SLURM User Group 2017

Some slightly unconventional use cases

Chris Hill, Director Research Computing, MIT

+ Rajul Kumar (Northeastern), Evan Weinberg (BU), Naved Ansari (BU), Tim Donahue.

Talk topics

- Extending a Slurm production environment for a couple of use cases that are little beyond "standard users"

  Part 1 - incorporating local virtualized resources (in our case from a local Openstack instance[†]), and using this as a way to do fully automated, transparent to user, checkpoint-restart for high-throughput computing (HTC) workloads.

  Part 2 - creating a "self-service" (i.e. no admin involvement) Slurm reservation system for managing temporary loaning of hardware to non-production environments.

- Goals in coming to Slurm user meeting
  - Connect to a great community that has produce software that is technically very capable, and automates some vexing equitable sharing challenges
  - discuss and get feedback on a couple of feature ideas that have emerged from our work

[†] audience participation question. Who does not have a bunch of folks trying to run Openstack in their data center?

Our production environment

- MIT operates two shared pools (A,B) of capacity at MGHPCC. Both under Slurm for research community that centers on MIT but spans many other institutions (local and global)
- 1500 compute nodes, >30,000 x86 cores (60,000 with HT etc..), 5+ PiB high-speed storage, ~20% nodes have some GPU (K20, K80, P100; 2 DGX-1 nodes), Infiniband FDR and EDR connected.
- >1000+ active individual users; gateway clients for Open Science Grid; MIMIC2 etc… 🡐 10,000+ pool users)
- Global access for projects around MIT research community, includes collaborators in every continent.
- Typical workload is broad mix of tightly coupled (multiple fully non-blocking zones up to 4500 cores, one OPA zone) and ensemble parallelism. Some workloads combine both.
- Multiple 100Gb/s ingress, egress paths to multiple login, viz and data transfer devices.
- Application domains range from music, economics, quantitative social science 🡐🡒 basic physics, Earth and planetary, fusion, chem, materials
- Thousands of Slurm jobs every day, several different groupings of node configuration (mem, CPU, accelerators)
- Located within MGHPCC research-computing data center

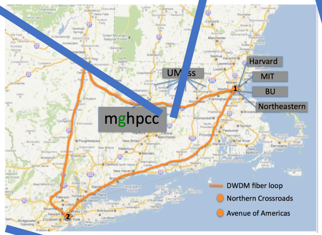
About MGHPCC

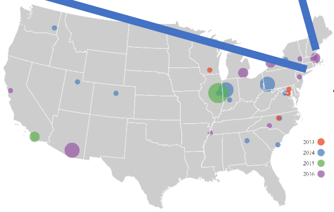
Feed up to 35MW. >95% non-fossil fuel.

Heat into water to/from computer room floor.

Currently hot-aisle with IRC (up to 30KW rack), begin to look at direct liquid cool (up to 80KW rack).

72 dark fiber pairs





Peering ESnet, I2, AWS, M$, RENs, GEANT, etc.. in Boston and Manhattan

http://www.mghpcc.org

(3 of Tues, Sept 26 Slurm talks also originate from MGHPCC, Slurm an invaluable resource at facility)

Part 1 - incorporating local virtualized resources (in our case from a local Openstack instance[†]), and using this as a way to do fully automated, transparent to user, checkpoint-restart for high-throughput computing (HTC) workloads.

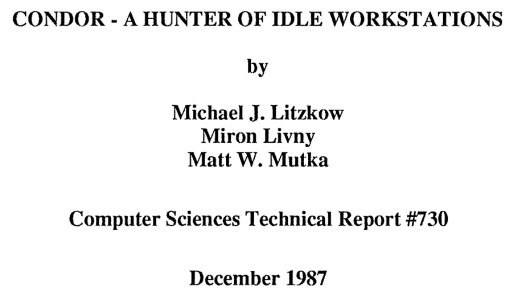
- Using OStack/KVM to enable transparent (to user/job) checkpoint to disk (freeing CPU, network, memory resources).
- Tweak Slurm "state" model (ALLOCATED…DOWN etc..) to create an alternate to either disruptive or memory resource keeping pre-emption
  - Motivation is to increase the useful work achieved by baseload back-fill high-throughput computing (HTC) jobs

- Motivation is to increase the useful work achieved by baseload back-fill high-throughput computing (HTC) jobs

- HTC jobs are a great approach to ensuring a quasi-infinite supply of relevant tasks that can fill idle slots created by complex job mix
- Utilize a low-priority “opportunistic” partition that is forcefully preempted by higher-priority partition jobs that target the same resource (node).
- This can lead to increased actual utilization (i.e. more real work gets done), but it can also increase apparent utilization (i.e. amount of time cluster is running work that is may have already run).
- Pathology/pattern for this is when opportunistic HTC jobs are <u>repeatedly</u> interrupted before they can checkpoint their progress to somewhere persistent.

An OSG experiment

- Open Science Grid (OSG - https://www.opensciencegrid.org) is a global high-throughput computing framework that grew out of the CONDOR project in Wisconsin



CONDOR - A HUNTER OF IDLE WORKSTATIONS

by

Michael J. Litzkow
Miron Livny
Matt W. Mutka

Computer Sciences Technical Report #730

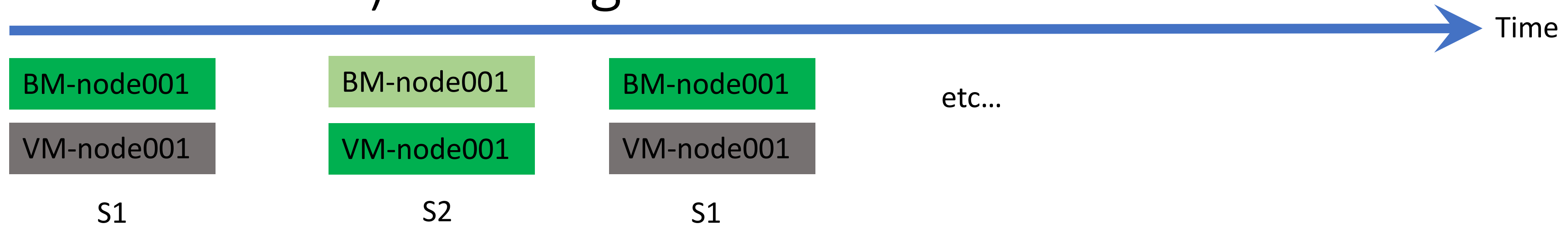
December 1987

- Today OSG supports a distributed ecosystem of science analysis centered on the Large Hadron Collider with an extensive system of authorization, credits and resources sharing.
- The OSG Slurm Glidein system allows a Slurm cluster to accept OSG jobs queued up by some or all of the OSG research community, subject to some controls/specification.
- OSG jobs are some unit of computation expressed in a script in which the OSG framework + Glidein installation provides an environment. This allows the work to be relatively portable and access global software and data inputs
- In general OSG jobs do not require a built in checkpoint frequency.
- A Slurm cluster providing opportunistic cycles does not have a way to determine if OSG jobs are doing any checkpointing.
- Consequently preempted a OSG jobs may repeatedly enter the backfill pipeline and repeat exactly the same steps they previously computed, replacing actual useful utilization with apparent utilization.

Introduce VMs with full checkpointing to disk for OSG

- Broadly we want to try and create Slurm “nodes” that are virtual machines (VMs) and then
  - transfer “bare-metal” nodes to a VM host role when there is no work in queue for bare metal resources
  - trigger the VMs to instantiate on the bare-metal “hosts”
  - have VM accepting work from opportunistic partition
  - trigger a VM suspend to disk when work appears for the bare-metal resource
  - trigger VM to restart (from where it left off) when bare-metal resource becomes free
  - do this relatively transparently to Slurm and OSG pipelines

Pictorially we want to have two nodes (e.g. BM-node001, VM-node001) running on same hardware



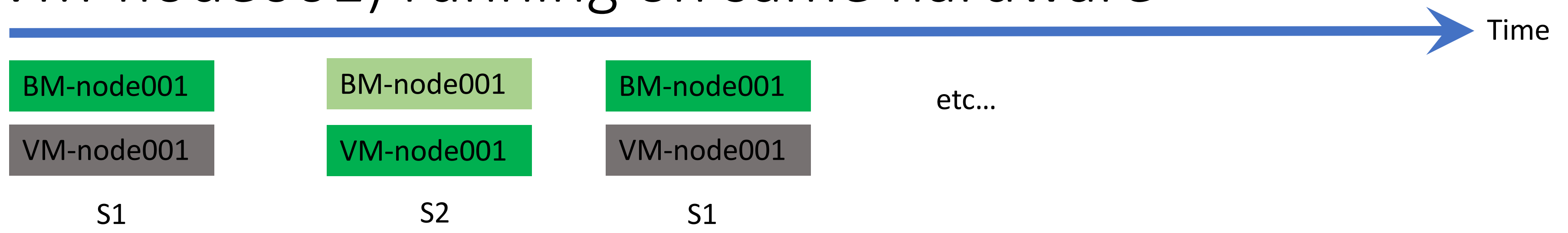
S1: `node=BM-node001 State=ALLOCATED, node=VM-node001 State=DOWN`

S2: `node=BM-node001 State=DOWN, node=VM-node001 State=ALLOCATED`

Transition between S1 and S2 is handled by a daemon that watches what is queued to higher priority partition(s) holding BM-node001,… and what is queued to “opportunistic” partition(s) holding VM-node001,…

Same daemon can handle restart/suspend of VM-node001,… to disk.

Pictorially we want to have two nodes (e.g. BM-node001, VM-node001) running on same hardware

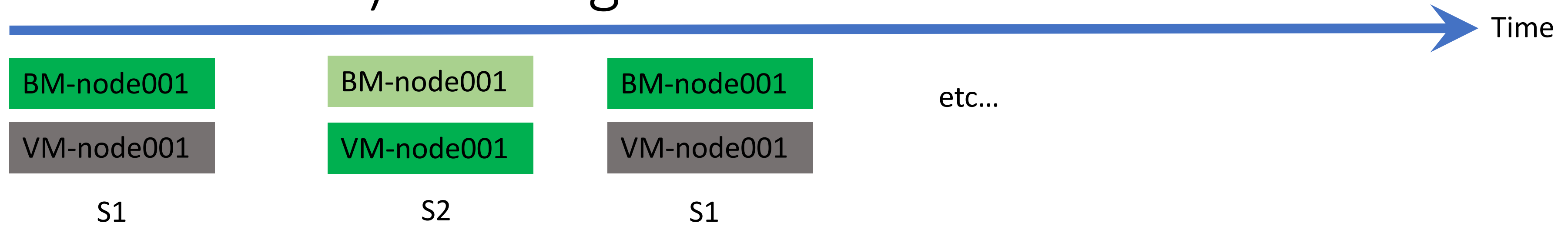


```
S1:  node=BM-node001 State=ALLOCATED, node=VM-node001 State=DOWN
S2:  node=BM-node001 State=DOWN, node=VM-node001 State=ALLOCATED
```

Transition S1 ➔ S2 whenever there is nothing pending to higher priority
Transition S2 ➔ S1 whenever we see something pending in higher priority

Can almost do this with Slurm and scontrol as is except....

Pictorially we want to have two nodes (e.g. BM-node001, VM-node001) running on same hardware



```
S1:  node=BM-node001 State=ALLOCATED, node=VM-node001 State=DOWN
S2:  node=BM-node001 State=DOWN, node=VM-node001 State=ALLOCATED
```

If we force VM-node001 DOWN when a job is running then Slurm heartbeat notices. It then either requeues jobs, or marks it failed.
For now we have modified Slurm code to allow heartbeat checks for nodes in namespace VM- to be turned off.
If we do this then this sort of cycle can be supported, potentially increasing actual useful utilization for back-fill style HTC (depending on a sites preemption cycles)

Part 1 conclusion - a possible new “feature”

- This is a potentially valuable workflow to support we believe.
- Looking for some feedback
- Unilaterally disabling heartbeat is clearly not a good idea and will end badly!
  - Would a node “SLEEPING” state in Slurm help this workflow?
  - How to distinguish between SLEEPING and DOWN
  - Would it need a timeout (in case node dies in SLEEP etc…, i.e. SLEEPING UNTIL…?)

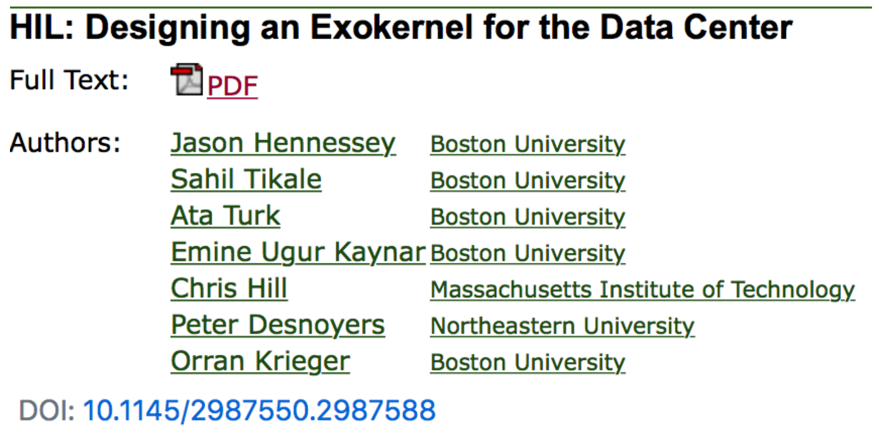
- Similar debate occurred around parallel images in Fortran 2015. They can be RUNNING; STOPPED; FAILED
  - How this works in practice for large systems of images is TBD though!

- More

https://www.youtube.com/watch?v=yRtyvfzDt94	Rajul explaining how this works with OpenStack VMs
https://github.com/CCI-MOC/hpc	Includes experimental modifications to Slurm heartbeat

Part 2 - creating a "self-service" (i.e. no admin involvement) Slurm reservation system for managing temporary loaning of hardware to non-production environments.

- Slurm as a way to automate handling requests for loaning hardware outside of Slurm cluster.
- These requests occur for us around
  - Temporary use of "bare metal" hardware for special software stack interests – custom OS hacks (I/O, network, memory management experiments)
  - Temporary use of "bare metal" hardware for private data experiments
  - Beyond capabilities of KVM, Singularity etc…. to manage incompatible software stacks
- Exploring Slurm as a useful *master scheduler* for this working in conjunction with a proxy tool (Hardware Isolation Layer – **HIL**) for managing controlled access to privileged network and out-of-band actions



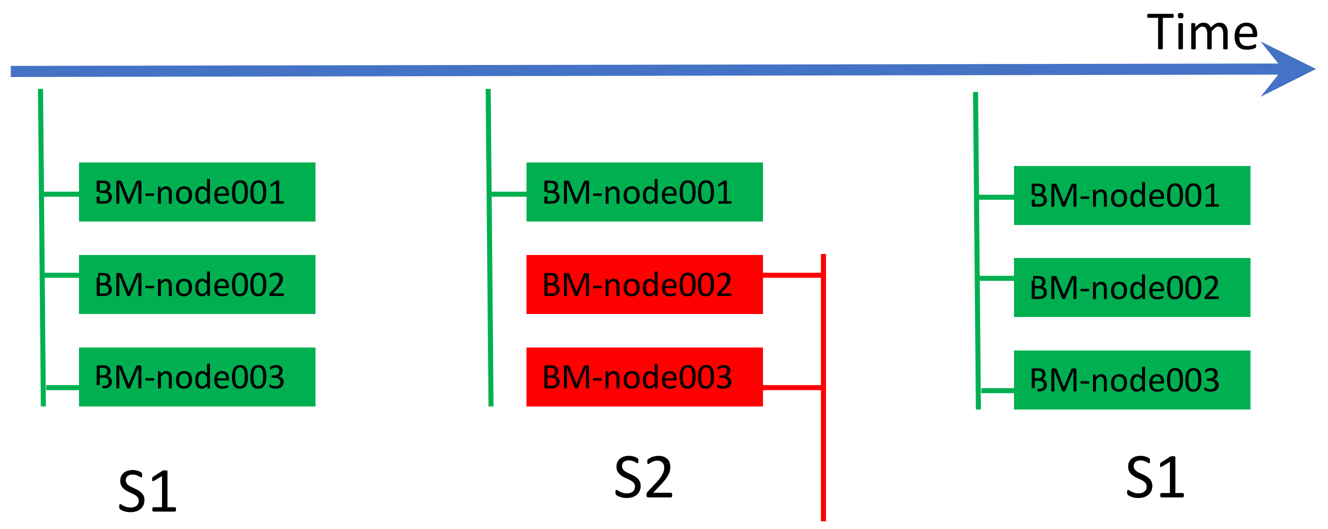
HIL: Designing an Exokernel for the Data Center

Full Text: PDF

Authors:		
	Jason Hennessey	Boston University
	Sahil Tikale	Boston University
	Ata Turk	Boston University
	Emine Ugur Kaynar	Boston University
	Chris Hill	Massachusetts Institute of Technology
	Peter Desnoyers	Northeastern University
	Orran Krieger	Boston University

DOI: 10.1145/2987550.2987588

General “Hardware Isolation” Scenario



S1: All nodes part of main production cluster/network. Standard software etc…

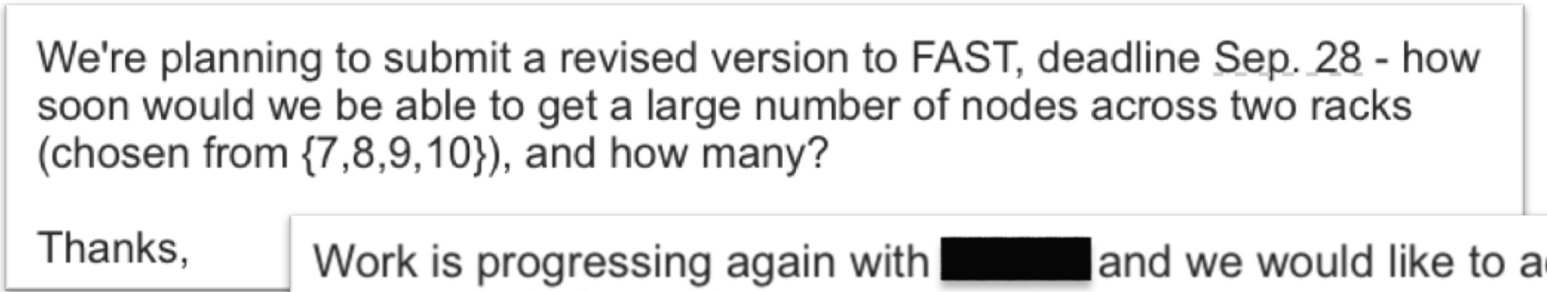
S2: Red nodes are temporarily part of some other cluster/project/network. Unknown software etc…

Use Slurm as base for master control for self-service, user controlled node transfer requests (S1 ➔ S2)

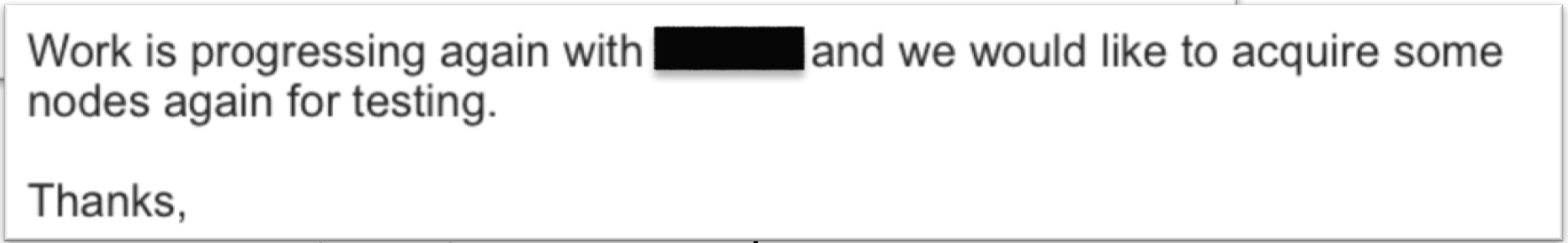
Slurm provides rules and manages transfer and “repo” automatically (S2 ➔ S1) †

† human nature 1 - nobody ever returns nodes “eagerly”, need to automate repo.

Typical human style requests


We're planning to submit a revised version to FAST, deadline Sep. 28 - how soon would we be able to get a large number of nodes across two racks (chosen from {7,8,9,10}), and how many?

Thanks,


Work is progressing again with [redacted] and we would like to acquire some nodes again for testing.

Thanks,

In both cases requests are for software stacks/privacy needs that are incompatible with Slurm cluster general setup.

We want these users to be able to use Slurm to automate. Let user request transfer of nodes via Slurm; trigger transfer and repossession of nodes. Transfer and repo † steps are other software, but we want Slurm to be the master scheduler.

† human nature 2 – we have never, ever received a message to say "we are done with resources".

Approach – two part (Slurm and outside of Slurm)

Slurm piece

1. Create partition that controls which nodes and which users can request hardware isolation, isolation duration limits etc….
2. Users can submit requests (via sbatch) to that partition and use Features to request nodes from a certain subset (e.g. some nodes from a particular rack).
3. A prolog is triggered by partition name that creates Slurm reservations corresponding to the transfer and repossession actions.

```
ReservationName=cnh-tufts-physical-expt StartTime=2017-07-13T09:52:55 ...

ReservationName=flexalloc_moc_20170410_rack7_8 StartTime=2017-09-01T11:12:27 …
```

A reservation name PREFIX_GROUP is recognized by external software that then applies appropriate isolation steps.

Approach – two part (Slurm and outside of Slurm)

Interface to non Slurm pieces

1. Slurm reservations are monitored by daemon that triggers transfer actions using separate software (HIL and other tools)
    - Specific VLAN, PKEY and enable/disable application on network interfaces
    - Node shutdown
2. Slurm reservation name prefix are used to indicate to daemon which reservations are Hardware Isolation requests
3. Two reservations are created for each request. One triggers transfer when it becomes active, one triggers repossession when it becomes active.

Conclusions

- Status – actively being tested as a replacement for cumbersome hand process
- See More –
  - https://github.com/mghpcc-projects/user_level_slurm_reservations
  - https://github.com/CCI-MOC/hil

- Alternates exist in CS world
  - Chameleon - Blazr
  - Ostack – Ironic
  - Emulab
  - All appear much harder to see how they would fit in a production HPC environment
  - Slurm as “master” + HIL strategy seems particularly elegant (to us!).

- Probably not supposed to use Slurm for this (or even do this) but it looks like it will work rather nicely especially when paired with the HIL proxy for privileged access/operations.

Questions?