Directions in Workload Management

Alex Sanchez and Morris Jette
SchedMD LLC

HPC Knowledge Meeting 2016

Areas of Focus

- Scalability
  - Large Node and Core Counts
  - Power Management
  - Failure Management
  - Federated Clusters
- Data management
  - Burst Buffer Support
- New Architectures
  - Intel Knights Landing Processor

Scalability

- Systems with several thousand nodes common today with some much larger systems:
  - ~100,000 nodes (Sequoia at Lawrence Livermore Lab)
  - ~3,100,000 cores (Tianhe-2 at National University of Defense Technology)
- Systems will generally grow in node and core count with both direct and indirect consequences
  - E.g. direct: ability of Slurm and PMI to manage/bootstrap MPI jobs with > 10^6 tasks
  - E.g. indirect: failures are more common → how can long running jobs be executed

Data Structures and Scheduling

- Sequential operations are not very scalable (e.g. performing tests on each node)
- Slurm makes extensive use of bitmaps for scheduling purposes, which is very scalable
- Systems several times the size of today's largest systems have been tested using Slurm emulation
- Various data structures will require some modification as system sizes grow, but these changes are straightforward

Power Management

- HPC systems today can consume 20+ MegaWatts
- Availability of power can vary through time (e.g. more power available at night)
- Slurm includes a dynamic power capping mechanism and per job power management options

System Power Management

- Capture current power usage in near real-time
- Dynamically adjusts per-node power caps to maximize effective use of system-wide power cap (Cray only today)
  - Reduce power caps on nodes using less than their current cap
  - Increase power caps on nodes near their cap
  - Options to set equal power cap for all nodes associated with each job
- Idle nodes can be powered down and then restarted as needed

Job Power Management

- Jobs can specify desired minimum and maximum CPU frequency and CPU governor
- These frequencies can be used to estimate likely power consumption of pending jobs and defer their initiation when appropriate
- Jobs can profile power consumption through time for tuning

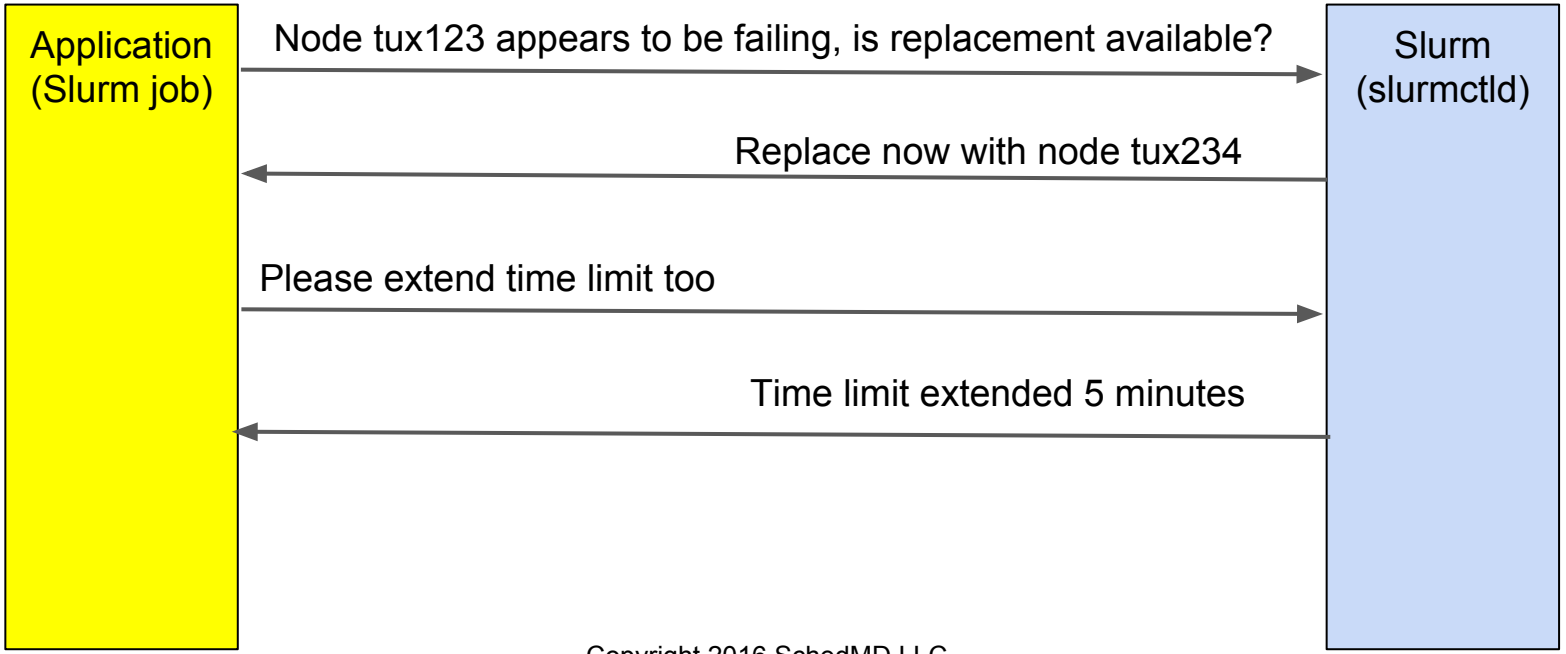
Future Power Management Work

- Extend system power management to non-Cray systems
- Add support for a power floor and controlled rate of change
- Improve algorithms to manage initiation of pending jobs based upon power availability and expected power consumption
- Consider temperature when scheduling jobs
  - Nodes higher in a rack may be hotter (heat rises) and be forced to use a lower frequency
  - Schedule a job on resources that can use a uniform power cap

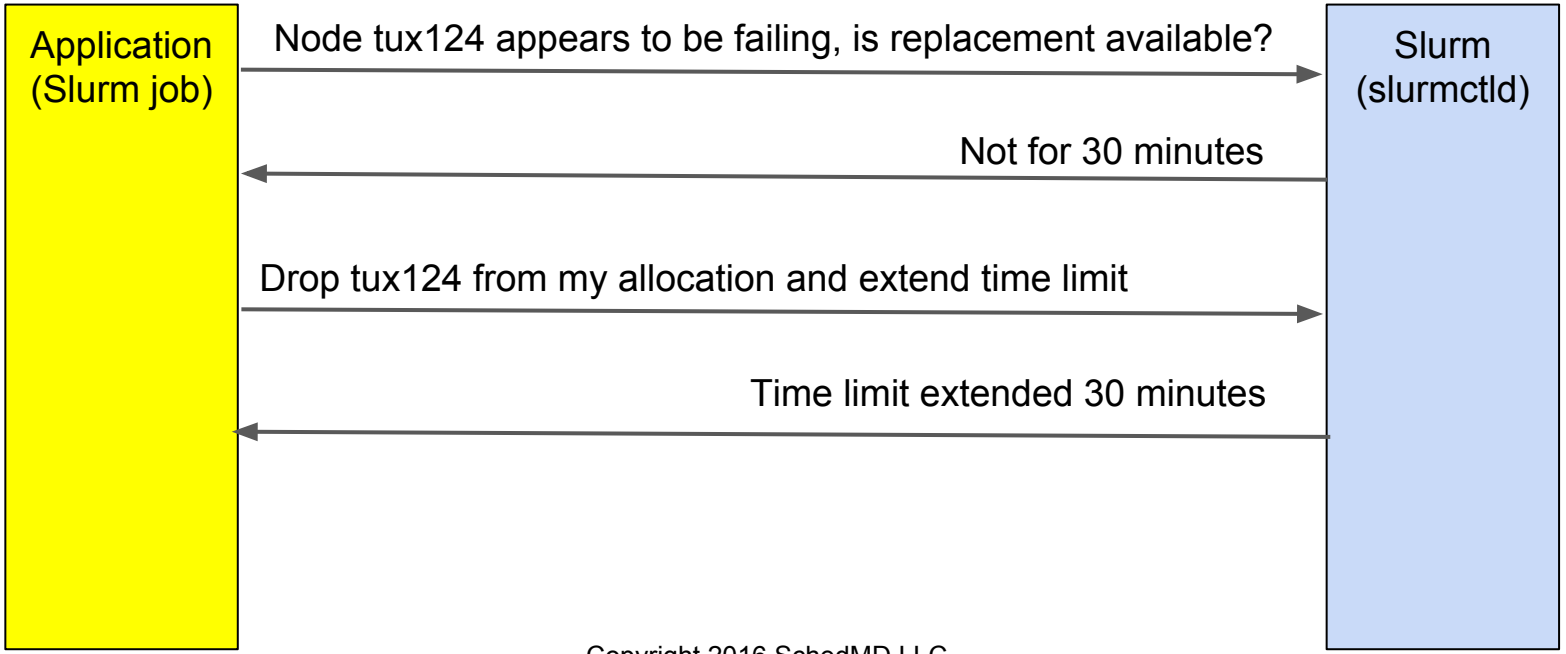
Failure Management

- Multiple failures per day are likely on large systems
- Slurm supports API for effective failure management
  - User input about suspected node failures
  - “Hot spare” resources available to running jobs as needed
  - Jobs can extend time limit for failure recovery

Failure Management Example 1



Failure Management Example 2



Federated Clusters

- Single large cluster might be managed as several clusters for improved performance
- Multiple clusters might be scheduled as a single “federated cluster” for improved throughput and utilization
- Slurm development work currently underway

Slurm Federated Clusters

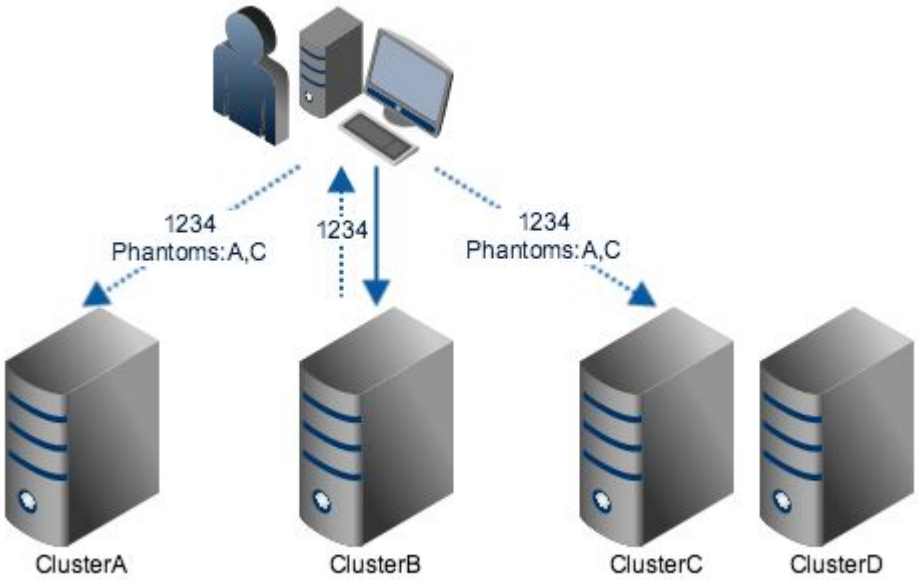
- Tools can operate either locally on a cluster or provided unified view across all federated clusters
- Uniform job ID space across the clusters
- Jobs routed to one or more clusters
  - “Phantom” copies of the job routed to additional clusters for fault tolerance
  - Job will start on whichever cluster can do so earliest
- Job dependencies across multiple clusters
- Expected availability early 2017

Unified Views

- Provide unified view of federated clusters by default
- Subset of federated clusters can be requested using -M option
- squeue, sinfo, sprio will output cluster name in separate column
  - squeue can sort by cluster name

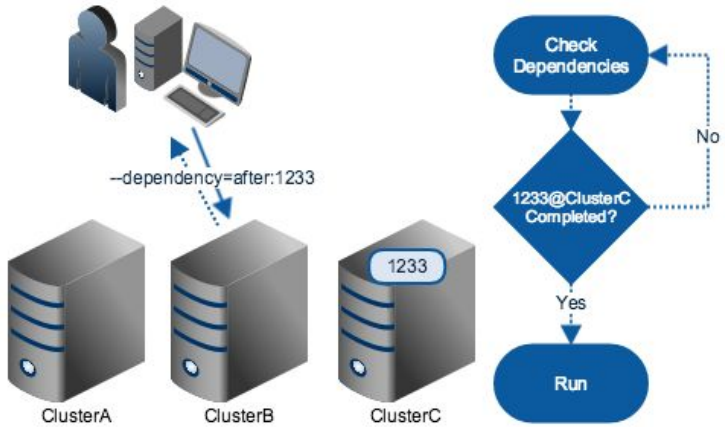
```
 CLUSTER      JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
ClusterA 2164339463     debug     wrap    brian PD       0:00     10 (Resources)
ClusterA 2164329686     debug     wrap    brian PD       0:00     10 (Priority)
ClusterB 2197893831     power     wrap    brian PD       0:00     10 (Resources)
ClusterA 2164333731      long     wrap    brian PD       0:00     10 (Resources)
ClusterC 2265002695     debug     wrap    brian PD       0:00     10 (Priority)
ClusterC 2265002695     debug     wrap    brian  R       0:36      1 c11
ClusterA 2164333936     debug     wrap    brian  R       0:36     10 a[01-10]
ClusterB 2197893831     power     wrap    brian  R       0:36      1 b180
ClusterC 2265002695     debug     wrap    brian  R       0:36    102 c[1-10,15,51-100,120-140,143-160]
ClusterA 2164333932     short     wrap    brian  R       0:39      1 a52
ClusterA      12345     short     wrap    brian  R       0:39      1 d79
ClusterA 2164333933     short     wrap    brian  R       0:39      1 a113
```

Job Submission

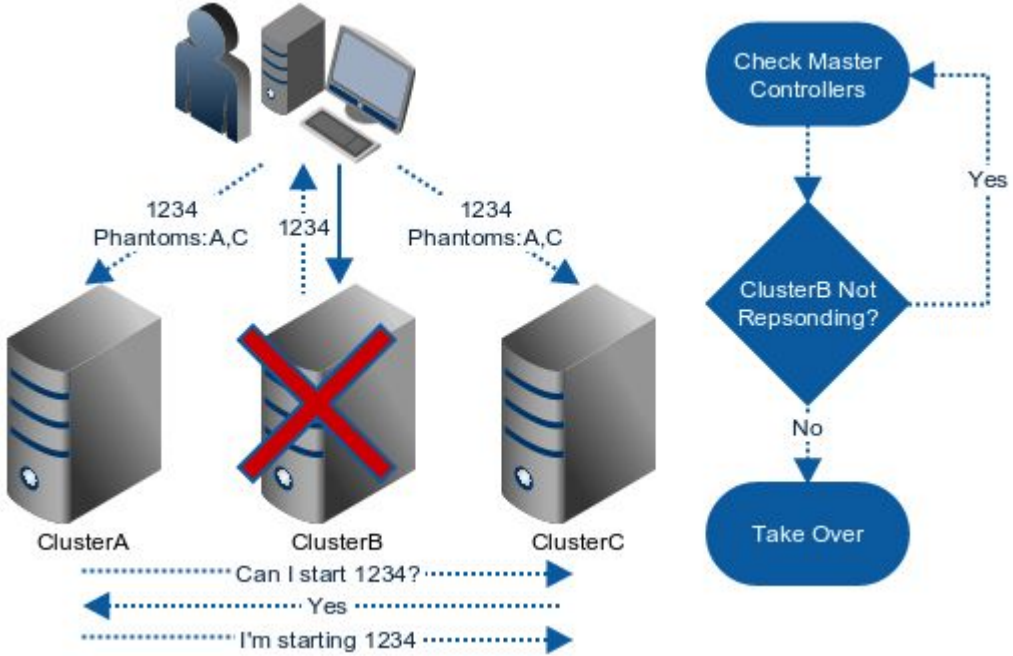


Cross-Cluster Job Dependencies

- Dependencies created with standard --dependency= syntax
- “Master” controller will check status of remote jobs on other clusters



Fault Tolerance / Job Migration



Burst Buffers

- Data management an important factor in HPC
- Burst buffers are a high-speed data store
- Buffers can persist across multiple jobs (e.g. a database)
- Buffers can be job specific
- Buffer typically allocated before compute resources and persist afterwards for data staging
- Slurm currently support for Cray systems today, generic burst buffer support in early 2017

Burst Buffer Workflow

- Job submission specifies burst buffer requirements
- Slurm allocates burst buffer resources to the jobs expected to start soonest
  - May not always be highest priority if using backfill scheduler
  - Allocations may be revoked for higher priority jobs if the expected job start order changes
- Stage-in of files starts
- Compute nodes may be allocated after stage-in completes
- Job begins, provided access to allocated burst buffer
- Stage-out begins after completion of job
- Slurm job-record retained until stage-out completed

Intel Knights Landing (KNL)

- The GRES mic plugin dedicates a whole mic to one job and does not reboot the node. We can probably use this for KNL co-processor mode.
- It seems everyone will use KNL as host processor, so:
  - Slurm needs to reboot nodes in different NUMA and MCDRAM modes
  - Allocate the MCDRAM resources to different jobs
  - Allocate cores to different jobs taking into consideration the changing NUMA modes

Intel Knights Landing (KNL)

- NUMA architecture and cache size can change per BIOS settings
- KNL has 72 cores (not a power of 2) → NUMA domains not symmetric (in ‘quad’ NUMA mode, some NUMA will have 17 cores, others will have 18 cores)

Slurm KNL Support

- Slurm configuration options:
  - Who can change NUMA and cache mode
  - What NUMA and cache modes are available to users
- Slurm job options:
  - Desired NUMA and MCDRAM mode (e.g. “--constraint=hemi,cache”)
  - Desired High Bandwidth Memory (e.g. “--gres=hbm:2g”)
- Under development
  - Support for KNL on non-Cray systems
  - Task layout for heterogeneous NUMA domains

Summary

- Exascale computing involves numerous challenges, but no intractable problems yet identified
- Questions?