BERKELEY LAB
LAWRENCE BERKELEY NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

SLURM Migration Experience at Lawrence Berkeley National Laboratory

Presented by
Jacqueline Scoggins
System Analysts, HPCS Group

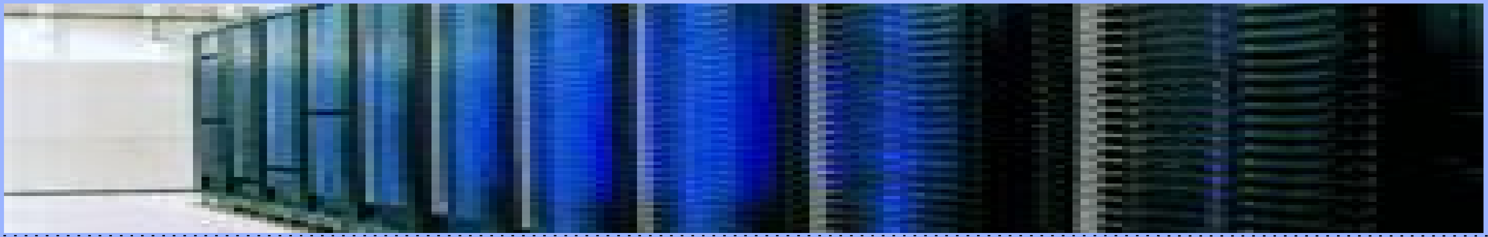
Slurm Conference
Lugano, Switzerland

September 22-23, 2014

National Laboratory Bringing Science Solutions to the world

HPCS Group
High Performance Computing Services

- We began providing HPCS in 2003 supporting 10 departmental clusters.
- Today we manage throughout LBNL campus and a subset of UC Berkeley campus
  - over 30 clusters – ranging from 4 node clusters to 240 node clusters.
  - over 2000 user accounts
  - over 140 projects

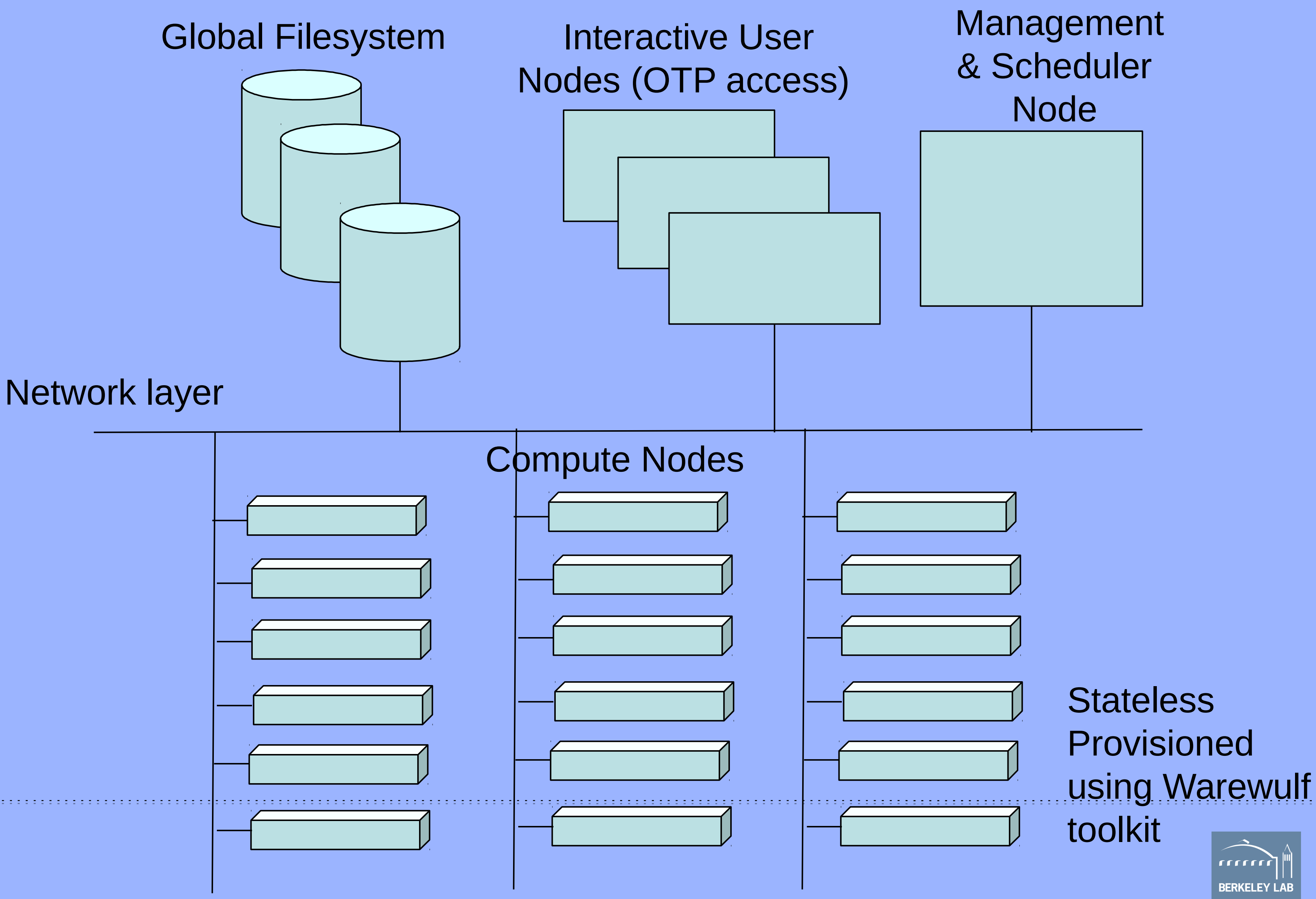


OUR ENVIRONMENT

We use the term - “SuperCluster”

- Single Master Node - provisioning node and scheduler system
- Multiple interactive login nodes
- Multiple Clusters
  - Institutional and Departmental purchased nodes
- Global Home Filesystems
- Flatten Network Topology so everything communicate with the management/interactive nodes

ARCHITECTURE DESIGN


Global Filesystem
Interactive User Nodes (OTP access)
Management & Scheduler Node
Network layer
Compute Nodes
Stateless Provisioned using Warewulf toolkit

OUR SET UP

- Nodes are stateless and provisioned using Warewulf (interactive, compute and storage nodes)
- Jobs could not span across clusters
- All user accounts exist on the interactive nodes
- All user accounts exist on the scheduler node but with nologin access.
- User accounts are only added to the compute nodes they are allowed to run on.

How was moab configured?

Moab was configured with

- **Queues** - many queues for every department and for each requirement they had – rather it is serial, parallel, debug, or high priority – one queue for each scenerio
- **Nodes** - Each node had a line with its configuration information with specific features to group nodes in cluster or by uniqueness, charge rates were set on nodes that were charging for CPU cycles

Moab Configuration (cont’d)

- **ACL** – on users/groups resources they could use in the CLASSCFG and SRCFG. They were also configured within torque for each queue
- **Limits/Policies** - Every class had limits/ policies set and some where the same but no way of combining these together because of their uniqueness. (QoS did not do it because of access controls were needed)
- **Fairshare** (globally setup)

Moab Configuration (cont'd)

- **Account Allocations** – Gold banking for charging CPU cycles used, setting up rates for node types/features
- **Backfill**
- **Standing Reservations**
- **Condo Computing**

What made us migrate?

- Too many unpredictable scheduler issues
- Existing Support contract was up for renewal
- Slurm had the features and capabilities we
- needed
- Slurm architectural design was less complex
- Easier to use and to manage

What did we need to migrate?

- Single scheduler capable of managing department owned and institutional owned nodes
- Scheduler needed to be able to separate users access from nodes they do not have accounts on.
- Scheduler needed to be able to avoid scheduling conflicts with users from other groups

What did we need to migrate?

- Scheduler needed to be able to have limits on jobs/partitions/nodes independently
- Needed an Allocation management scheme to be able to continue charging users
- Need to be able to use Node Health Check method for managing unhealthy nodes

What were some of our Challenges?

- Migrating over 500+ users and 17 different clusters within a 4 month period to a new system
- Implementation design
  - How will we configure the queues?
  - How will we get fairshare and backfill to work?
  - How will we implement our Accounting/Charging model? (Condo vs Non-condo usage on institutional cluster)
  - Can we control User Access to the nodes?

How did we configure slurm?

- Jobs run on nodes exclusively or shared
- Multiple Partitions to separate the clusters from each other
- QoS's to setup the limits for the jobs/partition and which one a user can use within a partition
- Backfill
- Fairshare
- Priority Job Scheduling
- Network topology configured to group the nodes and switches to best place the jobs

How did we migrate?

We ran both schedulers during the 4 month process simultaneously

We migrated a set of clusters to slurm or we did a single cluster over to slurm

Create scripts for adding accounts and users

We had to educate our users

Problems Encountered

Accounting – had to create a separate mechanism for managing charging. Used or existing user data text file by adding a few fields to identify a user with all of the projects they were associated with. We need to have key values: username Project id and account name for billing the users. Slurm does not have the association in it – WKEYs did not get the job done.

Node naming – our existing cluster names were n0000.clustername any clustername without a number at the end of it did not work. So n0000.jbs would not work in slurm. So we had to rename all of our nodes by appending the number 0 to any cluster that did not have a number at the end of it’s name. So jbs became jbs0.

Problems encountered

Backfill was not properly getting all of the queued jobs that could fit into the scheduler. Had to change some of the parameters

TreeWidth = 4096

SchedulerParameter = bf_continue

Default/Min job limits missing – the ability to set the default or min limits are not available. It would be good to have at least the default setting available for Wallclock limits in case a user forgets to request it.

Condo vs Non-Condo Computing

- Condo Computing - a PI purchase nodes, and we incorporate them into the compute node environment. The users of the condo are subject to run for free only on their contribution of nodes. If they run outside of their contributed condo resources, they are subjected to the same recharge rates as all other users.

Non-condo Computing – Institutional compute nodes + condo compute nodes not in use by the condo users. Users are charged a recharge rate of $0.01 per Service Unit (1 cent per service unit, SU).

NOTE: We reduce the charge rate by 25% for each generation of nodes

lr1 - 0.50 SU per Core CPU hour ($0.005 per core/hr)

mako and lr2 - 0.75 SU per Core CPU hour ($0.0075 per core/hr)

lr3 – 1 SU per Core CPU hour ($0.01 per core/hr)

Plus side of SLURM

Capabilities for separating users resource pool via accounting association

Capabilities for setting up Fairshare from a parent level and the children inherit equal shares and get individually lower priority if they use the resources over a window of time

So far it has been simple and easy to use