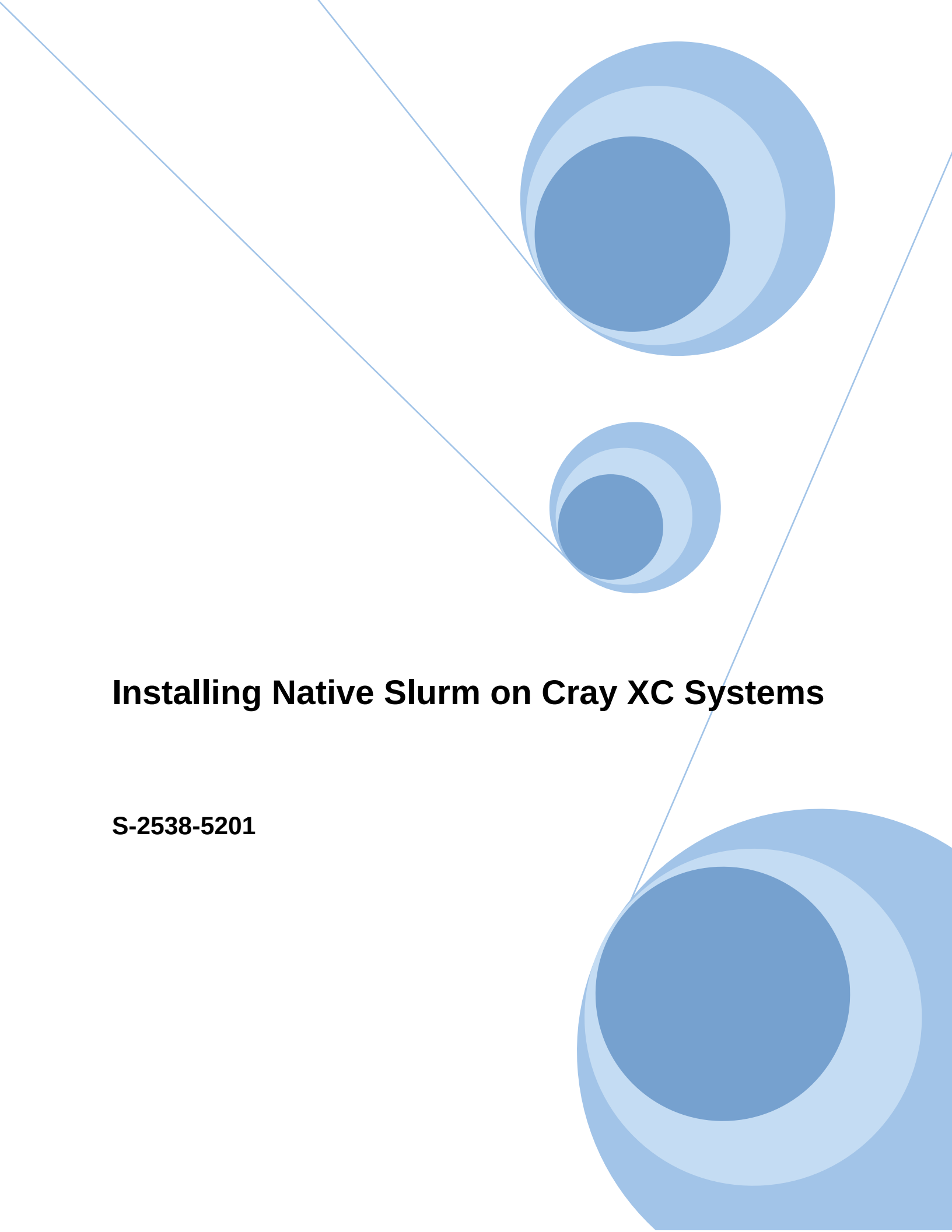

Installing Native Slurm on Cray XC Systems

S-2538-5201

U.S. GOVERNMENT RESTRICTED RIGHTS NOTICE

TRADEMARKS

Direct requests for copies of publications to:

Mail: Cray Inc. Logistics
PO Box 6000
Chippewa Falls, WI 54729-0080
USA
E-mail: spares@cray.com
Fax: +1 715 726 4602

Direct comments about this publication to:

Mail: Cray Inc.
Technical Training and Documentation
P.O. Box 6000
Chippewa Falls, WI 54729-0080
USA
E-mail: ttd_online@cray.com
Fax: +1 715 726 49

Record of Revision

Publication Number	Description
S-2538-5101	Initial publication
S-2538-5201	Documents changes including installing from source and the required **gres.conf** file

Contents

Tables

1. Introduction

Cray provides the required infrastructure to support running the workload manager Slurm natively on Cray systems. SchedMD, the Slurm vendor, will provide any Slurm plug-ins required to run on a Cray system machine. The Slurm-ALPS hybrid model continues to be supported, but only one workload manager at a time can run, either the Slurm-ALPS hybrid or native Slurm.

New daemons

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

- There is one of each daemon per Cray system, not a 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 automatically started through install tool activities, but only if the active WLM is not ALPS (see “Set the active native WLM to Slurm”, page 15).

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 logfiles, 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
```

2. Files

Scripts

Scripts shown in Table 1 are automatically invoked for native Slurm on service nodes during system boot when these services are enabled with **chkconfig**, as shown in "Enable required services", page 12.

Table 1. Script files

File	Starts and stops this daemon:
/etc/init.d/aeld	**aeld** daemon
/etc/init.d/apptermd	**apptermd** daemon
/etc/init.d/ncmd	**ncmd** daemon
/etc/init.d/slurm	**slurmctld** daemon
/etc/init.d/slurmdbd	**slurmdbd** daemon

The following scripts are automatically invoked for native Slurm on compute nodes during compute node boot.

- `/etc/init.d/cnrte`

  Creates or removes required mount points

- `/etc/init.d/wlm_switch`

  Starts and stops all services listed within `/etc/opt/cray/cnrte/compute-dsl-services.conf` (currently **dbus**, **rpcbind**, **nscd**, and **munge**) by invoking the applicable `/etc/init.d` script and starts **slurmd** by calling `/etc/init.d/slurm`. The **wlm_switch** script is run within the **initramfs**, executing **compute-dsl-services** and `/etc/init.d/slurm` from the **chroot** environment.

- Following are the contents of the **compute-dsl-services.conf** file:

```
# compute-dsl-services.conf
#
# Cray WLM Compute Node DSL Service List
# Copyright 2013 Cray Inc. All Rights Reserved.
#
/etc/init.d/dbus
/etc/init.d/rpcbind
/etc/init.d/nscd
/etc/init.d/munge
```

- `/usr/sbin/cpuset_release_agent`

  (Installed by the **wlm_switch** RPM.) 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
```

Logs

Logs and used by Slurm and their locations are listed in Table 2.

Table 2. Log files

Logs	Node	Path
slurmctld	sdb node	`/var/spool/slurm/slurmctld.log`
slurmd	compute node	`/var/spool/slurmd/nid`*xxxxx*`.log`
slurmdbd	Login node	`/var/spool/slurm/slurmdb.log`

The native Slurm support daemons write logs to the following:

- `/var/opt/cray/aeld/log/aeld.log` on the **boot** node.
- `/var/opt/cray/apptermd/log/apptermd.log` on the **sdb** node.
- `/var/opt/cray/ncmd/log/ncmd.log` on the **sdb** node.
- `/var/log/munge/munged.log` on all nodes.

IMPORTANT: Your site must monitor the size of the **slurmctld** and **slurmdbd** logfiles on service nodes, as well as the **slurmd** logfiles on compute nodes.

Using logrotate to control file size

Cray recommends using logrotate to control the size of Slurm log files. To set up logrotate, in **xtopview** on the boot node, run

1. Run:

   ```
   /sbin/chkconfig --add cron
   ```

2. `Add /etc/init.d/cron` to `/etc/opt/cray/cnrte/compute-dsl-services.conf`.

3. Enter:

   ```
   chmod +x /etc/cron.daily/logrotate
   ```

4. Set up `/etc/logrotate.d/slurm`. An example file can be found in the **slurm.conf** man page.

Note that the log file should be owned by root, and that the log paths are different from those shown in the example.

The **slurm.conf** man page provides an example configuration file.

3. Installation

The basic installation of Slurm consists of obtaining and installing RPMs.

Installation Instructions

To build and install Slurm to the Cray, you must first have at least version 14.11.0-pre1 installed. Then do the following:

1. Obtain the latest Slurm source from SchedMD at:

   http://www.schedmd.com/#repos

2. Build RPMs by entering the following commands:

```
crayadm@smw:~/> scp slurm-14.03.3-2.tar.bz2 root@boot:/usr/src/packages/SOURCES/
crayadm@smw:~/> ssh root@boot
boot:~ # rpmbuild --with cray --with debug \
    --define "_slurm_sysconfdir /etc/opt/slurm" \
    --define "_prefix /opt/slurm/14.03.3-2" \
    -ta /usr/src/packages/SOURCES/slurm-14.03.3-2.tar.bz2
```

3. Install RPMs by entering the following commands:

```
boot:~ # cp /usr/src/packages/RPMS/x86_64/slurm*.rpm /rr/current/software/
boot:~ # xtopview
default/:/ # rpm -ivh /software/slurm*.rpm
default/:/ # update-alternatives --install /opt/slurm/default slurm
    /opt/slurm/14.03.3-2 140303
default/:/ # exit
```

4. Configuration

Enable required services

Native Slurm requires the following services to be running:

- **slurm**: Starts the **slurmctld**, which provides scheduling, placement, and accounting. See the final paragraph following this list.
- **munge**: Must be running on all service nodes where Slurm daemons are running or where Slurm commands will be used. **Munge** is used for user authentication for Slurm commands.
- **slurmdbd**: Required if you use an accounting database. Starts the daemon, **slurmdbd**, that communicates with the accounting database. Because the database is off the Cray system, **slurmdbd** must run on a login node to allow communication. Install as shown below.

To ensure that the above services are running, run the following on the boot node.

- The first **xtopview** command and **chkconfig** calls enables the `/etc/init.d/`*xxx* scripts to be executed as part of the system boot activity. The corresponding scripts start the **slurm** and **munge** daemons.
- boot activity. The corresponding scripts will start the slurm and munge daemons.
- The second **xtopview** command and **chkconfig** installs **slurmdbd** only on the specified login node.

```
boot:~ # xtopview
default/:/ # chkconfig munge on
default/:/ # chkconfig slurm on
default/:/ # exit
```

```
boot:~ #  xtopview -n node_id_of_login_node
default/:/ # chkconfig slurmdbd on
default/:/ # exit
```

slurmctld, activated by **slurm**, must run on a service node accessible to all login nodes; Cray recommends that you use the **sdb** node for this purpose. This is controlled by the **ControlMachine** value in **slurm.conf**; on that node, the **slurm** startup script starts only **slurmctld**.

The `/etc/opt/cray/cnrte/compute-dsl-services.conf` file contains a list of services to be started on each compute node within the shared root upon boot.

Slurm configuration files

Use the following steps to create and configure **slurm.conf**, and edit **topology.conf** and **cgroup.conf**:

1. Create a **slurm.conf** file as follows: In **xtopview** on the boot node, edit `/etc/opt/slurm/slurm.conf.template` and update any configuration parameters you wish to change. Values enclosed in curly brackets are automatically replaced by the **slurm.conf** generator script.

   Following are values to use in **slurm.conf**:

   – Required:

   - `SlurmUser=root`
   - `SwitchType=switch/cray`
   - `TaskPlugin=task/cgroup,task/cray`
   - `SelectType=select/cray`
   - `CoreSpecPlugin=cray`
   - `JobSubmitPlugins=cray`
   - `ProctrackType=proctrack/cray`
   - `SelectTypeParameters` can have any of the following values:
     - `CR_CPU_Memory, other_cons_res` (use hyperthreads by default)
     - `CR_Core_Memory, other_cons_res` (don't use hyperthreads by default)

   – Recommended:

   - `ControlMachine=`*sdb_hostname*
   - `AuthType=auth/munge`
   - `CryptoType=crypto/munge`
   - `DefMemPerCPU=` *minimum (total node memory)/(cores) for all nodes*
   - `MaxMemPerCPU=` *maximum (total node memory) for all nodes*
   - `MessageTimeout=60`
   - `PropagateResourceLimits=ALL`

2. Run the following script to write the configuration:

   ```
   /opt/slurm/default/bin/slurmconfgen.py >/etc/opt/slurm/slurm.conf
   ```

3. Note that the configuration file generator does not perform any topological reordering of nodes. Cray recommends that you set up a hierarchical topology with each "switch" (a group of four compute nodes that share an Aries switch, configured as a switch in file **topology.conf**). Instructions for setting up topology can be found at the following URL:

   http://slurm.schedmd.com/topology.html

4. Edit `/etc /opt/slurm/default/sbin/slurmconfgen.py etc/opt/slurm/cgroup.conf` and set these parameters:

   ```
   CgroupAutomount=yes
   CgroupMountpoint="/dev"
   ConstrainCores=yes
   ConstrainRAMSpace=yes
   TaskAffinity=yes
   ```

   In the last line above, the **yes** value for **TaskAffinity** enables the `srun --cpu_bind` option and requires **hwloc** to be installed on the service node image. Cray requires the use of the **task/cgroup** plugin for compute-node cleanup.

5. Create **/etc/opt/slurm/gres.conf** with these contents:

   ```
   Name=craynetwork Count=4
   ```

   You can specify additional generic resources if you wish, but they will be identical for every node. The lines for GPUs and MICs are created on the compute nodes with those resources by **wlm_switch**.

Slurm Affinity options

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/cgroup,task/cray`

  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/cgroup,task/affinity,task/cray`

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

Agreement with wlm_switch script file

Certain values must match between the **slurm.conf** file and those defined within the compute node script `/etc/init.d/wlm_switch`. This script starts Slurm on the compute

node. The following default values within the **wlm_switch** script must match the Slurm **slurm.conf** configuration, or else Slurm will not start on the compute nodes.

```
SLURMD_BIN="/opt/slurm/default/sbin/slurmd"
SLURMD_SPOOLDIR="/var/spool/slurmd"
MUNGED_BIN="/usr/sbin/munged"
```

Using an external accounting database

To use an accounting database off the Cray, send the accounting data from **slurmctld** on the **sdb** node through a node with external network access (such as a login node) out to the external node. Cray recommends that you use **slurmdbd** running on a login node for this purpose.

Follow the accounting set up instructions from the Slurm accounting configuration after build section here. Make sure that **AccountingStorageHost** in **slurm.conf** and **DbdHost** in **slurmdbd.conf** are set to the hostname of a node with external network access.

Set the active native WLM to Slurm

Once Slurm is configured on the system, you can set the active native workload manager to Slurm using one of the methods shown in "Setting the value of **active_wlm**", below.

> **NOTE**: The use of native Slurm requires a change to the default value of the **active_wlm** boot parameter.

The **active_wlm** boot parameter identifies the current application launch and management services provider. Currently, the only options are ALPS or Slurm. This parameter is used by parts of Cray code that perform differently based on whether ALPS or Slurm is the system's native workload manager. The default value is ALPS, which must be overridden for native Slurm to function correctly. This value affects the following:

- **wlm_switch** startup script on the compute nodes
- Some **/etc/init.d** scripts
- Application information provided to Cray PMI
- Node Health Checker

Determining the current value of **active_wlm**

To query the current value of the **active_wlm** parameter, use the following command from any node on the Cray:

```
/opt/cray/wlm_detect/default/bin/wlm_detect
SLURM
```

Setting the value of **active_wlm**

When the WLM is not ALPS, you can set the **active_wlm** value to an appropriate WLM value such as Slurm. To set the parameter, do one of the following:

- Change the string `active_wlm=ALPS` to `active_wlm=SLURM` in these files:

File:	In:
`/boot/parameters-snl`	Service node image
`/boot/parameters-cnl`	Compute node image

  Note: After making theses changes, you must regenerate the compute node boot image.

- Edit the autoboot file as below:

```
# Set the active native WLM to Slurm
lappend actions [list crms_boot_loadfile SNL0 bootnode $data(idlist) linux active_wlm=Slurm]
lappend actions {crms_boot_sdb_loadfile SNL0 active_wlm=Slurm}
lappend actions [list crms_boot_loadfile SNL0 service $data(idlist) linux active_wlm=Slurm]
lappend actions [list crms_boot_loadfile CNL0 compute $data(idlist) linux active_wlm=Slurm]
```

If **active_wlm** has the wrong value

If **active_wlm** is set to ALPS on a native Slurm system, the **ncmd**, **aeld**, **apptermd**, and **slurmd** daemons do not start, and **sinfo** shows all nodes as down. Messages like the following appear in the console log:

```
aeld not configured to start (active native WLM is ALPS)
```

5. Troubleshooting

This section shows the use of the **viewcookies**, **totalview**, and **DDT** utilities with Slurm.

NOTE: On a native Slurm system, do not use **xtprocadmin** to set the compute node state. Slurm has no knowledge of node states set by **xtprocadmin**, so jobs can still be scheduled on nodes downed by **xtprocadmin**. Instead, use **scontrol** (see the **scontrol** man page for details).

Viewcookies command

If issues are suspected with the use of network cookies by **ncmd**, 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, the command displays all allocated cookies, with columns for their owners, domains, and expiration and reuse times. You can filter the results using the **Owner** and **Domain** values by using the `--owner` and `--domain` arguments, respectively. Table 3 lists fields available with the `viewcookies` command.

Table 3. Viewcookies command field

Field	Shows
Owner	The process that reserved the cookie. Currently this will show Slurm but could show a different workload manager in future implementations.

Field	Shows

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`

Cookie	Identifier code for the cookie. The cookie contains a unique pKey.
ID	Counter of the cookies actively assigned at that moment. Currently, Aries requires two cookies to be assigned per app launch
Type	Expiration time, using the following values: **Expired**: The cookie is not returned to the pool of available cookies until the time in the Until column. **Allocated**: The cookie is currently allocated to a job and expires at the time in the Until column. **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.
Until	Duration for the cookie to be reused.

Example 1. The following example 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 # 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 the following example, the step ID was zero, so it is not displayed as part of the **Domain** identifier provided by Slurm.

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

The **viewcookies** 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:1327 Multiple errors
   while trying to create a socket: #1 (src/lib/alpscomm_sn/
ookie.c:1380 socket on /var//opt/cray/alpscomm/ncmd.uds: No such file or
directory) #2 (src/lib/alpscomm_sn/cookie.c:1609 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.

Running Totalview with Slurm

To run **totalview** with Slurm :

1. Create a `$HOME/.tvdrc` file with the following contents:

```
if {![catch {exec scontrol ping >& /dev/null}]} {
#
# Enable debug server bulk launch: Checked
dset -set_as_default TV::bulk_launch_enabled true

# Command:
# Beginning with TV 7X.1, TV supports Slurm and %J.
dset -set_as_default TV::bulk_launch_string {srun --mem-per-cpu=0 -
    N%1 -n%N -w`awk -F. 'BEGIN {ORS=","} {if (NR==%N) ORS=""; print
    $1}' %t1` -l --input=none %B/tvdsvr%K -callback_host %H -
    callback_ports %L -set_pws %P -verbosity %V  -working_directory %D
    %F}

# Temp File 1 Prototype:
# Host Lines:
# Slurm NodeNames need to be unadorned hostnames. In case %R returns
# fully qualified hostnames, list the hostnames in %t1 here, and use
# awk in the launch string above to strip away domain name suffixes.
dset -set_as_default TV::bulk_launch_tmpfile1_host_lines {%R}
}
```

2. Invoke **totalview**:

```
totalview srun -a -n 16 -N 4 ./mpit
```

Slurm is used to launch both the app and TV:

```
4 S root 9932 1 0 80 0 - 45578 wait 11:32 ?  00:00:00 slurmstepd: [122359.0]
4 t dah 9947 9932 0 80 0 - 34474 ptrace 11:32 ?  00:00:00 _ /ufs/home/users/dah/./mpit
4 t dah 9948 9932 0 80 0 - 34474 ptrace 11:32 ?  00:00:00 _ /ufs/home/users/dah/./mpit
4 t dah 9949 9932 0 80 0 - 34474 ptrace 11:32 ?  00:00:00 _ /ufs/home/users/dah/./mpit
4 t dah 9950 9932 0 80 0 - 34474 ptrace 11:32 ?  00:00:00 _ /ufs/home/users/dah/./mpit
4 S root 9953 1 0 80 0 - 45512 wait 11:32 ?  00:00:00 slurmstepd: [122359.1]
4 S dah 9959 9953 0 80 0 - 9317 poll_s 11:32 ?  00:00:00 _
   /opt/toolworks/totalview.8.12.0-1/linux-x86-64/bin/..
```

Running DDT with Slurm

Run DDT as follows:

1. Use the following commands:

```
module load ddt
salloc -n xx ....
ddt ./executable
```

2. Do the following in the **Run** window:

   a. Change the MPI implementation to Slurm.

b. Clear the submit job through the queue in the job submission tab.

c. Click **OK**.

d. Click **Run**.

DDT uses **srun** to launch the app and DDT:

```
4 S root 8612    1 0 80 0 - 45518 wait   11:13 ? 00:00:00 slurmstepd: [122334.3]
4 S dah  8618 8612 0 80 0 - 13032 poll_s 11:13 ? 00:00:00  \_ /opt/cray/ddt/4.1.1.0/bin/ddt-debugger --ddthost gala
0 S dah  8639 8618 0 80 0 - 21761 poll_s 11:13 ? 00:00:00  |   \_ /opt/cray/ddt/4.1.1.0/libexec/gdb --fullname --nx
0 t dah  8645 8639 0 80 0 - 34475 ptrace 11:13 ? 00:00:00  |       \_ /ufs/home/users/dah/mpit
4 S dah  8619 8612 0 80 0 - 13033 poll_s 11:13 ? 00:00:00  \_ /opt/cray/ddt/4.1.1.0/bin/ddt-debugger --ddthost gala
0 S dah  8627 8619 0 80 0 - 10598 poll_s 11:13 ? 00:00:00  |   \_ /opt/cray/ddt/4.1.1.0/libexec/ddt-treeserver --dd
0 S dah  8632 8619 0 80 0 - 21761 poll_s 11:13 ? 00:00:00  |   \_ /opt/cray/ddt/4.1.1.0/libexec/gdb --fullname --nx
0 t dah  8634 8632 4 80 0 - 34475 ptrace 11:13 ? 00:00:04  |       \_ /ufs/home/users/dah/mpit
4 S dah  8620 8612 0 80 0 - 13826 poll_s 11:13 ? 00:00:00  \_ /opt/cray/ddt/4.1.1.0/bin/ddt-debugger --ddthost gala
0 S dah  8638 8620 0 80 0 - 21761 poll_s 11:13 ? 00:00:00  |   \_ /opt/cray/ddt/4.1.1.0/libexec/gdb --fullname --n
0 t dah  8644 8638 0 80 0 - 34475 ptrace 11:13 ? 00:00:00  |       \_ /ufs/home/users/dah/mpit
4 S dah  8621 8612 0 80 0 - 13032 poll_s 11:13 ? 00:00:00  \_ /opt/cray/ddt/4.1.1.0/bin/ddt-debugger --ddthost gala
0 S dah  8637 8621 0 80 0 - 21761 poll_s 11:13 ? 00:00:00      \_ /opt/cray/ddt/4.1.1.0/libexec/gdb --fullname --nx
0 t dah  8643 8637 0 80 0 - 34475 ptrace 11:13 ? 00:00:00          \_ /ufs/home/users/dah/mpit
```