Slurm Operator

Skyler Malinowski
Alan Mutschelknaus
Marlow Warnicke

Slurm User Group 2024

slurm | SCHEDMD

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

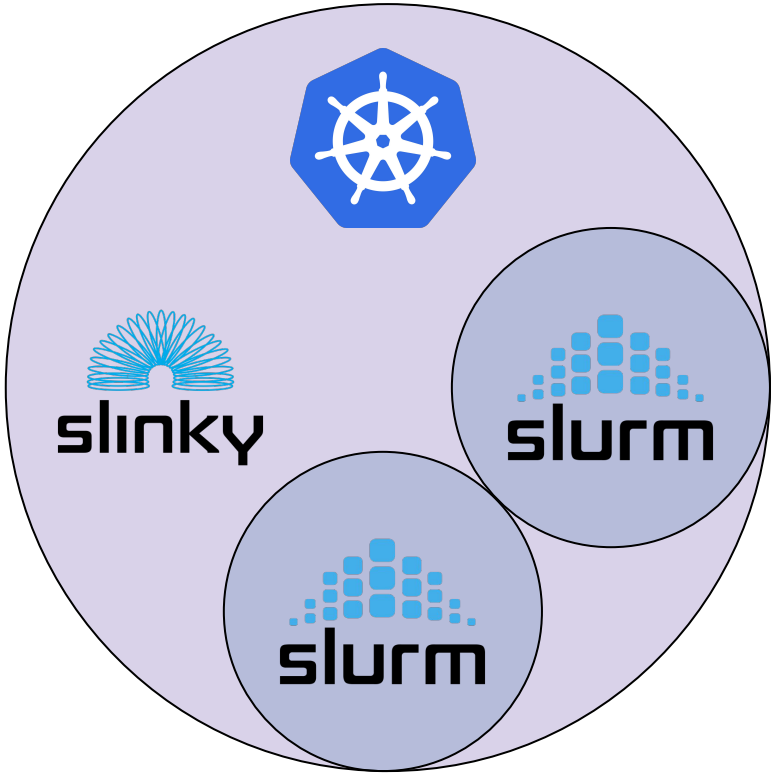
- Underlying software is mutable
  - Users assume fine-grained control
- Users are often systems experts that understand infrastructure
  - Have a tolerance for complexity
- Access to compute handled by a resource manager or scheduling system
- Users own the node entirely during computation
- Assumption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



Why Slurm Operator

- 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 Operator

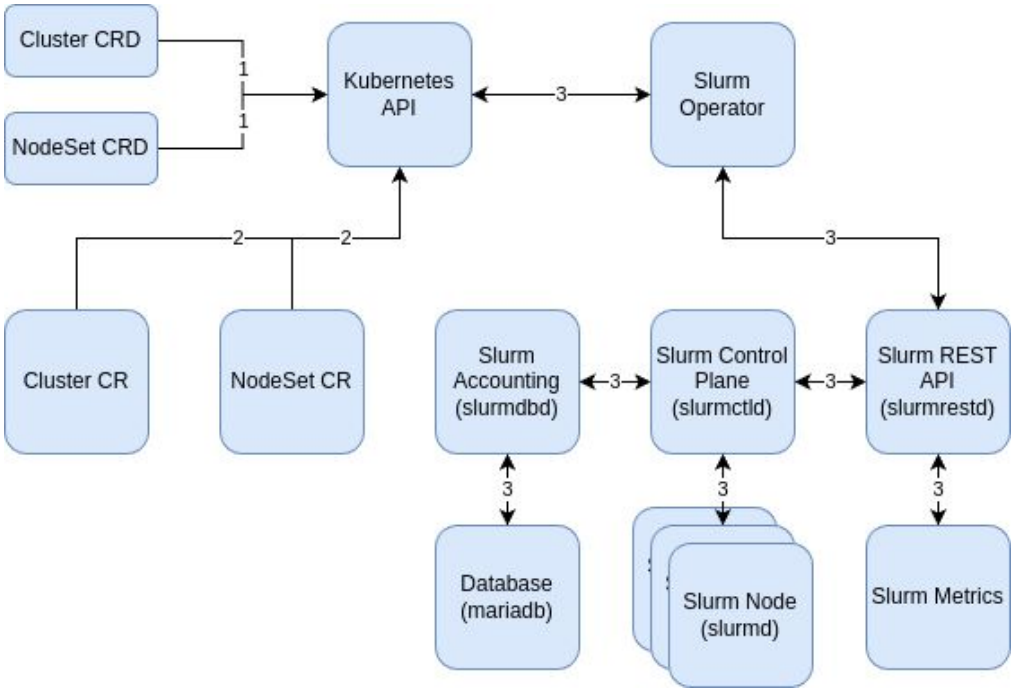
Requirements

- Can run Slurm and Kubernetes workloads on pools of nodes
- Reconcile Kubernetes and Slurm as the source of truth
  - Propagate Slurm node state bidirectionally
- Support dynamic scale-in and scale-out of Slurm nodes
- Support most Slurm Scheduling features

Restrictions

- Configure Kubernetes with static CPU management policy, which only allows for pinned cores, but not positioning or affinity
  - Properly constrain hwloc view of the node
- Disable cgroups within Slurm
  - Kubernetes does not natively allow delegation of cgroup sub-tree to pod
  - Slurmd cannot constrain slurmstepd via cgroups
- Should configure Slurm partitions with OverSubscribe=Exclusive
  - The slurmd (pod) can get Out of Memory (OOM) and killed because of user jobs!
- Pod-to-pod connections will still be through the regular Container Network Interface (CNI)

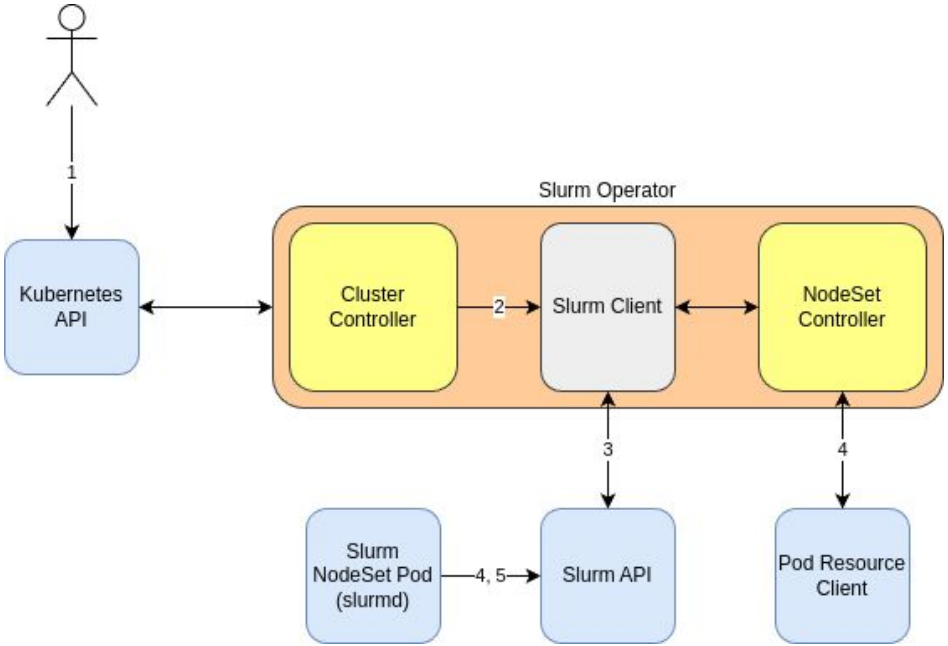
Big Picture



1. Install Slinky Custom Resource Definitions (CRDs)
2. Add/Delete/Update Slinky Custom Resource (CR)
3. Network Communication

Slurm Operator

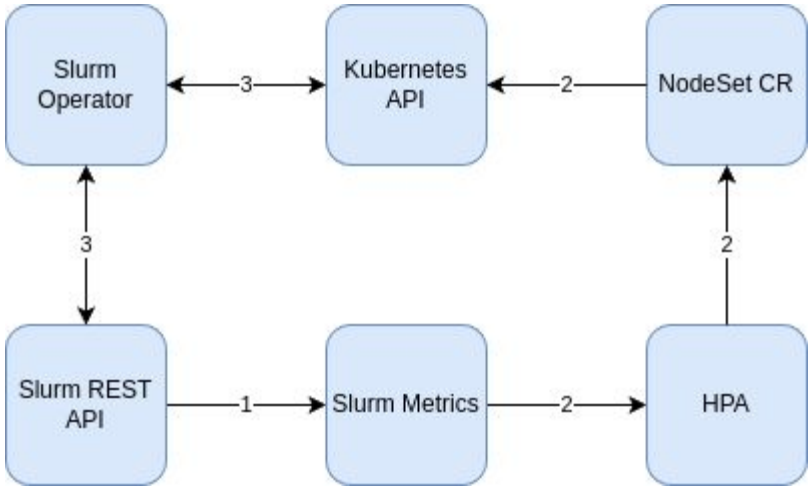
1. User installs Slinky CRs
2. Cluster Controller creates Slurm Client from Cluster CR
3. Slurm Client starts informer to poll Slurm resources
4. NodeSet Controller creates NodeSet Pods from NodeSet CR
   a. The `slurmd` registers to `slurmctld` on startup
5. NodeSet Controller terminates NodeSet Pod after fully draining Slurm node
   a. NodeSet Pod deletes itself from Slurm on preStop



Slurm Cluster Scaling

Auto-Scale NodeSet

1. Metrics are gathered and exported.
2. HPA scales CR replicas based on read metrics and defined policy.
3. Slurm-Operator reconciles CR changes, scaling in or out NodeSet Pods.



Demo Screenshots

```
Every 1.0s:     kubectl exec -n slurm statefulset/slurm-controller -- squeue; echo;     kubectl...  bluemachine: Mon Jul 29 19:19:24 2024

             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               221    purple     wrap    slurm PD       0:00      2 (Resources)
               224    purple     wrap    slurm PD       0:00      2 (Resources)
               226    purple     wrap    slurm PD       0:00      2 (Resources)
               227    purple     wrap    slurm PD       0:00      2 (Resources)
               229    purple     wrap    slurm PD       0:00      2 (Resources)
               231    purple     wrap    slurm PD       0:00      2 (Resources)
               232    purple     wrap    slurm PD       0:00      2 (Resources)
               234    purple     wrap    slurm PD       0:00      2 (Resources)
               235    purple     wrap    slurm PD       0:00      1 (Resources)
               236    purple     wrap    slurm PD       0:00      2 (Resources)
               237    purple     wrap    slurm PD       0:00      2 (Resources)
               238    purple     wrap    slurm PD       0:00      1 (Resources)
               216    purple     wrap    slurm  R       0:38      2 kind-worker,kind-worker2

PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
purple*      up   infinite      2  alloc kind-worker,kind-worker2

NAME                           READY   STATUS    RESTARTS        AGE   IP            NODE           NOMINATED NODE   READINESS GATES
slurm-compute-purple-55gch     1/1     Running   0               4d    10.244.2.11   kind-worker2   <none>           <none>
slurm-compute-purple-xgdnb     1/1     Running   5 (3d23h ago)   4d    10.244.1.9    kind-worker    <none>           <none>
slurm-controller-0             2/2     Running   0               4d    10.244.2.12   kind-worker2   <none>           <none>
slurm-metrics-79c86f5978-s5wdv 1/1     Running   0               4d    10.244.2.9    kind-worker2   <none>           <none>
slurm-restapi-79f44bff7d-9pmqr 1/1     Running   0               4d    10.244.1.7    kind-worker    <none>           <none>
```


Nodes
3
2
1
0
18:59:00
19:00:00
19:01:00
19:02:00
19:03:00
19:04:00
19:05:00
19:06:00
19:07:00
19:08:00
19:09:00
19:10:00
19:11:00
19:12:00
19:13:00
Total
Allocated
Idle
Mixed
Partition CPU Allocation
100%
Partition Nodes
4
3
2
1
0
19:00
19:05
19:10
purple
Jobs
50
40
30
20
10
0
18:59:00
19:00:00
19:01:00
19:02:00
19:03:00
19:04:00
19:05:00
19:06:00
19:07:00
19:08:00
19:09:00
19:10:00
19:11:00
19:12:00
19:13:00
Name
Min
Max
Mean
Last
Pending (purple) 3 52 29 10
Running (purple) 0 12 3 1
Total (purple) 11 53 33 11

Future Work

Future Work

- Slurm scheduler component
- Slurm finer-grained management of kubelet resource allocations (e.g. CPUs, GPUs, Core pinning)
  - Current Kubernetes cannot mix pinned and unpinned cores, let alone more complex versions of core assignment
  - Increase pluggable infrastructure of Kubernetes - current CPU and memory manager leaves much to be desired
- Network Topology Aware Scheduling in Slurm
  - Using NFD combined with Slurm internals
- Add Slurm scheduling extension to handle resource scheduling for the cluster
  - Map current scheduling concepts not in Slurm, e.g. affinity/anti-affinity

Questions?


SCHEDMD
The Slurm Company

Extended Reading

Use Cases - Immediate

- Ephemeral Slurm Clusters in the Cloud
  - Consistent user experience regardless of cloud vendor
  - Easy to plug in underlying infrastructure and just work
- Running traditional HPC workloads without needing to translate into Kubernetes pods
  - Currently, many workloads in this space, including: weather; genomics; scientific computing
  - Fine grained resource allocation and management
  - Efficient execution of multi-node workloads
    - E.g., AI/ML Training

Initial Slinky demo demonstrates these use cases by running an AI Benchmark on an ephemeral Slurm cluster

Use Cases - Immediate

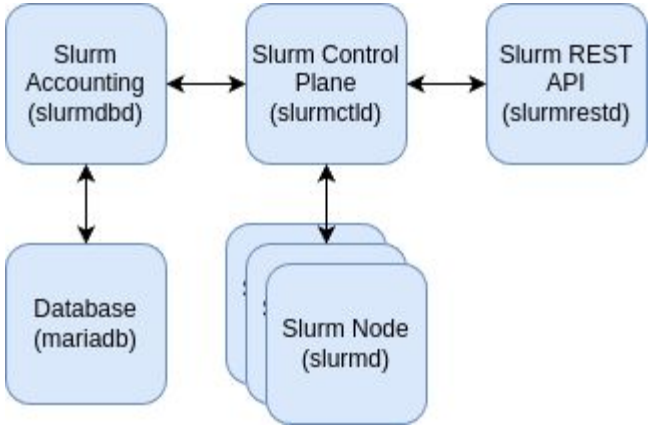
- For a hybrid compute environment, coordinate workloads running in Kubernetes and Slurm to allow for efficient sharing of resources
  - Intended approach is to provide a Kubernetes scheduling plugin that defers scheduling decisions to Slurm, allowing Slurm to have a complete view of both K8s and Slurm workloads

Use Cases - Future

- Schedule AI/ML Training, Single and Multi-node Inference in Kubernetes Clusters with minimal translation
  - Longer-term, support training operations in a Cloud-Native environment
  - Key obstacles:
    - fine-grained native resource allocation and management
    - fine-grained accelerator allocation and management
    - DRA headed in this direction
- Optimal resource use
  - Bin packing - maximize utilization of node resources
  - CPU Affinity management - avoid conflicts between pods

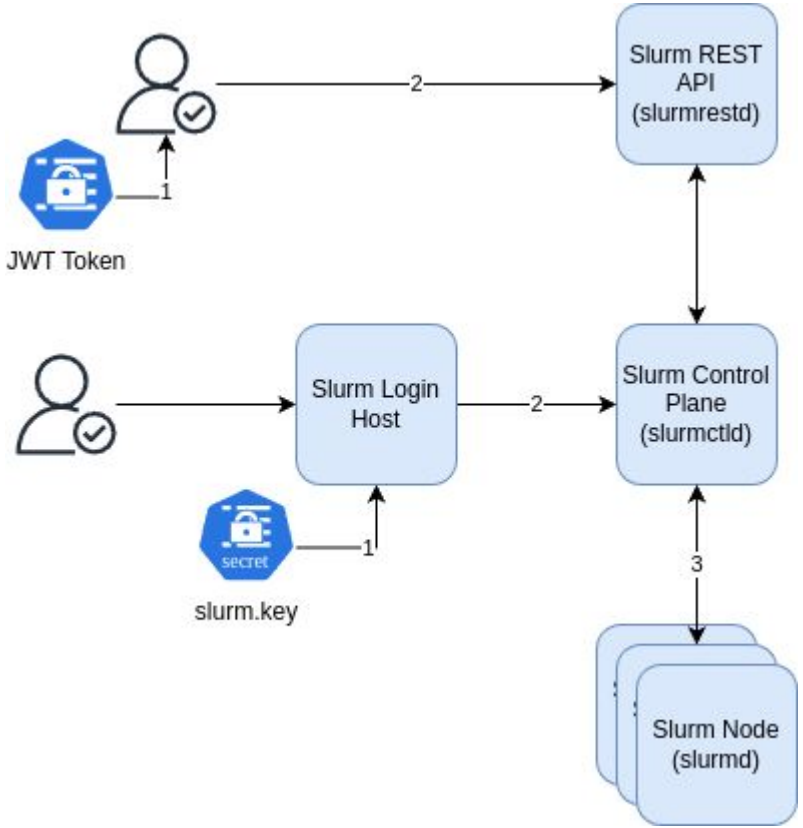
Slurm Daemons

- Slurmctld
  - Slurm Control-Plane
  - Slurm API
    - Slurm Daemon
    - Client Commands
- Slurmd
  - Slurm Compute Node Agent
- Slurmstepd
  - Slurm Job Agent
- Slurmrestd
  - Slurm REST API
- Slurmdbd
  - Slurm Database Agent
- Sackd
  - Slurm Auth/Cred Agent

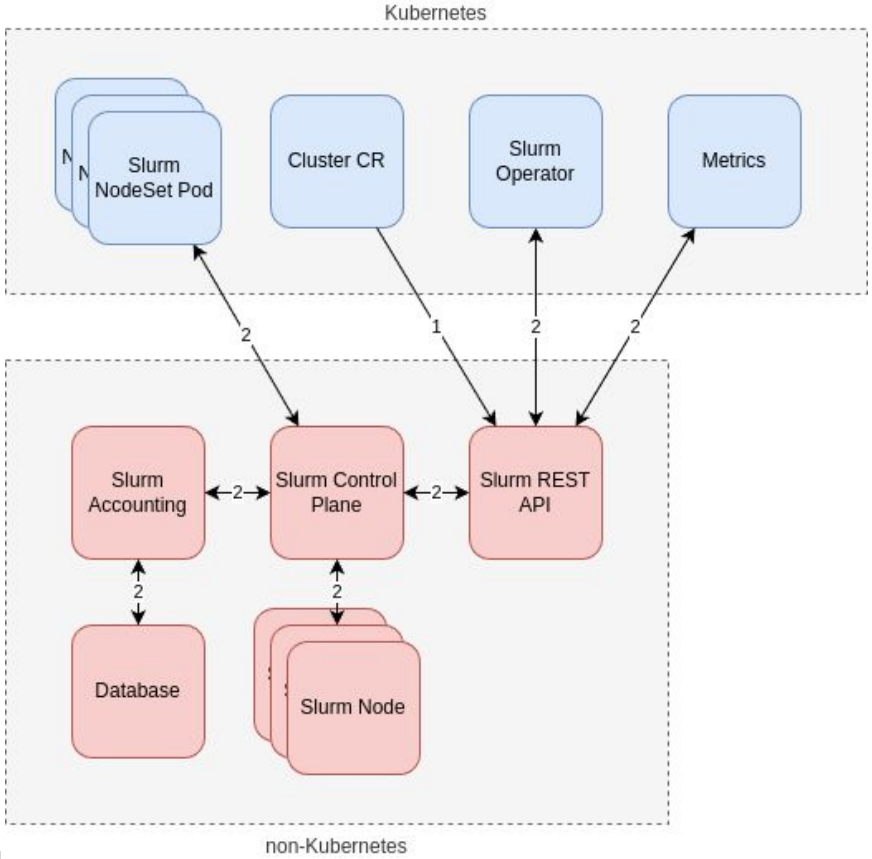


Jobs

1. User can be authenticated with Slurm
2. User submits a Slurm job.
3. Job runs until completion.



Slurm: Kubernetes + non-Kubernetes



1. References a resource
2. Network Communication

- Slurm components (e.g. slurmctld, slurmd, slurmrestd, slurmdbd) can reside anywhere
  - Kubernetes
  - Bare-metal
  - Virtual Machine
- Communication is key!

Slurm Helm Chart