This should only be done during a suitable maintenance interval or when the Mesos cluster is quiescent to avoid job failure.

7.8 Authentication Mechanisms

Table 23. Authentication Mechanisms

Application	Authentication Mechanism
Cray Application Management UI	LDAP. Users can also log in with the default account shipped with the system. This account has the following credentials: username: `admin` password: `admin`
Urika-GX Applications Interface	Not available.
Documentation and Learning Resources UI	Not available.
Grafana UI	LDAP. The system also ships with a default account that can be used to log on to Grafana. The credentials of this account are: username: `admin` password: `admin`
Jupyter Notebook UI	LDAP. The system also ships with a default account that can be used to log on to Jupyter. The credentials of this account are: username: `crayadm` password: `initial0`
HUE UI	LDAP. The system also ships with a default admin account that can be used to log on to HUE. The credentials of this account are: username: `admin` password: `initial0`
Hadoop/Spark related UIs, such as Spark History Server, Hadoop History Server, YARN Resource Manager, etc.	Not available
Marathon UI	Not available.
Mesos UI	User name: LDAP user name Password: Mesos secret, found in the `/security/secrets/`*`userName`*`.mesos` file, located on all the nodes.
Spark Thrift Server	Urika-GX ships with LDAP authentication enabled for Spark Thrift server. SSL authentication can be set up if required. For instructions, refer to *Urika®-GX System Administration Guide*. Storage based authorization is supported for Spark Thrift Server.

Application	Authentication Mechanism
HiveServer2	LDAP authentication for HiveServer2 can be enabled if required. Storage based and SQL standard based authorizations are supported for HiveServer2. For more information, refer to *Urika®-GX System Administration Guide*
Nagios UI	Default credentials: ● User name: `crayadm` ● Password: `initial0`

7.9 Change Default Passwords

Change the SMW's Password

Follow the instructions documented in *Change the Default System Management Workstation (SMW) Passwords* on page 219.

Change the iDRAC's Password

Follow the instructions documented in *Change the Default iDRAC8 Password* on page 217

Change the Password of the Cray Application management UI

Default credentials:

- username: `admin`
- password: `admin`

Select **Change Password** from the **admin** drop down menu to change the default password.

Change HUE's Default Password

Default credentials:

- username: `admin`
- password: `initial0`

Change the default HUE password from the HUE shell using the following commands on both login nodes as either user `'admin'` or `'hue'`.

```
# /usr/lib/hue/build/env/bin/hue shell
# from django.contrib.auth.models import User
# user = User.objects.get(username='admin')
# user.set_password('new password')
# user.save()
```

Change Jupyter Notebook's Default Password

Default credentials:

- username: `crayadm`

- password: `initial0`

The admin user for Jupyter is `crayadm`, which is an LDAP user. Change the Jypyter password by changing the LDAP password, which typically looks like:

```
# ldappasswd -H ldap_server_or_IP -x -D " user_dn" -W -A -S
```

In the preceding command:

- `ldap_server_or_IP` = server name, domain or IP
- `user_dn` = LDAP user name

Usage of this command will connect to the provided LDAP server and authenticate with the `user_dn` entry. The user will be asked to provide and confirm the old password, the new password, and will be asked to supply the old password again for the actual bind to take place. After that, the password will change. If a new user is to be designated as the `admin` user, add that user to the `c.Authenticator.admin_users` in the `/etc/jupyterhub/jupyterhub_config.py` file.

Change Grafana's Default Password

Default credentials:

- username: `admin`
- password: `admin`

Select **Profile** from the **admin** menu and then select the **Change Password** button to change the default password.

Additional Enabled Accounts

Table 24. Enabled System Accounts

Account Name	Default Password
`builder`	`initial0`
`marathon`	`initial0`

Passwords for both the `builder` and `marathon` accounts can be changed using the `passwd` command.

LDAP Accounts

Credentials for Urika-GX's internal LDAP accounts are:

- Admin account credentials:
  - username: `crayadm`
  - password: `initial0`
- Regular user account/non-admin credentials:
  - username: `crayusr`
  - password: `initial0`

These accounts can be used for logging into Jupyter as well. The `ldappasswd` command can be used to change the password for these internal LDAP accounts. Use site specific procedures for changing external LDAP passwords.

7.9.1 Default Urika-GX System Accounts

Default System Management Workstation (SMW) Accounts

Table 25. Default SMW Accounts

Account Name	Password
`root`	`initial0`
`crayadm`	`crayadm`

The SMW account should not be set up with the local Urika-GX LDAP for security. The SMW can be connected directly to the corporate LDAP.

Table 26. Default iDRAC Account

Account Name	Password
`root`	`initial0`

MariaDB/MySQL Accounts

Table 27. Default MariaDB/MySQL Accounts

Account Name	Password
`hssds`	`hssds`
`mysql`	Empty string
`mariadb`	Empty string

Default Nagios Account

Table 28. Default Nagios Account

Account Name	Password
`crayadm`	`initial0`

Additional Accounts

Table 29. Addition Accounts and Associated Passwords

Account Name	Password
`builder`	`initial0`
`marathon`	`initial0`

Local accounts/accounts used for running services

Following are the accounts that have password login disabled:

```
apache, spark, docker, grafana, hdfs, hive , hue,
influxdb, jupyterhub, kafka, ldap, munge, nscd, nslcd, ntp,
oozie, sqoop, sssd, yarn, zookeeper, adm, bin, daemon, ftp,
games, halt, lp,,mail, nobody, operator, shutdown, sync,
usbmuxd, unbound, tss, tcpdump, systemd-network, systemd-bus-proxy,
sshd, smmsp, setroubleshoot, saslauth, rtkit, rpcuser, rpc,
radvd, rabbitmq, qemu, pulse, postgres, postfix, polkitd,
pcp, oprofile, ntp, nginx, nfsnobody, mysql, memcached,
mailnull, libstoragemgmt, haproxy, gnome-initial-setup,
geoclue, gdm, epmd, dovenull, dovecot, dhcpd, dbus,
colord, chrony, avahi-autoipd, avahi, abrt
```

LDAP

Credentials for Urika-GX's internal LDAP accounts are:

- Admin account credentials:
  - username: `crayadm`
  - password: `initial0`
- Regular user account/non-admin credentials:
  - username: `crayusr`
  - password: `initial0`

These accounts can be used for logging into Jupyter as well. The `ldappasswd` command can be used to change the password for these internal LDAP accounts.

Use site specific procedures for changing external LDAP passwords.

7.9.2 Change the Default Nagios Password

Prerequisites

This procedure requires root privileges.

About this task

The default credentials for login on to the Nagios UI on the Urika-GX system are:

- User name: `crayadm`
- Password: `initial0`

Procedure

1. Log on to the SMW as root.

   ```
   # ssh root@machine-smw
   ```

2. Stop the HTTPD service if it is running.

   ```
   # service httpd stop
   ```

3. Stop the Nagios service if it is running.

   ```
   # service nagios stop
   ```

4. Change the default Nagios password

   ```
   # htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
   ```

5. Start the HTTPD service.

   ```
   # service httpd start
   ```

6. Start the Nagios service.

   ```
   # service nagios start
   ```

7. Verify that the password has been changed by logging on with the new password to the Nagios UI, which is located at `http://`*`machine`*`-smw/nagios`

7.9.3 Change the Default iDRAC8 Password

About this task

After accessing the iDRAC8's web interface, it is recommended to change the default password using the following instructions:

Procedure

1. Bring up a web browser.
2. Go to: `https://`*`cray-drac`*, where *cray-drac* is used as an example for the iDRAC's name, such as https://*system-smw-ras*

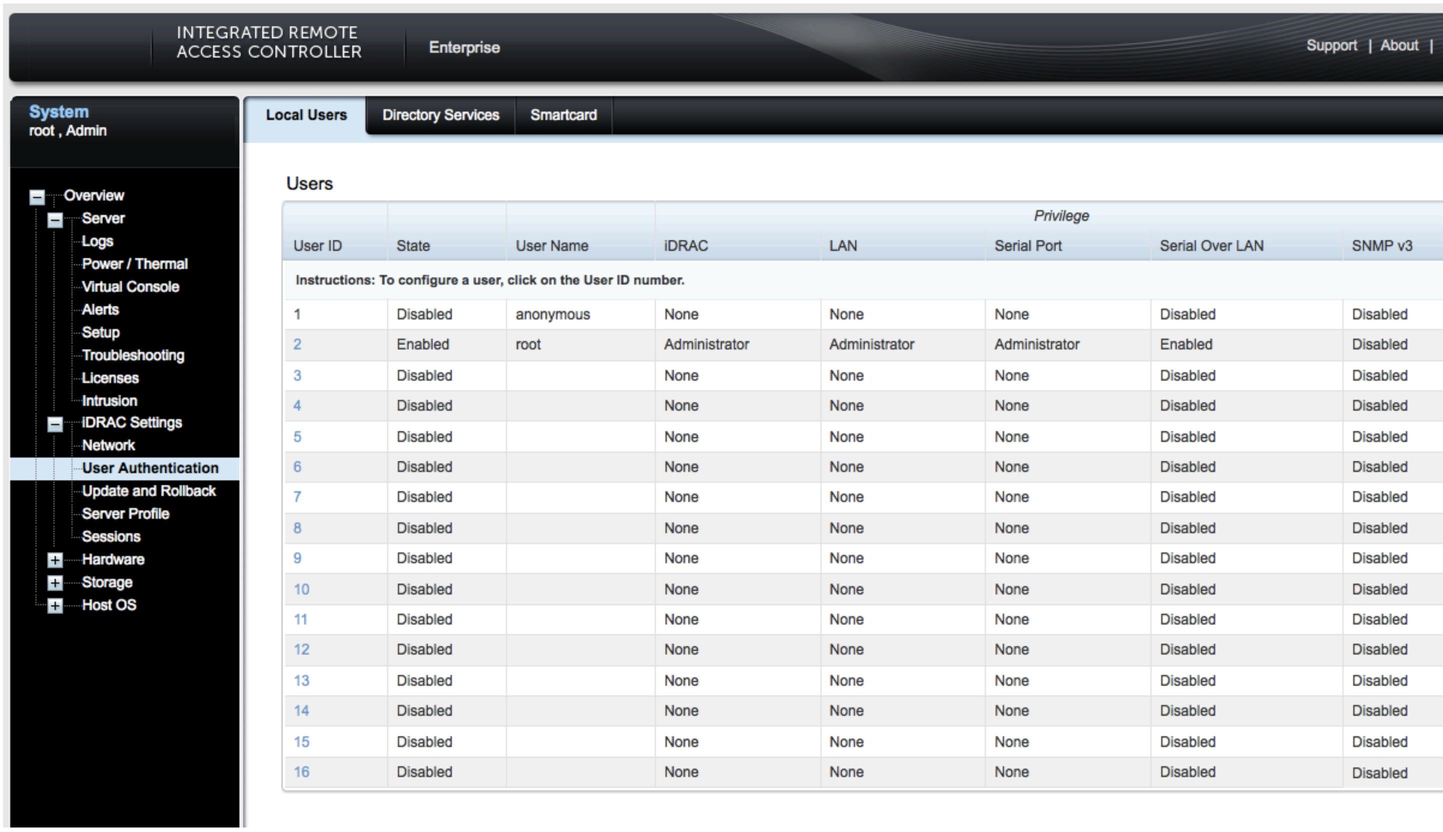
   The iDRAC's login screen appears.
3. Enter `root` and `initial0` as the default user name and password on the iDRAC's log in screen.
4. Select the **Submit** button

5. Select **iDRAC settings** from the left navigation menu bar.

6. Select **User Authentication**

7. Select the **User ID** for the user that needs to have the password changed.

Figure 39. Change the Default Password Interface



Users

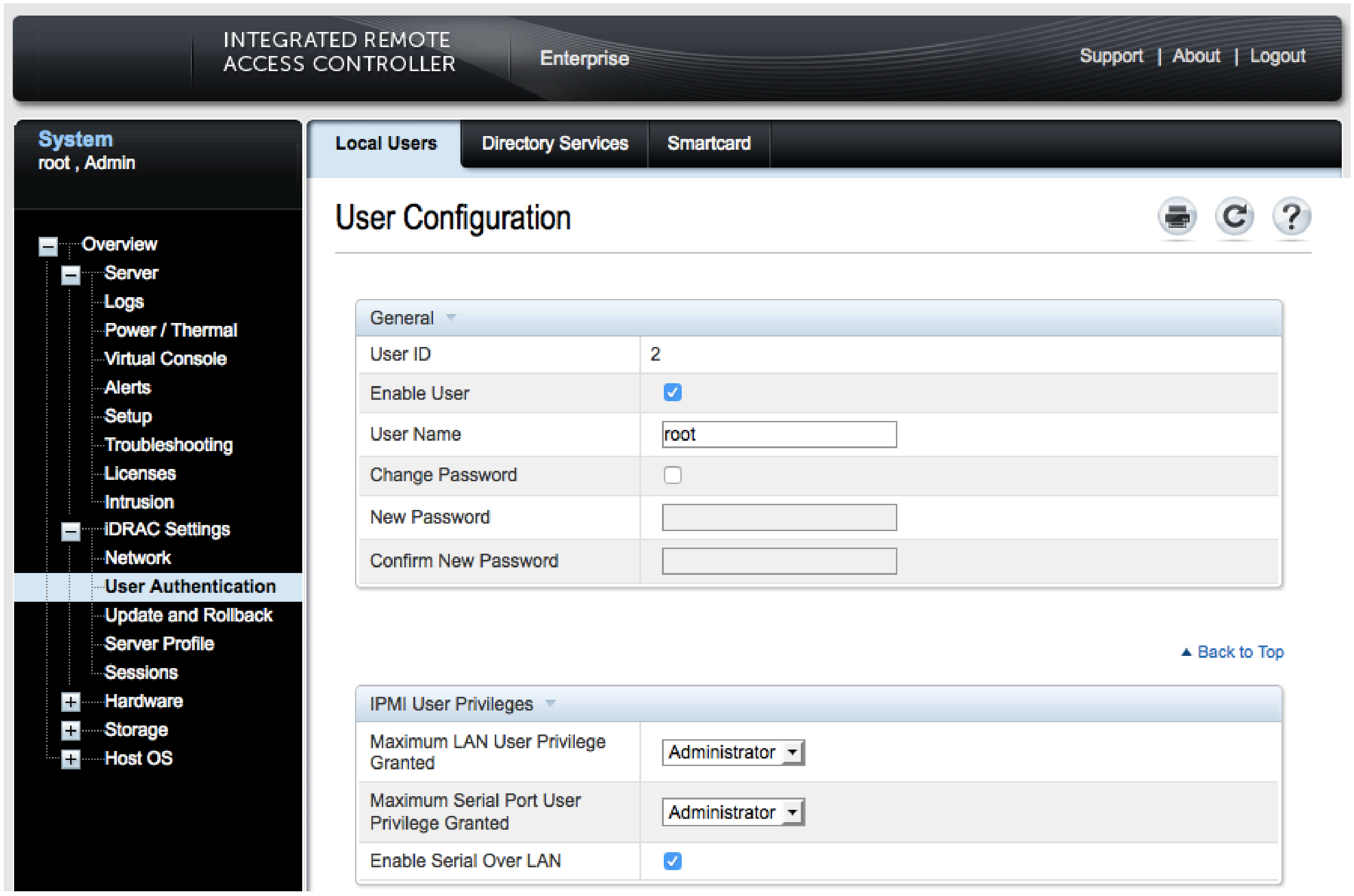
Instructions: To configure a user, click on the User ID number.

			Privilege				
User ID	State	User Name	iDRAC	LAN	Serial Port	Serial Over LAN	SNMP v3
1	Disabled	anonymous	None	None	None	Disabled	Disabled
2	Enabled	root	Administrator	Administrator	Administrator	Enabled	Disabled
3	Disabled		None	None	None	Disabled	Disabled
4	Disabled		None	None	None	Disabled	Disabled
5	Disabled		None	None	None	Disabled	Disabled
6	Disabled		None	None	None	Disabled	Disabled
7	Disabled		None	None	None	Disabled	Disabled
8	Disabled		None	None	None	Disabled	Disabled
9	Disabled		None	None	None	Disabled	Disabled
10	Disabled		None	None	None	Disabled	Disabled
11	Disabled		None	None	None	Disabled	Disabled
12	Disabled		None	None	None	Disabled	Disabled
13	Disabled		None	None	None	Disabled	Disabled
14	Disabled		None	None	None	Disabled	Disabled
15	Disabled		None	None	None	Disabled	Disabled
16	Disabled		None	None	None	Disabled	Disabled

8. Select the **Next** button on the next interface

9. Select the **Change Password** check-box on the **User Configuration** interface.

Figure 40. User Configuration Interface



User Configuration

General

User ID	2
Enable User	✓
User Name	root
Change Password	☐
New Password	
Confirm New Password	

IPMI User Privileges

Maximum LAN User Privilege Granted	Administrator
Maximum Serial Port User Privilege Granted	Administrator
Enable Serial Over LAN	✓

10. Enter the new password in the **New Password** and **Confirm New Password** fields.

11. Select **Apply** to complete the password change.

7.9.4 Change the Default System Management Workstation (SMW) Passwords

Prerequisites

Ensure that the SMW is accessible. This procedure requires root access.

About this task

After logging on to the SMW for the first time, Cray recommends changing the default passwords, as described in the following instructions.

Procedure

1. Log in to SMW as root.

2. Change default passwords on the SMW by executing the following commands.

   ```
   smw# passwd root
   ```

   ```
   smw# passwd crayadm
   ```

   ```
   smw# passwd mysql
   ```

3. Update the password in the same manner on all the nodes.

For rack-mount SMWs, such as that used in the Urika-GX system, it is also necessary to change the default iDRAC password.

7.9.5 Change LDAP Password on Urika-GX

Prerequisites

This procedure requires root privileges.

About this task

Follow the instructions in this procedure to change the LDAP password on Urika-GX

Procedure

1. Log on to `nid00030`, which is a login node.

2. Edit the `slapd.conf` file and add in ACLs to allow users to modify the LDAP password.

   ```
   [root@nid00003~]# vim /usr/local/openldap/etc/openldap/slaps.conf
   # Define global ACLs to disable default read access.
   access to *
    by self write
    by * read
    by anonymous auth
   ```

3. Restart the `slapd` daemon, which listens for LDAP connections.

   ```
   [root@nid00003~]# systemctl restart slapd
   ```

4. Login as `crayusr` to test changing the password by entering it when prompted.

   ```
   [crayusr@nid00003~]# ldappasswd -H ldap://127.0.0.1 -x -D \
   "uid=crayusr,ou=users,dc=urika,dc=com" \
   -w old_password -a new_password -S
   New password: Re-enter new
   password:
   ```

7.9.6 Reset a Forgotten Password for the Cray Application Management UI

Prerequisites

This procedure requires root privileges and assumes that it is being carried out on a 48-node system. The node IDs should be replaced with actual ones to reflect actual system configuration.

About this task

Follow the instructions in this procedure if the password has been forgotten and needs to be reset.

Procedure

1. Log on to login node 1.

   ```
   # ssh nid00030
   ```

2. Switch to the `/opt/cray/ui-application-management/default` directory.

   ```
   # cd /opt/cray/ui-application-management/default
   ```

3. Run the `./manage.py` script to reset the password and enter the new password when prompted.

   ```
   # ./manage.py changepassword admin
   Changing password for user 'admin'
   Password:
   ```

7.9.7 Reset an Administrator LDAP Password on Systems Using Urika-GX 1.2UP01 and Earlier Releases

Prerequisites

This procedure:

- requires root privileges.
- assumes that the system is running Urika-GX 1.2UP01 or an earlier release.
- assumes that the LDAP host server is running on nid00030, which is login node 1 on a 48 node system.

About this task

This procedure provides instructions for updating the LDAP admin server password for the root domain name, i.e., `cn=crayadm,dc=urika,dc=com`. This can be used to reset the password in case it is forgotten.

This procedure contains instructions for systems that have their LDAP running in non-OLC mode. For such systems, the `/etc/default/slapd.conf` file (located on login node 1, which is the LDAP server host) does not contain the following entry:

```
SLAPD_CONF_DIR="/usr/local/openldap/etc/openldap/slapd.d"
```

Procedure

1. Log on to the LDAP host server as root.

2. Generate a new hashed password.

   ```
   # slappasswd
   New password:
   Re-enter new password:
   {SSHA}ZNYj4jyMpTo3xfln0lxpirj0ZyuKVa24
   ```

3. Make a back up copy of the `/usr/local/openldap/etc/openldap/slapd.conf` file.

4. Edit the `/usr/local/openldap/etc/openldap/slapd.conf` file, replacing values for the two '`rootpw`' entries with the hashed password output of the `slappasswd` command.

   ```
   #local database urika
   database        bdb
   idlcachesize 50000
   suffix         "dc=urika,dc=com"
   rootdn         "cn=crayadm,dc=urika,dc=com"
   rootpw         {SSHA}ZNYj4jyMpTo3xfln0lxpirj0ZyuKVa24
   cachesize 50000
   dirtyread
   dbnosync
   checkpoint 128 15
   idlcachesize 50000
   index objectClass eq

   #database meta - COMBINES the LDAP DATABASES
   database meta
   suffix  "dc=local"
   rootdn  "cn=crayadm,dc=local"
   rootpw  {SSHA}ZNYj4jyMpTo3xfln0lxpirj0ZyuKVa24
   ```

5. Restart the `slapd` service.

   ```
   # service slapd restart
   ```

6. Verify that the password has been reset by executing the `ldap` command and using the new password for `cn=crayadm,dc=urika,dc=com`, provided to the `slappasswd` command above.

   ```
   # ldapsearch -D "cn=crayadm,dc=urika,dc=com" -W -p 389 -h localhost -b "ou=groups,dc=urika,dc=com" -t cn
   Enter LDAP Password: (enter new password here)
   ```

7.9.8 Reset an Administrator LDAP Password when the OLC Schema Password is Unknown

Prerequisites

- This procedure requires root privileges.

- This procedure requires the LDAP server (slapd) to be listening at "`ldapi://`" (socket) and the appropriate permission to be set in the LDAP schema to allow for roots `uid` and `gid` to perform the necessary `ldif` operations
- This procedure assumes that the LDAP host server is running on nid00030, which is the login node on a 48 node system.
- This procedure requires Urika-GX 2.0UP00 or latter installed on the system.

About this task

This procedure contains steps for resting a forgotten LDAP administrator password in case the OLC scheme is not known. This procedure provides instructions for updating the LDAP admin server password for the root domain name, i.e., `cn=crayadm,dc=urika,dc=com`.

This procedure contains instructions for systems that have their LDAP running in OLC mode. For such systems, the `/etc/default/slapd.conf` file contains the following entry:

```
SLAPD_CONF_DIR="/usr/local/openldap/etc/openldap/slapd.d"
```

If the LDAP passwords for "`cn=admin,cn=config`" and "`cn=crayadm,dc=urika,dc=com`" are not known, it is possible to change either or both as root logged into the LDAP server host with no LDAP password(s) but this requires the LDAP server (slapd) to be listening at "`ldapi://`" (socket) and the appropriate permission to be set in the LDAP schema to allow for roots `uid` and `gid` to perform the necessary `ldif` operations, however this is not configured in the default Urika-GX installation, therefore, setting this up for existing systems requires knowing the OLC admin password.

Below are the configuration changes to enable `ldapi://` and allow the Linux root user access to make changes to the LDAP server without needing any LDAP passwords using `ldapi` (via `ladpmodify` and `ldif`). These require having the `cn=admin,cn=config` LDAP password for the initial setup.

Procedure

1. Log on to the LDAP host server as root.
2. Change the LDAP passwords using `ldif` over `ldapi://`, as shown in the following examples.

```
# ldapmodify -Y EXTERNAL -H ldapi://%2Fusr%2Flocal%2Fopenldap%2Fvar%2Frun%2Fldapi -f crayadmpw_bdb.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "olcDatabase={1}bdb,cn=config"
```

```
# ldapmodify -Y EXTERNAL -H ldapi://%2Fusr%2Flocal%2Fopenldap%2Fvar%2Frun%2Fldapi -f crayadmpw_meta.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "olcDatabase={2}meta,cn=config"
```

3. Edit the `global` file, located under the `/etc/sysconfig/uxtenant` directory, to configure the new password in the environment variable `UXTENANT_GLOBAL_LDAP_ROOT_PW`

   For example, if the password requested by `slappasswd` was entered as "`changeme`" then the variable would need to be set as:

```
UXTENANT_GLOBAL_LDAP_ROOT_PW="changeme"
```

7.9.9 Reset an Administrator LDAP Password when the OLC Scheme Password is Known

Prerequisites

- This procedure requires root privileges.
- The `cn=admin,cn=config` LDAP OLC schema password needs to be known while carrying out this procedure.
- This procedure requires Urika-GX 2.0UP00 or latter installed on the system.

About this task

This procedure provides instructions for updating the LDAP admin server password for the root domain name, i.e., `cn=crayadm,dc=urika,dc=com`. It can be used to reset the password in case it is forgotten.

This procedure contains instructions for systems that have their LDAP running in OLC mode. For such systems, the `/etc/default/slapd.conf` file contains the following entry:

```
SLAPD_CONF_DIR="/usr/local/openldap/etc/openldap/slapd.d"
```

Procedure

1. Log on to the LDAP host server as root.
2. Generate a new hashed password.

   ```
   [root@nid00030]# slappasswd
   New password:
   Re-enter new password:
   {SSHA}I1/QKXFN+RjLFlJwdPBtBuf7q46+2dmX
   ```

3. Create the `crayadmpw_bdb.ldif` file with the newly hashed password.

   ```
   dn: olcDatabase={1}bdb,cn=config
   changetype: modify
   replace: olcRootPW
   olcRootPW: {SSHA}I1/QKXFN+RjLFlJwdPBtBuf7q46+2dmX
   ```

4. Create the `crayadmpw_meta.ldif` file with the newly hashed password.

   ```
   dn: olcDatabase={2}meta,cn=config
   changetype: modify
   replace: olcRootPW
   olcRootPW: {SSHA}I1/QKXFN+RjLFlJwdPBtBuf7q46+2dmX
   ```

5. Execute the following commands and enter the `cn=admin,cn=config` password when prompted for the LDAP password.

   ```
   # ldapmodify -x -W -D "cn=admin,cn=config" -H ldap://localhost -f crayadmpw_bdb.ldif
   Enter LDAP Password:
   modifying entry "olcDatabase={1}bdb,cn=config"
   ```

6. Execute the following commands and enter the `cn=admin,cn=config` password when prompted for the LDAP password.

```
# ldapmodify -x -W -D "cn=admin,cn=config" -H ldap://localhost -f crayadmpw_meta.ldif
Enter LDAP Password: (enter the cn=admin,cn=config password)
modifying entry "olcDatabase={2}meta,cn=config"
```

The LDAP password for "`cn=crayadm,dc=urika,dc=com`" will now be reset to the new password that was provided to `slappasswd`.

7. Log on to the SMW as root.
8. Edit the `global` file, located under the `/etc/sysconfig/uxtenant` directory, to configure the new password in the environment variable `UXTENANT_GLOBAL_LDAP_ROOT_PW`

   For example, if the password requested by `slappasswd` was entered as "`changeme`" then the variable would need to be set as:

   ```
   UXTENANT_GLOBAL_LDAP_ROOT_PW="changeme"
   ```

7.10 Tableau Authorization and Authentication Mechanisms

Urika-GX ships with LDAP authentication enabled for Spark Thrift Server.

CAUTION: Enabling LDAP authentication for HiveServer2 may result in misconfiguration of HUE. To resolve this issue, please follow the instructions documented at *http://gethue.com*.

SSL authentication for Tableau can be set up using instructions documented in Enable SSL on Urika-GX. Urika-GX ships without authorization enabled. To enable storage based authorization for connecting to HiveServer2, follow the instructions documented by Hive, visit *https://cwiki.apache.org*

To learn more about using Tableau, visit *http://www.tableau.com/*.

7.11 Enable SSL

Prerequisites

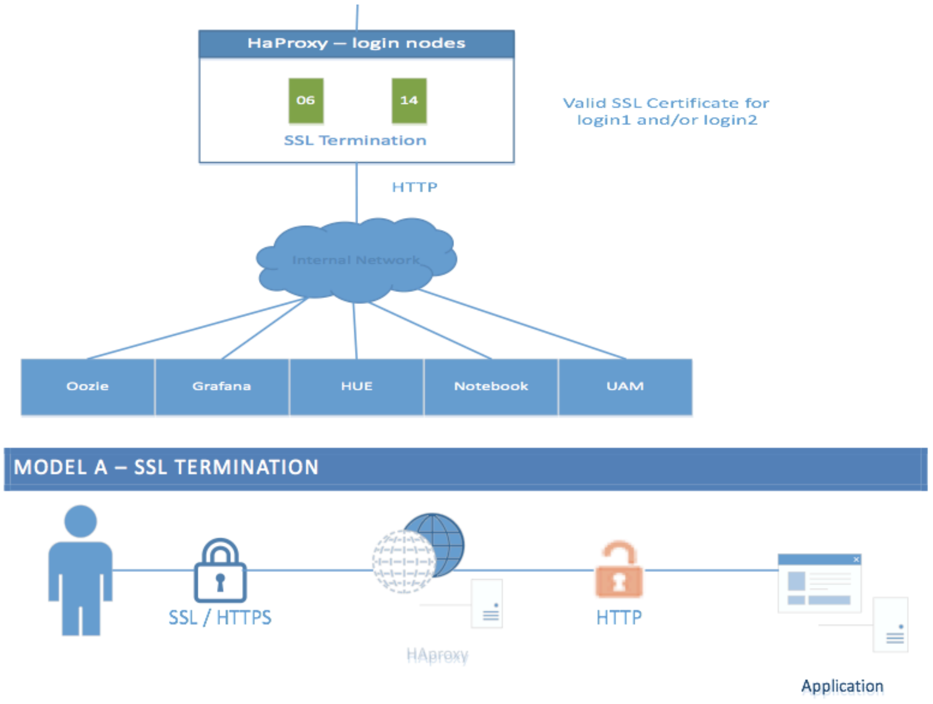
This procedure requires root privileges and needs the host name of the machine to be known. The host name should resolve to the site's DNS.

About this task

Users access applications by using the login1 node's `https://IP:PORT` using SSL. HA Proxy decrypts SSL and then forwards the HTTP communication to back end servers over the internal Aries or management network. Back-end applications send their communication back to HA Proxy via HTTP. HA Proxy then encrypts the response and send it back to the user using SSL. HA or redundant models are also supported so it is possible to multiple back-ends available, as long as the data is in sync.

Urika-GX's SSL setup is depicted in the following figure:

Figure 41. Urika-GX SSL setup



Information sent over the Internet is passed through various servers before it reaches the destination. Any computer located between the source and destination can see user names, passwords, and other sensitive information if it is not encrypted with an SSL certificate. Therefore, it is recommended to enable SSL, as described in this procedure.

Major steps involved in installing SSL include:

1. Acquire and install a valid SSL certificate.
2. Edit the HA Proxy configuration file by uncommenting the settings to enable SSL.
3. Edit the settings for the **Urika Applications Interface** by uncommenting some settings to enable SSL.

In the following instructions, it is assumed that the SSL certificate is being installed on Urika-GX system containing 48 nodes and `nid00030` is used as the node ID of login node 1.

CAUTION: SSL is only supported with certificates issued by a trusted Certificate Authority(CA), not with self-signed certificates

In the following instructions and code samples, *`hostname`* is used as an example and should be replaced with the actual host name of the system. This procedure assumes that it is being carried out on a 48 node system.

Procedure

1. SSH into login node 1.

```
# ssh nid00030
```

2. Uncomment the following lines in `/etc/haproxy/haproxy.cfg` configuration file and replace all occurrences of `/etc/ssl/certs/filename.pem` with the full path to the SSL certificate.

```
frontend hive_ssl
  bind *:29207 ssl crt /etc/hive/conf/server.pem
  mode tcp
  option forwardfor
  reqadd X-Forwarded-Proto:\ https
  default_backend hive_ssl_backend

backend hive_ssl_backend
  mode tcp
  balance source
  server server1 192.168.0.33:10000

frontend sparkthrift_ssl
  bind *:29208 ssl crt /etc/hive/conf/server.pem
  mode tcp
  option forwardfor
  reqadd X-Forwarded-Proto:\ https
  default_backend sparkthrift_ssl_backend

backend sparkthrift_ssl_backend
  mode tcp
  balance source
  server server1 192.168.0.33:10015

frontend grafana_ssl
  bind *:29201 ssl crt /etc/ssl/certs/filename.pem
  mode http
  option forwardfor
  option http-server-close
  option httpclose
  reqadd X-Forwarded-Proto:\ https
  default_backend grafana_ssl_backend

backend grafana_ssl_backend
  mode http
  balance source
  server server1 192.168.0.47:3000

frontend hue_ssl
  bind *:29202 ssl crt /etc/ssl/certs/filename.pem
  mode http
  option forwardfor
  option http-server-close
  option httpclose
  reqadd X-Forwarded-Proto:\ https
  reqadd X-Forwarded-Protocol:\ https
  default_backend hue_ssl_backend

backend hue_ssl_backend
  mode http
  balance source
  server server3  192.168.0.31:8888

frontend jupyter_ssl
  bind *:29204 ssl crt /etc/ssl/certs/filename.pem
  mode http
  option forwardfor
  option http-server-close
  option httpclose
  reqadd X-Forwarded-Proto:\ https
  default_backend jupyter_ssl_backend

backend jupyter_ssl_backend
  mode http
  balance source
  server server4  192.168.0.31:7800

frontend urika-app-magmt_ssl
  bind *:29203 ssl crt /etc/ssl/certs/filename.pem
  mode http
  option forwardfor
  option http-server-close
  option httpclose
  reqadd X-Forwarded-Proto:\ https
  reqadd X-Forwarded-Protocol:\ https
  default_backend urika-app-magmt_ssl_backend

backend urika-app-magmt_ssl_backend
  mode http
  balance source
  server server1  192.168.0.31:8000
```

In the above file, `192.168.0.31` is the IP address of the node where HUE is running. 8888 is the port where HUE is running. If SSL is enabled, the HUE UI would be available at `https://`*`hostname`*`-login1:29202`. HUE would still be available at `http://`*`hostname`*`-login1:8888`, but this URL not secure. It is recommended to use `https://`*`hostname`*`-login1:29202`.

The preceding example is specific to a 48 node system. The IP address should be used according to the system configuration. The following list of IP addresses can be used as a reference:

```
hive_ssl_backend
16 node: 192.168.0.3
32 node: 192.168.0.9
48 node: 192.168.0.33
sparkthrift_ssl_backend
16 node: 192.168.0.1
32 node: 192.168.0.9
48 node: 192.168.0.33
grafana_ssl_backend
16 node: 192.168.0.15
32 node: 192.168.0.31
48 node: 192.168.0.47
hue_ssl_backend
16 node: 192.168.0.7
32 node: 192.168.0.15
48 node: 192.168.0.31
jupyter_ssl_backend
16 node: 192.168.0.7
32 node: 192.168.0.15
48 node: 192.168.0.31
csms_ssl_backend
urika-app-mgmt_ssl_backend
16 node: 192.168.0.7
32 node: 192.168.0.15
48 node: 192.168.0.31
```

NOTE: CSMS is not used in post Urika-GX 1.2UP00 releases.

3. Save the `/etc/haproxy/haproxy.cfg` configuration file.

4. Edit the `/etc/httpd/conf.d/ssl.conf` file to replace the path in following line with the path to the actual `pem` file.

```
SSLCertificateFile /etc/ssl/certs/filename.pem
```

```
SSLCertificateFile /etc/pki/tls/certs/localhost.crt
```

5. Restart HAProxy

```
# service haproxy restart
```

6. Uncomment the following lines and make sure they are set to `true` in `/etc/hue/conf/hue.ini`

```
use_x_forwarded_host=true
secure_proxy_ssl_header=true
```

7. Restart the HUE service.

```
# service urika-hue restart
```

8. Uncomment the following lines (or add if they do not already exist) in `/opt/cray/ui-application-management/default/application_management/settings.py`

```
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
USE_X_FORWADED_HOST = True
```

9. Set USE_SECURE_URLS to `True` in `/opt/cray/ui-application-management/default/application_management/settings.py`, as shown in the following example. This allows the **Urika-GX Applications Interface** page to load secure URLs (configured in the preceding steps) when the HUE/Grafana/Jupyter/Urika-GX Application Management UIs are accessed from the **Urika-GX Applications Interface** page. If there is any change in the HAProxy port numbers, the following URLs in `settings.py` need to be updated:

```
USE_SECURE_URLS = False
USE_SECURE_URLS == True:
GRAFANA_SERVER = "https://" + LOGIN1  + ":29201"
JUPYTER = "https://" +  LOGIN1 + ":29204"
HUE = "https://" +  LOGIN1 + ":29202"
UAM = "https://" + LOGIN1 + ":29203/applications"
```

10. Update the `/etc/httpd/conf.d/uam.conf` configuration file as follows, replacing the URL '`hostname-login1.us.cray.com`' with the FQDN of login node 1.

```
WSGISocketPrefix run/wsgi
WSGIScriptAlias / /opt/cray/ui-application-management/default/
application_management/apache/wsgi.py
Alias /static/ /opt/cray/ui-application-management/default/
application_management/app_monitoring/static/
<Directory /opt/cray/ui-application-management/default/application_management/
apache>
  Require all granted
</Directory>
```

```
<VirtualHost *:80>
  Redirect  /applications https://hostname-login1.us.cray.com:29203/
applications
</VirtualHost>
Listen 8000
<VirtualHost *:8000>
</VirtualHost>

<Directory /opt/cray/ui-application-management/default/application_management/
app_monitoring/static>
  Require all granted
</Directory>
```

11. Restart Apache on login node 1.

    ```
    # service httpd restart
    ```

12. SSH on to the SMW.

    hostname is used in the following example as the machine name.

    ```
    # ssh hostname-smw
    ```

13. Restart Apache on the SMW.

    ```
    # service httpd restart
    ```

14. Verify that all the URLs of services are accessible.

7.12 Install a Trusted SSL Certificate on Urika-GX

Prerequisites

This procedure needs to be performed as root. The hostname needs to be known and needs to resolve to the site's DNS.

About this task

This procedure provides instructions for installing a SSL certificate that has been issued by a trusted Certificate Authority (CA). In the following instructions, `nid00030` is used as an example for login node 1's ID. Replace '*hostname*`-login1.us.cray.com`' in the following examples with the FQDN of login node 1.

Procedure

1. Log on to login node 1 as root.

2. Generate a key. Following is an example for generating a key using openssl

    ```
    [root@nid00030~]# openssl genrsa -out hostname-login1.us.cray.com.key 2048
    ```

3. Generate a Certificate Signing Request (CSR). The system will prompt to enter information that will be incorporated into the certificate request.

```
[root@nid00030~]#  openssl req -new-key hostname-login1.us.cray.com.key -out
 hostname-login1.us.cray.com.csr
```

4. Store the CSR in a secure location on the system, such as `/opt/cray/certificate`

5. Send the CSR to the IT department to obtain a certifcate

6. Create a `PEM` file, which is the `crt + key` combined into 1 file.

   WARNING: The `PEM` file needs a linefeed bewteen crt and key. Use a vi editor or the `cat` command to combine the crt and key

   ```
   [root@nid00030~]# cat server.crt server.key > server.pem
   ```

7. Save the PEM file as `hostname-login1.us.cray.com.pem` under `/opt/cray/certificate`.

7.13 Enable LDAP Authentication on Urika-GX

Prerequisites

- This procedure requires root access.
- Ensure that the storage LDAP client points at login node 1, which is the LDAP server on Urika-GX. This ensures that the Urika-GX system and storage are authenticating to the same source.

  NOTE: This examples used in this procedure are intended for a 3 sub-rack system. Replace node IDs as needed when executing the following commands if using a system containing less than 3 sub-racks.

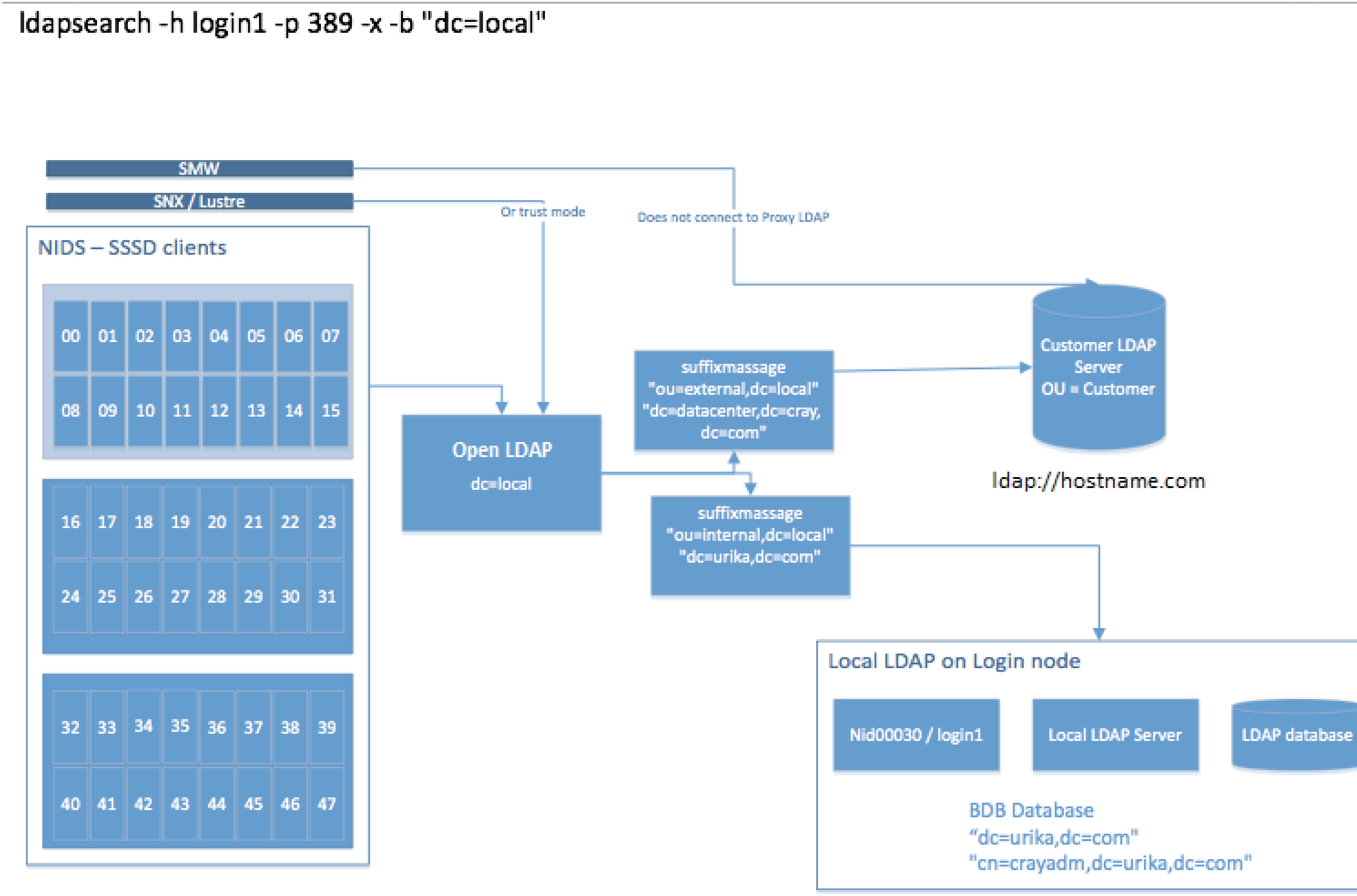
About this task

Urika-GX uses an internal LDAP server that comes with two basic user types:

- crayusr – This is the standard user type
- crayadm – Users of this type have root/admin privileges.

The Urika-GX LDAP setup is depicted in the following figure:

Figure 42. Urika-GX LDAP Setup



The following procedure can be used to enable the LDAP server for every Urika®-GX node. In the following instructions, *`login-1`* is used as an example for the name of login node 1, which is where the LDAP service runs.

CAUTION: Cray Support should be involved for all LDAP related procedures and changes.

Procedure

1. Log on to login node 1 as root.

```
# ssh root@login-1
```

2. Edit the `/usr/local/openldap/etc/openldap/slapd.conf` file to uncomment the following lines, and replace `[myldap]` with the actual LDAP server's URL:

```
[root@login-1 ~]# vi /usr/local/openldap/etc/openldap/slapd.conf #uri"ldap://[myldap]/ou=external,dc=local"
#lastmod off
#suffixmassage"ou=external,dc=local" "[dc=datacenter,dc=cray,dc=com]"
```

NOTE: The square brackets, i.e. `[ ]` need to be omitted when replacing `[myldap]` with the actual LDAP server's URL.

For example, if the LDAP server's URL is `hostname.com`, the line:

```
# uri"ldap://[myldap]/ou=external,dc=local"
```

should be uncommented and customized to look like the following:

```
uri"ldap://hostname.com/ou=external,dc=local"
```

3. Save the `/usr/local/openldap/etc/openldap/slapd.conf` file.
4. Restart the `slapd` daemon, which listens for LDAP connections.

```
[root@login-1 ~]# service slapd restart
```

5. Restart the `oddjobd` daemon, which provides a means for invoking and taking limited control of applications.

```
[root@login-1 ~]# service oddjobd restart
```

6. Ensure that LDAP has been set up properly
   a. Log on to the System Management Workstation (SMW) as root.
   b. Verify that the users registered with LDAP are listed on all the nodes.

      In the following example, *`system-smw`* is used as an example for the SMW's hostname and *`username`* is used as an example of an LDAP user's username.

```
# root@system-smw pdsh -w nid000[00-47] 'id username'
```

   c. Log on to login node 1.

```
# ssh root@login-1
```

   d. Verify that all LDAP users are listed when the `ldapsearch` command is executed for all entries:

```
[root@nid00030 ~]# ldapsearch -h 127.0.0.1 -D "cn=crayadm,dc=local" -w initial0 -b "dc=local"
```

For information about internal LDAP or the LDAP server, contact Cray Support. For advanced configuration settings, see the OpenLDAP Software Administrator's guide at *http://www.openldap.org*.

7.13.1 Enable LDAP for Connecting Tableau to HiveServer2

Prerequisites

This procedure requires root privileges.

About this task

This procedure provides instructions for enabling LDAP for connecting HiveServer2 to Tableau or other external clients.

Procedure

1. Log on to the SMW.

2. Stop the HiveServer2 service.

```
# urika-stop -s hive
```

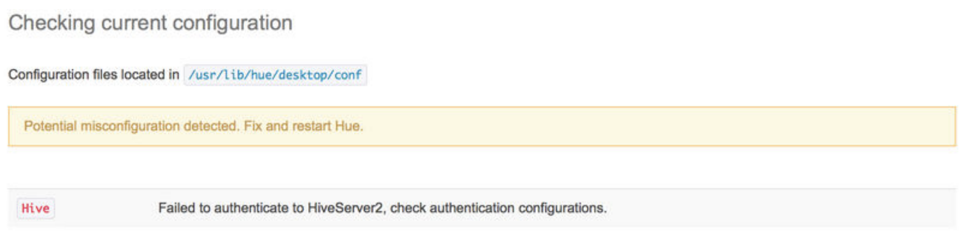
3. Connect to a node where the HiveServer2 service is running.

4. Edit the `/etc/hive/conf/hive-site.xml` file and change the authentication mode from `NONE` to `LDAP`

```
<property>
 <name>hive.server2.authentication</name>
 <value>NONE</value>
</property>
```

CAUTION: Enabling LDAP authentication for HiveServer2 may result in misconfiguration of HUE, as shown in the following figure. To resolve this issue, please follow the instructions documented at *http://gethue.com*.

Figure 43. Error Message


Checking current configuration

Configuration files located in `/usr/lib/hue/desktop/conf`

Potential misconfiguration detected. Fix and restart Hue.

Hive Failed to authenticate to HiveServer2, check authentication configurations.

5. Log on to the SMW.

6. Restart HiveServer2 service.

```
# urika-start -s hive
```

7.14 Enable SQL Standard based Authorization for HiveServer2

Prerequisites

This procedure requires root privileges.

About this task

SQL standard based authentication for connecting to HiverServer2 is not enabled by default on Urika-GX. Follow the instructions in this procedure to enable it.

Procedure

1. Log on to the SMW as root.

2. Stop the Hive service.

   ```
   # urika-stop -s hive
   ```

3. Follow the Hive instructions documented at *https://cwiki.apache.org/confluence/display/Hive/SQL+Standard+Based+Hive+Authorization#SQLStandardBasedHiveAuthorization-ForHive0.14andNewer* with the following exceptions:

   1. Edit the `hive-site.xml` file and set the value of `hive.metastore.uris` to `thrift://192.168.0.33:9083`
   2. Use either or both of the values defined in the Hive documentation for the `hive.security.authorization.manager` configuration parameter.
   3. Edit `hive-site.xml` to set values of properties, as opposed to the Hive documentation, which instructs users to set these properties in `hiveserver2-site.xml`

      NOTE: Even though the documentation at *https://cwiki.apache.org/confluence/display/Hive/SQL+Standard+Based+Hive+Authorization#SQLStandardBasedHiveAuthorization-ForHive0.14andNewer* instructs users to put some settings into `hiveserver2-site.xml`, all of the properties listed in the linked documentation should go in `hive-site.xml`.

   There is no need to modify other parameters of the `hive-site.xml` file that are documented in the Hive documentation, as they are already configured on the Urika-GX system.

4. Start the Hive service.

   ```
   # urika-start -s hive
   ```

5. Reconnect to HiveServer2.

7.15 File System Permissions

This section provides details on user restrictions and access controls implemented on Urika-GX for the three Urika-GX file systems.

- **Internal file systems**
  - Hadoop Distributed File System (HDFS) - Apache Hadoop HDFS implements authentication and authorization model similar to POSIX model, where each file is associated with an owner and a group. Urika-GX implements a simple mode of user authentication. In this mode, the identify of a client process is determined by the OS's user name, which can be retrieved via the `whoami` command. The user authentication is managed by Open LDAP server running on login node 1. Groups and user credential are inherited from the Open LDAP.
  - Network File System (NFS) - The SMW provides the NFS mount, which has the same groups and user permissions as those of the host operating system.
- **External file system (Lustre)** - If Lustre is used on Urika-GX, it inherits Linux group permissions from the login node.

7.16 Urika-GX Security Quick Reference Information

Configuration of LDAP Settings on Urika-GX

The Open LDAP server on Urika-GX runs on login node 1. For example, if the system's hostname is `hostnatme`, then the Open LDAP service would run on `hostname-login1`. The particular ID of the login node (such as `nid00030` on a 48 node system) would change, based on the number of nodes in the system i.e. 16N, 32N or 48N. The Open LDAP server and can be connected with the site corporate LDAP server from which, groups, permissions and user credentials will be inherited. Please refer to the site specific LDAP server documentation for details on setup and configuration.

LDAP and User management Information

For details of how to add, enable and manage users on LDAP please refer to *http://www.openldap.org*.

- **Configure LDAP settings** - The Open LDAP server on Urika-GX can be connected with the site/corporate LDAP server, from which groups, permissions, and user credentials will be inherited.
- **Open LDAP service node** - The Open LDAP service runs on login node 1 of Urika-GX system. For example, if the Urika-GX hostname is *`hostname`* then Open LDAP service is running on *`hostname`*-login1. The particular node (such as: nid00030) would change, based on the number of nodes in the system i.e. 16N, 32N or 48N
- **Authenticate users** - Urika-GX uses LDAP based authentication. System administrators can use LDAP servers to setup user authentication.
- **Add a user to a group** - Use the `usermod` command to add a user to an existing group.
- **Add a user to LDAP** - Use the `ldapadd` command to add a user to LDAP.
- **Create a group on each node** - In order to enable group permissions in HDFS, the changes on the Linux system need to take place on the server that is running the NameNode service. If it is unclear which node is running the NameNode service, run the `urika-inventory` command from the SMW. For consistency, it is recommended to set up group(s) from the SMW on all nodes via `pdsh`. Use the `groupadd` command to create a group on each node and the `hdfs groups` command to confirm that HDFS recognizes the new group.
- **Change user passwords** - Use the `ldappasswd` command to change user passwords.
- **Create HDFS home directory** - Use the `hdfs dfs -mkdir` command to create a HDFS home directory.
- **Change user permissions** - Use the `hadoop dfs -chown` command to change permissions on directory to make new user the owner of home directory

The user authorization list uses LDAP. If this list is moved to a different location, the following items will need to be modified to point to the new location:

- Tenancy related configuration file, located at `/etc/sysconfig/uxtenant/global` on the SMW

  Urika-GX tenant proxy PAM modules, located at `/etc/utp.d/pam_cray_utp.conf` on all the physical nodes and tenant VM

For more information, refer to *Authentication and Authorization* on page 171

7.17 Port Assignments

Table 30. Services Running on the System Management Workstation (SMW)

Service Name	Default Port
SSH	22

Table 31. Services Running on the I/O Nodes

Service Name	Default Port
SSH	22

Table 32. Services Running on the Compute Nodes

Service Name	Default Port
ssh	22
YARN Node Managers	8040, 8042, 45454, and 13562
Mesos slaves on all compute nodes	5051
DataNode Web UI to access the status, logs and other information	50075
DataNode use for data transfers	50010
DataNode used for metadata operations.	8010

Table 33. Services Accessible via the Login Nodes via the Hostname

Service	Default Port
Mesos Master UI	5050. This UI is user-visible.
Spark History Server's web UI	18080. This UI is user-visible.
HDFS NameNode UI for viewing health information	50070. This UI is user-visible.
Secondary NameNode web UI	50090. This UI is user-visible.
Web UI for Hadoop Application Timeline Server	8188. This UI is user-visible.
YARN Resource Manager web UI	8088. This UI is user-visible.
Marathon web UI	8080. This UI is user-visible.
Hive Server2	SSL - 29207

Service	Default Port
	Non-SSL - 10000
Hive Metastore	9083.
Hive WebHCat	50111.
Oozie server	11000. The Oozie dashboard UI runs on this port and is user-visible.
Hadoop Job History Server	19888 on nid00016. This is a user-visible web UI.
HUE server	8888 on login1 and login2. The web UI for the HUE dashboard runs on this port and is user-visible.
CGE `cge-launch` command	3750. See S-3010, "*Cray® Graph Engine Users Guide*" for more information about the `cge-launch` command or see the `cge-launch` man page.
CGE Web UI and SPARQL endpoints	3756
Spark Web UI	4040. This port is valid only when a Spark job is running. If the port is already in use, the port number's value is incremented until an open port is found. Spark Web UI runs on whichever login node (1 or 2) that the user executes `spark-submit/spark-shell/spark-sql/pyspark` on. This UI is user-visible.
InfluxDB	8086 on login2. InfluxDB runs on nid00046 on three sub-rack, and on nid00030 on a two sub-rack system.
InfluxDB port for listening for `collectl` daemons on compute nodes	2003. InfluxDB runs on login node 2 on the Urika-GX system.
InfluxDB cluster communication	8084
Grafana	3000 on login2 (login node 2). The Grafana UI is a user-visible.
Web UI for Jupyter Notebook	7800. Jupyter Notebook internally uses HTTP proxy, which listens to ports `7881` and `7882`
Urika-GX Applications Interface	80 on login1 (login node 1).
Urika-GX Application Management	80 on login1 (login node 1).
Spark SQL Thrift Server	SSL - 29208 Non-SSL - 10015
Kubernetes Master	6443 on login1 (login node 1).

Additional Ports and Services

Table 34. Additional Services and Ports They Run on

Service	Port
Apache ZooKeeper	2181

Service	Port
Kafka (not configured by default)	9092
Flume (not configured by default)	41414
Port for SSH	22

8 Troubleshooting

8.1 System Management Log File Locations

Table 35. System Log File Locations

Application/Service	Log File Location	Additional Notes
Logs of the dual Aries Network Card (dANC) `syslog` and Hardware Supervisory System (HSS) daemons	Located on the SMW at `/var/opt/cray/log/controller/`*`dancname`*	`syslogs`, forwarded by LLM
Logs of the Rack Controller (RC) `syslog` and HSS	Located on the SMW at `/var/opt/cray/log/controller/`*`rackname`*	`syslogs`, forwarded by LLM
Logs of SMW HSS monitor daemons	Located at `var/opt/cray/log/p0-default` on the SMW. The following log files are stored in this directory: ● `console-`*`date`* - Contains node console logs ● `consumer-`*`date`* - Contains ERD event collector (`xtconsumer`) logs ● `netwatch-`*`date`* - Contains logs generated by the `xtnetwatch` command ● `nlrd-`*`date`* - Contains logs generated by the `xtnlrd` command ● `pcimon-`*`date`* - Contains PCIe monitor logs ● hwerrlogd-*`date`* -Contains logs generated by the `hwerrlogd` command In the above list, *`date`* is the date stamp with a format of *`yyyymmdd`*.	All HSS daemons and processes log here, except `xtremoted` and `erd`. Logs within this directory are rotated daily.
SMW `syslogs` logs	Located on the SMW at `/var/opt/cray/log/`*`smwmessages-datestamp`*	SMW `syslogs`, rotated daily
Logs generated erd/ erdh event logs	Located on the SMW at `/var/opt/cray/log/events-`*`datestamp`*	`capmc` remote agent. Logs a level up from HSS daemons.

Application/Service	**Log File Location**	**Additional Notes**
		These logs are rotated daily.
Logs generated by the State Manager, ERD, ERFS, NID mgr, `xtremoted` stdout/ stderr	Located on the SMW at `/var/opt/cray/log/`*`daemon-datestamp`*	State Manager, ERD, ERFS, NID mgr,`xtremoted` stdout/stderr. Rotated daily.
HSS command logs	Located on the SMW at `/var/opt/cray/log/commands`	Log of all HSS commands executed, with time stamps, exit status, etc. Rotated daily.
`rsyslog` logs	Located on the SMW at `/var/opt/cray/log/logsystem-`*`datestamp`*	Log of `rsyslog` actions. Rotated daily.
`xtdiscover` logs	Located on the SMW at `/var/opt/cray/log/xtdiscover/log.`*`datestamp`*	`xtdiscover` log files. Rotated daily, or when `xtdiscover` is run, whichever is less often.
Nagios	Default location of logs is `/var/log/nagios/nagios.log` on the SMW. This location can be configured if desired	N/A
Cobbler	Located in the Cobbler KVM at `/var/log/cobbler`	N/A
Grafana	Located at `/var/log/grafana/grafana.log` on login node 2	N/A
Tenant proxy logs	`utp-server` logs are located on the host node under `/var/log/utp/`	N/A
Urika-GX security manager logs	The default log file for the `usm-sync-users` and `usm-recreate-secret` commands is located at `/var/log/usm/urika-secret-manager.log`	N/A
Kubernetes	Logging for individual pods/jobs exists in the associated containers.	N/A

8.2 Default Log Settings

The following table lists the default log levels of various Urika-GX analytic components. If a restart of the service is needed, please first stop services using the `urika-stop` command, change the log level, and then restart services using the `urika-start` command.

Table 36. Default Log Levels

Component	Default Log Level	Restarting service required after changing log level?
Spark	Default log levels are controlled by the `/opt/cray/spark/default/conf/log4j.properties` file. Default Spark settings are used when the system is installed, but can be customized by creating a new `log4j.properties` file. A template for this can be found in the `log4j.properties.template` file.	No
Hadoop	Default log levels are controlled by the `log4j.properties` file. Default Hadoop settings are used when the system is installed, but can be customized by editing the `log4j.properties` file.	Yes
Mesos	`INFO`	Yes
Marathon	`INFO`. Log levels can be modified by editing the `log4j.properties` file.	Yes
Grafana	`INFO`. Log properties can be modified by editing the `/etc/grafana/grafana.ini` file.	Yes
Jupyter Notebook	Log levels are controlled by the `Application.log_level` configuration parameter in `/etc/jupyterhub/jupyterhub_config.py`. It is set to 30 by default.	Yes
Cray Graph Engine (CGE)	`INFO`. The `log-reconfigure –log-level` *`number`* command can be used to modify the log level. Use the drop down on the CGE UI to set the log level for the specific action being performed, i.e. `query`, `update` or `checkpoint`. Use the drop down on the **Edit Server Configuration** page to set the log level. Changing the log level in this manner persists until CGE is shut down.	No. Restarting CGE reverts the log level to `INFO`
Flex scripts: ● `urika-yam-status` ● `urika-yam-flexup` ● `urika-yam-flexdown` ● `urika-yam-flexdown-all`	`INFO`. Changing the log level for these scripts is not supported.	NA
Spark Thrift server	`INFO`.	Yes
HiverServer2	`INFO`.	Yes
Tenant proxy logs	`DEBUG`	No
Urika-GX security manager logs	`INFO`	No

8.3 Analytic Applications Log File Locations

Log files for a given service are located on the node(s) the respective service is running on, which can be identified using the `urika-inventory` command. For more information, see the `urika-inventory` man page.

Table 37. Analytics Applications Log File Locations

Application/Script	Log File Location
Mesos	`/var/log/mesos`
Marathon	`/var/log/messages`
HA Proxy	`/var/log/haproxy.log`
Mesos frameworks: ● Marathon ● Spark	/var/log/mesos/agent/slaves/. Within this directory, a framework's output is placed in files called `stdout` and `stderr`, in a directory of the form `slave-`*`X`*`/fw-`*`Y`*`/`*`Z`*, where *X* is the slave ID, *Y* is the framework ID, and multiple subdirectories *Z* are created for each attempt to run an executor for the framework. These files can also be accessed via the web UI of the slave daemon. The location of the Spark logs is determined by the cluster resource manager that it runs under, which is Mesos on Urika-GX.
Grafana	`/var/log/grafana/grafana.log`
InfluxDB	`/var/log/influxdb/influxd.log`
collectl	collectl does not produce any logging information. It uses logging as a mechanism for storing metrics. These metrics are exported to InfluxDB. If collectl fails at service start time, the cause can be identified by executing the `collectl` command on the command line and observing what gets printed. It will not complain if the InfluxDB socket is not available.
Hadoop	The following daemon logs appear on the node they are running on: ● `/var/log/hadoop/hdfs/hadoop-hdfs-namenode-`*`nid`*`.log` ● `/var/log/hadoop/hdfs/hadoop-hdfs-datanode-`*`nid`*`.log` ● `/var/log/hadoop/yarn/yarn-yarn-nodemanager-`*`nid`*`.log` ● `/var/log/hadoop/yarn/yarn-yarn-resourcemanager-`*`nid`*`.log` In the above locations, *`nid`* is used as an example for the node name. Application specific logs reside in HDFS at `/app-logs`
Spark	● Spark event logs (used by the Spark History server) reside at: `hdfs://user/spark/applicationHistory` ● Spark executor logs (useful to debug Spark applications) reside with the other Mesos framework logs on the individual compute nodes (see above) at: `/var/log/mesos/agent/slaves/`
Jupyter Notebook	`/var/log/jupyterhub.log`
Flex scripts:	`/var/log/urika-yam.log`

Application/Script	Log File Location
• `urika-yam-status` • `urika-yam-flexdown` • `urika-yam-flexdown-all` • `urika-yam-flexup`	
ZooKeeper	`/var/log/zookeeper`
Hive Metastore	`/var/log/hive`
HiveServer2	`/var/log/hive`
HUE	`/var/log/hue`
Spark Thrift Server	`/var/log/spark`

Spark Audit Logs

A per-user Spark audit log that details start and stop of applications is located at `/var/log/spark/k8s/`*`username`*`.log` with entries of the following form:

```
Tue Apr 03 07:54:05 CDT 2018 username spark-test-1522760043061-driver START \
Application Started with 1 driver plus 5.0 executors using 6.0 cores and 496.0GB memory
Tue Apr 03 07:54:38 CDT 2018 username spark-test-1522760043061-driver STOP \
Application Stopped
Tue Apr  3 08:15:51 CDT 2018 username username-shell-159738-5d5b87c8b-82td6 \
START spark-shell Shell Started with 1 driver using 16.0 cores and 60GB memory
Tue Apr  3 08:16:36 CDT 2018 username username-shell-159738-5d5b87c8b-82td6 \
STOP spark-shell Shell Stopped
Wed Apr 04 04:28:45 CDT 2018 username spark-test-1522834122688-driver START \
Application Started with 1 driver plus 7.0 executors using 8.0 cores and 688.0GB memory
Wed Apr 04 04:29:38 CDT 2018 username spark-test-1522834122688-driver STOP \
Application Stopped
Wed Apr 04 04:30:09 CDT 2018 username spark-test-1522834207396-driver START \
Application Started with 1 driver plus 2.0 executors using 3.0 cores and 208.0GB memory
Wed Apr 04 04:30:42 CDT 2018 username spark-test-1522834207396-driver STOP \
Application Stopped
Wed Apr 04 04:32:05 CDT 2018 username spark-test-1522834323513-driver START \
Application Started with 1 driver plus 2.0 executors using 17.0 cores and 208.0GB memory
Wed Apr 04 04:32:28 CDT 2018 username spark-test-1522834323513-driver STOP \
Application Stopped
```

These log files will be located on whatever node an application is submitted from, typically login1, though maybe elsewhere depending how the system enables users to access the system.

This log has the general format *`date username driver-pod-name action message`*, where:

- *`driver-pod-name`* is the Kubernetes pod name that is the driver for the application that can be used to link this information to more detailed information from Kubernetes.
- *`action`* is either `START` or `STOP`
- *`message`* contains informational content, such as resources requested by the users application.

CAUTION: In the event of a failed or cancelled job (i.e. user executes a Ctrl+C/kill on the job) there may be no corresponding `STOP` event registered for a job reported as started.

8.4 Security Related Troubleshooting Information

User Authorization Issues

While logging on to Urika-GX (either a tenant or a physical node), the system may return the following message:

```
You are not authorized to log into this system -- to obtain access please contact
your system administrator
```

This can mean several things. The administrator should review the logs found in `/var/log/secure` on the node where the message was seen and look for log messages from the `pam_cray_utp` PAM module. These will have the string `pam_cray_utp` as a prefix to the log message.

- If the system returns the following message:

  ```
  pam_cray_utp: look up of user '<username> on LDAP server with local_ldap_host = 'login1', \
  local_ldap_port number = 389 and local_ldap_base_dn = 'ou=crayusers,dc=urika,dc=com' produced an empty result
  ```

  The user is not yet registered in the user authorization list. Refer to *Authentication and Authorization* on page 171 for more information. If the user is supposed to be authorized and this is a physical node (e.g. a login node), add the user to the user authorization list as follows:

  ```
  smw# ux-tenant-add-user -u username
  smw# ux-tenant-relax username
  ```

  If this was seen on a tenant VM and the user is supposed to be authorized for that tenant, add the user to the user authorization list as follows:

  ```
  smw# ux-tenant-add-user -u username tenant-name
  ```

  The user will be added as a member of the tenant and restricted from logging on to physical nodes.

- If the system returns the following message:

  ```
  pam_cray_utp: user '<username>' attempted to log into Tenant VM 'tenant-name' \
  but does not have a matching crayTenant attribute: failing account authorization
  ```

  The user is in the authorized user list, but is not yet a member of the tenant who owns the VM. If the user is supposed to be authorized for this tenant, add the user to the tenant as follows:

  ```
  smw# ux-tenant-add-user -u username tenant-name
  ```

  The user will be added as a member of the tenant and restricted from logging on to physical nodes.

User Access Issues

A user may attempt to log on to a physical node (e.g. a login node) and appear to be logged in but then be dropped out again with the following message:

```
interactive login not permitted
```

This message may be followed by the login attempt being closed. This means the user is configured with restricted user access in the user authorization list. The user is not currently permitted to log on to physical nodes. The user may be authorized for a tenant VM, but not a physical node.

If the user is supposed to have access to physical nodes, then the user must be granted relaxed user access, as follows:

```
smw# ux-tenant-relax username
```

Kerberos Issues

When a user logs on to a physical node or tries to run commands through the Urika tenant proxy from a tenant VM, the user may see the following message:

```
kinit failed: operations requiring kerberos will not work during this session --
please contact your system administrator
```

In all probability, the login or `utp-launch` attempt itself will succeed, but commands requiring access to HDFS or to the user's Kerberos keytab file will fail.

To diagnose this, the administrator needs to look in `/var/log/secure` and look for messages coming from the `pam_cray_keytab` PAM module. These will have the string `pam_cray_keytab:` preceding the text of the log message. Most likely, this may be caused by Kerberos recognizing or finding a keytab file. The log messages will be similar to the following:

```
Oct 16 13:34:30 nid00030 sshd[143134]: pam_cray_keytab: user = '<username>', kinit reported: \
'kinit: Client '<username>@local' not found in Kerberos database while getting initial credentials'
Oct 16 13:34:30 nid00030 sshd[143134]: pam_cray_keytab: kinit failed for user '<username>', see previous logs
```

There may be other errors reported by `kinit` as well. If the logs contain such information, first check whether there is a keytab currently set up for the user. This can be done by looking for a file named *`username`*`.keytab` in the `/security/secrets` directory. If that file does not exist, the problem is most likely that the `usm-sync-users` command has not been re-run to create the user's secrets since the time that user was either added to the authorized users list or added as a Linux user. Re-run `usm-sync-users` to troubleshoot the problem:

```
smw# usm-sync-users
```

If the file does exist, most likely the user's keytab file has gotten out of sync with what Kerberos assumes as the user's keytab. In this case, regenerate the user's keytab (and Mesos secrets) by running:

```
smw# usm-recreate-secret username
```

This will generate a new set of secrets for the user and install them.

Note that while the above actions alone will probably fix the Kerberos issue, it will still be needed to restart Urika-GX services by executing `urika-stop` and then executing `urika-start` to sync up the created Mesos secrets and give the user access to Mesos for job launch. Since stopping and starting Urika-GX services can cause running jobs to be killed, schedule that action carefully.

Tenant Management Issues

When the Urika-GX tenant management package is updated, the update procedure attempts to identify configuration that is out of date without damaging the configuration. If configuration settings are removed in a new version of the software, the system will return a message of the following form when the `ux-tenant-*` commands are executed:

```
The following configuration settings are no longer used
and can be removed from the Urika Tenant Management
configuration:
UXTENANT_GLOBAL_LDAP_DEFAULT_TENANT_USER_PASS
```

```
UXTENANT_GLOBAL_LDAP_GROUP_OU
Clean up all references to these to avoid seeing this
warning again.
```

Though this message is harmless, it can be prevented by locating the files containing the offending settings (all settings have a key in them that identifies the type of file they come from:

- `UXTENANT_GLOBAL` settings come from `/etc/sysconfig/uxtenant/global`
- `UXTENANT_TENANT` settings come from configuration files in `/etc/sysconfig/uxtenant/tenants`
- `UXTENANT_HOST` settings come from files in `/etc/sysconfig/uxtenant/hosts`
- `UXTENANT_MOUNT` settings come from files in `/etc/sysconfig/uxtenant/mounts`

Edit all the affected files and remove the listed configuration settings and the comments at the end of the file. Also remove the `UXTENANT_UNUSED_SETTINGS` setting at the very end of the file containing the list of outdated settings.

Along the same lines, the system may return the following message:

```
ux-tenant-list-users(error): configuration is out of date with the current
version
The following configuration files need to be brought
up to date with the current version of Urika Tenant
Management. These need to be fixed before tenant
management can proceed:
tenants/default
Fix all of these files before trying this command again
```

This indicates that some set of configuration files are out of date. `ux-tenant-*` commands cannot be executed until the configuration is brought up to date by adding the required settings, which are listed as a comment in each of the affected files. Edit the file to put the setting in place. Use the `ux-tenant-config(5)` man page to determine the values for new settings.

Similarly, the system may return the following message:

```
The following configuration settings are new with the
current version of Urika Tenant Management and need to
be set up correctly before tenant management can proceed:
UXTENANT_TENANT_IP_DNS1
UXTENANT_TENANT_IP_DNS2
UXTENANT_TENANT_IP_DOMAIN
UXTENANT_TENANT_IP_GATEWAY
Fix all references to these before trying this command again
```

This will indicate the configuration settings that need to be brought up to date. Edit the affected files and set up the missing configuration, then remove the comments and the `UXTENANT_NEW_SETTINGS` setting at the end of the file(s). After that, re-run the failing command.

About Using su

If a root attempts to switch to a restricted users via `su`, the user's shell will respond with the message, "`interactive logins not permitted`" and then exit. To get around this, root can execute:

```
# su -s /bin/bash - restricted-user-name
```

Public Network Access from Tenant Virtual Machines

If the tenant VM is up and running and all of the network configuration looks like it should be working correctly (the public interface on `ens4` has an IP address and the interface is up, the gateway and DNS configuration is correct and so forth) look on the host node (probably one of the login nodes) and make sure that the `br1` network bridge is up and has the IP address being used for the public IP address of the login node. Also make sure that no other interface (for example, the Ethernet onto which `br1` is bound) has that same IP address on it. If an interface other `br1` has that same IP address, check the network configuration files in `/etc/sysconfig/network-scripts`. Look first in `ifcfg-br1`. It should looks something like this:

```
DEVICE=br1
TYPE=Bridge
BOOTPROTO=static
ONBOOT=yes
NM_CONTROLLED=no
IPADDR=172.30.51.237
NETMASK=255.255.240.0
GATEWAY=172.30.48.1
DNS1=172.30.84.40
DNS2=172.31.84.40
DOMAIN=us.cray.com
```

Then look at `ifcfg-enp8s0f1`. This is the configuration for the Ethernet device plugged into the public network and should look something like this:

```
NAME="enp8s0f1"
DEVICE="enp8s0f1"
ONBOOT=yes
BOOTPROTO=static
TYPE=Ethernet
NM_CONTROLLED=no
BRIDGE=br1
```

Notice a few different things about this interface config:

- It has no IP address and is configured with `BOOTPROTO=static`
- It is not controlled by network-manager: `NM_CONTROLLED=no`
- It is bound to the bridge device `br1`:`BRIDGE=br1`
- It is set up to be brought up at boot: `ONBOOT=yes`

All of these things need to be true for the tenant VM to work correctly on this host node.

Now, look at the running interface state for both of these interfaces. An IP address on `enp8s0f1` indicates a stale network state from the time the user switched to bridging. Clear this out using the following command:

```
# ifdown br1; ifdown enp8s0f1; ifup enp8s0f1; ifup br1
```

Lastly, look at the running network interface state, which should indicate that `br1` has an IP address, `enp8s0f1` does not, and both are up. At this point, the tenant VM should be able to reach the external network.

Working with kinit

When a user logs into a system running in the secure service mode, the login procedure uses kinit and the user's keytab to assign a Kerberos ticket granting ticket to the user. If the user stays logged into the system for too long, this ticket may expire and the user may lose access to HDFS and other services. To see the status of the ticket granting ticket, including when it was granted and when it will expire, the user may use the following command:

```
$ klist
Ticket cache: FILE:/tmp/krb5cc_10003
Default principal: crayusr@local

Valid starting       Expires              Service principal
10/23/2017 07:55:16  10/30/2017 07:55:16  krbtgt/local@local
   renew until 11/22/2017 06:55:16
```

To obtain a new ticket granting ticket the user may log out and log back in, or issue the following command:

```
$ kinit -A -k -t /security/secrets/username.keytab
```

For users logged into tenant VMs, there is no ticket granting ticket generated for the tenant VM login session. A ticket granting ticket is generated each time a command is run on a physical node through the tenant proxy mechanism.

Logs

- **Urika-GX Tenant Management Logs** - `utp-server` logs are located in the `/var/log/utp/utp.log` file. These logs are only accessible to the administrator and have a default log level of `DEBUG`
- **Urika-GX Security Manager Logs** - The default log file for the `usm-sync-users` and `usm-recreate-secret` commands is located at `/var/log/usm/urika-secret-manager.log`. These logs have a default log level of `INFO`. The default log level and format can also be optionally reconfigured by modifying `/opt/cray/urika-secret-manager/default/logconf.d/usm-logging-dict.json`

8.4.1 Save and Restore Tenant Information

Prerequisites

This procedure requires root privileges on the SMW.

About this task

When making changes to Urika-GX that result in swapping, re-deploying or wiping out the contents of tenant VM host nodes or the node on which HDFS name nodes reside (NID 0), critical tenant data can be destroyed, resulting in loss of the tenant VM environment or users' HDFS data.

The procedure described here provides a means for backing up this tenant data and reconstructing the affected tenants so that HDFS data and the tenant VM environment are preserved.

Procedure

1. Log on to the SMW as root.
2. Stop the name nodes.

   ```
   # urika-stop
   ```

   For more information, refer to the `urika-stop` man page.

3. Remove the tenant VMs

   ```
   # ux-tenant-remove --vm-only tenant-name
   ```

   For more information, refer to the `ux-tenant-remove` man page.

4. Back up data/files.

   While the HDFS data on Urika-GX is spread across data nodes and is stored redundantly to prevent loss, the name nodes for both tenant and non-tenant data store the metadata on NID 0. This means that if NID 0 loses its data, the user data may persist, but the ability to identify and retrieve it will be lost. To prevent this when swapping, redeploying or wiping NID 0, it is necessary to bring all name nodes to a quiescent (`shutdown`) state, and then copy the data found in the directory tree `/mnt/hdd-2/hdfs` to a safe place. Shutting down the name nodes is achieved by running the `urika-stop` command. The operating environment for any given tenant VM resides in a disk image file found on the host node (the node named in the `/etc/sysconfig/uxtenant/hosts/`*`host-name`* file named in the tenant configuration) at `/qemu/`*`tenant-name`*`.img`.

   a. Copy the files from `/mnt/hdd-2/hdfs` on NID 0 to a safe place.

   b. Copy the tenant disk image(s) from `/qemu/`*`tenant-name`*`.img` on the affected host node(s) to a safe place.

5. Perform the required operations on the node(s).

6. Bring the nodes back up, making sure they are completely deployed and integrated. Contact Cray Support for guidance on node redeployment.

7. Put back the name node data on NID 0 at `/mnt/hdd-2/hdfs`

8. Put back the tenant VM disk image files on the affected host nodes.

9. Restore the tenant VMs.

   ```
   # ux-tenant-create --use-image-file path  tenant-name
   ```

   In the preceding command, *`path`* is the path to the disk image file on the host node where the disk image is put back. For more information, refer to the `ux-tenant-create` man page.

10. Restart the name nodes

    ```
    # ux-tenant-start
    ```

    For more information, refer to the `ux-tenant-start` man page.

8.4.2 LDAP Server Start-up Issues

When there is an external network connectivity issue, the configured DNS server(s) on the external network cannot be reached. This results in a DNS lookup timeout (somewhere from 45 seconds to two minutes, possibly longer) completing an `su -` to a different user. Similar delays are seen logging into login1 (or other login nodes) from the SMW. For the most part, the place where this DNS lookup seems to take place is in creating a Kerberos ticket from a keytab in `kinit`. This takes place in the `pam_cray_kinit.so` PAM module. It is possible that other networking related problems could arise from this situation as well, since we do not know all of the paths that could result in network timeouts during LDAP start-up. One consequence of this can be that the `slapd` (LDAP) server on login1 will time out trying to start and fail start-up.

Likely Cause

When there is an external network connectivity issue, the configured DNS server(s) on the external network cannot be reached. This results in a DNS lookup timeout (somewhere from 45 seconds to two minutes, possibly longer) completing an `su -` to a different user. Similar delays are seen logging into login1 (or other login nodes) from the SMW. For the most part, the place where this DNS lookup seems to take place is in creating a Kerberos ticket from a keytab in `kinit`. This takes place in the `pam_cray_kinit.so` PAM module. It is possible that other networking related problems could arise from this situation as well, since we do not know all of the paths that could result in network timeouts during LDAP start-up. One consequence of this can be that the `slapd` (LDAP) server on login1 will time out trying to start and fail start-up.

Resolution

To troubleshooting this issue, check and fix the network connectivity from the login node to the external network over the BR1 network bridge on the login node.

To check network connectivity, the admin needs to check that the BR1 network bridge is up, has an IP address configured on it, and is bridging the correct Ethernet device for the external network. Also make sure that the underlying Ethernet device does NOT have an IP address configured on it. Then verify that the physical network connection is plugged into Login1 in the correct location and that the Ethernet connection is showing a good link light. If all of this is true, verify that the DNS servers listed in `/etc/resolv.conf` are reachable (usually this can be done by pinging them, unless they have ICMP replies disabled). Correct all the issues found until the DNS connectivity to the configured DNS servers is ensured.

If the problem persists with all of this verified to be working, Contact Support for additional guidance.

8.5 Modify the Secret of a Mesos Framework

Prerequisites

This procedure requires root privileges.

About this task

The `urika-mesos-change-secret` command is used to change the secret of Urika-GX service-level Mesos secrets, such as, `marathon`, `haproxy`, etc. The following list of items, which is subject to change, needs to be kept under consideration when creating a new secret for a framework:

- All lowercase and uppercase letters and numbers are allowed in secrets. `-`, `_`, and `.` are allowed at any position in the secret
- ! can be used, but should not be the first character
- Other punctuation characters can cause authentication issues and are not recommended
- White space in the secret is not allowed. White space at the end of the string will get trimmed
- An empty space secret will not authenticate.

For more information, refer to the `urika-mesos-change-secret` man page.

Procedure

1. Log on to the SMW as root.
2. Stop the Mesos cluster.

   ```
   # urika-stop -s mesos_cluster
   ```

3. Execute the `urika-change-mesos-secret` script, specifying the framework and new secret.

   Modify Spark's secret:

   ```
   # urika-change-mesos-secret -spark secret
   ```

   Modify Marathon's secret:

   ```
   # urika-change-mesos-secret -marathon secret
   ```

   The system will return an error if no secret is specified or if the secret is not entered in plain text.

4. Start the Mesos cluster.

   CAUTION: The Mesos cluster needs to be restarted after running the `urika-mesos-change-secret` command, otherwise the framework will not authenticate.

   ```
   # urika-start -s mesos_cluster
   ```

8.6 Clean Up Log Data

As jobs are executed on the system, a number of logs are generated, which need to be cleaned up, otherwise they may consume unnecessary space. Log data is useful for debugging issues, but if it is certain that this data is no longer needed, it can be deleted.

- **Mesos logs** - Mesos logs are stored under `var/log/mesos`, whereas the Mesos framework logs are stored under `/var/log/mesos/agent/slaves`. These logs need to be deleted manually.
- **Marathon logs** - Marathon logs are stored under `var/log/message` and need to be deleted manually.
- **HA Proxy logs** - HA Proxy logs are stored under `var/log/message` and need to be deleted manually.
- **Jupyter logs** - Juypter log file are located at `var/log/jupyterhub/jupyterhub.log` and need to be deleted manually.
- **Grafana and InfluxDB logs** - Grafana logs are stored under `var/log/grafana`, whereas InfluxDB logs are stored under `var/log/influxdb`. Influxdb log files are compressed. Both Grafana and InfluxDB use the `logrotate` utility to keep log files from using too much space. Log files are rolled daily by default, but if space is critical, logs can be deleted manually.
- **Spark logs** - Shuffle data files on the SSDs is automatically deleted on Urika-GX. Spark logs need to be deleted manually and are located at the following locations:
  - Spark event logs - Located at `hdfs://user/spark/applicationHistory`
  - Spark executor logs - Located on individual compute nodes at `/var/log/mesos/agent/slaves/`

- **Hadoop logs** - Hadoop log files are located in the following locations and need to be deleted manually:
  - Core Hadoop - Log files are generated under the following locations:
    - `var/log/hadoop/hdfs`
    - `var/log/hadoop/yarn`
    - `var/log/hadoop/mapreduce`
  - ZooKeeper - ZooKeeper logs are generated under `var/log/zookeeper`
  - Hive (metastore and hive server2) - These logs are generated under `var/log/hive`
  - Hive Webhcat - These logs are generated under `var/log/webhcat`
  - Oozie - Oozie logs are stored under `/var/log/oozie`
  - HUE - HUE logs are generated under `/var/log/hue`
- **Flex scripts (urika-yam-status, urika-yam-flexup, urika-yam-flexdown, urika-yam-flexdown-all)** - These scripts generate log files under `/var/log/urika-yam.log` and need to be deleted manually.
- **`mrun`** - `mrun` does not generate logs.
- **Cray Graph Engine (CGE) logs** - The path where CGE log files are located is specified via the `-l` parameter of the `cge-launch` command. Use the `cge-cli log-reconfigure` command to change the location after CGE is started with `cge-launch`. CGE logs need to be deleted manually. Users can also use `--log-level` argument to CGE CLI commands to set the log level on a per request basis. In addition, the `cge.server.DefaultLogLevel` parameter in the `cge.properties` file can be used to set the log level to the desired default.

8.7 Diagnose and Troubleshoot Orphaned Mesos Tasks

Prerequisites

This procedure requires root access and the system to be running in the default service mode. Execute the `urika-state` command to ensure that Mesos is running and that the system is operating in the default service mode.

About this task

The metrics displayed in Mesos UI can also be retrieved using CURL calls. Cray-developed scripts (for flexing up a YARN sub-cluster) and `mrun` use these curl calls in as they interoperate with Mesos for resource brokering. If the metrics displayed by Mesos UI and the metrics that the curl calls return different results Mesos may not work correctly and all the Mesos frameworks will be affected. As such, the aforementioned Cray-developed scripts and `mrun` will not be able to retrieve the needed resources. This behavior can be identified when:

- there is a disconnect between the CURL calls and the Mesos UI. Specifically, there will be an indication of orphaned Mesos tasks if the CURL call returns a higher number of CPUs used than that returned by the UI. Cray-developed scripts for flexing YARN sub-clusters use curl calls, and hence do not allow flexing up if there are not enough resources reported.
- there are orphaned Mesos tasks, as indicated in the Mesos Master and Mesos Slave logs at `/var/log/mesos`. Mesos Master will reject task status updates because it will not recognize the framework those tasks are being sent from.

If this behavior is encountered, follow the instructions listed in this procedure:

Procedure

1. Log on to the System Management Workstation (SMW) as root
2. Clear the slave meta data on all the nodes with Mesos slave processes running

   The following example can be used on a 3 sub-rack system:

   ```
   # pdsh -w nid000[00-47] -x nid000[00,16,30,31,32,46,47] \
   'rm -vf /var/log/mesos/agent/meta/slaves/latest'
   ```

3. Stop the cluster

   ```
   # urika-stop
   ```

4. Start the cluster

   ```
   # urika-start
   ```

   After following the aforementioned steps, the system should be restored to its original state. For additional information, contact Cray Support.

8.8 Troubleshoot Common Analytic and System Management Issues

The following table contains a list of some common error messages and their description. Please note that this is not an exhaustive list. Online documentation and logs should be referenced for additional debugging/ troubleshooting. For a list of Cray Graph Engine error messages and troubleshooting information, please refer to the *Cray® Graph Engine User Guide*.

Table 38. System Management Error Messages

Error Message	Description	Notes/Resolution
ERROR: unauthorized command 'cat' requested by client	This message is returned when a restricted user, logged in to a tenant VM, attempts to execute a command that is not part of set of the white listed commands.	Only white listed commands can be executed by restricted users who are logged into tenant VMs. For more information, refer to the '*Urika®-GX System Administration Guide*'.
Error message: `ERROR: tag(s) not found in playbook: non_existent_service. possible values: collectl,grafana,hdfs,hdfs_dn,hdfs_nn,hdfs_sn,hdp_app_timeline,hdp_hist,hive,hive2,hive_met,hive_web,hue,influxdb,jupyterhub,marathon,`	Description: User has specified a service that does not exist to the `urika-stop` or `urika-start` command.	Resolution: Use the correct name of the services by selecting one of the options listed in the error message.

Error Message	Description	Notes/Resolution
`mesos,nodemanager,oozie,spark_hist,yarn,yarn_rm, zookeeper`		
Error message: You are not authorized to log into this system -- to obtain access please contact your system administrator su: Permission denied	This message may be returned by the `urika-state` command on a freshly deployed system if the `hdfs` user has not yet been created.	Wait until the `main_run/ main_update` playbooks have finished creating the `hdfs` user

Table 39. Spark Error Messages

Error Message	Description	Resolution
INFO mesos.CoarseMesosSchedulerBackend: Blacklisting Mesos slave 20151120-121737-1560611850-5050-20795-S0 due to too many failures; is Spark installed on it? INFO mesos.CoarseMesosSchedulerBackend: Mesos task 30 is now TASK_FAILED	There may be something preventing a Mesos slave from starting the Spark executor. Common causes include: ● The SSD is too full ● The user does not have permission to write to Spark temporary files under `/var/spark/tmp/`	Refer to Spark logs.
ERROR mesos.MesosCoarseGrainedSchedulerBackend: Mesos error: Master refused authentication Exiting due to error from cluster scheduler: Master refused authentication	This message appears when Spark is not able to authenticate with Mesos.	The user may not have Mesos credentials, in which case the user would need to be added via the `usm-sync-users script` script. In other cases, this error may be a result of the user's credentials being in a bad state, in which case the admin would need to run the `usm-recreate-secret` script to create a new secret for the user. For more information, refer to the '*Urika®-GX System Administration Guide*'.
Lost executor # on *`host`*	Something has caused the Spark executor to die. One of the reasons may be that there is not enough memory allocated to the executors.	Increase the memory allocated to executors via one of the following parameters: ● `--executor-memory` ● `spark.executor.memory configuration` Refer to Spark logs for additional information.

Table 40. Flex Scripts Error Messages

Error Message	Description	Resolution
The number of nodes requested to flex up is greater than the total number of resources available. Please enter a valid number of nodes	The user is attempting to flex up more nodes than are available which using the `urika-yam-flexup` command.	Enter a lower number of nodes for the flex up request.
No time out specified by user through commandline argument, setting the timeout from /etc/urika-yam.conf file. in /etc/urika-yam.conf val: 15 minutes	The user has not specified a timeout while using the `urika-yam-flexup` command.	This error message can safely be ignored if it is required to use the default timeout value, which is 15 minutes. Otherwise, please specify the desired value when using the `urika-yam-flexup` command.
ID names can only contain alphanumeric, dot '.' and dash '-' characters. '@' not allowed in jhoole@#$. Usage: urika-yam-flexup --nodes #nodes --identifier name --timeout timeoutInMinutes	The user has specified an incorrect identifier/application name when using the `urika-yam-flexup` command.	Reenter the command with the correct identifier.
Minimum timeout is 5 minutes. A timeout less than the minimum timeout cannot be requested, with an exception of zero timeout. Please note that you can request a zero timeout (set value of timeout to 0) by which you do not call timeout, you chose to flex down the nodes manually using `urika-yam-flexdown`. Please submit a new flex up request with valid timeout.	Incorrect minimum timeout was specified.	Submit a new flex up request with valid timeout (Request for timeout greater than minimum timeout).
Currently only "x" nodes are available in the cluster. Please wait till the number of nodes you require are available Or submit a new flex up request with nodes less than "x"	This error is seen when the number of nodes requested to flex up is not available.	Either wait till the number of nodes required are available Or submit a new flex up request with nodes less than "x".
Invalid app name. Your app name can consist of a series of names separated by slashes. Each name must be at least 1 character. The name may only contain digits (0-9), dashes (-), dots (.), and lowercase letters (a-z). The name may not begin or end with a dash.	This error is seen when the identifier provided by user for the flex up request is invalid.	Follow the rules mentioned there and re-submit a new flex up request.
Total number of resources not set in the /etc/urika-yam.conf file, please re-check the configuration file	In `/etc/urika-yam.conf` file, the number of resources is set by default. The total number of resources may not have been set.	Re-check the status of mesos cluster.

Error Message	Description	Resolution
Hostname is not set in the /etc/ urika-yam.conf file, please re-check the configuration file.	In `/etc/urika-yam.conf` file, the parameter `hostname` is set by default. The value set may not be correct or may not have been set.	Ensure that this parameter is set, and the value is the same as default value.
Mesos port is not set in the /etc/ urika-yam.conf file, please re-check the configuration file	In `/etc/urika-yam.conf` file, the parameter `marathon_port` is set by default. This parameter may not have been set or value set may not be set to the same as the default value.	Ensure that this parameter is set, and the value is the same as default value.
Marathon port is not set in the /etc/ urika-yam.conf file, please re-check the configuration file.	In `/etc/urika-yam.conf` file, the parameter marathon_port is set by default. This parameter may not have been set or value set may not be set to the same as the default value..	It should be ensured that this parameter is set, and the value is the same as default value.
The number of nodes you requested to flex up is greater than the total number of resources available. Please enter a valid number of nodes	This error is seen when the number of nodes requested to flex up is more than the total number of nodes available in the cluster	Submit a new flex up request with nodes less than or equal to the number of nodes available in the cluster.
App '/$marathon_app_name' does not exist. Please re-check the identifier corresponding nodes you flex up, that you would like to flex down	The identifier provided for flex down does not exist.	Re-check the usage: if operating as the root user, please provide the complete name as seen in urika-yam-status or as a non-root user, ensure to provide the same identifier used at the time of flex up. In addition, check if `/var/log/urika-yam.log` reflects any log messages where timeout criteria has been matched and there was a flex down of the app already.
Looks like there is some problem with flex up. Please try urika-yam-status or look at the logs to find the problem	The job failed to launch.	Review logs (stored at `/var/log/urika-yam.log` on login nodes) or execute the urika-yam-status command to identify if there is any problem. Please check if there are any issues related to Mesos and/or Marathon. If the Mesos and/or Marathon web UI cannot be accessed, contact the administrator, who should verify that the Mesos and Marathon daemons are up and running. If any of these daemons are not running for some reason, report the logs to Cray Support and restart the Mesos

Error Message	Description	Resolution
		cluster using the `urika-start` command. For more information, see the `urika-start` man page.
Could not find the script urika-yam-start-nodemanager in hdfs. Looks like there is an error with your urika-yam installation Please contact your sysadmin	The `urika-yam-start-nodemanager` script is a component of the Cray developed scripts for flexing up a YARN cluster and is installed as part of the installation of these flex scripts.	If this issue is encountered, the administrator should verify that: ● HDFS is in a healthy state ● Marathon and Mesos services are up and running. The status of the aforementioned services can be checked using the `urika-state` command. For more information, see the `urika-state` man page. Contact support for additional information about resolving this issue.
INFO: can not flexup YAM in secure mode	This message indicates that the user is attempting to execute the `urika-yam-flexup` script while the system is in the secure service mode, instead of in the default mode.	The `urika-yam-flexup` script can only be used in the secure service mode. For more information, refer to the Urika-GX System Administration Guide.
INFO: can not flexdown YAM in secure mode	This message indicates that the user is attempting to execute the `urika-yam-flexdown` script while the system is in the secure service mode, instead of in the default mode.	The `urika-yam-flexdown` script can only be used in the secure service mode. For more information, refer to the Urika-GX System Administration Guide.
INFO: can not flexdown-all YAM in secure mode	This message indicates that the user is attempting to execute the `urika-yam-flexdown-all` script while the system is in the secure service mode, instead of in the default mode.	The `urika-yam-flexdown-all`script can only be used in the secure service mode. For more information, refer to the Urika-GX System Administration Guide.
INFO: can not get YAM status in secure mode	This message indicates that the user is attempting to execute the `urika-yam-status` script while the system is in the secure service mode, instead of in the default mode.	The `urika-yam-status` script can only be used in the secure service mode. For more information, refer to the Urika-GX System Administration Guide.

Table 41. Marathon/Mesos/`mrun` Error Messages

Error Message	Description	Resolution
ERROR:mrun: Force Terminated job /mrun/ 2016-193-12-13-03.174056 Cancelled due to Timeout Examples: ● error("mrun: --immediate timed out while waiting") ● error("mrun: Timed out waiting for mrund : %s" % appID) ● error("mrun: Force Terminated job %s Cancelled due to Timeout" %	These errors indicate timeout and resource contention issues, such as the job timed out, the machine is busy, too many users running too many jobs, a user waiting for their job to start, but previous jobs have not freed up nodes, etc. Additionally, if a user set a job timeout's to 1 hour, and the job lasted longer than 1 hour, the user would get a `Job Cancelled` timeout error.	Ensure that the there are enough resources available and that the timeout interval is set correctly.
HWERR[r0s1c0n3][64]:0x4b14:The SSID received an unexpected response:Info1=0x191000000000003:Info2=0x7	Mesos is not able to talk to the Zookeeper cluster and is attempting to shut itself down.	Restart Mesos using the `urika-start` command.
WARN component.AbstractLifeCycle: FAILED SelectChannelConnector@0.0.0.0:4040: java.net.BindException: Address already in use	User is attempting to execute a job on a port that is already in use.	This message can be safely ignored.
ERROR:Unexpected 'frameworks' data from Mesos ● Examples: ○ error("Mesos Response: %s" % ret) ○ error("Unexpected 'frameworks' data from Mesos") ○ error("mrun: Getting mrund state threw exception - %s" % ○ error("getting marathon controller state threw exception - %s" % ○ error("Unexpected 'apps' data from Marathon") ○ error("mrun: Launching mrund threw exception - %s" % (str(e)))	These errors occur when `mrun` is not able to connect/communicate with Mesos and/or Marathon.	Refer to online Mesos/Marathon documentation.

Error Message	Description	Resolution
○ error("mrun: unexpected 'app' data from Marathon: exception - %s" % (str(e))) ○ error("mrun: startMrund failed") ○ error("mrun: Exception received while waiting for "		
● error("mrun: select(): Exception %s" % str(e)) ● error("mrun: error socket") ● error("mrund: ● error %r:%s died\n" % (err,args[0])) ● error("mrund: select(): Exception %s\n" % str(e))	These errors may be encountered in situations where an admin physically unplugs an Ethernet cable while a CGE job was running, or a node died, etc.	Ensure that the Ethernet cable is plugged while jobs are running.
● NCMD: Error leasing cookies MUNGE: ● Munge authentication failure [%s] (%s).\n	These error only occur if the specific system services have failed.	Refer to log messages under `/var/log/messages` on the node the message was encountered on.
ERROR:Not enough CPUs for exclusive access. Available: 0 Needed: 1 Examples ● parser.error("Only --mem_bind=local supported") ● parser.error("Only --cpu-freq=high supported") ● parser.error("Only --kill-on-bad-exit=1 supported") ● parser.error("-n should equal (-N * --ntasks-per-node)") ● parser.error("-N nodes must be >= 1") ● parser.error("-n images must be >= -N nodes") ● parser.error("No command specified to launch"); error("Not enough CPUs. " ● error("Not enough CPUs for exclusive access. "	These errors are typically caused by user errors, typos and when not enough nodes are available to run a job.	Ensure that there are enough nodes available and there are no typos in the issues command.

Error Message	Description	Resolution
• error("Not enough nodes. " • parser.error("name [%s] must only contain 'a-z','0-9','-' and '.'" • parser.error("[%s] is not executable file" % args[0])		
ERROR: Zero read: Scaling DOWN not supported	This error message is returned when `mrun` is not expecting one of the remote nodes to abruptly close its socket. When `mrun` goes to read the socket, it detects the error, generates the message, and the application dies (or is killed).	Marathon UI to scale down `mrun` jobs is not supported. This message can occur in the following conditions: • The user used the –`kill-on-scaledown` option of the `mrun` command. • The first remote compute node did not exit successfully Thus, this error message can only occur if the application is killed or if someone used the Marathon REST API in an attempt to scale the application down. No action needs to be taken in either case.
mrun: error: Users may only cancel their own mrun jobs	This message comes when a non-root user tries to use `mrun --cancel` to cancel any Marathon job that was not launched by that user using `mrun`.	User `root` is allowed to use "mrun --cancel" to kill any Marathon-started job. All other users should only kill the Marathon jobs they launched using the `mrun` command.
INFO: Can not run this application in secure mode, cancelling app	This message indicates that the user is attempting to use `mrun` while the system is in the secure service mode, instead of in the default mode.	`mrun` can only be used in the default service mode. For more information, refer to the Urika-GX System Administration Guide.

Table 42. Hadoop Error Messages

Error Message	Description	Resolution
org.apache.hadoop.hdfs.server.\ namenode.SafeModeException: Cannot create or delete a file. Name node is in safe mode.	During the start up, the NameNode goes into a safe mode to check for under replicated and corrupted blocks. A Safe mode for the NameNode is essentially a read-only mode for the HDFS cluster, where it does not allow any modifications to file system or blocks. Normally, the NameNode disables the safe mode	Force the NameNode out of safe mode by running the following command as a HDFS user: `$ hdfs dfsadmin -safemode leave`

Error Message	Description	Resolution
	automatically, however, if there are there are too many corrupted blocks, it may not be able to get out of the safe mode by itself.	
Too many underreplicated blocks in the NameNode UI	Couple of dataNodes may be down. Please check the availability of all the dataNodes	If all the DataNodes are up and still there are under replicated blocks. Run the following 2 commands in order as a HDFS user: `$ hdfs fsck / | grep 'Under replicated' |` `awk -F':' '{print $1}' >> \` `/tmp/under_replicated_files` `$ for hdfsfile in ` + "`" + `cat /tmp/` `under_replicated_files` + "`" + `; \` `do echo "Fixing $hdfsfile :" ; \` `hadoop fs -setrep 3 $hdfsfile; \` `done`
Too many corrupt blocks in name node UI	The NameNode might not have access to at least one replication of the block.	Check if any of the DataNodes are down. If all the DataNodes are up and the files are no longer needed, execute the following command: `$ hdfs fsck / -delete`
org.apache.hadoop.ipc.\ RemoteException(java.io.IOException): \ File /tmp/test could only be replicated to \ 0 nodes instead of minReplication (=1).	HDFS space may have reached full capacity. Even though Urika-GX has a heterogeneous file system, the default storage type is DISK unless explicitly set to use SSD. The user might have filled up the default storage, which is why HDFS would not be able to write more data to DISK.	To identify the used capacity by storage type, use the following commands:For both `DISK` and `SSD`, calculate the sum of usage on all the DataNodes. For `DISK`: `$ df /mnt/hdd-2/hdfs/dd | awk 'NR==2{print $3}'` For SSD: `$ df /mnt/ssd/hdfs/dd | awk 'NR==2{print $3}'`
YARN job is not running. You can see the status of the job as ACCEPTED: waiting for AM container to be allocated, launched and register with RM.	The NodeManagers may not be running to launch the containers.	Check the number of available node managers by executing the following command: `$ yarn node -list`

Additional Tips and Troubleshooting Information

- If JupyterHub processes owned by the user remain running after the user has logged out from Jupyter these processes can be manually killed using the Linux kill command.
- The system will return the message, "`Service '`*`serviceName`*`' is not supported in the current security mode`" if it is attempted to start a service that is not supported in the current service mode. Use the `urika-state` or `urika-service-mode` commands to check which service mode the system is running in. For more information, refer to *Urika-GX Service Modes* on page 173

- If for any reason, Marathon does not start after a system crash, as a result of the queue reaching full capacity, use the `urika-stop` command, followed by the `urika-start` command to resolve the issue.
- In Urika-GX's multi tenant environment, individual tenant members are restricted from overriding the global Hadoop configuration directory and from specifying a specific NameNode on the CLI. As such, certain arguments passed to HDFS commands on the CLI are ignored to ensure security of tenant data. If these arguments are passed to the CLI, the system will return a warning indicating that it detected an argument that is not allowed for restricted users and that the argument is being removed
- Use one of the following mechanisms if it is required to kill Spark jobs:
  - Kill the job using the Spark UI - Click on the text **(kill)** in the **Description** column of the **Stages** tab.
  - Kill the job using the Linux `kill` command.
  - Kill the job using the `Ctrl+C` keyboard keys.
- The system will return the following error if a user attempts to view help information for an unsupported Lustre `lfs` sub-command:

  ```
  The sub-command command is either unknown or not supported for tenant users. For
  more information on tenant user rules try 'lfs help tenant-rules'.
  ```
- When modifying the number of CPUs or memory for a tenant VM, the system will return an error if it is attempted to allocate more than the acceptable value of CPU or memory to a tenant VM via the `ux-tenant-alter-vm` command. For more information, refer to the `ux-tenant-alter-vm` man page.
- In rare cases, switching from the secure to default mode may result in some Romana network policy information that is not translated into the appropriate IP table rules. This allows a recently created pod to ping a pod in a different Kubernetes name space. Contact Cray support if this problem is encountered.

8.9 Troubleshoot `mrun` Issues

Some common `mrun` error messages and their cause(s) are listed as following:

Issue related to Mesos/Marathon

Potential cause: - These errors occur when `mrun` is not able to connect/communicate with Mesos and/or Marathon. To troubleshoot these issues, refer to online Mesos/Marathon documentation.

- Format: `Mon Jul 11 2016 11:39:43.601145 UTC[][mrun]:ERROR:Unexpected 'frameworks' data from Mesos`
- Examples:
  - `error("Mesos Response: %s" % ret)`
  - `error("Unexpected 'frameworks' data from Mesos")`
  - `error("mrun: Getting mrund state threw exception - %s" % )`
  - `error("getting marathon controller state threw exception - %s" %)`
  - `error("Unexpected 'apps' data from Marathon")`
  - `error("mrun: Launching mrund threw exception - %s" % (str(e)))`
  - `error("mrun: unexpected 'app' data from Marathon: exception - %s" % (str(e)))`
  - `error("mrun: startMrund failed")`

- `error("mrun: Exception received while waiting for ")`

Command-line options Errors

Potential cause - These errors are typically caused by user errors, typos and when not enough nodes are available to run a job.

- Format: `Mon Jul 11 2016 11:47:22.281972 UTC[][mrun]:ERROR:Not enough CPUs for exclusive access. Available: 0 Needed: 1`
- Examples:
  - `parser.error("Only --mem_bind=local supported")`
  - `parser.error("Only --cpu-freq=high supported")`
  - `parser.error("Only --kill-on-bad-exit=1 supported")`
  - `parser.error("-n should equal (-N * --ntasks-per-node)")`
  - `parser.error("-N nodes must be >= 1")`
  - `parser.error("-n images must be >= -N nodes")`
  - `parser.error("No command specified to launch");`
  - `error("Not enough CPUs. "`
  - `error("Not enough CPUs for exclusive access. " )`
  - `error("Not enough nodes. " )`
  - `parser.error("name [%s] must only contain 'a-z','0-9','-' and '.'" )`
  - `parser.error("[%s] is not executable file" % args[0])`

Timeout errors

Cause- The errors indicate timeout and resource contention issues, such as, the job timed out, the machine is busy, too many users running too many jobs, a user waiting for their job to start, but previous jobs have not freed up nodes, etc. Additionally, if a user set a job timeout's to 1 hour, and the job lasted longer than 1 hour, they would get a `Job Cancelled` timeout error.

- Format: `Mon Jul 11 2016 12:13:08.269371 UTC[][mrun]:ERROR:mrun: Force Terminated job /mrun/2016-193-12-13-03.174056 Cancelled due to Timeout`
- Examples:
  - `error("mrun: --immediate timed out while waiting")`
  - `error("mrun: Timed out waiting for mrund : %s" % appID)`
  - `error("mrun: Force Terminated job %s Cancelled due to Timeout" %)`

Network errors, such as socket, switch, TCP, node failure

Cause - These errors may be encountered in situations where an admin physically unplugs an Ethernet cable while a CGE job was running, or a node died, etc.

- Examples:
  - `error("mrun: select(): Exception %s" % str(e))`
  - `error("mrun: error socket")`
  - `error %r:%s died\n" % (err,args[0]))`
  - `error("mrund: select(): Exception %s\n" % str(e))`

System service errors

Cause - These errors only occur if the specific system services have failed. The cause of the issue may be identified by looking at the log messages under `/var/log/messages` on the node the message was encountered on.

- Examples:
  - `NCMD: Error leasing cookies MUNGE:`
  - `Munge authentication failure [%s] (%s).\n`

For more information, see the `mrun(1)` man page.

8.10 Troubleshoot: Application Hangs as a Result of NFS File Locking

About this task

Applications may hang when NFS file systems are projected through DVS and file locking is used. To avoid this issue:

Procedure

Specify the `nolock` option in the NFS mount point on DVS servers.

See the `nfs(5)` man page for more information on the `nolock` option.

8.11 Troubleshoot: DVS does not Start after Data Store Move

About this task

If DVS fails after the Cray system's data store is moved to a shared external Lustre file system, verify that DVS has the correct `lnd_name` that uniquely identifies the Cray system to the LNet router. The default value for `lnd_name` on a single-user Lustre file system is `gni`. Each system sharing an external Lustre file system must have a unique `gni*` identifier, such as `gni0`, `gni1`.

Procedure

Modify the `/etc/modprobe.d/dvs.conf` file to add the following line:

```
options dvsipc_lnet lnd_name=gnix
```

where `gnix` is the unique LNet identifier for the Cray system.

8.12 Troubleshoot: DVS Ignores User Environment Variables

If the `nouserenv` option has not been specified in the DVS entry, and a DVS user environment variable that was set does not override the associated DVS mount option, it appears as if DVS is ignoring user environment variables. This can be caused by the addition of a large number of user environment variables. Due to the nature of Linux, if a user adds a large number of user environment variables (large enough that the kernel needs to store that information somewhere other than the usual location), DVS may not be able to find and apply those user environment variables, producing unexpected results.

To define a large number of user environment variables, Cray recommends that users include those definitions in the user's shell so that they are available at startup and stored where DVS can always locate them.

8.13 Clear Leftover hugetlbf Files

Prerequisites

This procedure requires root privileges

About this task

Follow the instructions in this procedure if the system displays the error message, "`LIBDMAPP ERROR during create in _dmappi_get_sheap_addr_hugepages: Permission denied`". The node IDs shown in these examples are for a system containing 3 sub-racks. If a system with 2 sub-racks is being used, replace these with the node IDs for a 2 sub-rack system.

Procedure

1. Log on to the System Management Workstation (SMW) as root
2. Execute the following command to clear out any leftover `hugetlbf` files

   ```
   # pdsh -w 'nid000[00-47]' '/bin/rm -f /var/lib/hugetlbfs/global/*/*DMAPP*'
   ```

3. Verify that the leftover `hugetlbf` files have been cleared

   ```
   # pdsh -w 'nid000[00-47]' '/bin/ls -l /var/lib/hugetlbfs/global/*/*DMAPP*' 2>/dev/null|wc
   0       0       0
   ```

8.14 Remove Temporary Spark Files from SSDs

Prerequisites

This procedure requires root privileges.

About this task

Spark writes temporary files to the SSDs of the compute nodes that the Spark executors run on. Ordinarily, these temporary files are cleaned up by Spark when its execution completes. However, sometimes Spark may fail to fully clean up its temporary files, such as, when the Spark executors are not shut down correctly. If this happens too many times, or with very large temporary files, the SSDs may begin to fill up. This can cause Spark jobs to fail or slow down.

Urika-GX checks for any idle nodes once per hour, and cleans up any left over temporary files. This is handled by a cron job running on one of the login nodes that executes the `/usr/sbin/cleanupssds.sh` script once per hour. Follow the instructions in this procedure if this automated clean up ever proves to be insufficient.

Procedure

1. Log on to one of the login nodes as root.
2. Kill all the processes of running Spark jobs.
3. Execute the `/usr/sbin/cleanupssds.sh` script.

   ```
   # /usr/sbin/cleanupssds.sh
   ```