Slurm and/or/vs Kubernetes

Tim Wickberg
SC'22

Opening Questions

- Who's running Slurm today?
  - Who's running Slurm in the cloud?
- Who's running Kubernetes for infrastructure?
- Who's using Kubernetes for research workloads?

Difference in Perspectives

- K8s built to manage long-running processes
    - Coordinating multiple microservices - scaling, managing availability
    - Usually in support of one or more web services
- Cloud-native systems assume "infinite" resources are available
    - Prioritization not a central aspect of cloud orchestration
    - This is reflected in the scheduling semantics
        - "Affinity" vs. "Anti-Affinity" settings don't translate to batch workflows

Difference in Perspectives

- HPC systems assume system size is fixed
  - … but workload is infinite
  - Queue prioritization critical for most large-scale systems
- "Slurm is a policy engine"
- Slurm covers several related HPC systems management tasks
  - Job queuing and prioritization
  - Job accounting
  - User access control to compute resources (cgroups, pam_slurm_adopt)
  - Large-scale job launch (MPI, PMIx, nss_slurm, sbcast)

Current Kubernetes Batch Support

- K8s has limited support for batch workflows
    - Modeled as either individual "pods", or as "jobs"
    - Most workflows use "pods" due to issues around the "jobs" model
- Prioritization models are limited
    - FIFO is most common

Current Kubernetes Batch Support

- MPI-style workload support is weak
    - Concurrent pod scheduling is not guaranteed by default K8s components
- "MPI Operator" is the most commonly used component to ensure pods launch concurrently
    - Does not scale - struggles to launch above 80 (!) ranks
        - Citation - https://sc22.supercomputing.org/presentation/?id=ws_canopie106&sess=sess438

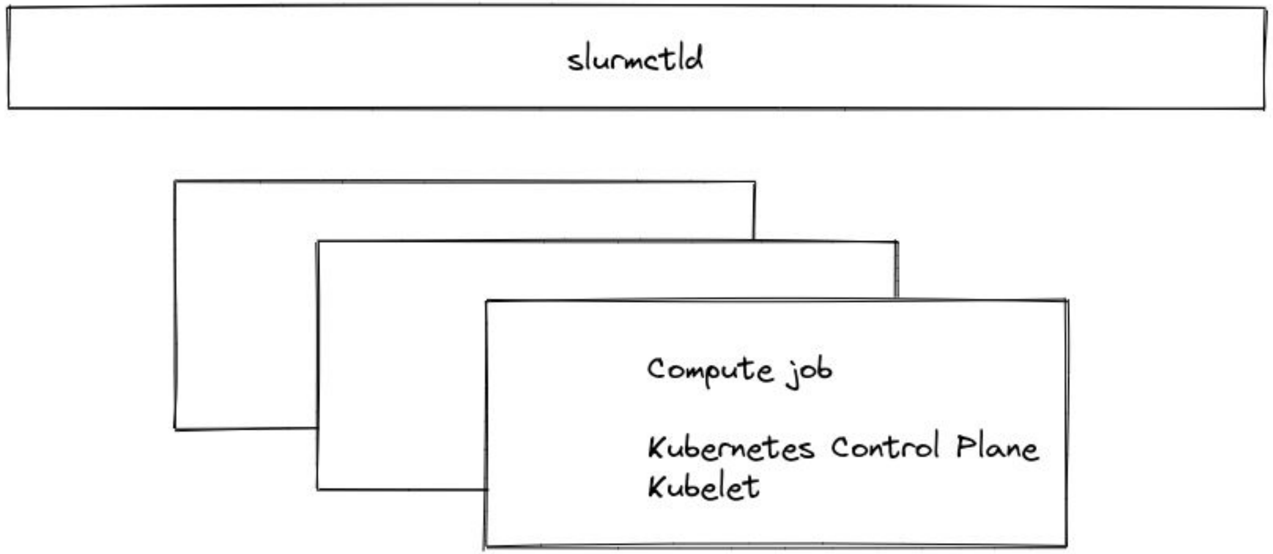
Convergence of HPC and Cloud-Native

- So… why should we care?
    - Opportunity to bridge the gap between HPC and Cloud-Native workloads
    - Find a way to bring familiar commands, tooling, prioritization models into newer architectures

Possible Models

- Potential approaches, from Slurm's perspective:
  - "Over"
  - "Adjacent"
  - "Under"

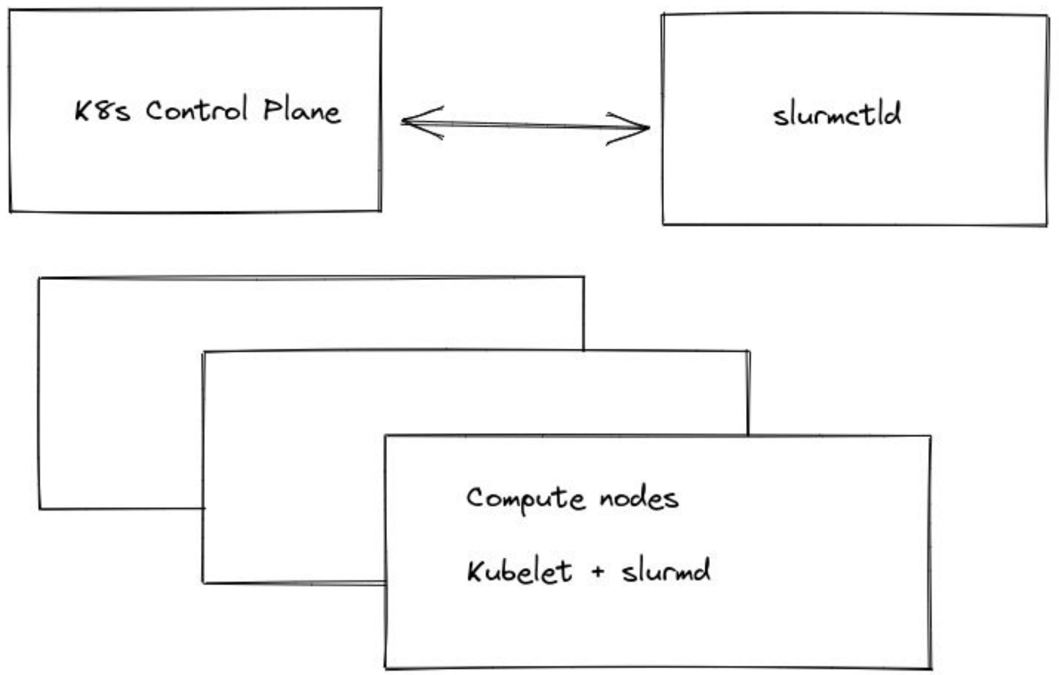
"Over"



"Over"

- Slurm manages resources
- K8s clusters created ephemerally within batch jobs
- K8s control plane unavailable until job launches…
- Not particularly useful beyond test / development

"Adjacent"



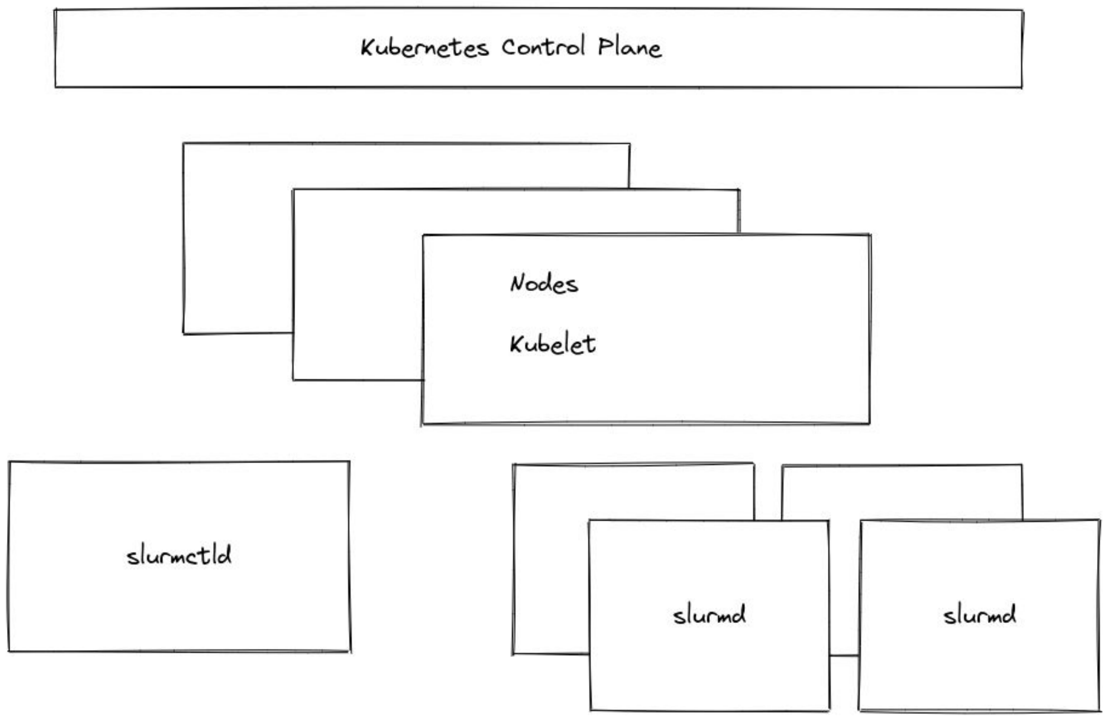
"Adjacent"

- Overlap both control planes
- Install Slurm K8s scheduler plugin
  - Have Slurm prioritized and schedule both Slurm and K8s workloads
- K8s jobs run through kubelet
  - Have full access to K8s capabilities
- Slurm jobs run through Slurm
  - High-throughput, allows for large-scale MPI work

"Adjacent"

- Proof-of-concept working already
  - Through a "slurm-k8s-bridge" K8s scheduler plugin
- Limitations
  - K8s scheduling only cares about nodes
    - No further granularity available currently
      - Working on this, but changes are difficult to push upstream
  - Slurm scheduling plugin ignores affinity/anti-affinity, other K8s scheduling mechanisms, as they don't directly model to traditional HPC concepts

"Under"


Kubernetes Control Plane
Nodes
Kubelet
slurmctld
slurmd
slurmd

"Under"

- Run Slurm cluster(s) within a K8s environment
- K8s-native cloud providers are already emerging
- Long-lived "login" nodes (pods) provide for traditional user experience
- Auto-scaling can be used to shift resources to/from Slurm's control
    - Slurm 22.05 - dynamic nodes - greatly simplifies this experience

"Under"

- Pros
  - Traditional experience for Slurm users
  - Allows for higher throughput, and full MPI support
- Cons
  - K8s workloads outside of Slurm's purview
  - Prioritization between Slurm and K8s workloads difficult
    - All limitations of K8s scheduling apply

"Under"

- Usable today
- But plenty of places to improve
  - Provide reference deployments for a Slurm containerized control plane
  - Develop a K8s Operator to auto-scale the Slurm environment

Open Forum

- A few starting questions:
  - When your researchers tell you they "need Kubernetes", what do they mean?
    - They have a deployment from somewhere they want to run?
    - They need long-lived services for their workflow alongside their compute?
  - Do you expect your researchers to need CNI, Sidecars, or other extended K8s features?