XC™ Series Slurm Installation Guide

(CLE 6.0.UP07)

S-2538

Contents

1 About *XC Series™ Slurm Installation Guide (CLE 6.0.UP07)*

This publication describes how to install and configure the Slurm workload manager as native software on Cray XC™ systems.

Title Change

This manual was formerly titled *Slurm Software Installation Guide for Cray XC™ Series Systems*.

Scope and Audience

The procedures presented in this manual are to be carried about by technicians at sites where Cray XC™ systems are installed, employed by either Cray Inc., or the customer organization.

Typographic Conventions

`Monospace`	A `Monospace` font indicates program code, reserved words or library functions, screen output, file names, path names, and other software constructs
`Monospaced Bold`	A **`bold monospace`** font indicates commands that must be entered on a command line.
`Oblique` or *`Italics`*	An *`oblique`* or *`italics`* font indicates user-supplied values for options in the syntax definitions
Proportional Bold	A **proportional bold** font indicates a user interface control, window name, or graphical user interface button or control.
`Alt-Ctrl-f`	`Monospaced` hypenated text typically indicates a keyboard combination

Record of Revision, publication S-2538

CLE release	Date
5.1 UP01	December 2013
5.2 UP01	June 2014
5.2 UP02	October 2014
5.2 UP04	August 2015
6.0 UP01	June 2016
6.0 UP02	November 2016
6.0 UP03	February 2017

CLE release	Date
6.0 UP04	June 2017
6.0 UP05	October 2017
6.0 UP06	February 2018
6.0 UP07	July 2018

2 Introduction

Cray provides the required infrastructure to support running the workload manager Slurm natively on Cray systems. SchedMD, the Slurm vendor, provides any Slurm plug-ins required to run on a Cray system.

Cray Daemons

Cray supplies the service node daemons listed below, to support the native Slurm model. Characteristics:

- There is one of each daemon per Cray system, not one daemon per service node.
- These three daemons need to execute on the listed Cray service node, not on nodes external to a Cray.
- These daemons are started automatically by Ansible plays.

aeld Provides job information to HSS to use in congestion-management decisions. This daemon needs to run on a service node which has connectivity to the SMW. The `aeld` daemon must run on the boot node.

apptermd Upon receipt of certain compute node events, initiates killing the applications which are assigned to that compute node. It is recommended that the `apptermd` daemon run on the SDB node.

ncmd Manages the assignment and release of Aries network cookies required per application launch. The `ncmd` daemon must run on the node where the WLM scheduler executes, which is the SDB node.

Each of the daemons listed above has its own logfile in the `/var/opt/cray/daemon/log` directory on the service node where the daemon is executing. Log rotation is used to manage log files, as shown in the following directory listings on the boot and sdb nodes:

```
boot:~ # ls /var/opt/cray/aeld/log/
aeld.log aeld.log-20131116.gz aeld.log-20131118.gz aeld.log-20131119.gz
```

```
sdb:~ # ls /var/opt/cray/apptermd/log
apptermd.log apptermd.log-20131116.gz apptermd.log-20131118.gz
apptermd.log-20131119.gz
```

```
sdb:~ # ls /var/opt/cray/ncmd/log
ncmd.log ncmd.log-20131116.gz ncmd.log-20131118.gz ncmd.log-20131119.gz
```

3 Install Slurm

IMPORTANT: The following commands must be repeated when updating Slurm versions. They can be skipped for other updates.

Build Slurm RPMs

1. Obtain the source for Slurm.

   Find out which version(s) of Slurm are compatible with this CLE release on the CrayPort website at *http://crayport.cray.com*. Then obtain the source for the latest compatible, stable version of Slurm from *SchedMD*.

2. Create a `~crayadm/wlm_install/build-slurm.sh` file with these contents:

```
#!/bin/bash
export PATH=/usr/local/bin:/usr/bin:/sbin:/usr/sbin:/bin
rpmbuild --with cray --with debug --define "_slurm_sysconfdir /etc/opt/slurm" \
         --define "_prefix /opt/slurm/$1" \
         -ta ${IMPS_POSTBUILD_FILES}/slurm-$1.tar.bz2
```

3. Make the file executable:

```
chmod +x ~crayadm/wlm_install/build-slurm.sh
```

4. Create the Slurm build image recipe for a CLE 6.0.up07 x86-64 system:

```
smw# export REPOS="-r common_cle_6.0.up07_sles_12sp3_ari
         -r common_cle_6.0.up07_sles_12sp3_ari_updates
         -r lustre-2.7_cle_6.0.up07_sles_12sp3_ari
         -r lustre-2.7_cle_6.0.up07_sles_12sp3_ari_updates
         -r passthrough-common_cle_6.0.up07_sles_12sp3
         -r passthrough-common_cle_6.0.up07_sles_12sp3_updates
         -r sle-module-legacy_12
         -r sle-module-legacy_12_updates
         -r sle-sdk_12sp3
         -r sle-sdk_12sp3_updates
         -r sle-server_12sp3
         -r sle-server_12sp3_updates"
smw# recipe create slurm_build
smw# recipe update -c slurm-build_cle_6.0.up07_sles_12sp3 $REPOS slurm_build
```

5. If building Slurm for aarch64 (ARM) nodes, add this line:

```
smw# recipe update -l aarch64 slurm_build
```

6. Edit `/etc/opt/cray/imps/image_recipes.d/image_recipes.local.json`, adding `postbuild_chroot` and `postbuild_copy` sections to the `slurm_build` recipe:

```
"slurm_build": {
    "postbuild_chroot": [
        "${IMPS_POSTBUILD_FILES}/build-slurm.sh <version>"
```

```
    ],
    "postbuild_copy": [
        "/home/crayadm/wlm_install/build-slurm.sh",
        "/home/crayadm/wlm_install/slurm-version.tar.bz2"
    ],
```

7. Build Slurm and add RPMs to a repository:

```
# image create -r slurm_build slurm_build
# repo create -t SLES12 slurm
# repo update -a \
'/var/opt/cray/imps/image_roots/slurm_build/usr/src/packages/RPMS/x86_64/slurm*.rpm' \
slurm
```

8. If building Slurm for aarch64 (ARM) nodes, also do this:

```
# image create -A aarch64 -r slurm_build slurm_build_aarch64
# repo update -a \
'/var/opt/cray/imps/image_roots/slurm_build_aarch64/usr/src/packages/RPMS/aarch64/slurm*.rpm' \
slurm
```

9. Create Slurm image recipes using the following commands. Use the `-l aarch64` argument only if installing for use the aarch64 (ARM) nodes. The *version* argument is either `17.02` or `17.11` depending on the version of Slurm being installed.

```
smw# recipe create initrd-slurm_login-large

smw# recipe update [-l aarch64] initrd-slurm_login-large \
-i initrd-login-large_cle_6.0.up07_sles_12sp3_ari \
-r slurm ${REPOS}

smw# recipe create slurm_login-large

smw# recipe update [-l aarch64] slurm_login-large \
-i login-large_cle_6.0.up07_sles_12sp3_ari \
-c login-slurm_version_cle_6.0.up07_sles_12sp3 \
-r slurm ${REPOS}

smw# recipe create initrd-slurm_compute-large

smw# recipe update [-l aarch64] initrd-slurm_compute-large \
-i initrd-compute-large_cle_6.0.up07_sles_12sp3_ari \
-r slurm ${REPOS}

smw# recipe create slurm_compute-large

smw# recipe update [-l aarch64] slurm_compute-large \
-i compute-large_cle_6.0.up07_sles_12sp3_ari \
-c compute-slurm_version_cle_6.0.up07_sles_12sp3 \
-r slurm ${REPOS}

smw# recipe create slurm_service

smw# recipe update slurm_service \
-i service_cle_6.0.up07_sles_12sp3_ari \
-c service-slurm_version_cle_6.0.up07_sles_12sp3 \
-r slurm ${REPOS}
```

10. If the `sjstat` or `sjobexit` commands are needed, they should be added to the image recipe(s) where needed.

```
recipe update slurm_login-large -p slurm-contribs
```

Package the Recipes Using `imgbuilder`

1. If installing for use on x86-64 nodes, add image recipes to `imgbuilder` by editing `/etc/opt/cray/config/global/config/cray_image_groups.yaml`:

```
slurm:
- recipe: "initrd-slurm_compute-large"
  dest: "initrd-slurm_compute-large{note}_cle_{cle_release}-build{cle_build}
{patch}_sles_12-created{date}.cpio"
  nims_group: "compute"
- recipe: "initrd-slurm_login-large"
  dest: "initrd-slurm_login-large{note}_cle_{cle_release}-build{cle_build}
{patch}_sles_12-created{date}.cpio"
  nims_group: "login"
- recipe: "slurm_service"
  dest: "slurm_service{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12-
created{date}.cpio"
  nims_group: "service"
```

2. If installing for use on aarch64 nodes, add these entries to `/etc/opt/cray/config/global/config/cray_image_groups.yaml`:

```
slurm:
- recipe: "initrd-slurm_compute-large"
  dest: "initrd-slurm_compute-large{note}_cle_{cle_release}-build{cle_build}
{patch}_sles_12-aarch64-created{date}.cpio"
  arch: "aarch64"
  nims_group: "compute_aarch64"
- recipe: "initrd-slurm_login-large"
  dest: "initrd-slurm_login-large{note}_cle_{cle_release}-build{cle_build}
{patch}_sles_12-aarch64-created{date}.cpio"
  aarch: "aarch64"
  nims_group: "login_aarch64"
```

3. Run `imgbuilder` to package the images.

```
smw# imgbuilder -g slurm --map --partition part
```

4 Configure Slurm

In the following instructions replace *cfgset* with the actual config set being configured (e.g. slurm_p0), and *version* with the version of Slurm obtained from SchedMD (e.g. 15.08.6).

Create Slurm Configuration Files

IMPORTANT: These commands must be repeated when system hardware changes. They can be skipped for other updates.

Before you begin, you must know the *control_machine* hostname and *partition*. The *control_machine* is a service node running the slurm_service or slurm_login-large image and it is site-specific. Contact your system administrator to determine this information.

1. Copy the file slurm-*version*.tar.bz2 to ~crayadm/wlm_install/ on the SMW. (If the ~crayadm/wlm_install/ directory does not exist, create it.)
2. Extract files from the tarball:

   ```
   smw# tar -xf slurm-version.tar.bz2
   ```

3. Run the following commands on the SMW as root to set up Slurm configuration files.

   ```
   # SLURM_CONF_DIR=/var/opt/cray/imps/config/sets/cfgset/files/simple_sync/common/files/etc/opt/slurm
   # mkdir -p $SLURM_CONF_DIR
   # SLURM_DIR=/home/crayadm/wlm_install/slurm-version
   # python $SLURM_DIR/contribs/cray/csm/slurmconfgen_smw.py control_machine partition \
       -t $SLURM_DIR/contribs/cray/csm/ -o $SLURM_CONF_DIR
   ```

   where *control_machine* is the control machine hostname.

4. Add the following to the top of file $SLURM_CONF_DIR/slurm.conf, where *name* is a descriptive name of the system:

   ```
   ClusterName=name
   ```

5. Create $SLURM_CONF_DIR/cgroup.conf with the following contents:

   ```
   # Slurm cgroup.conf for Cray XC systems
   CgroupAutomount=yes
   CgroupMountpoint="/dev"
   ConstrainCores=yes
   ConstrainRAMSpace=yes
   TaskAffinity=no
   AllowedRAMSpace=95
   ```

6. Create this file:

   $SYNC_DIR/etc/sysconfig/slurmd

   with the following contents:

```
SLURM_OOM_ADJ=-1000
SLURMSTEPD_OOM_ADJ=-1000
```

The above steps create a basic working Slurm configuration. Customize Slurm configuration files in `$SLURM_CONF_DIR` as needed.

7. If using Slurm 17.02.3 or later, copy the ansible playbook into the config set:

```
cp /var/opt/cray/imps/image_roots/slurm_build/usr/src/packages/BUILD/slurm-*/
contribs/cray/csm/slurm_playbook.yaml \
/var/opt/cray/imps/config/sets/<cfgset>/ansible/
```

Configure `logrotate`

If configuring Slurm 17.02, create this file:

`$SYNC_DIR/etc/logrotate.d/slurm`

with the following contents:

```
/var/spool/slurm/*log {
  compress
  missingok
  nocopytruncate
  nocreate
  nodelaycompress
  nomail
  notifempty
  noolddir
  rotate 5
  sharedscripts
  size=5M
  create 640 root root
  postrotate
    for daemon in \$(/opt/slurm/default/bin/scontrol show daemons)
    do
      killall -SIGHUP \$daemon
    done
  endscript
}
```

If configuring Slurm 17.11, create the same `$SYNC_DIR/etc/logrotate.d/slurm` file, but replace the `SIGHUP` with `SIGUSR2`:

```
/var/spool/slurm/*log {
  compress
  missingok
  nocopytruncate
  nocreate
  nodelaycompress
  nomail
  notifempty
  noolddir
  rotate 5
  sharedscripts
  size=5M
  create 640 root root
  postrotate
    for daemon in \$(/opt/slurm/default/bin/scontrol show daemons)
    do
```

```
      killall -SIGUSR2 \$daemon
    done
  endscript
}
```

Configurator interface

If you are not familiar with the configurator interface, this is a short description. Also see *XC Series System Configurator User Guide*.

On the interactive menu that is displayed by the configurator, note the number (such as 4) for an option that is to be changed. Enter this number or enter one of the option groupings shown under **Select Options**, and press **Return**. Each succeeding prompt shows the command or result that is executed by pressing **Return** again. (Other allowed commands are listed under **Actions on Selected** and **Other Actions**.) For each option, enter a value or press **Return** to enter the default value (unless the interactive menu shows `default=(none)`). Options that have been set are indicated by a symbol or highlight in the interactive list. Example using the following excerpt of the interactive menu:

```
4)    ccm_wlm          [ unconfigured, default=pbs ]
5)    ccm_queues       [ unconfigured, default=(none) ]
6)    cray_batch_var   [ unconfigured, default=/var/spool/PBS ]
```

To enter the default value for the first option above: enter **4** and press **Return** three times. This sets the value `pbs` for the `ccm_wlm` option.

Configure CLE

Update the Slurm config set using the following commands and settings.

`smw# cfgset update -s cray_wlm_detect -m interactive -l advanced cfgset`

- Set `cray_wlm_detect.enabled` to `true`.
- Set `cray_wlm_detect.settings.common.data.active_wlm` to `SLURM`.
- Set `cray_wlm_detect.settings.common.data.slurm_id_multiplier` to `10000000000` if using Slurm 17.02 and earlier, or `4294967296` if using Slurm 17.11 and later.

`smw# cfgset update -s cray_munge -m interactive -l advanced cfgset`

- Set `cray_munge.enabled` to `true`.

`smw# cfgset update -s cray_user_settings -m interactive -l advanced cfgset`

- Add slurm to the `default_modules login` and `service` lists.

`smw# cfgset update -s cray_eproxy -m interactive -l advanced cfgset`

- Set `cray_eproxy.enabled` to `true`.
- Set `cray_eproxy.settings.wrapped.data.slurm` to `true`.

`smw# cfgset update -s cray_persistent_data -m interactive -l advanced cfgset`

- Add an entry for `/var/spool/slurm` to the mounts list.

`smw# cfgset update -s cray_node_groups -m interactive cfgset`

- Define a node group containing the Slurm control machine, if required. An existing node group can be used if appropriate.

```
smw# cfgset update -s cray_auth -m interactive cfgset
```

- Add the control machine node group to the `cray_auth.settings.access.data.config_id_service_groups` list.

```
smw# cfgset update -s cray_rsip -m interactive -l advanced cfgset
```

- Add the control machine node group to the `cray_rsip.settings.service.data.node_groups_as_client` list.

Validate and Apply the Changes

After all the config set changes have been applied, validate the config set.

```
smw# cfgset validate cfgset
```

```
smw# cfgset validate -c cfgset
```

```
smw# cfgset validate varname -c cfgset
```

```
smw# cfgset validate -p varname -c cfgset
```

```
smw# cfgset validate -p varname -c cfgset
```

```
smw# cfgset validate cfgset
```

```
smw# cfgset validate -c cfgset
```

```
smw# cfgset validate varname -c cfgset
```

```
smw# cfgset validate -p varname -c cfgset
```

```
smw# cfgset validate -p varname -c cfgset
```

If the config set passes validation, use the `cnode` command to update the CLE nodes.

```
smw# cnode update -p part -c cfgset '*'
```

Use an External Accounting Database

To use an accounting database off the Cray, the accounting data must be sent from `slurmctld` on the SDB node through a node with external network access (such as a login node) and on to the external node. Cray recommends using `slurmdbd` running on a login node for this purpose.

Follow the accounting setup instructions in the *Accounting and Resource Limits document*, under the heading *Slurm Accounting Configuration After Build* . Make sure that `AccountingStorageHost` in `slurm.conf` and `DbdHost` in `slurmdbd.conf` are set to the hostname of a node with external network access.

Stack Sizes

Some applications which run successfully under ALPS may require more stack size to run successfully under Slurm. This problem will cause segmentation faults in applications which use large amounts of stack space. For example, the HPC Challenge Benchmark requires at least 32MB of stack space to run successfully.

To resolve the issue, manually increase the stack size limit with `ulimit -s` before running the application.

5 Update Slurm

About this task

This procedure describes how to update to a newer version of Slurm.

Procedure

1. Obtain the latest Slurm source from SchedMD at *http://www.schedmd.com/#repos* and copy it to `~crayadm/wlm_install/` on the SMW.

   NOTE: If this file does not already exist, create it.

2. Update the version in the `postbuild_chroot` and `postbuild_copy` sections of the `slurm_build` recipe in `/etc/opt/cray/imps/image_recipes.d/image_recipes.local.json`:

```
"slurm_build": {
     "postbuild_chroot": [
         "${IMPS_POSTBUILD_FILES}/build-slurm.sh version"
      ],
      "postbuild_copy": [
          "/home/crayadm/wlm_install/build-slurm.sh",
          "/home/crayadm/wlm_install/slurm-version.tar.bz2"
      ],
```

3. Build Slurm and update the slurm repository.

```
# image create -r slurm_build --force slurm_build
# repo update -a
'/var/opt/cray/imps/image_roots/slurm_build/usr/src/packages/RPMS/x86_64/
slurm*.rpm' slurm
```

4. Run `imgbuilder` to package the images:

```
smw# imgbuilder -g slurm --map --partition part
```

6 Use Slurm with CCM

This topic describes the use of Cluster Compatibility Mode (CCM) and native Slurm to provide `ssh` launch and use of the Aries network on Cray XC systems. CCM provides a typical cluster environment enabling Slurm to work with the following:

- Third-party MPI and ISV applications
- Serial workloads
- X11 (X Windows)
- Compilation on compute nodes in Cray XE/XC systems
- `ssh` launching

Slurm with Cray modifications runs natively on Cray systems and uses the Aries network with no need for code changes, for applications launched by `srun` and other third-party MPI application launchers that use `srun`.

Cray uses ISV Application Acceleration (IAA) and modified libraries and files to launch applications that do not use Cray PMI. IAA-related modified libraries and files are linked during the RPM install. These libraries and files are always available on compute nodes at the following standard locations:

- `/usr/lib64/libibgni.so.1.0.0`
- `/etc/dat.conf`
- `/usr/sbin/ibstat`
- `/usr/lib64/libibverbs.so.1.0.0`
- `/usr/lib64/libibumad.so.3.0.2`
- `/usr/lib64/librdmacm.so.1.0.0`

Slurm and CCM Configuration

Use the following steps to configure Slurm for use with CCM.

Prerequisites:

- NHC must be enabled, which is the default.
- In the `slurm.conf` file, the `SelectTypeParameters` cannot be set to `NHC_NO` or `NHC_NO_STEPS`.
- CCM requires use of the NHC plugin.

The `Nodes` setting in `slurm.conf` can be the same as the `Nodes` setting of other partitions. That is, the nodes defined for the CCM-only partition are not exclusive to this partition. The `Nodes` value is different on every system and is automatically generated.

You will use the Configurator interface, described below. The procedure specifies the step details.

Configurator interface

If you are not familiar with the configurator interface, this is a short description. Also see *XC Series System Configurator User Guide*.

On the interactive menu that is displayed by the configurator, note the number (such as 4) for an option that is to be changed. Enter this number or enter one of the option groupings shown under **Select Options**, and press **Return**. Each succeeding prompt shows the command or result that is executed by pressing **Return** again. (Other allowed commands are listed under **Actions on Selected** and **Other Actions**.) For each option, enter a value or press **Return** to enter the default value (unless the interactive menu shows `default=(none)`). Options that have been set are indicated by a symbol or highlight in the interactive list. Example using the following excerpt of the interactive menu:

```
4)    ccm_wlm          [ unconfigured, default=pbs ]
5)    ccm_queues       [ unconfigured, default=(none) ]
6)    cray_batch_var   [ unconfigured, default=/var/spool/PBS ]
```

To enter the default value for the first option above: enter **4** and press **Return** three times. This sets the value `pbs` for the `ccm_wlm` option.

1. Log in to the SMW and edit the following file:

   ```
   /var/opt/cray/imps/config/sets/cfgset/files/simple_sync/common/files/etc/ \
      opt/slurm/slurm.conf
   ```

2. Configure the `ccm.conf` file so that the following fields are set appropriately to work with native Slurm:

   - `CCM_QUEUES=`[list of one or more Slurm exclusive partitions for CCM use]
   - `CCM_WLM=slurm`
   - `CCM_BATCH_VAR=`[full path of Slurm spool]

3. Configure the `slurm.conf` file to define one or more exclusive-access partitions for CCM use. To make a new partition, create a new line starting with `PartitionName`, customizing the values as needed.

   Following is an example of two partitions that have the same nodes, but one partition is exclusive for CCM use, and the other is shared for general use. The `slurm.conf` and `ccm.conf` configuration files need to agree on the *`ccm_queue`* partition name.

   ```
   PartitionName=workq Nodes=nid000[32-35,41-44] Shared=EXCLUSIVE Priority=1 \
      Default=YES DefaultTime=60 MaxTime=24:00:00 State=UP
   PartitionName=ccm_queue Nodes=nid000[32-35,41-44] Shared=EXCLUSIVE \
      Priority=1 Default=NO State=UP
   ```

4. Change to `root`:

   ```
   > sudo su -u root -i
   root's password:
   # password
   ```

5. Launch the configurator for CCM:

   ```
   # cfgset update -m interactive -l advanced -s cray_ccm ccm_queue
   ```

   Where *`ccm_queue`* is the name defined in file `slurm.conf`.

6. Ensure that the cray_ccm config service is enabled (`[ status: enabled ]` displayed at top of the Service Configuration Menu). If it is not enabled, enter **E**, **Return** until it is.

```
Cray CCM Configuration Service Menu [default: save & exit - Q] $ E
```

7. Enter **4**, **Return**, **C**, and **Return**. Enter **slurm** at the following prompt, and then press **Return**:

```
cray_ccm.settings.base.data.ccm_wlm
[<cr>=set 'pbs', <new value>, ?=help, @=less] $ slurm
```

8. Enter **5**, **Return**, **C**, and **Return**. Enter **ccm_queue** at the following prompt, and then press **Return**:

```
cray_ccm.settings.base.data.ccm_queues
[<cr>=set '"ccm_queue vce_queue"', <new value>, ?=help, @=less] $ ccm_queue
```

9. Enter **6**, **Return**, **C**, and **Return**. Enter **/var/spool/slurm** at the following prompt, and then press **Return**:

```
cray_ccm.settings.base.data.cray_batch_var
[<cr>=set '/var/spool/PBS', <new value>, ?=help, @=less] $ /var/spool/slurm
```

10. Verify that the interactive menu shows these three selected values:

```
4)       ccm_wlm             slurm
5)       ccm_queues          ccm_queue
6)       cray_batch_var      /var/spool/slurm
```

Use Cases

The following are typical use cases. The ccm modules file needs to be loaded along with any site-specific module file related to CCM which may set environment variables to define the CCM partition name(s).

Use Case 1

Use `salloc` to reserve eight compute nodes for 64 tasks from the configured CCM partition.

1. The partition name can be specified on the `salloc` command line or set with the `SALLOC_PARTITION` environment variable by the user or within a site specific file.

```
salloc -n 64 -N 8 --partition=ccm_queue SALLOC_PARTITION=ccm_queue
salloc -n 64 -N 8
```

2. From within the `salloc` interactive session, use `ccmrun` with a third-party MPI launcher to start an application using `ssh` launch and the Aries interconnect.

```
ccmrun /cray/css/ostestdata/isv/mpi/platform_mpi/9.1.2/bin/mpirun -v -prot \
   -IBV -e MPI_REMSH=ssh -hostfile /home/users/username/.crayccm/ccm_nodelist.9 \
   -np 64 /cray/css/ostestdata/isv/apps/imb/3.2.4/platform_mpi/9.1.2/IMB-MPI1 \
   gather -off_cache -1
```

3. From within the `salloc` interactive session, use `ccmlogin` to `ssh` to the head compute node.

4. Use the third-party MPI launcher directly to start an application using `ssh` launch and the Aries interconnect.

```
/cray/css/ostestdata/isv/mpi/platform_mpi/9.1.2/bin/mpirun -v -prot -IBV \
   -e MPI_REMSH=ssh -hostfile /home/users/username/.crayccm/ccm_nodelist.7 \
   -np 64 /cray/css/ostestdata/isv/apps/imb/3.2.4/platform_mpi/9.1.2/IMB-MPI1 \
   allreduce -off_cache -1
```

Use Case 2

Use sbatch to reserve eight compute nodes for 64 tasks from the configured CCM partition. Use `sbatch` to reserve eight compute nodes for 64 tasks from the configured CCM partition. The partition name can be specified on the sbatch command line or set within the batch job script or set with the `SBATCH_PARTITION` environment

variable by the user or within a site specific file. As noted in Use Case 1 above, `ccmrun` can be called from within the batch job script.

```
sbatch -n 64 -N 8 --partition=ccm_queue batch-job-script
SBATCH_PARTITION=ccm_queue sbatch -n 64 -N 8 batch-job-script
```

Use Case 3

For a system configured with eight compute nodes and two partitions (one shared for general use and one for CCM use only) with the same eight compute nodes assigned to each partition, use `salloc` to reserve all eight compute nodes from the `ccm_queue`.

An `srun` to the general use partition remains queued because there are no available resources from the general partition. Because the nodes are assigned to both partitions, the nodes are "allocated" from both partitions. The nodes were allocated from an exclusive partition, so those nodes are not available when requested from the shared general use partition.

```
salloc -n 64 -N 8 --partition=ccm_queue
sinfo

PARTITION  AVAIL  JOB_SIZE  TIMELIMIT   CPUS  S:C:T     NODES STATE  NODELIST
workq*     up     1-infini  1-00:00:00  16+   1+:8:2 8  allocated    nid000[32-35,40-43]
ccm_queue  up     1-infini  1-00:00:00  16+   1+:8:2 8  allocated    nid000[32-35,40-43]

srun -n 1 hostname
srun: job 65 queued and waiting for resources
```

7 Usage and Troubleshooting

7.1 Slurm Affinity Options

Compute unit affinity gives greater control for placing applications on compute nodes.

Three possible `Taskplugin` configurations are available on native Slurm systems, each with different effects. The choice affects how the `--cpu_bind` and `--mem_bind srun` options work. For details on how to use those options, consult the `srun` man page. To change the configuration, set the `TaskPlugin` option in `slurm.conf` accordingly

- `TaskPlugin=task/cray,task/cgroup`

  With this option, CPU binding succeeds when the node is reserved in exclusive mode, but memory binding options are not supported. The `cgroup` information is used by node health and to perform memory compaction after the task has completed.

- `TaskPlugin=task/affinity,task/cray,task/cgroup`

  When using this option, set `TaskAffinity=no` in `cgroup.conf`. When exclusive mode is used, both CPU and memory binding succeed.

7.2 Network Performance Counters

Request access to blade-level or system-level network performance counters by providing the `--network=`*`blade`* or `--network=`*`system`* argument to `srun`, `sbatch`, or `salloc`. When access to performance counters is requested, the `--exclusive` argument too must be used. For more information on using network performance counters, see *Aries Hardware Counters*.

7.3 `viewcookies` Command

The `viewcookies` command provides information about cookies that are managed by the `ncmd` daemon. Plugins make common library calls to request cookies per application launch and release them on exit. Cookie information, including protection keys, is provided through Slurm to the assigned compute nodes.

If issues are suspected with `ncmd`'s use of network cookies, the `viewcookies` command can be used to view the currently allocated cookies. It must be run as root, with the `alpscomm` module loaded. By default, it displays all allocated cookies, along with their owner, domain, and an expiration/reuse time. The results can be filtered by Owner and Domain with the `--owner` and `--domain` arguments, respectively.

If the Type is "Expired", the cookie is not returned to the pool of available cookies until the time in the "Until" column. If the Type is "Allocated", the cookie is currently allocated to a job and expires at the time in the "Until" column. If the type is "Infinite", the cookie never expires, and can only be explicitly released. The current implementation sets infinite cookie leases and explicitly releases cookies when the job is complete.

Syntax

```
# viewcookies [-h,--help] [-d,--domain domain] [-o,--owner owner]
```

Table 1. viewcookies Command Field

Field	Shows
[-h, --help]	Displays help message showing these options, and exits.
[-d *domain*, --domain *domain*]	Identifier provided by the requesting process. For Slurm, this value is the batch job ID and any non-zero step ID in the following format: **Step ID * 10000000 + Job ID** **Example:** **Step ID: 1** **Job ID: 456** **Domain: 10000456**
[-o *owner*,] --owner *owner*]	The process that reserved the cookie. Currently this shows Slurm but could show a different workload manager in future implementations.

Example 1

This output shows cookie numbering while there are "Expired" cookies plus currently used cookies, followed by output showing numbers after the expired cookies are removed from the actively managed list of cookies.

```
galaxy:/home/users/smith # module load alpscomm
galaxy:/home/users/smith # viewcookies
Owner     Domain      Cookie    Id       Type                  Until
Slurm      141339    8388608     1     Expired    2013-12-02T13:51:42
Slurm      141339    8454144     2     Expired    2013-12-02T13:51:42
Slurm      141341    8519680     3     Expired    2013-12-02T13:51:38
Slurm      141341    8585216     4     Expired    2013-12-02T13:51:38
Slurm 10000141341    8650752     5     Expired    2013-12-02T13:51:45
Slurm 10000141341    8716288     6     Expired    2013-12-02T13:51:45
Slurm      141342    8781824     7    Infinite                Never
Slurm      141342    8847360     8    Infinite                Never
... # elapsed time
galaxy:/home/users/smith # viewcookies
Owner      Domain     Cookie    Id       Type                  Until
Slurm      141342    8781824     7     Infinite                Never
Slurm      141342    8847360     8     Infinite                Never
```

Example 2

In this example, the step ID was zero, so it is not displayed as part of the Domain identifier provided by Slurm.

```
boot-p2:~ # module load alpscomm
boot-p2:~ # viewcookies
Owner   viewcookies  Domain      Cookie    Id         Type      Until
Slurm      82159     8388608     1     Infinite      Never
Slurm      82159     8454144     2     Infinite      Never
```

The command requires privilege to execute and can only be successfully run by root. Any other users see the following error message from `viewcookies`:

```
Error retrieving cookie information: src/lib/alpscomm_sn/cookie.c:1329
Multiple errors while trying to create a socket:
#1 (src/lib/alpscomm_sn/cookie.c:1382 socket on /var//opt/cray/ncmd/ncmd.uds: No
such file or directory)
#2 (src/lib/alpscomm_sn/cookie.c:1611 Unable to create socket to sdb:8765)
```

Root users will see the same error shown above if the `ncmd` process is not running either on the **sdb** node or the local node.

Example 3

This ouput shows a line generated by `drcs`, a new system component for obtaining cookies.

```
opal-p2:/opt/cray/rdma-credentials/default/man # viewcookies
   Owner     Domain          Cookie    Id       Type                Until
   SLURM      15939      2461728768 37436    Expired  2016-05-12T14:15:57
   SLURM      15939      2461794304 37437    Expired  2016-05-12T14:15:57
    drcs          0      2461859840 37438   Infinite                Never
   SLURM      15940      2461925376 37439  Allocated  2016-05-19T14:16:38
   SLURM      15940      2461990912 37440  Allocated  2016-05-19T14:16:38
```

7.4 Scripts

Scripts shown in the following table are automatically invoked for native Slurm on service nodes during system boot.

Table 2. Script Files

File	Starts and Stops this Daemon
`/usr/lib/systemd/system/aeld.service`	`aeld`
`/usr/lib/systemd/system/apptermd.service`	`apptermd`
`/usr/lib/systemd/system/ncmd.service`	`ncmd`
`/user/lib/systemd/system/slurmctld.service`	`slurmctld`

The following script is installed for native Slurm during compute node boot and invoked when the `cgroup` is released.

`/usr/sbin/cpuset_release_agent`

The script is used as the release agent for the `/dev/cpuset cgroup` in native Slurm so that any `cpuset` for a completed job is cleaned up. The file's contents are shown below:

```
#!/bin/sh
/bin/rmdir /dev/cpuset/$1
```

7.5 Logs

Logs used by Slurm and their locations are listed in the following table.

Table 3. Log Files

Logs	Node	Path
`slurmctld`	`sdb`	`/var/spool/slurm/slurmctld.log`
`slurmd`	`compute`	`/var/spool/slurmd/nid`*xxxxx*`.log`
`slurmdbd`	`Login`	`/var/log/slurm/slurmdbd.log`
`aeld`	`boot`	`/var/opt/cray/aeld/log/aeld.log`
`apptermd`	`sdb`	`/var/opt/cray/apptermd/log/apptermd.log`
`ncmd`	`sdb`	`/var/opt/cray/ncmd/log/ncmd.log`
`munged`	`all`	`/var/log/munge/munged.log`

Your site must monitor the size of the `slurmctld` and `slurmdbd` log files on service nodes, as well as the `slurmd` log files on compute nodes.

To control file size, the **logrotate** utility is run by default on service nodes. See *Configure logrotate* on page 10.