Slurm Bridge

slinky

Skyler Malinowski
Alan Mutschelknaus
Marlow Warnicke

Slurm User Group 2024

What is Slinky?

A collection of projects and initiatives to enable Slurm on Kubernetes:

- Slurm-operator
  - Manage Slurm nodes in Kubernetes
- **Slurm-bridge**
  - Enable Slurm scheduling of Kubernetes Pods
- Kubernetes Tooling
  - Helm Charts
  - Container Images
- Future work

HPC vs. Cloud Native - Historical Assumptions

HPC

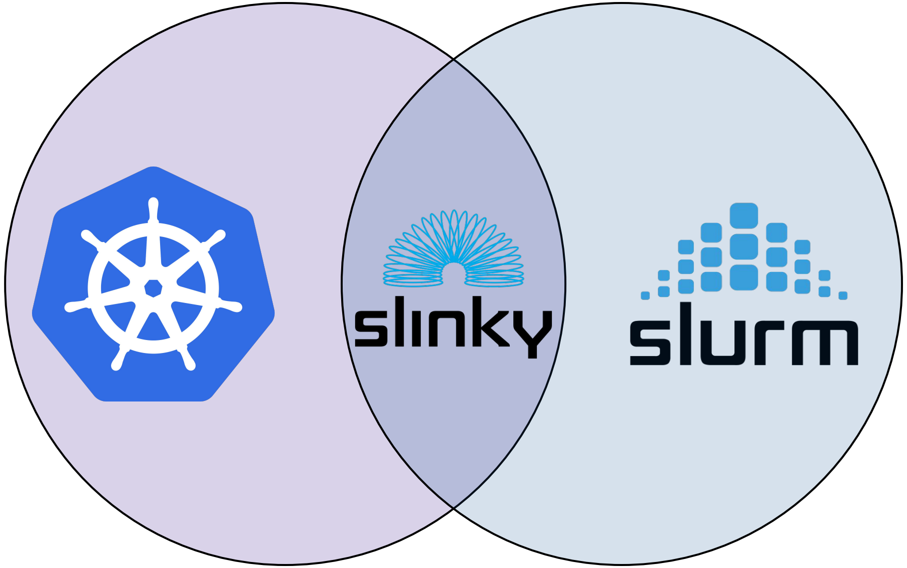
- Underlying software is mutable
  - Users assume fine-grained control
- Users are often systems experts that understand infrastructure
  - Have a tolerance for complexity
- Access to compute handled by a resource manager or scheduling system
- Users own the node entirely during computation
- Assumption of node homogeneity

Cloud Native

- Underlying software is immutable
- Users are not systems experts, do not think in terms of parallel
  - Limited tolerance for complexity
- Users share nodes
  - Can introduce jitter
  - Can blow through bandwidth
- Assumption of heterogeneous nodes
- Not a ton of attention given to network topology

Domain Pools

- Kubernetes manages its nodes, running a kubelet
- Slurm manages its nodes, running a slurmd
- Slinky tooling will manage the overlapping nodes
  - **Slurm Bridge**



Why Slurm Bridge

- Kubernetes lacks fine-grained control of native resources (CPU, Memory)
    - HPC and AI training workloads are inefficient
    - Need to build the infrastructure to get this capability
- Ability to have fast scheduling that is not possible in kubelet
- Ability to use both Kubernetes and Slurm workloads on the same set of nodes
    - Do not need to separate the clusters!

Slurm Bridge

Requirements

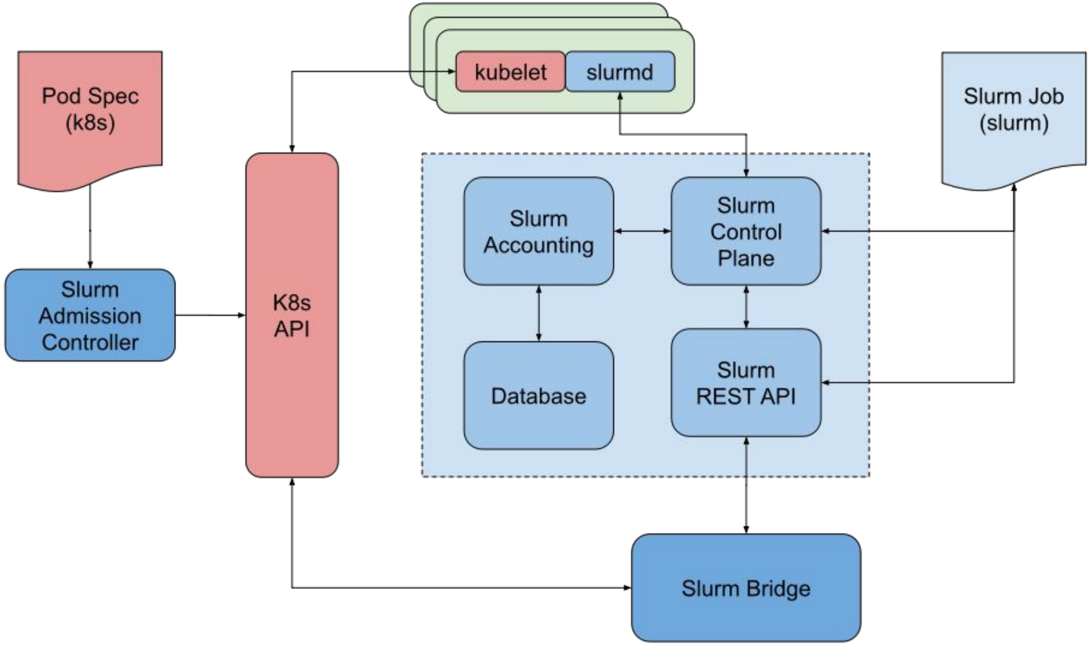
- Can run Slurm and Kubernetes workloads on pools of nodes
- Slurm bridge will translate resource requirements for Kubernetes workloads and make sure appropriate resources are available and schedule within the cluster
- Handle Device Plugins, such as GPUs
- Will filter nodes that Slurm is not to manage, through the current set of labels provided
- Will filter pods out that we do not handle, like pods on control plane, via designated namespaces
- Will have an allow-list of namespaces we handle

Restrictions

- Each node can run Slurm **or** Kubernetes workloads, not both concurrently
    - The kubelet will manage on-node resource assignment for its workload
    - The slurmd will manage on-node resource assignment for its workload
- Configure Slurm with Multi-Category Security (MCS)
    - Use to enforce workload exclusivity (e.g. Kubernetes vs. Slurm workload)
    - Requires Slurm Accounting for user, account information
- Configure our plugin as Kubernetes scheduling profile
    - Our scheduling plugin will take control of all workloads in allow-list namespaces
    - The Default Scheduler will handle all other workload
- Device plugins will be supported, but not Dynamic Resource Allocation (DRA)
    - DRA is still in alpha and has a volatile API, currently

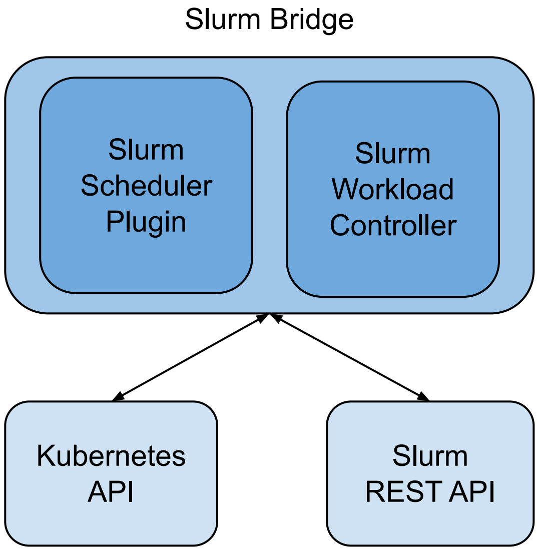
Big Picture

- Slurm uses Multi-Category Security (MCS) to label nodes with Kubernetes or Slurm workload, to enforce node workload exclusivity
- Slurm cluster can still run workloads on nodes with only a slurmd (no kubelet)
- The Admission Controller makes sure pods use the Slurm Scheduler Plugin



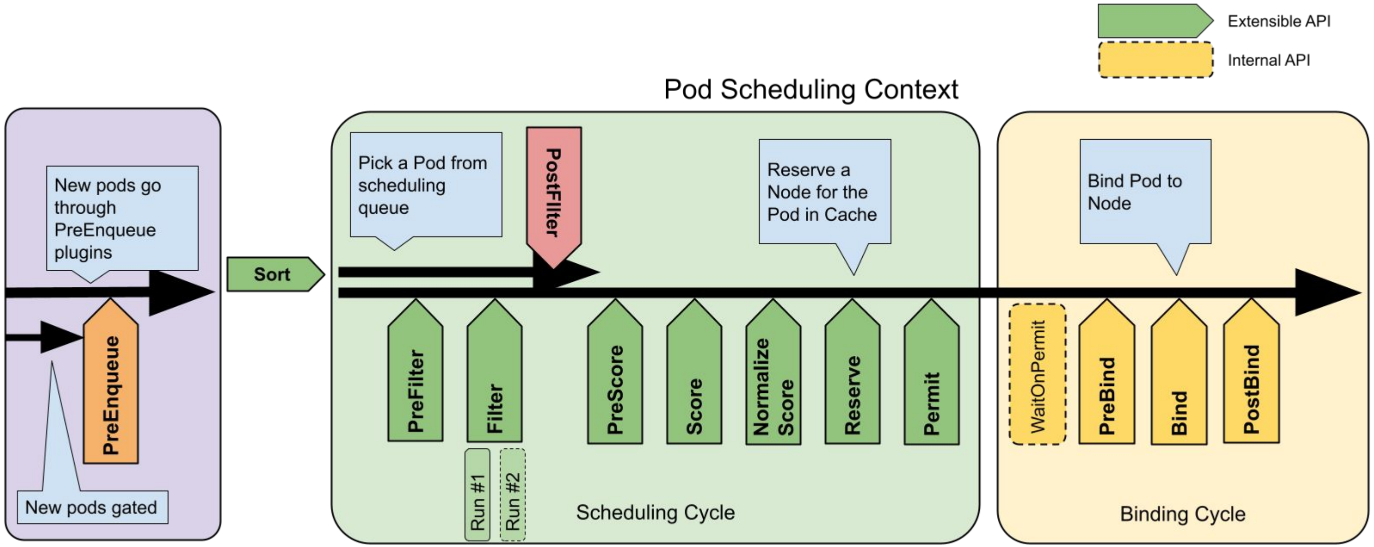
Slurm Bridge

- Responsible for managing Slurm as the source of truth and enforcing scheduling decisions from Slurm
- Slurm Scheduler Plugin
  - Implements the Kubernetes Scheduler Framework
  - Hooks into the Kubernetes scheduling API to utilize the Slurm Control Plane to make scheduling decisions
- Slurm Workload Controller
  - Reconciles pod drift/desync using Slurm as the source of truth for Slurm scheduled workloads



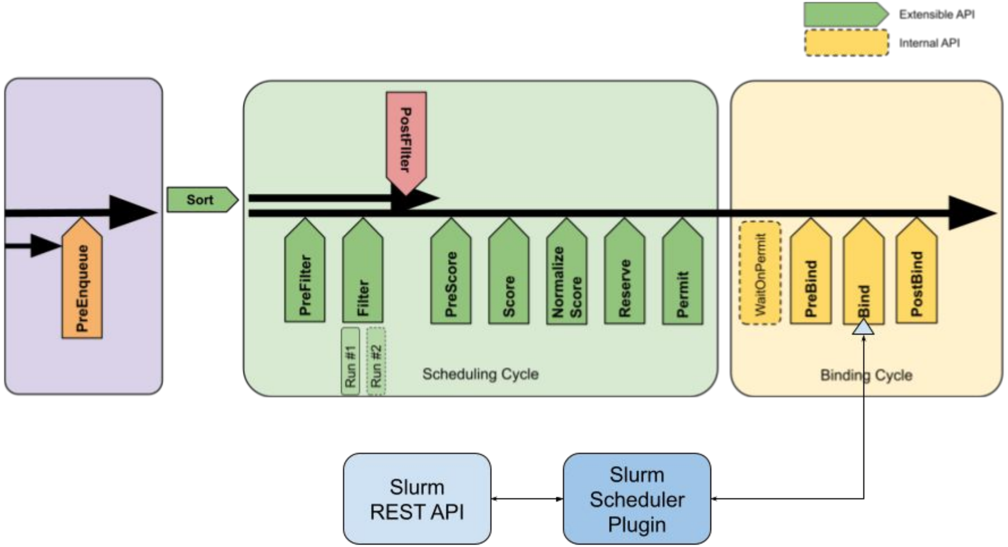
Slurm Scheduler Plugin

Kubernetes Scheduler Framework



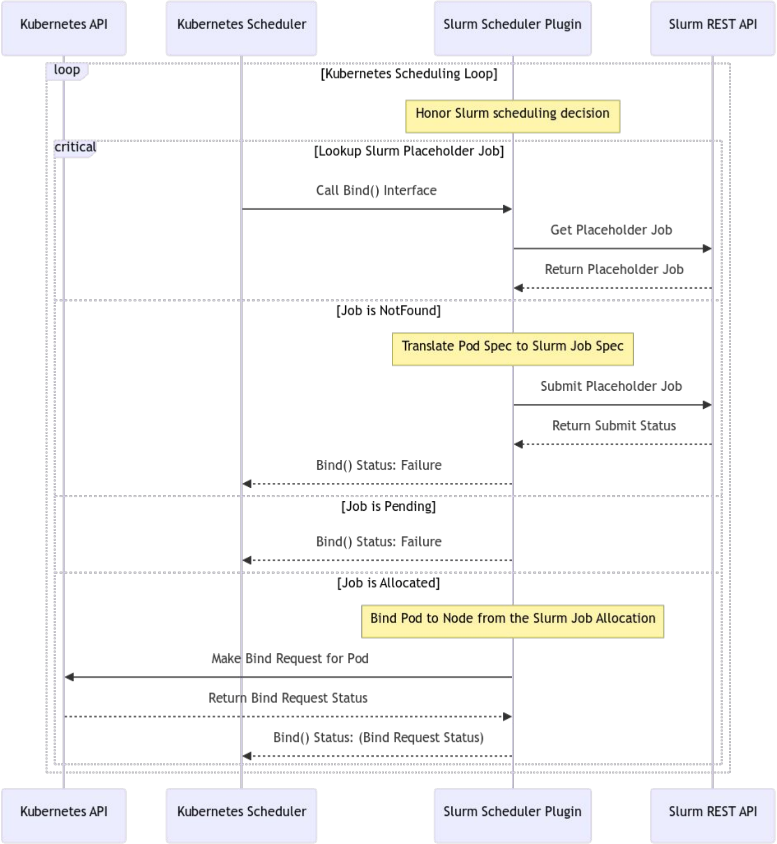
Slurm Scheduler Plugin

- Implement Bind() Interface
- Translate Pod spec into Slurm job spec, and submit as a placeholder Slurm job
- Bind pod to Kubernetes Node based on Slurm node allocation, from Slurm job
- Let kubelet handle the pod initialization and resources



Slurm Scheduler Plugin - Sequence

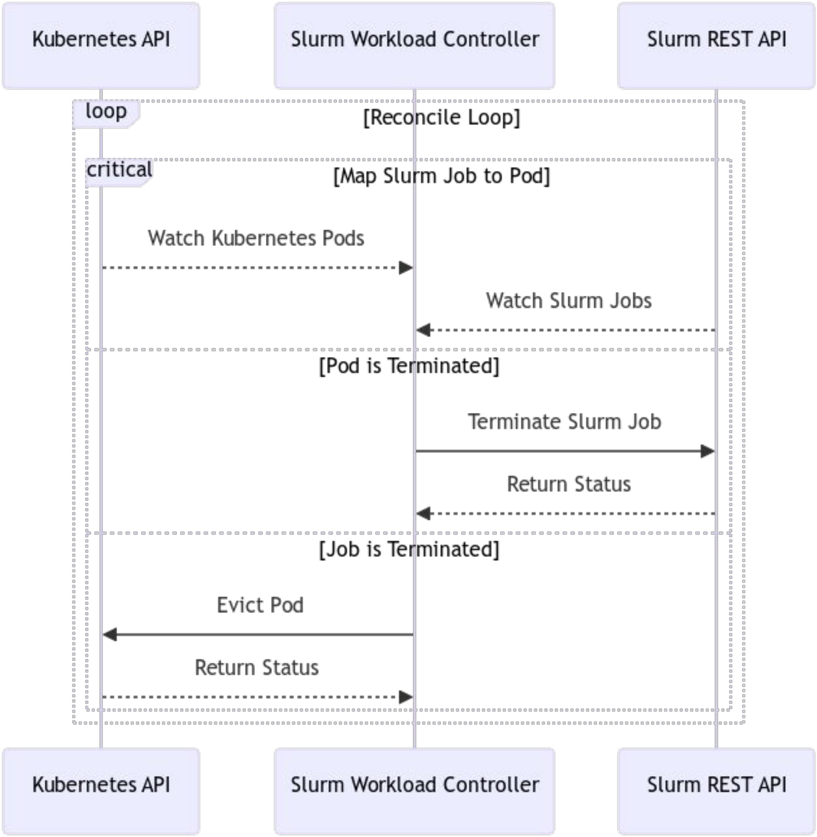
- Translate a pod spec to Slurm job spec
- Submit a placeholder job in Slurm for the pod
- Bind the pod to a node, given where the Slurm job was allocated in Slurm
    - Slurm will only allocate the placeholder Slurm job on nodes with both a slurmd and kubelet


Kubernetes API
Kubernetes Scheduler
Slurm Scheduler Plugin
Slurm REST API
loop
[Kubernetes Scheduling Loop]
Honor Slurm scheduling decision
critical
[Lookup Slurm Placeholder Job]
Call Bind() Interface
Get Placeholder Job
Return Placeholder Job
[Job is NotFound]
Translate Pod Spec to Slurm Job Spec
Submit Placeholder Job
Return Submit Status
Bind() Status: Failure
[Job is Pending]
Bind() Status: Failure
[Job is Allocated]
Bind Pod to Node from the Slurm Job Allocation
Make Bind Request for Pod
Return Bind Request Status
Bind() Status: (Bind Request Status)
Kubernetes API
Kubernetes Scheduler
Slurm Scheduler Plugin
Slurm REST API

Slurm Workload Controller

Slurm Workload Controller - Sequence

- Slurm is the source of truth for Nodes that overlap between Kubernetes and Slurm
- Requires the Slurm Scheduler Plugin to schedule pods
- The Slurm Bridge will consider the allow-list namespaces, which it tightly manages



Future Work

Future Work

- Work with the Kubernetes community to be able to handle fine-grained control and understanding of native resources
- Be able to handle Dynamic Resource Allocation (DRA)
- Allow Slurm to schedule Kubernetes workloads without slurmd needing to run alongside kubelet

Questions?

SCHEDMD

The Slurm Company

Extended Reading

Slurm Attribute Mapping

- Want to allow pods to map into Slurm job settings for:
  - CPUs / Cores
    - Slurm manages Cores. Need to map to Cores.
  - Memory
    - Slurm only has a hard limit, not soft limit
  - GPUs
    - device-manager syntax only initially
    - DRA has separate (evolving) syntax we want to ignore for now
  - Optional, should use defaults from Slinky-Bridge configuration:
    - Slurm Account
    - Slurm User*
      - Note: would need REST interface to run with root-level jwt to allow this to be manipulated. Running as a single service account would be preferable in phase 1.
    - Slurm Partition
    - Slurm Constraints
    - Slurm Job Name
    - GRES