Linux Clusters Institute: Scheduling

David King, Sr. HPC Engineer
National Center for Supercomputing Applications
University of Illinois
About me

• Worked in HPC since 2007
• Started at Purdue as a Student
  • Initially fixing nodes
  • Moved on to bigger projects
• Sysadmin at Idaho National Laboratory for a year
• Sr. HPC Engineer at Northwestern University for 5 years
  • Scheduling with Moab and Torque
  • Condo Cluster of 1000 nodes
• Sr. HPC Engineer at NCSA for about a year
  • Scheduling on Blue Waters
  • 27k node system using Torque/Moab
Scheduling Schedule

- Introduction
- Workload Management Overview
  - What is the goal?
- Review of Fundamentals of Workload Managers
  - Queues
  - Priority
  - FIFO
  - Fairshare
  - QOS
  - Reservations
- Feature Set Survey
  - PBS Pro
  - Torque
  - Moab/Maui
  - LSF
  - Other Open Source Tools
- Cloud and Scheduling
- Break
- Slurm Deep Dive
  - Components
  - Interface
- Limiting Resources
  - Overview
  - Cgroups
  - Processor affinity
  - Containers
- Cloud and Scheduling
- Accounting
  - Xdmod
  - Gold/Mam
  - Slurm
- Lunch
- Slurm Hands-on
Workload Management Overview

• What is the goal of workload management?
  • Efficient, effective use of all resources by the user community under normal use
  • Ability to accommodate unusual circumstances and special requests
Workload Management Overview

• What it looks like in practice to the admin:
Workload Management

• Workload management is software that “fairly” manages jobs running on an HPC system.

• Most can apply many different policies and many inputs including past usage and available allocation to determine priorities.

• What is deemed “fair” depends a great deal upon the point of view!
  • Top complaint: “Bob is hogging the system by submitting [too many|too big|too long running] jobs.”

• Generates the number one question from users:
  • “Why doesn’t my job start?”

• You can’t please all of the people all of the time!
Resource Managers

• Manage the sharing of the available resources on your system
  • At a minimum resources are managed at the granularity of a node.
  • You may also want to manage global resources such as licenses and shared file system space and node level resources such as memory and CPU cores.

• Keep track of requests
  • Requested resources, priority related fields, account, etc

• Provide tools to start/stop and monitor jobs
  • Ideally providing a highly scalable process launching/management capability.

• Automate Submission of Jobs
  • Define workflows and/or Job dependencies

• Prioritization or Queues to control execution order
Schedulers

• The scheduler takes the information on available resources and requests to “optimize” the order and mapping of requests to the resources.
  • Many possible inputs and tunable settings can impact the scheduling algorithm
  • Resource related – amount, type, duration
  • Account – user, group, project
  • Policies – size of job, project, fair share, duration of job, required turnaround time, etc

• Tuning the scheduler is an art!
  • Know what your goals are first!
  • Ongoing process to tune the policy adapting to actual use!
  • Scheduling is not “set it and forget it”
Features Set

- **Job Priority Factors**
  - Fairshare
  - QOS
  - Multi-factor Priority
  - Backfill
  - Preemption/Gang Scheduling

- **Reservations**

- **Job Arrays**

- **Topology Aware**

- **GPU/Intel Phi Support**

- **Power Capping**
Fairshare

• Takes historical resource utilization as a factor in job priority.
• The more you use, the less priority you get.
• If Bob uses the entire cluster on Monday, his priority will be less for the rest of the week.
• Can be set for users, groups classes/queues, and QoS.
• Multi-level targets
• Highly tunable for interval (duration of window), depth (number of days), decay (weighting of contribution of each day) and what metric to use.
QOS

• QoS is Quality of Service
• Provides special treatment for jobs based on specified criteria
• Examples are:
  • Give a group access to special resources or bump priority to specific jobs within the group
• A QOS can be used to:
  • Modify job priorities based on QOS priority
  • Configure preemption
  • Allow access to dedicated resources
  • Override or impose limits
  • Change “charge rate” (a.k.a. UsageFactor)
Multi-factor Priority

- Multi-factor priority is the use of multiple factors to determine the order in which a job will start compared to others
- Uses a weighted equation
- Allows for tunable parameters
- Slurm uses the following:
  - Job priority = (PriorityWeightAge) * (age_factor) + 
    (PriorityWeightFairshare) * (fair-share_factor) + 
    (PriorityWeightJobSize) * (job_size_factor) + 
    (PriorityWeightPartition) * (partition_factor) + 
    (PriorityWeightQOS) * (QOS_factor) + 
    SUM(TRES_weight_cpu * TRES_factor_cpu, TRES_weight_<type> * TRES_factor_<type>, ...)
Backfill

• If jobs were started in strict priority order, system utilization would be significantly lower and there would be less throughput.
• Backfill allows lower priority jobs to be started without affecting the start time of any higher priority jobs
• Example: Short small jobs that can fill in and finish before a large full system job
• Different schedulers handle backfill different ways
  • Moab looks at backfill on every iteration
  • Slurm has a special backfill iteration
• Highly tunable
• Time consuming as every job is taken into consideration
  • Use of limits highly suggested on clusters with lots of small jobs that are running into performance issues
Preemption/Gang Scheduling

- Preemption is stopping a "low-priority" job(s) to let a "high-priority" job run.
- Lower priority job can be cancelled, suspended or requeued
- A grace period should be implemented before preemptions to allow for check pointing
- Gang Scheduling is time slicing and oversubscribing a node to multiple workloads
  - Suspends jobs and balances between various workloads
  - Increases workload efficiency
  - Can create indeterminate workload times
  - Difficult to predict start of future jobs
Reservations

• Allows the ability to reserve advanced resources for users, accounts or groups
• Can also be used for future system maintenance
• Reservations can be standing such that they occur on a regular basis
• Can be used for cores, nodes, licenses and other resources
• Does not work with gang scheduling due to unpredictable end times of jobs
Reservation Examples

• Have a PM for 14 days out from 0800-1700, need to ensure no jobs are still running – **system reservation**

• Have a training event on Thursday, and the teacher/guest accounts will need 5 nodes reserved for 3 hours so that they can practice submitting jobs – **user reservation**

• Node 10 needs to be reserved for interactive jobs (devel & testing) M-F 0800-1800, but can do general work the rest of the time – **standing reservation**

• Need to troubleshoot a cluster-wide issue and am willing to let running work continue, but don’t want new jobs to start while troubleshooting
Job Dependencies

• The workflow may require the output/results of Job1 to be available before Job2 can run successfully.

• The scheduler will not normally force jobs to run in submission order due to other priority factors

• Job Dependency support allows the user to indicate that one job must finish before another can start.
  • Can be tricky to implement if trying to submit many jobs at once
Job Arrays

• Mechanism for submitting and managing collections of similar jobs quickly and easily
• Quickly submits thousands of jobs in seconds
• Used for jobs that have the same time limit, size and only small variations in executables or datasets
• Uses environment variables to implement variation within job
Topology Aware

- Network Topology aware scheduling can improve application performance and increase workload throughput
- Applications that are communication (latency or bandwidth) sensitive can improve with proper job placement
- Job locality can cause less cluster fragmentation and less communication contention between jobs
Hierarchical Topology Aware

- Most common network configuration is Fat-Tree
  - Nodes are at the edge
  - Switches are hierarchically
  - Uplink bandwidth between switches is higher
  - Bisection bandwidth could be less than within switch due to switch limitation or cost
  - Placing workload within a switch provides full bandwidth for workload
  - Placing workload as close as possible will reduce latency decreasing the amount of hops required between nodes

- Dragonfly networks are similar to fat-tree in grouping of nodes on edge
  - Groups of switches interconnected connected
  - Generally uses the same interface as Fat-Tree with a slightly different algorithm
3-D Torus Interconnect Topology Aware

• A 3-D Torus is a mesh network with no switches
• All sides are connected to each other
• Blue Waters at NCSA uses this topology
• Job placement is critical for throughput
• Network Congestion can be a problem if jobs are placed where communication is flowing through each other
• Locality is important for performance
• Not all links have to be the same speed
Accelerator Support

• Specialized hardware such as GPUs, FPGAs and Intel PHIs are increasing in demand
• Workload managers can monitor and assign these resources just like CPUs or memory
  • The difference is that some nodes may not have any of these resources
Power Capping and Green Computing

• The idea to limit the work done by a specific job to a fixed of total power consumed
• The implementation is usually a resource manager (RM) throttle of local cpu performance per node
  • May not account for MB power from DIMM’s, networking & any GPU’s
• Alternately, the RM may be able to monitor the total power consumed at the power supply (usually through BMC hooks) and simply terminate the job at the requested amount
• Node can also be shutdown while not in use
• Not all schedulers support this
Common Open Source and Commercial Schedulers and Workload Managers

- There are several commercial and open source choices for workload management
  - Portable Batch System derived
    - PBS Pro – commercial and open source product supported by Altair Engineering
    - Torque – open source maintained by Adaptive Computing
      - Very limited built-in scheduler
      - Can utilize the Maui Scheduler – open source
        - In maintenance mode
      - Moab scheduler is a commercial fork of Maui developed by Adaptive
  - Univa Grid Engine (UGE) (formally Sun Grid Engine) supported by Univa
  - Platform LSF – IBM Commercial product
  - SLURM – Open source with commercial support
PBS Pro

- Both a scheduler and Workload Manager
- Continued Development from the original PBS at NASA Ames
- Commercially Released in 2000
- Joined Altair Engineering in 2003
- Open Sourced in 2016
Torque

• Torque (Terascale Open-source Resource and Queue)
  • A fork of OpenPBS started in 2003
  • OpenPBS is itself a fork of the commercial PBS in 1998, but hasn’t seen continued development.
    • PBSPro also derived from this code base and has seen continued commercial development.
  • PBS (Portable Batch System) was started in 1991
    • Note that this is prior to the rise of large scale distributed memory systems. In certain ways these products all still show some shortcomings based on this original design!
    • Actively developed and commercially supported by Adaptive Computing
  • The included scheduler for the open source variants has always been quite simple.
    • FIFO + backfill (no reservations)
Maui

- Open Source Scheduler
- Maui development was started in the mid-1990’s by David Jackson working as a contractor to develop a better scheduler to run on top of LoadLeveler on the IBM SP at the Maui High-Performance Computing Center.
- Fairly quickly he developed the interfaces needed to run on top of PBS/OpenPBS.
- The key feature is use of reservations for all jobs as well as for blocking out sets of resources for sets of users along with Backfill.
- Unfortunately, support for Maui has been deprecated- last update was in 2015
Moab

• Commercially available from Adaptive Computing
• Moab (named for a location in Utah where David Jackson lives) is a commercial fork of Maui started in 2001.
  • Pretty much all enhancements since then have went into Moab.
• Keeps most all of the Maui features and adds more.
• One recent feature added specifically for Blue Waters is topology aware scheduling – very relevant for the Cray 3D torus, less so for typical InfiniBand clusters.
• Interfaces with multiple workload managers including Slurm, Torque and PBS Pro.
• Highly customizable.
Univa Grid Engine

• Originally Developed by Genias Software in 1999 as CODINE and GRD
• Acquired by Sun Microsystems and renamed Sun Grid Engine (SGE) and released the software as open source
• Oracle acquired Sun in 2010 and SGE became Oracle Grid Engine (OGE)
• In 2011, Univa started a fork of the open source SGE and later acquired all commercial rights to Grid Engine now Univa Grid Engine (UGE)
Platform LSF (Load Sharing Facility)

- Originally based on the Utopia Project at the University of Toronto
- Commercialized by Platform Computing
- Acquired by IBM in 2012
- Easy to upgrade with in place patching
- Lots of add support for analytic, dashboards, submission portals
- Supports multiple APIs for submitting jobs
  - Python, Perl, DRMAA, SAGA
Slurm

• Simple Linux Utility for Resource Management
  • No longer all that simple!
  • Started out as a collaborative effort between LLNL, Linux NetworX, HP and BULL in 2001.
  • Designed with distributed memory parallel systems as the primary target to address some of the shortcomings at the time with systems like PBS and Torque.
  • Open-source code with commercial support/development provided by SchedMD.
  • Extendible architecture means there are many plugins available to modify the default SLURM behavior.
  • SLURM use has grown dramatically over the past five years at all system sizes.
    • Partly as a result there is a large and active community engaged in the development of the product and add-ons.
Choosing the right Scheduler and Workload Manager

• What is your budget?
• What support level do you need?
• What is your experience with various scheduler?
• What is your workload?
  • High-throughput computing?
  • Number of jobs?
• Feature set needed
SLURM In Depth

- Architecture
- Daemons
- Configuration Files
- Key Configuration Items
- Node Configuration
- Partition Configuration
- Commands
- Test Suite
SLURM Architecture

- SLURM is designed with scalability and reliability as key goals.
- Optional redundancy for the management daemon.
- Node daemons form a hierarchical communication tree
- Optional database to manage accounting and other restrictions.
SLURM Daemons

- Daemons
  - slurmd – controller that handles scheduling, communication with nodes, etc – One per cluster (plus an optional HA pair)
  - slurmdbd – (optional) communicates with MySQL database, usually one per enterprise
  - slurmd – runs on a compute node and launches jobs
  - slurmdstepd – run by slurmd to launch a job step
  - munged – authenticates RPC calls (https://code.google.com/p/munge/)
    - Install munged everywhere with the same key

- Slurmd
  - hierarchical communication between slurmd instances (for scalability)

- slurmdctld and slurmdbd can have primary and backup instances for HA
  - State synchronized through shared file system (StateSaveLocation)
Slurm Prerequisites

• Each node in cluster must be configured with a MUNGE key and have the daemons running

• MUNGE generated credential includes
  • User id
  • group id
  • time stamp
  • whatever else it is asked to sign and/or encrypt
    • names of nodes allocated to a job/step
    • specific CPUs on each node allocated to job/step, etc.
SLURM Configuration Files

- Config files are read directly from the node by commands and daemons
- Config files should be kept in sync everywhere
- Exception slurmd.conf: only used by slurmd, contains database passwords
- DebugFlags=NO_CONF_HASH tell Slurm to tolerate some differences. Everything should be consistent except maybe backfill parameters, etc that slurmd doesn't need
- Can use “Include /path/to/file.conf” to separate out portions, e.g. partitions, nodes, licenses
- Can configure generic resources with GresTypes=gpu
- man slurm.conf
- Easy: http://slurm.schedmd.com/configurator.easy.html
- Almost as easy: http://slurm.schedmd.com/configurator.html
Key Configuration Items

• ClusterName can be set as desired, will need to use it later for accounting.
  • Prefer lower case names.

• On the head node where slurmctld will run:
  • Set ControlMachine to “hostname –s” output.
  • SlurmUser=slurm NodeName

• All nodes run slurmd:
  • SlurmdUser=root
Partition Configuration

• Partitions are configured in slurm.conf. Specify the nodes associated with each partition along with limits.
  • PartitionName=batch Nodes=n[001-072],g[01-18] Default=YES MaxTime=INFINITE State=UP RootOnly=YES
  • PartitionName=cpus Nodes=pp[05-16] MinNodes=1 MaxNodes=4 MaxTime=24:00:00 STATE=UP

• Nodes can be in multiple partitions. Each partition can specify min and max nodes and times.
Commands

- `squeue` – view the queue
- `sbatch` – submit a batch job
- `salloc` – launch an interactive job
- `srun` – two uses:
  - outside of a job – run a command through the scheduler on compute node(s) and print the output to stdout
  - inside of a job – launch a job step (i.e. suballocation) and print to the job's stdout
- `sacct` – view job accounting information
- `sacctmgr` – manage users and accounts including limits
- `sstat` – view job step information
- `sreport` – view reports about usage
- `sinfo` – information on partitions and nodes
- `scancel` – cancel jobs or steps, send arbitrary signals (INT, USR1, etc)
- `scontrol` – list and update jobs, nodes, partitions, reservations, etc
A Simple Sequence of Jobs

• `sbatch --ntasks=1 --time=10 preprocess.batch`
  submitted batch job 100
  • Run the preprocess.batch script on 1 task with a 10 minute time limit, resulting job has an id of 100

• `sbatch --ntasks=128 --time=60 --depend=100 work.batch`
  submitted batch job 101
  • Run the work.batch script on 128 tasks with a 60 minute time limit after job 100 completes.

• `sbatch --ntasks=1 --time=10 --depend=101 post.batch`
  submitted batch job 102
  • Run the post.batch script on 1 tasks for up to 10 minutes after job 101 completes.
Tasks versus Nodes

• Tasks are like processes and can be distributed among nodes as the scheduler sees fit.

• Nodes means you get that many distinct nodes.
  • Must add –exclusive to ensure you are the only user of the node!

• Run hostname as two tasks:
  • srun --ntasks=2 --label hostname

• Same on two whole nodes:
  • srun --nodes=2 --exclusive --label hostname
Interactive Jobs

• `salloc` uses a similar syntax to `sbatch`, but blocks until the job is launched and you then have a shell within which to execute tasks directly or with `srun`.

• `salloc --ntasks=8 --time=20 --pty bash`

  `salloc: Granted job allocation 104`

• Try `hostname` directly.

• Try `srun --label hostname`
sr

• sr can be used as a general purpose task launcher.
• Inside of a job it can be used to launch your tasks from a master script
• sr support launching multiple executables at once using a simple config file.

• Many MPI implementations either use sr directly or the mpirun ties into sr.
sacct

• sacct provides accounting information for jobs and steps
• Many filtering and output options
• Works with the accounting file or optional database
• Return accounting information on user bob
   sacct –u bob
• Return accounting information on the debug partition
   sacct –p debug
sacctmgr

- Manages the accounting database
  - Add/delete users, accounts, etc
  - Get/Set resource limits, fair share allocations, etc
- sprio – view factors comprising a jobs priority
- sshare – view current hierarchical fair share information
- sdiag – view stats on the scheduling module operation (execution time, queue length)
scancel Command

• Cancel a running or pending job or step
• Can send arbitrary signal to all processes on all nodes associated with a job or step
• Has filtering options (state, user, partition, etc)
• Has an interactive (verify) mode

scancel 101.2
scancel 102
scancel –user=bob –state=pending
sbcast

• Copy a file to a local disk on allocated nodes
  • Execute within an allocation
  • Data is transferred using hierarchical slurmd daemons
• Might be faster than a shared file system.
strigger

- SLURM can run an arbitrary script with certain events occur
  - Node goes down
  - Daemon stop or restarts
  - Job is close to time limit

- strigger command can be used to create, destroy or list event triggers.
Host Range Syntax

- Host range syntax is more compact, allows smaller RPC calls, easier to read config files, etc
- Node lists have a range syntax with [ ] using “,” and “-”
- Usable with commands and config files
- n[1-10,40-50] and n[5-20] are valid
- Comma separated lists are allowed:
  - a-[1-5]-[1-2], b-3-[1-16], b-[4-5]-[1-2,7,9]
* Want to see all **running jobs** on **nodes n[4-31]** submitted by all users in **account acctE** using **QOS special** with a certain set of **job names** in **reservation res8** but only show the **job ID** and the **list of nodes** the jobs are assigned then sort it by **time remaining then descending by job ID**?

* There's a command for that!

```
squeue -t running -w n[4-31] -A acctE -q special -n name1,name2 -R res8 -o "%.10i %N" -S +L,-i
```

* Way too many options to list here. Read the manpage.
sbatch, salloc, srun

- sbatch parses #SBATCH in a job script and accepts parameters on CLI
  - Also parses most #PBS syntax
- salloc and srun accept most of the same options
- **LOTS** of options: read the man page!
sbatch, salloc, srun

- Short and long versions exist for most options
- `-N 2` # node count, same as `--nodes=2`
  - In order to get exclusive access to a node add `--exclusive`
- `-n 8` # task count, same as `--ntasks=8`
  - default behavior is to try loading up fewer nodes as much as possible rather than spreading tasks
- `-t 2-04:30:00` # time limit in d-h:m:s, d-h, h:m:s, h:m, or m
- `-p p1` # partition name(s): can list multiple partitions
- `--qos=standby` # QOS to use
- `--mem=24G` # memory per node
- `--mem-per-cpu=2G` # memory per CPU
- `-a 1-1000` # job array
Job Arrays

• Used to submit homogeneous scripts that differ only by an index number
  • $SLURM_ARRAY_TASK_ID stores the job's index number (from -a)
  • An individual job looks like 1234_7 where
    ${SLURM_JOB_ID}_${SLURM_ARRAY_TASK_ID}
  • “scancel 1234” for the whole array or “scancel 1234_7” for just one job in the array

• Prior to 14.11
  • Job arrays are purely for convenience
  • One sbatch call, scancel can work on the entire array, etc
  • Internally, one job entry created for each job array entry at submit time
  • Overhead of job array w/1000 tasks is about equivalent to 1000 individual jobs

• Starting in 14.11
  • “Meta” job is used internally
  • Scheduling code is aware of the homogeneity of the array
  • Individual job entries are created once a job is started
  • Big performance advantage!
scontrol

• scontrol can list, set and update a lot of different things
  • scontrol show job $jobid # checkjob equiv
  • scontrol show node $node
  • scontrol show reservation

• scontrol <hold|release> $jobid # hold/release (“uhold” allows user to release)

• Update syntax:
  • scontrol update JobID=1234 Timelimit=2-0 #set 1234 to a 2 day timelimit
  • scontrol update NodeName=n-4-5 State=DOWN Reason=“cosmic rays”

• Create reservation:
  • scontrol create reservation reservationname=testres nodes=n-[4,7-10]
    flags=maint,ignore_jobs,overlap starttime=now duration=2-0 users=root

• scontrol reconfigure #reread slurm.conf

• LOTS of other options: read the man page
Reservations

• Slurm supports time based reservations on resources with ACLs for users and groups.

• A system maintenance reservation for 120 minutes:
  • `scontrol create reservation starttime=2009-02-06T16:00:00 duration=120
    user=root flags=maint,ignore_jobs nodes=ALL`

• A repeating reservation:
  • `scontrol create reservation user=alan,brenda starttime=noon duration=60
    flags=daily nodecnt=10`

• For a specific account:
  • `scontrol create reservation account=foo user=-alan partition=pdebug
    starttime=noon duration=60 nodecnt=2k,2k`

• To associate a job with a reservation:
  • `sbatch --reservation=alan_6 -N4 my.script`

• To review reservations:
  • `scontrol show reservation`
Node Configuration

• All compute nodes are defined in slurm.conf in the form:
  • NodeName=n[001-080] CPUs=12 RealMemory=48260 Sockets=2
    CoresPerSocket=6 ThreadsPerCore=1 State=UNKNOWN
  • Sockets, Cores, Threads help define the NUMA domains to aid in
    pinning processes to cores
  • RealMemory defines the “configured” memory for the node.
  • Can add “GRES=gpu:2” for a GPU resource
SLURM Test Suite

• SLURM includes an extensive test suite that can be used to calibrate proper operation
• include over 300 test programs
• executes thousands of jobs
• executes tens of thousands of steps
• change directory to testsuite/expect
• create file “globals.local” with installation specific information
• set slurm_dir “/home/moe/SLURM/install.linux”
• set build_dir “/home/moe/SLURM/build.linux”
• set src_dir “/home/moe/SLURM/slurm.git”
• Execute individual tests or run regression for all tests
Plugins

- Dynamically linked objects loaded at run time based upon configuration file and/or user options
- 80 plugins of 20 different varieties currently available
- Accounting storage: MySQL, PostgreSQL, text file
- Network topology: 3D-torus, tree
- Different versions of MPI:
  - OpenMPI, MPICH1, MVAPICH, MPICH2, etc.
- There is an API that is available for you to write your own plugin to make Slurm perform how you would like.
Robust Accounting

• For more robust accounting we need to setup the database connection, slurmdbd.


• Likely want:
  • `AccountingStorageEnforce=associations,limits,qos`

• We will talk about this more later
Database Use

• Accounting information written to a database plus
  • Information pushed out to live scheduler daemons
  • Quality of Service (QOS) definitions
  • Fair-share resource allocations
  • Many limits (max job count, max job size, etc)
  • Based on hierarchical banks
    • Limits by user AND by banks
Setup Accounts

• Setup a couple accounts for testing:
  • `sacctmgr add account none,test Cluster=gideontest Description="none"
    Organization="none"
  • Leaving off the cluster will add the account to all clusters in the slurmdbd

• Accounts are hierarchical
  • `sacctmgr add account science Description="science accounts"
    Organization=science
  • `sacctmgr add account chemistry,physics parent=science
    Description="physical sciences" Organization=science
Add Users to Accounts

• sacctmgr add user brett DefaultAccount=test
  • Adds user brett to the DB with a default account of test.
  • At this point user brett can run jobs again.

• sacctmgr add user brett account=chemistry
  • Add user brett to a second, non-default account

• sacctmgr modify account chemistry set GrpCPUMins=5000
  • Set a total usage limit

• sacctmgr show associations
  • Useful to inspect your account settings
Break Time

• Back in 30 minutes
Limiting and Managing Resources

- Cgroups
- Pam authentication modules
- Processor Affinity
- Containers
Control Groups (Cgroups)

• Cgroups are a Linux Kernel feature that limits, accounts for, and isolates the resource usage of a collection of processes

• Used to limit and/or track:
  • CPU
  • Memory
  • Disk I/O
  • Network
  • Etc...

• Features
  • Resource Limiting
  • Prioritization
  • Accounting
  • Control
Cgroup support within Workload managers

• Torque must be built using cgroups during configure time
  • Built using –enable-cgroups
  • Newer versions of hwloc is required which can be built locally
  • You must have cgroups mounted when compiling with cgroups
  • You cannot disable cgroup support on a live system

• Slurm Cgroup support is enable via multiple plugins
  • proctrack (process tracking)
  • task (task management)
  • jobacct_gather (job accounting statistics)
Pam module for compute node authentication

• A pluggable **authentication** module (**PAM**) is a mechanism to integrate multiple low-level **authentication** schemes into a high-level application programming interface (API). It allows programs that rely on **authentication** to be written independently of the underlying **authentication** scheme.

• Most schedulers have a PAM plugin module that can be used to restrict ssh access to compute nodes to only nodes where the user has an active job.

• This will not clean up any users that exist on the compute nodes
Pam_slurm

- Slurm provides a PAM plugin module that can be used to restrict ssh access to compute nodes to only nodes where the user has an active job.
- The pam_slurm PAM plugin is installed by the rpms.
- Need to add:
  
  ```
  auth include password-auth
  account required pam_slurm.so
  account required pam_nologin.so
  ```

  to `/etc/pam.d/sshd`

- Only do this on compute nodes! If you put it on the head node it will lock out users!
Processor Affinity

• Processor Affinity is binding of a process or thread to a CPU so that the process or thread will execute on the designated CPU or CPUs rather than any CPU.

• You are overriding the scheduling algorithm of the operating system

• Reasons to use Processor Affinity
  • Take advantage of remnants of a previous or existing process/thread may still reside in cache to speed up process by reducing cache misses
  • Varied tasks in a job might be scheduled on a single CPU that could be sped up if ran on 2 separate CPUs
  • Architecture of CPU such AMD Bulldozer
  • Multiple NUMA domains

• Issues with Processor Affinity
  • Does not solve load balancing issues
  • Challenging on non-uniform systems
Enabling Processor Affinity on Slurm

- The following are the parameters that need to be modified in the `slurm.conf`
  - `SelectType=select/cons_res`
  - `SelectTypeParameters=CR_Core`
  - `TaskPlugin=task/affinity TaskPluginParam=sched`

- An example would be:
  - `srun --nodes=1-1 --ntasks=6 --cpu_bind=cores ...`
  - Slurm allocates 3 CPUs on each socket of 1 node
  - Slurms distributes each task in a round robin configuration

- Many options to distribute tasks.
Containers

• Allow you to isolate applications with their entire runtime environments
• Uses the same kernel as host operating system
• Common container in HPC environments:
  • Docker
    • Mostly for DevOps, microservices, enterprise applications
    • Generally not great for HPC as requires escalated privileges
  • Shifter
    • Pulls images from Docker hub
    • No root escalation required
    • Compatible with Slurm via plugin and other workload managers
  • Singularity
    • Self-contained executable that is ran within a script
    • Allows importation of docker images
    • No root escalation or daemons required
Docker

• Leading container platform in the world
• Portable deployment across docker environments
• Easily local client install
• Images allow users to be root
• Create an image on your local machine and to cloud dock service
• Self contained image with bind-able host filesystems
• Large community of premade images on Dockerhub
• Not great in HPC environment as the docker daemon requires root with users having root equivalent permissions
• LSF, Torque integration
Shifter

• Developed by the National Energy Research Scientific Computing Center
• Leverages or integrates with public image repos such as Dockerhub
• Require no administrator assistance to launch an application inside an image
• Shared resource availability such as parallel filesystems and network interfaces
• Robust and secure implementation
• Localized data relieves metadata contention improving application performance
• “native” application execution performance
• Slurm integration via SPANK Plugin
Singularity

- Developed by Lawrence Berkeley Lab
- Packages entire application and environment within image
- No user contextual changes or root escalation allowed
- No root owned daemon processes
- Users can run Singularity containers just as they run any other program on the HPC resource
- No integration with scheduler required
- All standard input, output, error, pipes, IPC, and other communication pathways that locally running programs employ are synchronized with the applications running locally within the container.
- MPI integration
Scheduling and Cloud

• Using cloud resources can allow clusters to grow and shrink on demand.
• Can offer bursting or an independent self-contained cluster
• Moab, LSF, Slurm all support
• Slurm uses the power capping plugin to launch and shutdown nodes
• Reasons for using Cloud:
  • Limited resources
  • Cheaper compute during less demand
  • Some data is already in the cloud
Accounting

• XDMod
• MAM/Gold
XDMod

- Open XDMoD is an open source tool to facilitate the management of high performance computing resource
- Support for SLURM, TORQUE/PBS, UGE and LSF
- Used to inform scheduler changes
  - Infer areas where jobs aren’t getting through
  - Users that are ‘gaming’ the system
  - Users that are getting stuck
- Easy Reports

- Bluewaters XDmod
XDMoD

August 2017
Moab Accounting Manager

- Originally called Gold as Open Source
- Uses Postgres as database
- Built by Scott Jackson (now at Adaptive Computing) at Pacific Northwest Laboratory
- Similar to SLURM accounting functionality
- Track usage per user, group, project, or account
- Pre-pay or pay-as-you-go models
- Charge rates per resource, action, or quality of service
- Lien-based model
- Enforce budgets
Job Submission Portals

• PBS Pro (Compute Manager), Moab (Viewpoint) and LSF have webportals for jobs submission

• Users can:
  • Submit jobs
  • Transfer files
  • Check status of jobs
  • Use job templates

• Some have administrative dashboard
  • Also ability to modify jobs

• Also can incorporate into authentication scheme
Lunch
Hands On with Slurm

- You are given a university community cluster of 2 nodes with 2 cores a piece
- It has the latest version of SLURM installed
- Software Environment modules are installed
- There are 4 departments
  - Math, Physics, Biology and Astronomy
- The scheduler is setup with 1 default queue (normal) and first in first out scheduler
  - The queue is limited to 20 minute jobs
- Accounting is setup with all of the users and groups
  - There is no enforcements of any limits
Getting Setup on Nebula

• Grab handout
• SSH to host
• Find all hosts
• Users and groups
• Exploring SLURM
  • Run a simple job
• After you are done exploring, users of the cluster will start submitting jobs
Things to know

- There is a NFS mounted home directory
- RPMs are installed
- All SLURM configs and binaries are in default locations on all nodes (/etc,/bin,/sbin)
Exercise 1: Enabling Fairshare

• Oh no! A professor Bob is complaining about the fairness of the cluster.
  • They only run 4 core jobs a few times a week and other are running a ton of single core jobs
• Enable Fairshare where all users are given an equal share of all of the resources
• Fairshare should be a very short period, 10 minute period with a depth of 6.
• Start up more jobs
Exercise 2: Enable Fairshare for Groups and Users

• The professors have decided that all departments need to share the cluster evenly
• They also want all users to share within the account
• Setup hierarchical fairshare between users and between accounts
Exercise 3: Issues with Priority

• Professor Bob is back to having issues getting high priority jobs through
• He has a deadline on a paper for a conference
• Create a reservation for the next 30 minutes for him on both of the nodes to help satisfy him temporarily
• For a more permanent fix create another partition with higher starting priority
Exercise 4: Limiting Groups with Accounting

• The IT department has decided they want to sell the resources to each department to help fund the machine
• Limit each project to 20 CPU hours
• Modify partitions so that the high priority is double the cost of the normal queue
• Follow the instructions on the handout
Exercise 5: Enable Preemption for a Low Queue

• Users want the ability to submit low priority jobs to allow
• Make sure these jobs only backfill
• They should be half the cost of normal jobs
Extra Exercise 6: Singularity

- Biology wants to use a Docker image for their genomics project
- Follow the instructions on the hand out to pull a Ubuntu image from Dockerhub for Singularity
References

• Brett Bode
• https://en.wikipedia.org/wiki/Linux_PAM
• https://en.wikipedia.org/wiki/Platform_LSFP
• http://www.nersc.gov/research-and-development/user-defined-images/
• http://singularity.lbl.gov/
• http://clusterdesign.org/