

# SLURM Operation on Cray XT and XE



Morris Jette  
jette@schedmd.com

SchedMD LLC

# Contributors and Collaborators



This work was supported by the Oak Ridge National Laboratory Extreme Scale Systems Center.



Swiss National Supercomputing Centre performed some of the development and testing.



Cray helped with integration and testing.

# Outline



- Cray hardware and software architecture
- SLURM architecture for Cray
- SLURM configuration and use
- Status

# Cray Architecture



- Many of the most powerful computers built by Cray
- Nodes are diskless
- 2 or 3-dimension torus interconnect
  - Multiple nodes at each coordinate on some systems
- Full Linux on front-end nodes
- Lightweight Linux kernel on compute nodes
- Whole nodes must be allocated to jobs

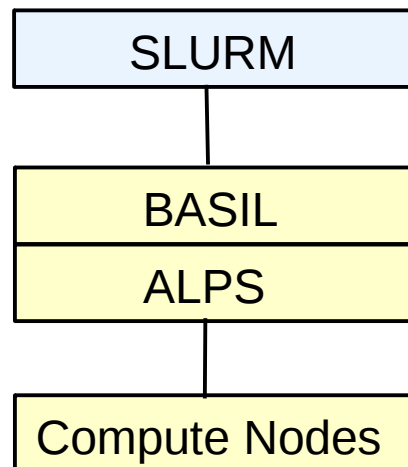
# ALPS and BASIL



- **ALPS – Application Level Placement Scheduler**
  - Cray's resource manager
  - Six daemons plus variety of tools
    - One daemon runs on each compute node to launch user tasks
    - Other daemons run on service nodes
  - Rudimentary scheduling software
    - Dependent upon external scheduler (e.g. SLURM, etc) for workload management
- **BASIL – Batch Application Scheduler Interface Layer**
  - XML interface to ALPS

# SLURM Architecture for Cray

- Many tools dependent upon ALPS
  - Use SLURM as scheduler layer above ALPS and BASIL, not a replacement



# SLURM and ALPS Functionality



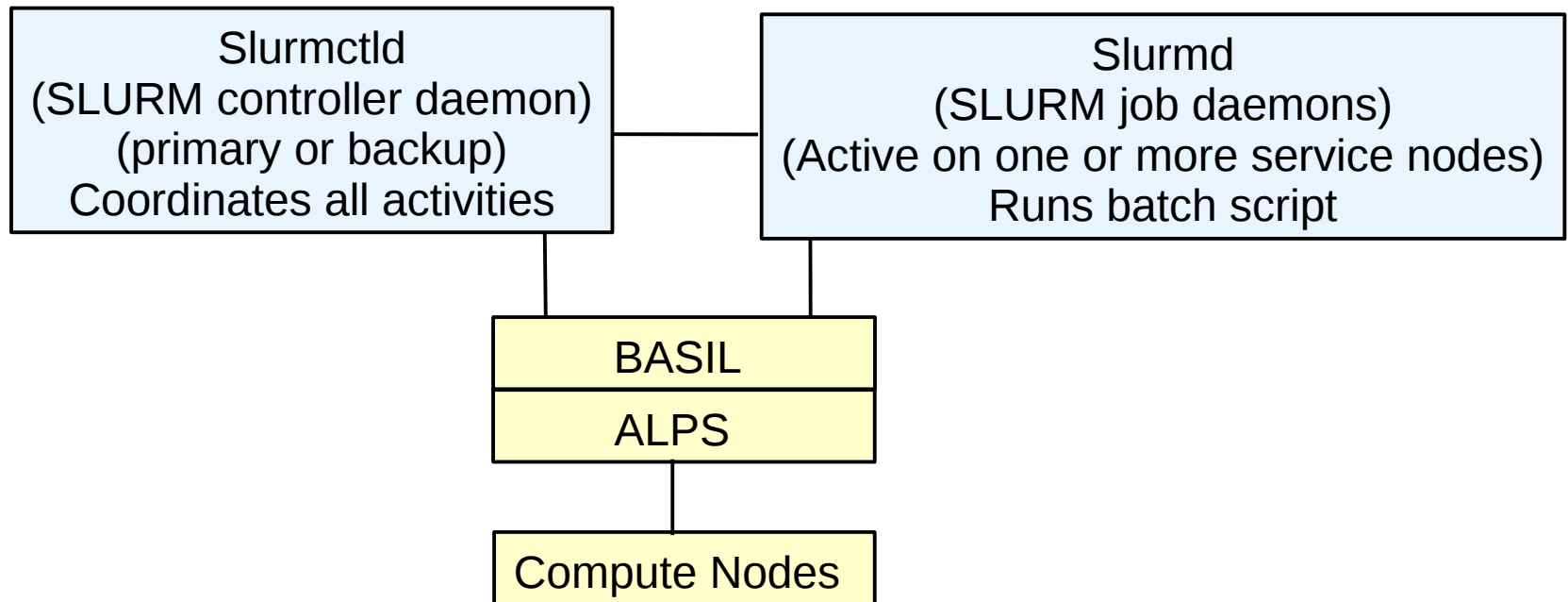
- SLURM

- Prioritizes queue(s) of work and enforces limits
- Decides when and where to start jobs
- Terminates job when appropriate
- Accounts for jobs

- ALPS

- Allocates and releases resources for jobs
- Launches tasks
- Monitors node health

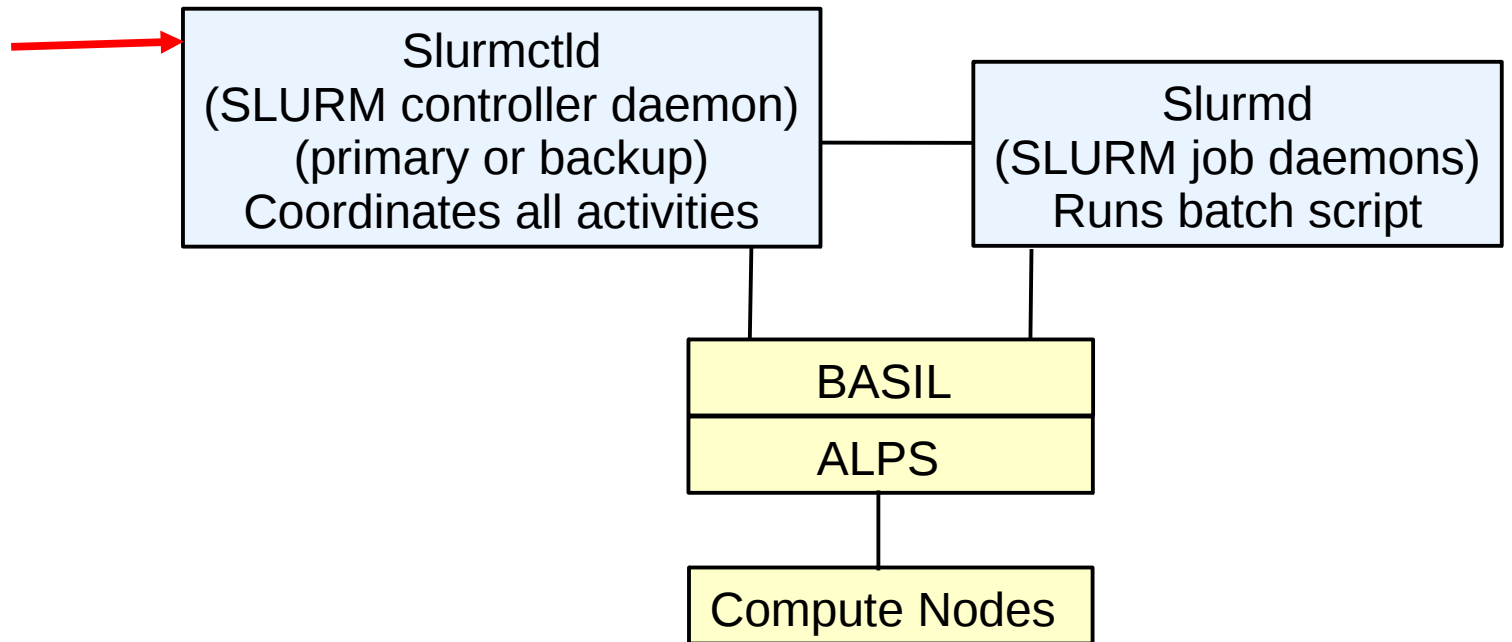
# SLURM Architecture for Cray (Detailed)





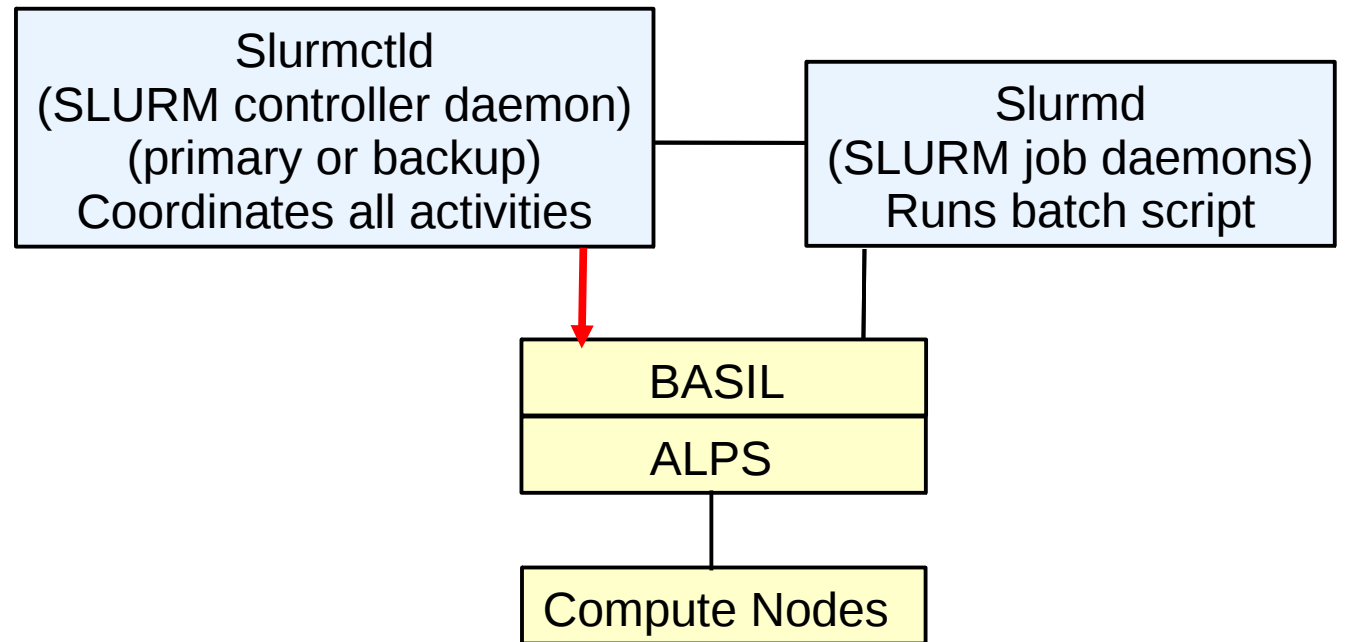
# Job Launch Process

1. User submits script



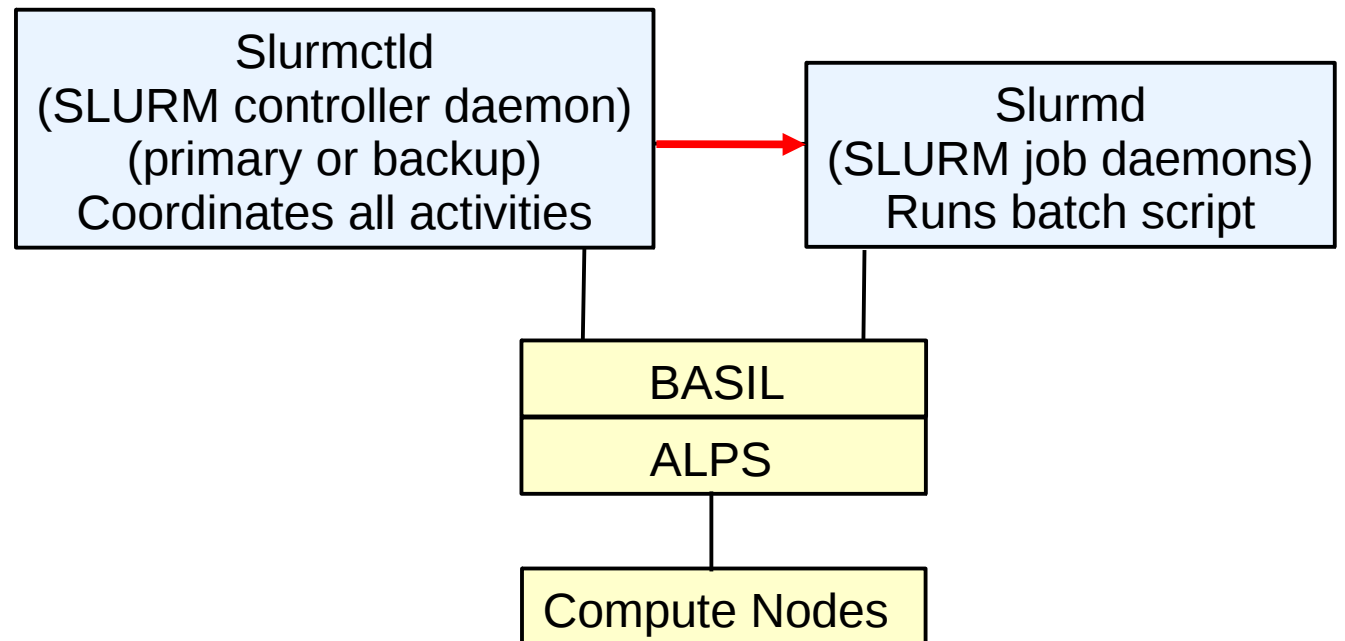
# Job Launch Process

1. User submits script
2. Slurmctld creates ALPS reservation



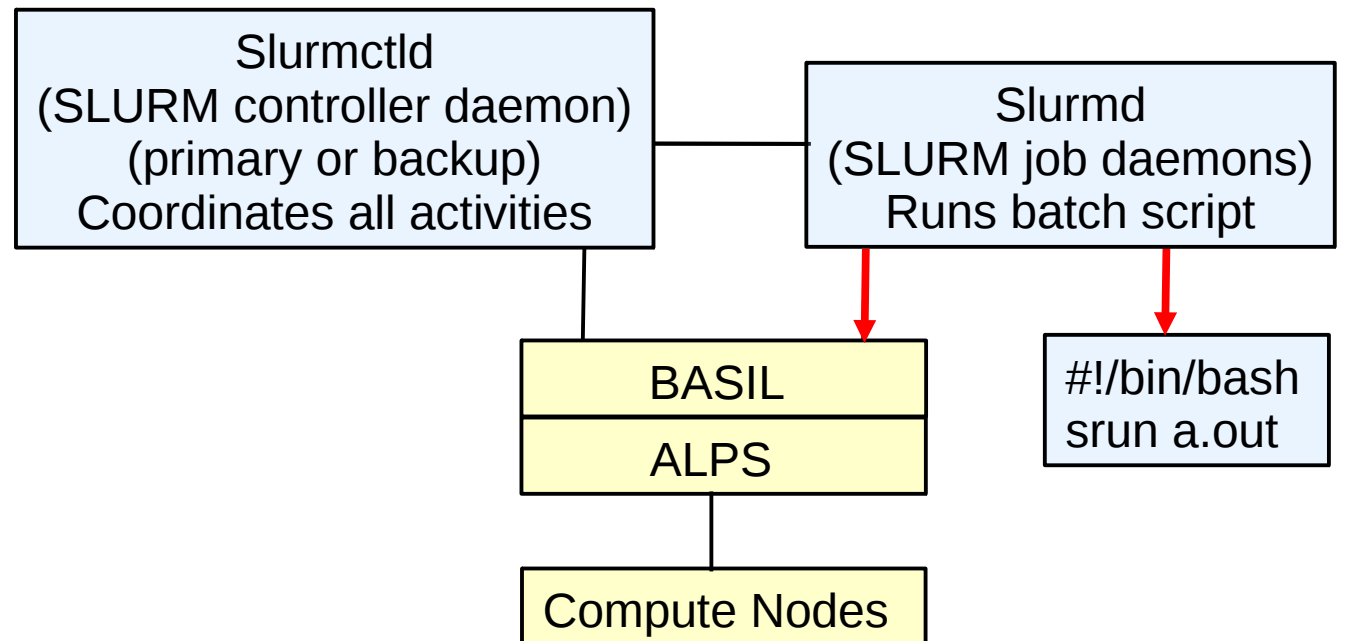
# Job Launch Process

1. User submits script
2. Slurmctld creates ALPS reservation
3. Slurmctld sends script to slurmd



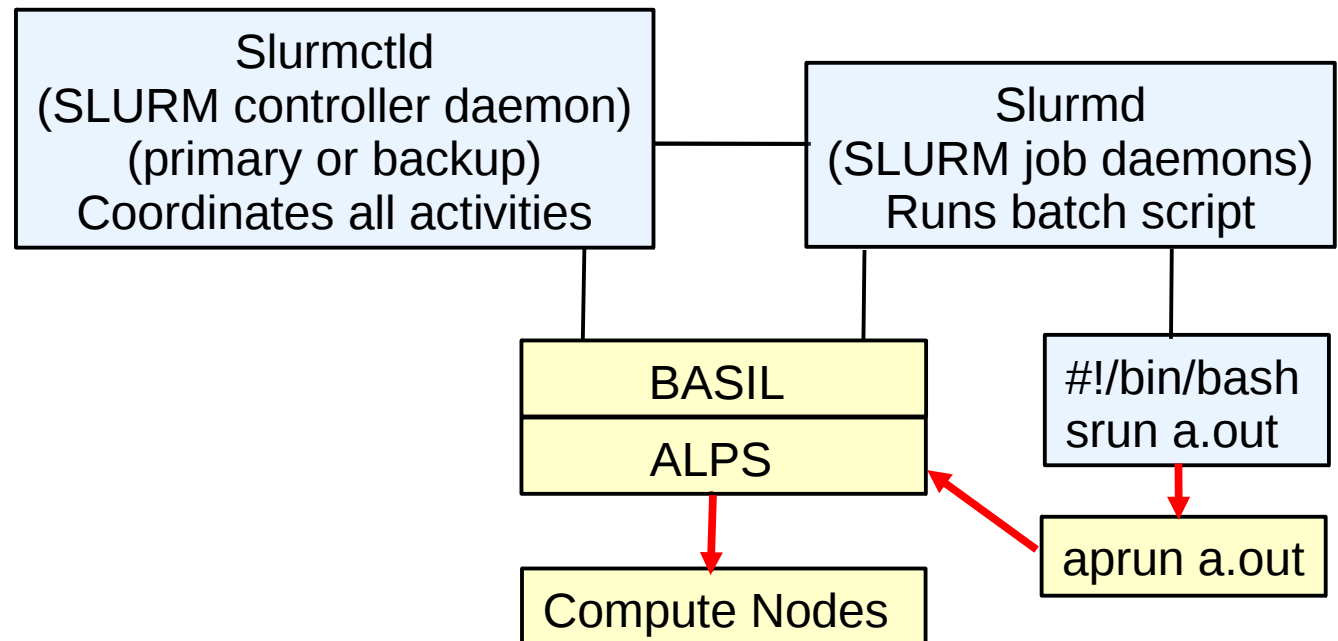
# Job Launch Process

1. User submits script
2. Slurmctld creates ALPS reservation
3. Slurmctld sends script to slurmd
4. Slurmd claims reservation for specific session ID and launches interpreter for script



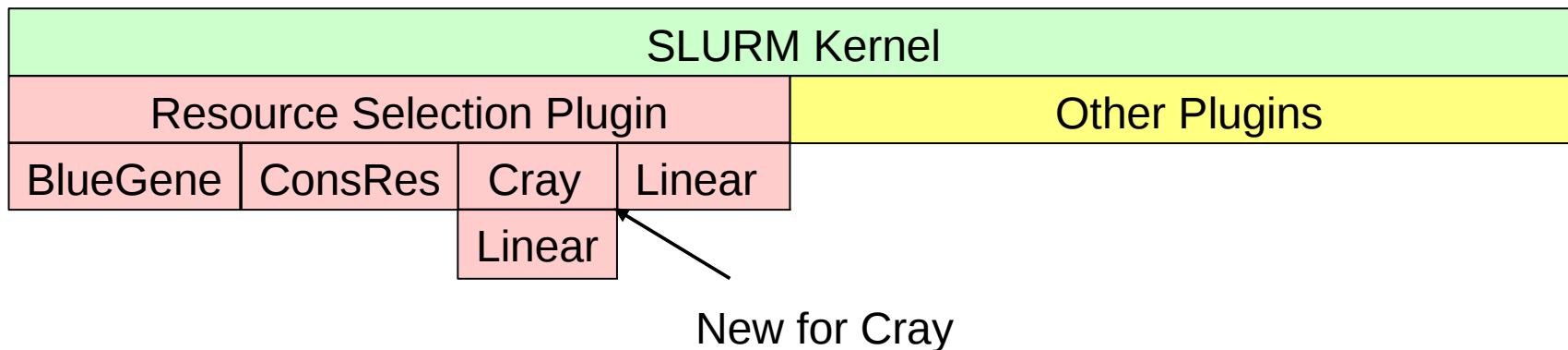
# Job Launch Process

1. User submits script
2. Slurmctld creates ALPS reservation
3. Slurmctld sends script to slurmd
4. Slurmd claims reservation for specific session ID and launches interpreter for script
5. aprun (optionally using the srun wrapper) launches tasks on compute nodes

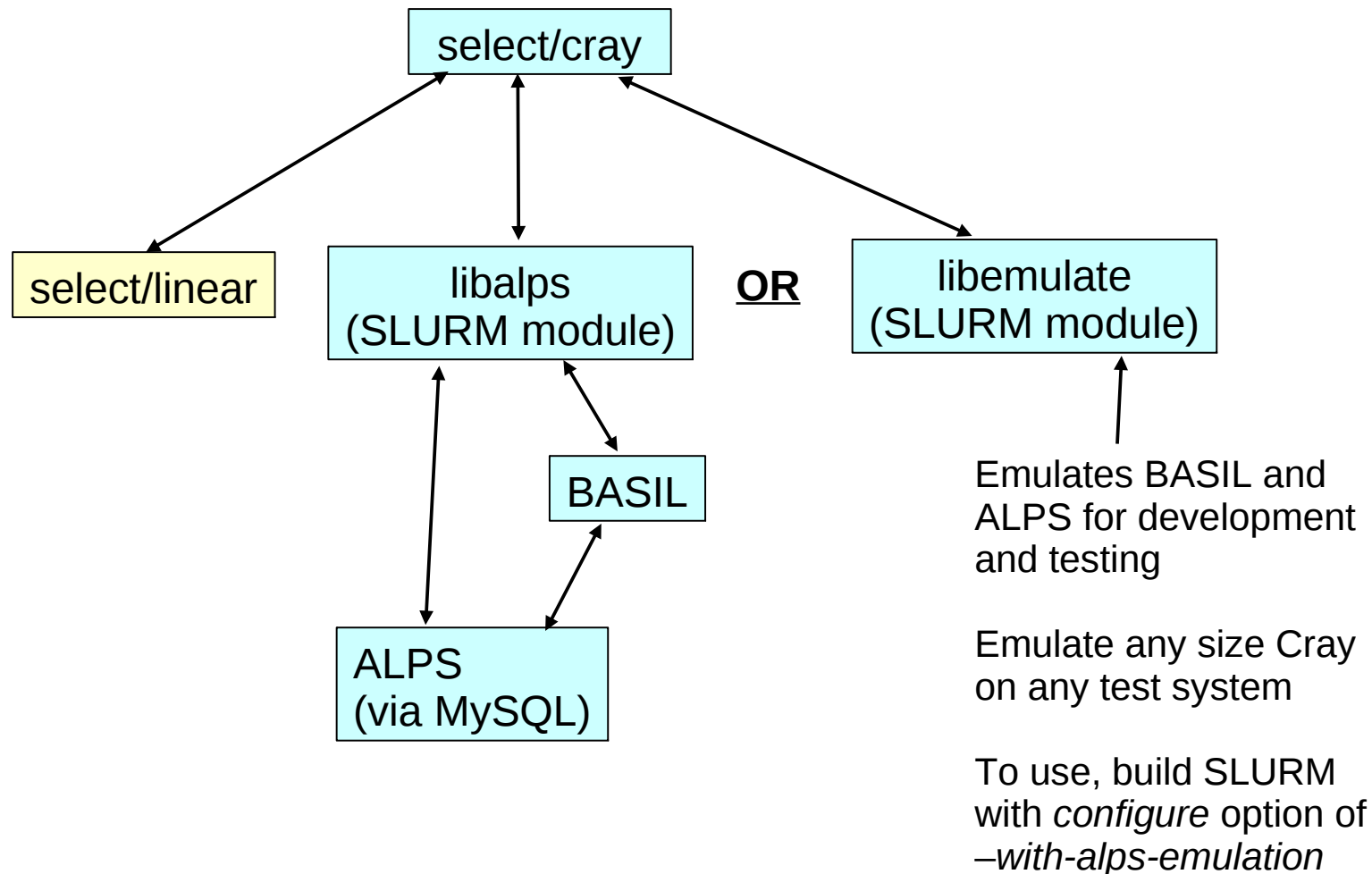


# SLURM Architecture for Cray

- Almost all Cray-specific logic is in a Resource Selection plugin (as SLURM does for IBM BlueGene systems)
- The *select/cray* plugin in-turn calls the *select/linear* plugin to provide full-node resource allocation support including job preemption, memory allocation, optimized topology layout, etc.



# SLURM's *select/cray* plugin



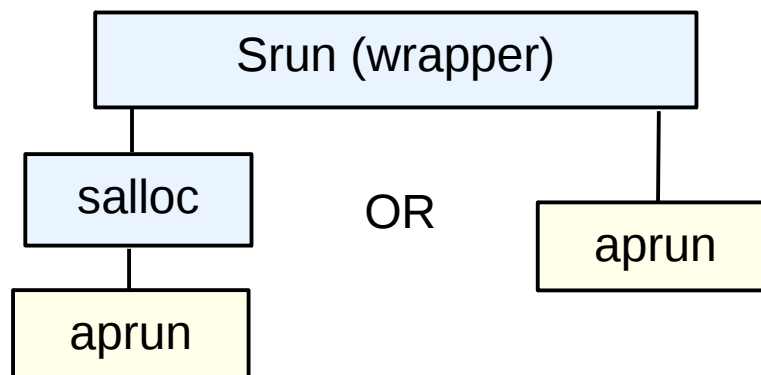
# SLURM Configuration

```
#  
# Sample slurm.conf file for Cray system  
# Selected portions  
#  
SelectType=select/cray          # Communicates with ALPS  
#  
FrontEndName=front[00-03]      # Where slurmd daemons run  
NodeName=nid[00000-01023]  
PartitionName=debug Nodes=nid[00000-00015] MaxTime=30  
PartitionName=batch Nodes=nid[00016-01023] MaxTime=24:00:00
```



# *srun* Command

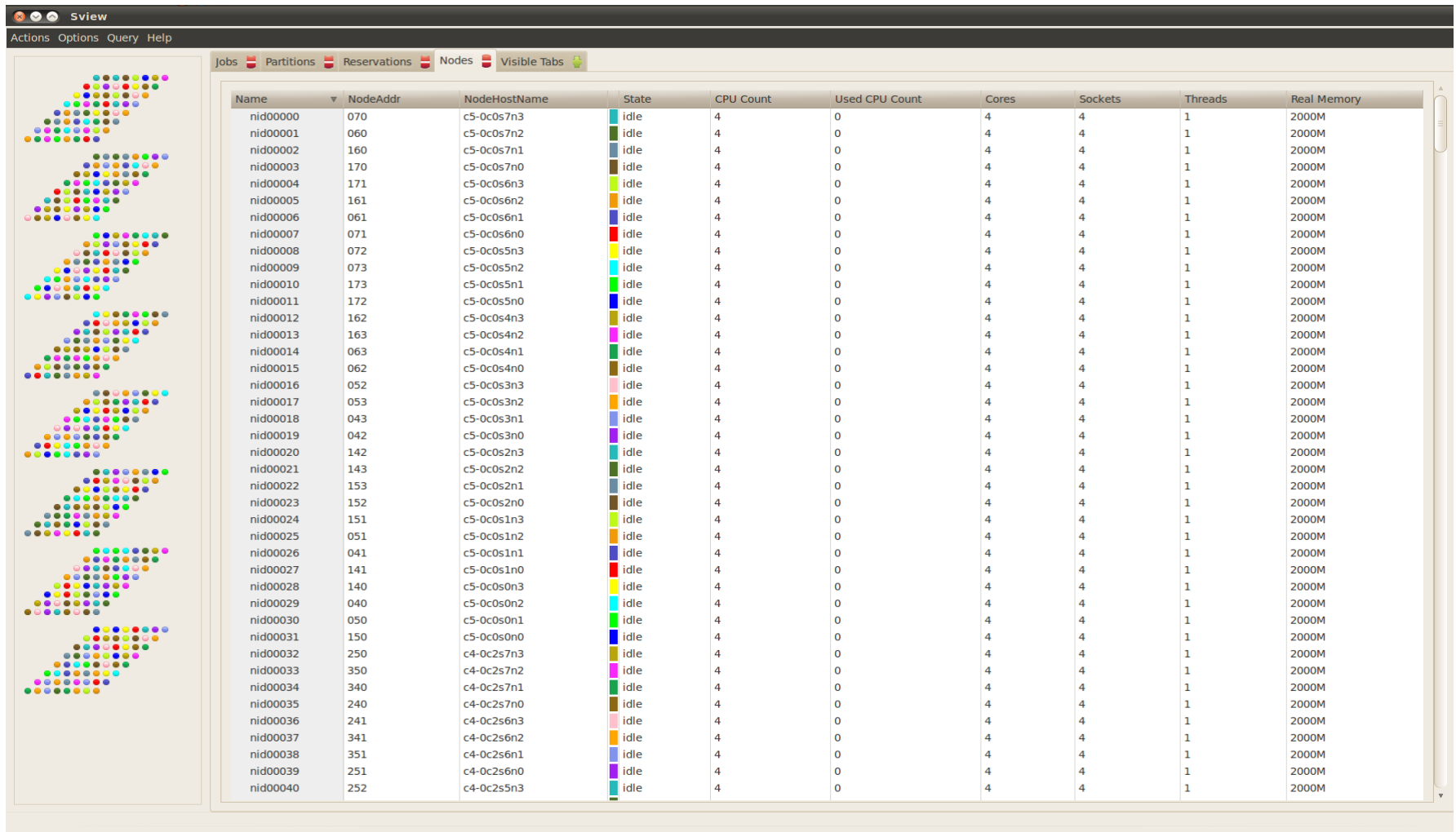
- SLURM has *srun* command for Cray systems that is a wrapper for both *salloc* (to allocate resources as needed) and *aprun* (to launch tasks)
- Options are translated to the extent possible
- Build SLURM with *configure* option *-with-srun2aprun* to build wrapper
  - Otherwise *srun* command advises use of *aprun* and exits



# *srun* Options

- If no allocation exists, the *srun* options are translated directly to *salloc* options to create a job allocation
  - Many *srun* option only apply when a job allocation is created
- After an allocation is created
  - Most common options are translated from *srun* to *aprun* (task count, node count, time limit, file names, support for multiple executables, etc.)
  - Some *srun* options lack *aprun* equivalent and vice-versa
  - *srun*'s “*-alps=*” option can pass any other options to *aprun*
- SLURM environment variables are not currently set
- There are fundamental differences in I/O
  - For example, ALPS does not support per-rank I/O streams

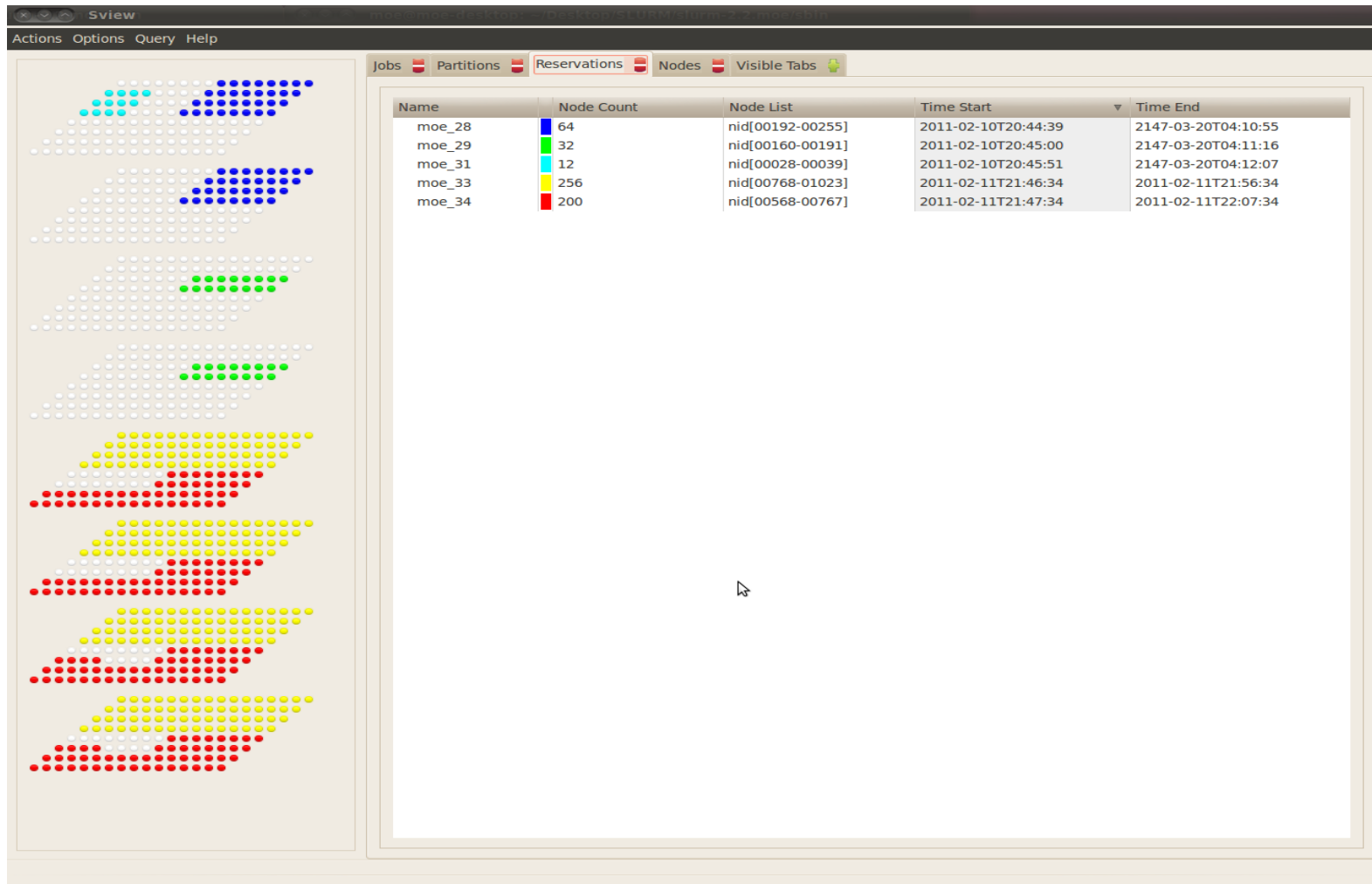
# *sview* of Emulated System



The screenshot displays the *sview* application window. On the left side, there is a grid of colored dots representing a system topology. On the right side, a table lists system metrics for various nodes.

Name	NodeAddr	NodeHostName	State	CPU Count	Used CPU Count	Cores	Sockets	Threads	Real Memory
nid00000	070	c5-0c0s7n3	idle	4	0	4	4	1	2000M
nid00001	060	c5-0c0s7n2	idle	4	0	4	4	1	2000M
nid00002	160	c5-0c0s7n1	idle	4	0	4	4	1	2000M
nid00003	170	c5-0c0s7n0	idle	4	0	4	4	1	2000M
nid00004	171	c5-0c0s6n3	idle	4	0	4	4	1	2000M
nid00005	161	c5-0c0s6n2	idle	4	0	4	4	1	2000M
nid00006	061	c5-0c0s6n1	idle	4	0	4	4	1	2000M
nid00007	071	c5-0c0s6n0	idle	4	0	4	4	1	2000M
nid00008	072	c5-0c0s5n3	idle	4	0	4	4	1	2000M
nid00009	073	c5-0c0s5n2	idle	4	0	4	4	1	2000M
nid00010	173	c5-0c0s5n1	idle	4	0	4	4	1	2000M
nid00011	172	c5-0c0s5n0	idle	4	0	4	4	1	2000M
nid00012	162	c5-0c0s4n3	idle	4	0	4	4	1	2000M
nid00013	163	c5-0c0s4n2	idle	4	0	4	4	1	2000M
nid00014	063	c5-0c0s4n1	idle	4	0	4	4	1	2000M
nid00015	062	c5-0c0s4n0	idle	4	0	4	4	1	2000M
nid00016	052	c5-0c0s3n3	idle	4	0	4	4	1	2000M
nid00017	053	c5-0c0s3n2	idle	4	0	4	4	1	2000M
nid00018	043	c5-0c0s3n1	idle	4	0	4	4	1	2000M
nid00019	042	c5-0c0s3n0	idle	4	0	4	4	1	2000M
nid00020	142	c5-0c0s2n3	idle	4	0	4	4	1	2000M
nid00021	143	c5-0c0s2n2	idle	4	0	4	4	1	2000M
nid00022	153	c5-0c0s2n1	idle	4	0	4	4	1	2000M
nid00023	152	c5-0c0s2n0	idle	4	0	4	4	1	2000M
nid00024	151	c5-0c0s1n3	idle	4	0	4	4	1	2000M
nid00025	051	c5-0c0s1n2	idle	4	0	4	4	1	2000M
nid00026	041	c5-0c0s1n1	idle	4	0	4	4	1	2000M
nid00027	141	c5-0c0s1n0	idle	4	0	4	4	1	2000M
nid00028	140	c5-0c0s0n3	idle	4	0	4	4	1	2000M
nid00029	040	c5-0c0s0n2	idle	4	0	4	4	1	2000M
nid00030	050	c5-0c0s0n1	idle	4	0	4	4	1	2000M
nid00031	150	c5-0c0s0n0	idle	4	0	4	4	1	2000M
nid00032	250	c4-0c2s7n3	idle	4	0	4	4	1	2000M
nid00033	350	c4-0c2s7n2	idle	4	0	4	4	1	2000M
nid00034	340	c4-0c2s7n1	idle	4	0	4	4	1	2000M
nid00035	240	c4-0c2s7n0	idle	4	0	4	4	1	2000M
nid00036	241	c4-0c2s6n3	idle	4	0	4	4	1	2000M
nid00037	341	c4-0c2s6n2	idle	4	0	4	4	1	2000M
nid00038	351	c4-0c2s6n1	idle	4	0	4	4	1	2000M
nid00039	251	c4-0c2s6n0	idle	4	0	4	4	1	2000M
nid00040	252	c4-0c2s5n3	idle	4	0	4	4	1	2000M

# *sview* of Emulated System



# *smap* of Emulated System

```
moe@moe-desktop: ~/Desktop/SLURM/slurm-2.2.moe/bin
File Edit View Terminal Help

BBBBBBBBCCCCCCC
BBBBBBBBCCCCCCC
BBBBBBBBCCCCCCC
BBBBBBBBCCCCCCC
.....EEEEEEEE
.....EEEEEEEE
.....EEEEEEEE
.....EEEEEEEE
.....EEEEEEEE

BBBBBBBBCCCCCCC
BBBBBBBBCCCCCCC
BBBBBBBBCCCCCCC
BBBBBBBBCCCCCCC
.....EEEEEEEE
.....EEEEEEEE
.....EEEEEEEE
.....EEEEEEEE

AAAAAAAADDDDDDD
AAAAAAAADDDDDDD
AAAAAAAADDDDDDD
AAAAAAAADDDDDDD
.....EEEEEEEE
.....EEEEEEEE
.....EEEEEEEE
.....EEEEEEEE

AAAAAAAADDDDDDD
AAAAAAAADDDDDDD
AAAAAAAADDDDDDD
AAAAAAAADDDDDDD
.....EEEEEEEE
.....EEEEEEEE
.....EEEEEEEE
.....EEEEEEEE

Sat Feb 12 19:26:08 2011
ID JOBID PARTITION RESV_ID USER NAME ST TIME NODES NODELIST
A 1993 debug 1 moe tmp R 00:00:31 64 nid[00448-00511]
B 1994 debug 2 moe tmp R 00:00:29 64 nid[00384-00447]
C 1995 debug 3 moe tmp R 00:00:29 64 nid[00320-00383]
D 1996 debug 4 moe tmp R 00:00:28 64 nid[00256-00319]
E 1997 debug 5 moe tmp R 00:00:11 128 nid[00128-00255]
```

# Caveats



- Some SLURM functionality has no ALPS equivalent
  - Independent I/O by task
  - Output labeled by task ID
- Some ALPS options have no SLURM equivalent
  - *srun* wrapper has *-alps* option to pass arbitrary arguments to *aprun*
- Some options are similar, but impossible to directly translate
  - Task binding syntax
  - Per-task vs. per-CPU limits

# Caveats (continued)

- SLURM environment variables are not set
  - Many need to be set on a per-node or per-task basis, so ALPS must do this
  - Under development by Cray

# Caveats

## (continued)

- SLURM GUIs (actually the *curses* and GTK libraries they use) have limited scalability
  - Scales to a few thousand nodes
  - Currently each position displayed represents a unique X, Y, Z-coordinate
  - If multiple nodes share an X, Y, Z-coordinate, the information for only one node is displayed
  - We found this better than displaying each node independently and providing confusing topology information, but could change this if desired



# Status



- SLURM has been running reliably over ALPS at Swiss National Supercomputer Centre since April 2011
- Validated by Cray in July/August 2011