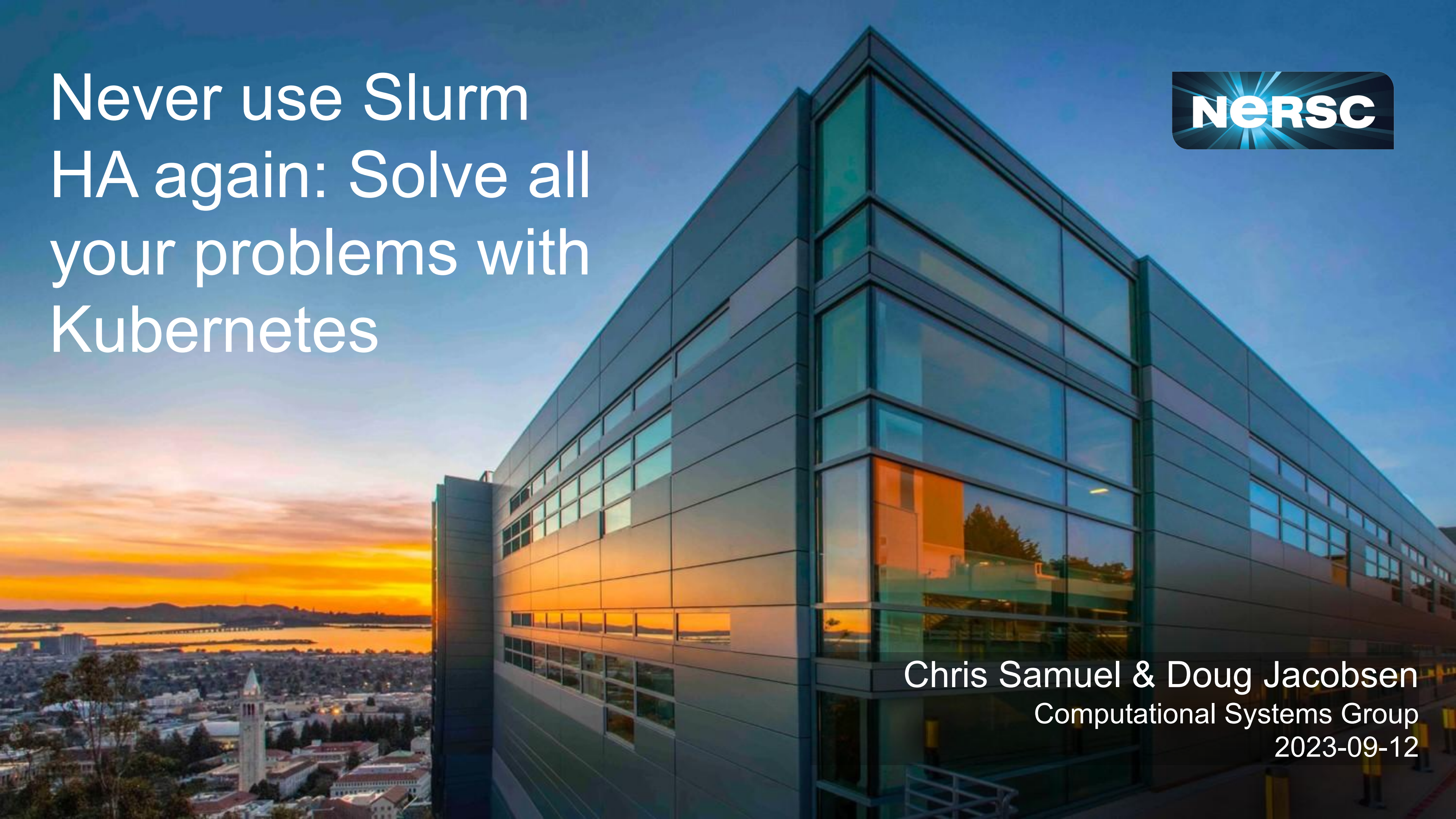
Never use Slurm HA again: Solve all your problems with Kubernetes



Chris Samuel & Doug Jacobsen
Computational Systems Group

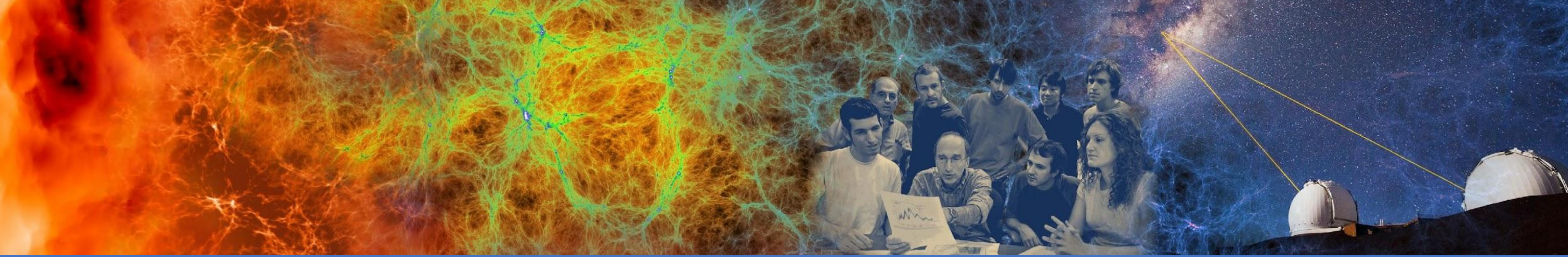
2023-09-12

I wish to acknowledge the Traditional Owners of the land on which we meet today, the peoples of the Timpagnos Nation and I pay my respects to their Elders past, present and emerging.

Caveats and Acknowledgements

- I am not an expert on kubernetes
  - This is based on what has worked for us
  - There may well be better ways to achieve some of this
  - Always happy to hear ideas and improvements!

- The initial starting point for our work was inspired by HPE’s work on the Shasta software stack (now called CSM - Cray Systems Management)
  - But evolved based on our needs ever since

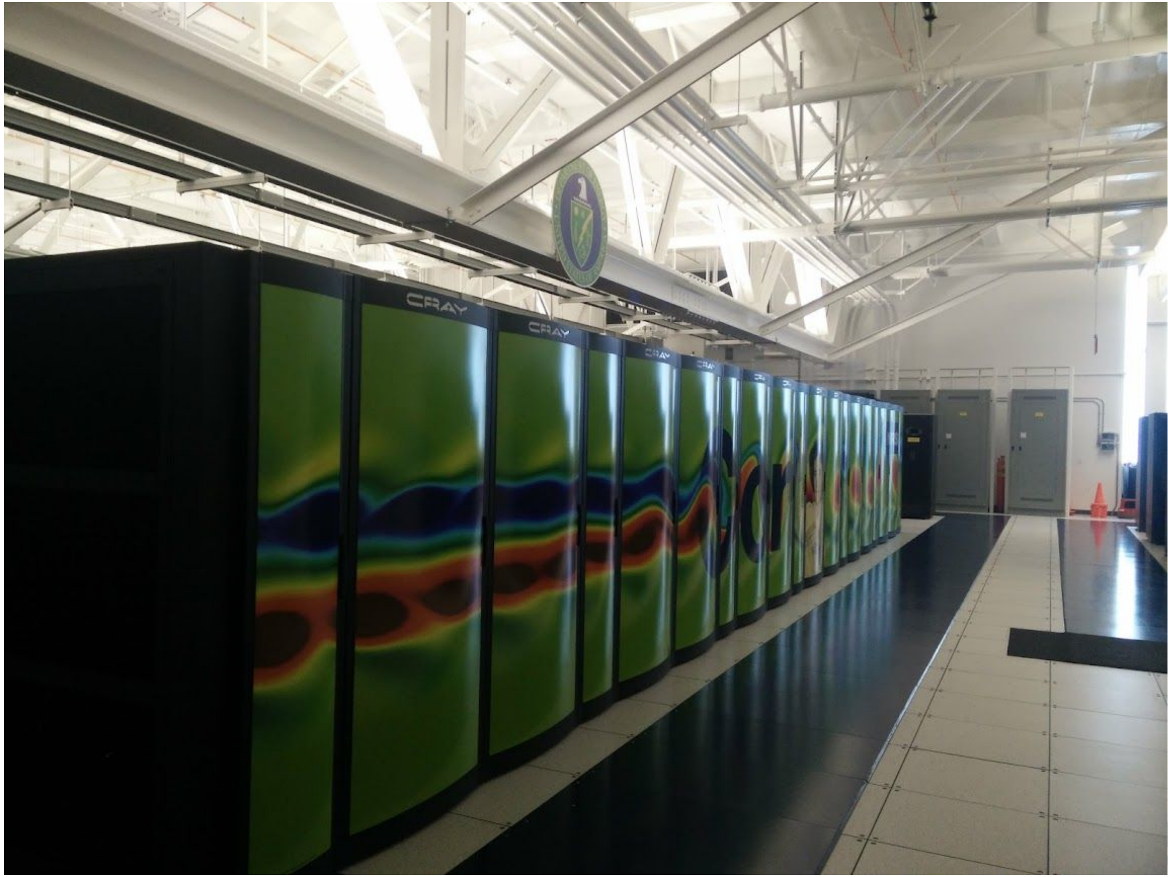


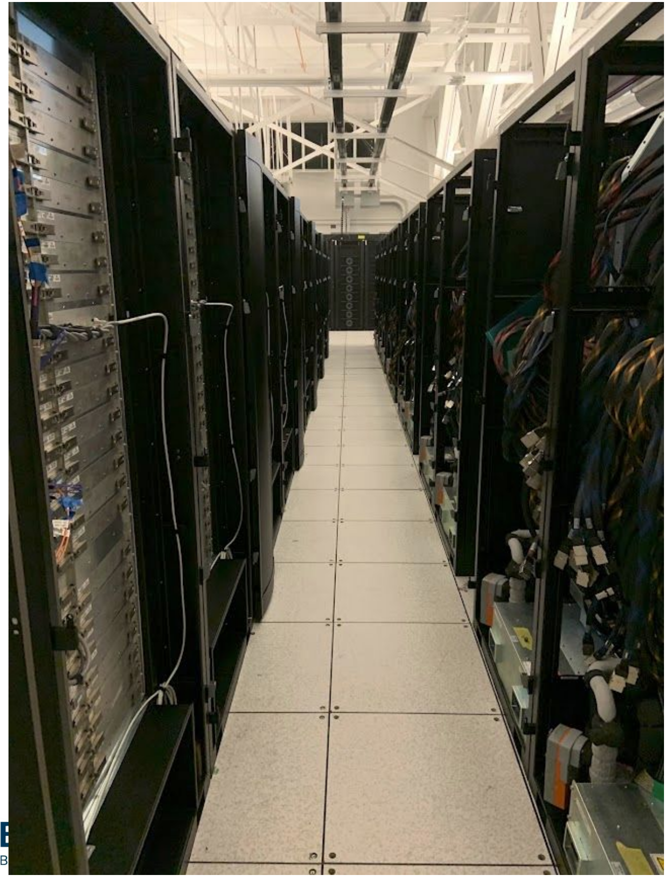
Where are we coming from?

A (very) brief overview of Slurm HA

- Why do we want High Availability?
  - A slurmctld crash shouldn’t stop users doing Slurm things
    - But they never happen, right? 😉
  - A node crash shouldn’t stop users doing Slurm things
    - These definitely do happen.. 😬
- Originally a pair of nodes - each running slurmctld, now can be more
  - Pre 18.08 used a BackupController/BackupAddr directive
  - Now uses a list given by SlurmctldHost
- The first SlurmctldHost is the primary, others are backups
- They must all share the same state directory
  - So has to be a distributed/network filesystem
- Admin can use “scontrol takeover” to switch which node is the primary

Cori - Cray XC40

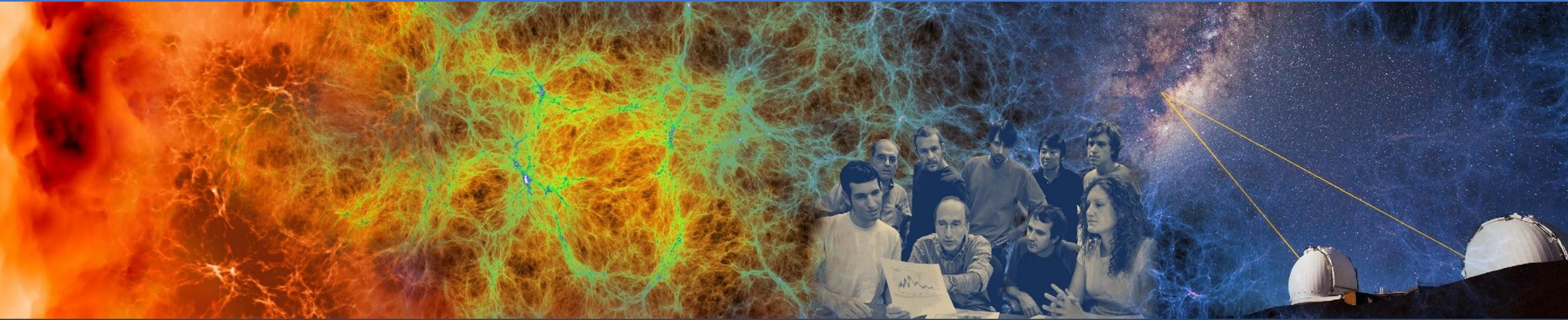




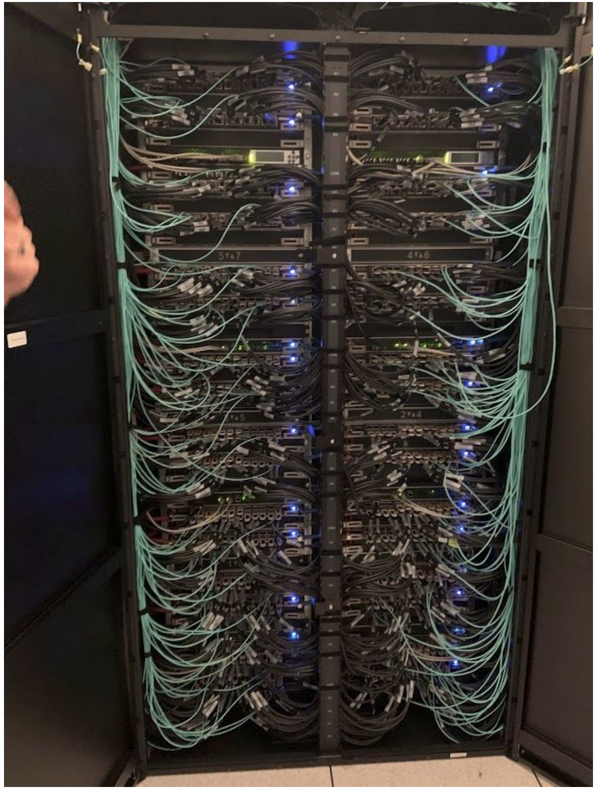
Slurm on Cori, our Cray XC system

- Tens of thousands of compute nodes (combination of Intel Haswell and KNL)
  - Retired mid year, #48 on Top 500 (~14PF, ~3.9MW)
- slurmctld running on slower Sandybridge “service nodes”
  - Restrictions due to architecture of system
  - No local storage but connected via Infiniband to NERSC GPFS fabric
  - state directory on GPFS
  - ctl1 and ctl2 as the HA pair
  - Failover used to (mostly) work up until Cray CLE7, then failover mostly didn’t
    - Hard to track down, appeared to be a hang in integration with Cray DataWarp plugin - Cray provided code
    - Decided rather than invest time in debugging only ran with it on ctl1, ctl2 had it shut down but present as a cold spare
    - Nagios updated to alert if absent or present on both. 😎

Perlmutter: The Next Generation



Perlmutter - HPE Cray EX



Basic structure of our Cray EX Systems

- Kubernetes Cluster - HPE Cray Systems Management (CSM) base
  - “Manager” nodes - running etcd clusters for Kubernetes
  - “Worker” nodes - running a myriad services for systems management
- Utility Storage - from CSM
  - Ceph cluster providing network filesystems, block storage and object storage
- Platform Storage - HPE E1000
  - Lustre filesystem
- High Speed Network
  - Slingshot 11 network - Rosetta switches, Cassini NICs
- Compute Hardware
  - Direct liquid cooled compute nodes
  - “Grizzly Peak” - AMD Milan CPU, 4 x A100 GPUs, 4 Cassini NICs
  - “Windom” - Dual AMD Milan CPUs, single Cassini NIC

Our family of EX systems

- Perlmutter - the production system
  - Currently #8 on the Top 500 (~70PF measured, ~2.5MW)
  - Runs from the "production" branch of our various git repositories
- Muller (Dr Perlmutter's PhD supervisor)
  - Pre-production Test and Development System (TDS)
  - Software only goes to Perlmutter after being tested here
  - 5 workers, 3 utility storage nodes, 3 manager nodes
  - A handful of each of the GPU and CPU nodes
  - Usually running from "development" branch, sometimes a feature branch.
- Alvarez (Dr Muller's PhD supervisor)
  - Experimental Test and Development System (TDS)
  - Exciting/Exhilarating/Exploding software testing and proving ground
  - Only system where we've corrupted the state directory (so far) - GPFS kernel bug
  - Successful testing here can feed into the "development" branch to go to Muller

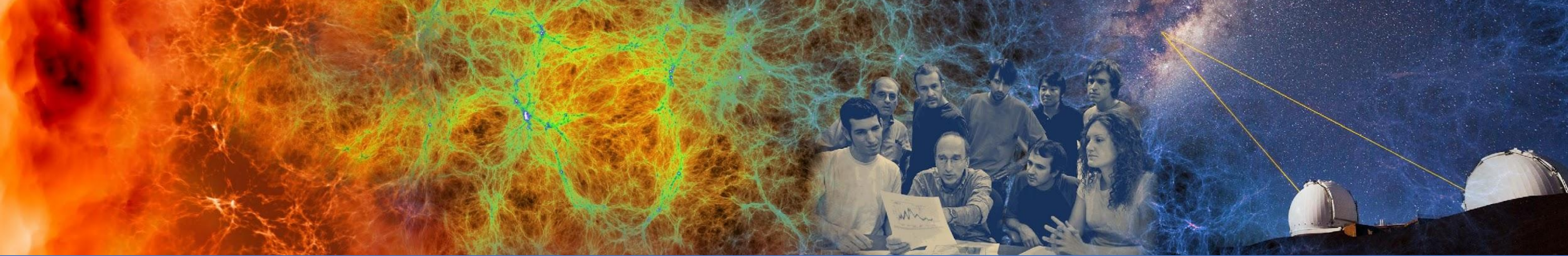
WHAT'S KUBERNETES
EVER DONE FOR US?
imgflip.com

A brief overview of what I know of k8s

- A system to deploy, manage and (potentially) expose persistent and transient container based workloads and services
- Workloads:
  - pods: one or more containers bound into a single unit to accomplish a task
  - deployment: one of many ways to manage a pod to provide a service, allows you to have multiple replicas of a pod
- Configurations (NB: appear to get atomically updated):
  - configmap - a text object made accessible as files in a directory
  - “secrets” - a base64 text object made accessible as files in a directory
- Storage:
  - Persistent Volume Claim: a request to use a storage (of an available PV type)
- Influencing placement of workloads:
  - labels: a text label you apply to a worker node for some reason
  - taints: used as a way to keep workloads that don’t tolerate a taint off a worker

A brief overview of what I know of k8s

- Operators
  - Software entities that are meant to encode the knowledge of how to deploy, rn and maintain a particular service
  - A deployment to manage other deployments, if you will
- Networking
  - Often kubernetes services are placed behind a reverse proxy style configuration referenced by a dynamic DNS style system which can add authentication, etc. However, this assumes an HTTP like protocol which Slurm most definitely does not have, hence we need a specialised..
  - CNI (Container Network Interface) allows you to use plugins (eg macvlan, ipvlan) to create a service exposed outside of kubernetes on a particular IP address

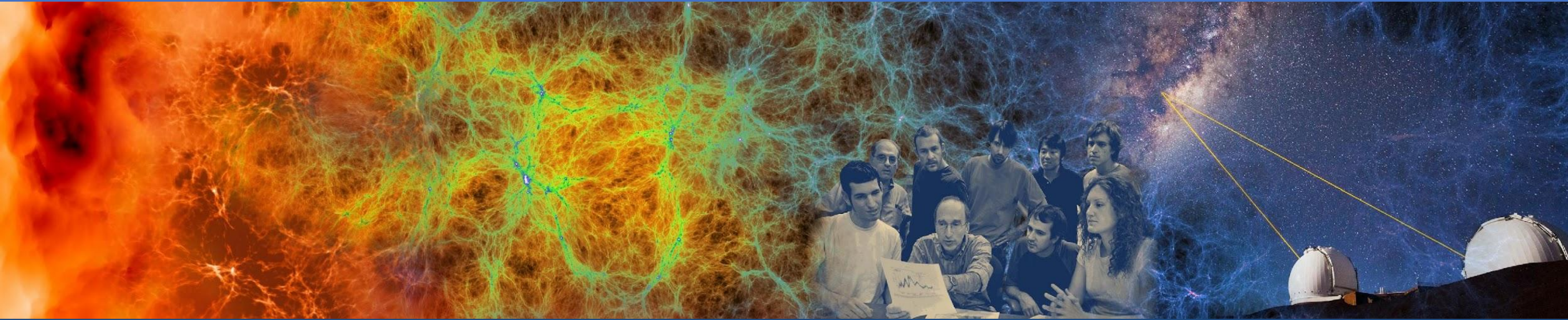


Putting this all together

How did we arrange our deployment?

- Percona XtraDB Cluster (PXC) operator to provide a MariaDB cluster with 5 replicas on Perlmutter, 3 on the TDS's. Also an haproxy front end. Not exposed externally.
- slurmdbd deployment (plus munge and sssd)
  - single replica
  - macvlan provisioned IP address exposed to users
  - no persistent storage (logs sent out to kubernetes as well as to local file)
- slurmctld deployment (plus munge, sssd, redis, nginx, iris, capmc)
  - single replica
  - macvlan provisioned IP address exposed to users
  - state directory & redis directory on a persistent CephFS mount (PVC)
- NB: after our Cori experience we included a couple of low core count, high clock frequency worker nodes dedicated to Slurm for Perlmutter. These are tainted to prevent other workloads, our Slurm deployments tolerate that and bind to labels.

This is where I let you all down (sort of)



OK, SO WHERE'S THE HA?
KUBERNETES
imgflip.com

Always has been..
Wait, it's all just Kubernetes?
imgflip.com

So where’s the High Availability?

- OK this is where the trick is - we don’t need to do anything special to get HA!
  - Kubernetes is all about automating the deployment and existence of services
- By default kubernetes will restart a failing container
  - This can make debugging slurmctld issues tricky so our existing startup script for slurmctld (needed to ensure sssd is happy prior to launching either slurmdbd or slurmctld) will restart it for us inside the same container with a sleep 5 between invocations
  - This also lets us swap slurmctld with a shell script that sleeps forever so we can run a crashing slurmctld with gdb inside the environment
- Node crashes are also handled by kubernetes, when it detects a node is unresponsive then it will start failing pods on it over to other workers that match their requirements.
  - Yes, we have had this happen and yes, it has worked!

No really, that’s it!

- To summarise
  - We were going to be using the kubernetes base (provided by HPEs CSM software) anyway as it was the way to manage the hardware
  - It provides built in handling of the failure modes we care about
    - Though for daemon failures we take a slightly different path, but from a choice to ease debugging
- But could you do Slurm HA in Kubernetes?
  - I suspect the answer is yes
  - Using a statefulset rather than a deployment
    - Pods will then have predictable names, essential for this
  - State dir PVC would need to be marked as/capable of RW for multiple mounts
  - Would need to allocate multiple IPs for each pod
  - Might complicate/slow upgrades

Caveats and possible future work

- Caveats
  - MacVLAN CNI has posed some issues
    - If a pod is deleted you can get stale IP assignments left on a worker node
    - If the pod tries to reschedule on that worker it will fail to start and just hang
    - Needs intervention to remove files (but cleaned on a reboot by CSM)
    - Scaling down to 0 pods does to the right thing
- Future work
  - Migrating to ipvlan instead of macvlan
  - slurmrestd deployment
    - much more amenable to kubernetes way of working
      - can put an authenticating proxy in front of it for instance
  - A slurm operator
    - Try and encode our knowledge of dealing with certain issues into it
    - Eg noticing it being rescheduled due to node failure & bringing partitions up

That's all Folks!