Orchestrate Next-Generation AI Workloads With Open-Source Slurm

Tim Wickberg, Director, Slurm and Slinky
Danny Auble, Senior Director, Slurm and Slinky

GTC 2026 | March 19, 2026

Agenda

SchedMD is now a part of NVIDIA

Slurm for AI

Slinky

Slinky - Slurm Operator

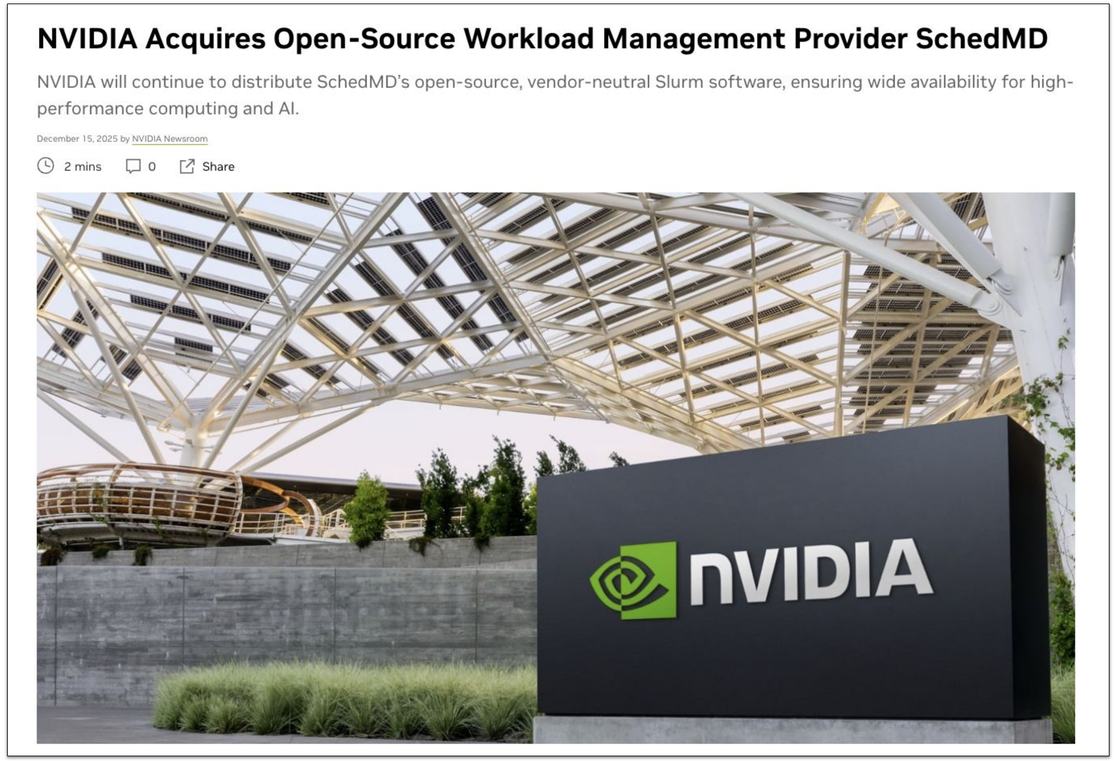
Slinky - Slurm Bridge

What’s Next

SchedMD is now a part of NVIDIA

SchedMD is now part of NVIDIA

Development and new contributions remain open-source, support and services now available from NVIDIA



1. Slurm and Slinky will remain open-source, including all new contributions

2. Direct-to-Engineering access, training, and other services are now available from NVIDIA Enterprise Support

3. Slurm engineers now have early access to NVIDIA's accelerated computing for faster development

Commitment to Open-Source Slurm

Slurm is not an acronym

Slurm releases are now directly available on GitHub

- As well as their usual location on SchedMD's website
- https://github.com/SchedMD/slurm

Pull Requests are now enabled on GitHub for Slurm

- Simpler process than through patch submission against SchedMD's Bugzilla

Issue boards are open on GitHub for Slurm

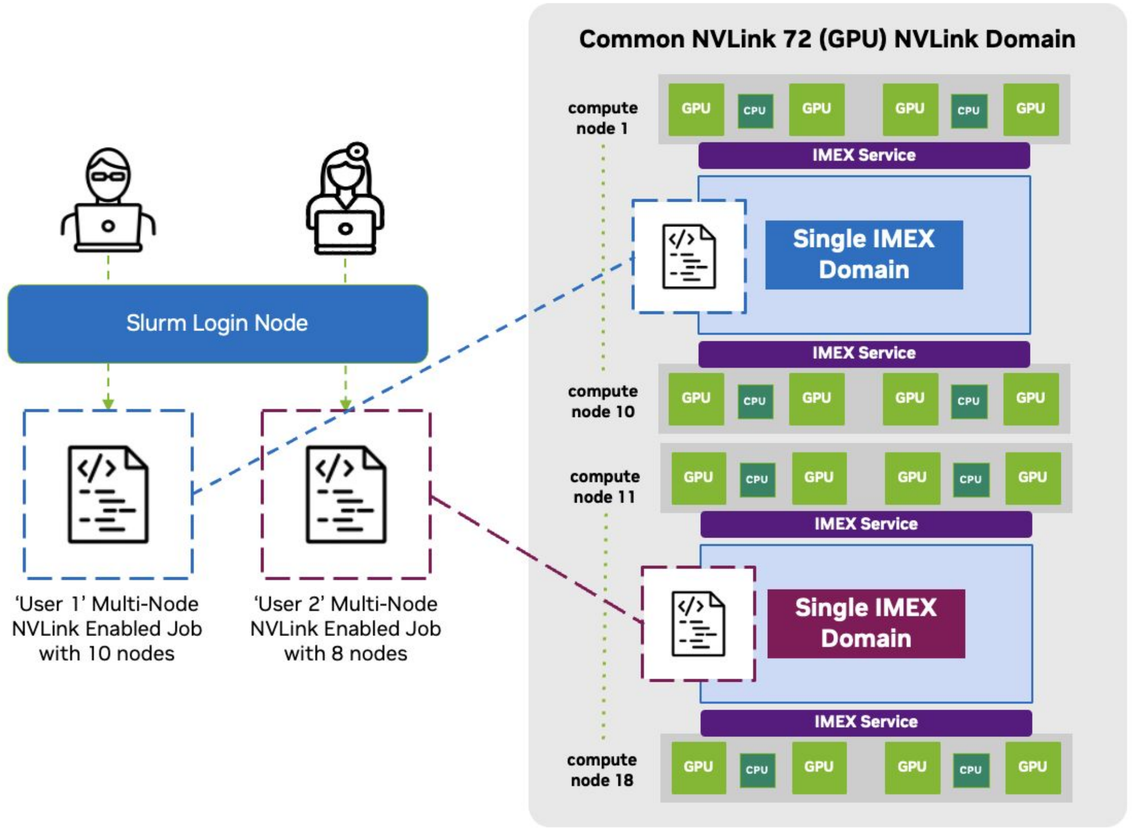
- More transparent management for public issues

Slurm for AI

Slurm's Advanced Topology Planning

With the “topology/block” plugin

- How do you plan for multi-node workloads on multi-layered interconnects?
  - … while they scale to thousands of nodes across hundreds of racks?
- NVL72 systems are built with two network tiers
  - NVLink for very-high-speed interconnections within the rack
  - NDR Infiniband for traffic crossing between racks
- Workload layout matters tremendously
  - Schedulers must plan accordingly



Slurm's Advanced Topology Planning

With the “topology/block” plugin

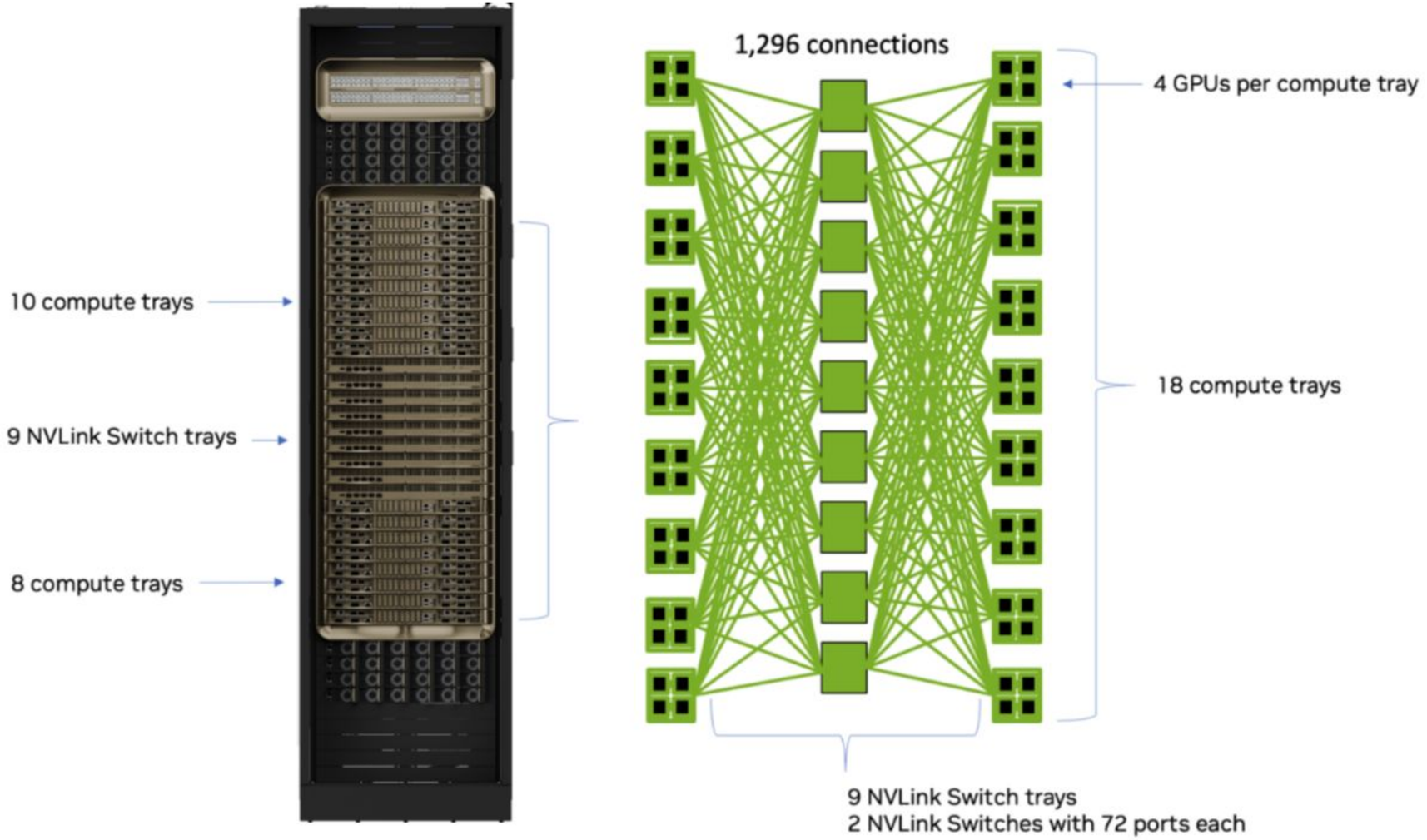
The “topology/block” plugin is designed to optimize workload placement on multi-tiered interconnects

- Designed for NVL72 and related interconnect designs

NVIDIA GB300 NVL72

Slurm's Advanced Topology Planning

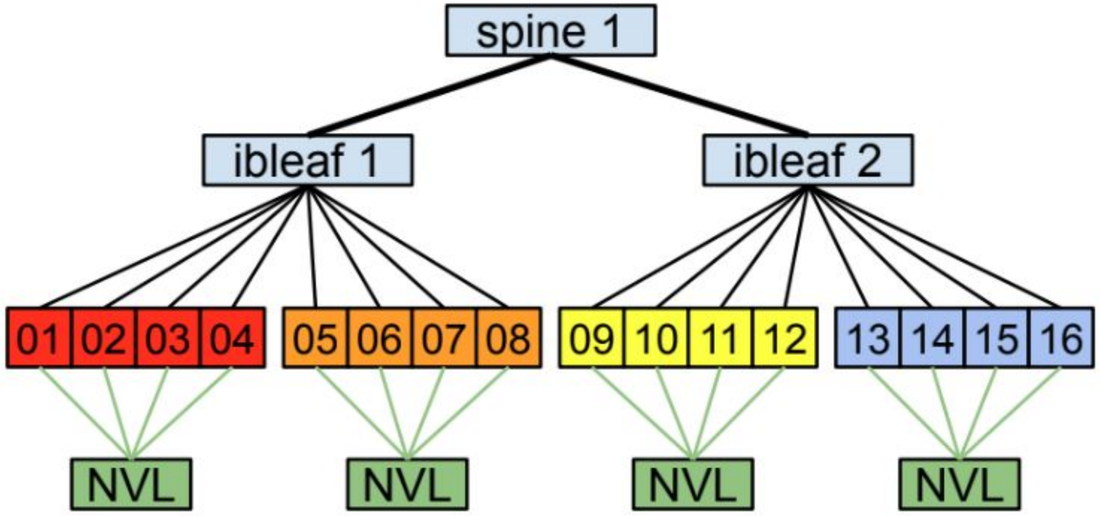
NVIDIA Vera Rubin NVL72 - Single rack layout



Slurm's Advanced Topology Planning

Example topology.conf configuration

```
##################################################################
# Slurm's network topology configuration file for use with
# the topology/block plugin
##################################################################
BlockName=block1 Nodes=node[01-04]
BlockName=block2 Nodes=node[05-08]
BlockName=block3 Nodes=node[09-12]
BlockName=block4 Nodes=node[13-16]
BlockSizes=4,8
```



Slurm's Advanced Topology Planning

With the “topology/block” plugin

Development of topology/block operation continues with new options:

- **--segment**
  - Instead of the configured BlockSize, define the grouping of nodes that’s best for the job
  - Works with --nodes to explicitly define the layout
  - Job receives (--nodes/--segment) groups of (--segment) nodes
- **--spread-segments**
  - Force each allocated segment to separate blocks
- **--consolidate-segments**
  - Ensure the densest possible packing within higher-order block sizes
- **--exclusive=block**
  - Prevent other workloads from occupying the same block
  - Ensure full cross-block bandwidth is reserved, although with lower node utilization

Slurm's Advanced Topology Planning

With the “topology/block” plugin

- **Support added for multiple topologies within a single Slurm cluster**
  - New configuration format - `topology.yaml` - instead of `topology.conf`
  - Documentation – https://slurm.schedmd.com/topology.yaml.html
  - Allow for separately defined topologies on each Slurm Partition

- **Upcoming Slurm 26.11 release includes new plugins for other network designs**
  - `topology/ring` – single-dimensional torus
  - `topology/torus3d` - three-dimensional torus

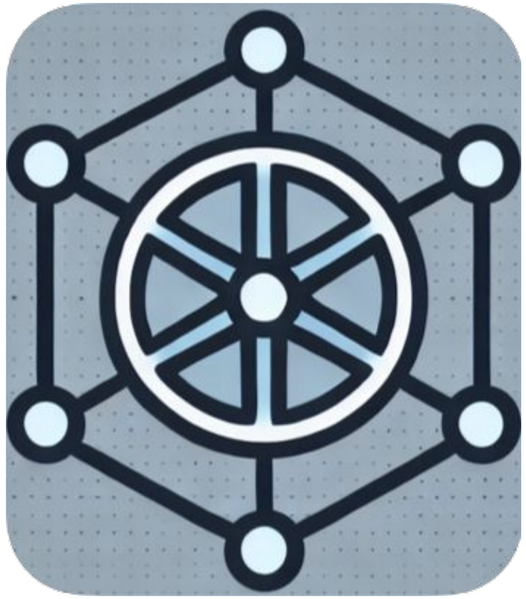
NVIDIA Topograph

Easy generation and management for `topology.conf` or `topology.yaml`

Automatic discovery of interconnect topology

- Supported on:
  - Most cloud providers
  - Bare-metal Infiniband
- Output available in Slurm's topology.conf / topology.yaml formats (Slurm and Slinky)
- Also supports native Kubernetes environments through node labels

https://github.com/NVIDIA/topograph



Expedited Requeue

Hero job support in the face of node instability

- Compute nodes – cpus, gpus, and memory – are not 100% reliable
- Training jobs – a.k.a. hero jobs – are likely to face disruption during their execution
  - Slurm previously required a 120-second delay between requeued job runs
  - And provided no special priority mechanism to ensure rapid restart

Expedited Requeue – new in Slurm 25.11 – is designed to address these issues

Expedited Requeue – Implementation Details

Hero job support in the face of node instability

- Must be explicitly enabled on the system:
  - `SlurmctldParameters=enable_expedited_requeue`

- Jobs must opt-in to this behavior with the `--requeue=expedite` option
  - Systems should consider limiting access to this option through `job_submit.lua`
  - And implementing alongside Preemption mechanisms to ensure replacement nodes are made available immediately

Expedited Requeue – Implementation Details

Hero job support in the face of node instability

- Expedited Requeue is then triggered in two situations:
  - Node failure detected by Slurm (leading to job cancellation)
  - Job failure – non-zero return code from the batch script – alongside the failure of one-or-more `Epilog` scripts

- When triggered, the job moves to a special `EXPEDITING` state
  - Absolute highest priority job in the system
  - Above-and-beyond the normal `Priority` level
  - Job is eligible to restart as soon as sufficient nodes are available

Hierarchical Resources

Planning for resources adjacent to traditional compute

- Hierarchical resources provide scheduling for resources adjacent to compute, but that aren't exclusive to specific compute nodes
- Slurm has traditionally supported planning for "licenses"
  - But assumes every license is accessible from every node

Three modes of hierarchical resource operation, each with their own topology implications:

- **Mode 1** – Resource must be available in any adjacent layer to the compute node the job runs on
  - One mode of operation for interconnect-offloaded network resources
- **Mode 2** – Equivalent resources must be available at all layers
  - Second mode of operation for other network resources
- **Mode 3** – Provides for resource aggregation as you move up through the hierarchy
  - Can model power capping limits within the datacenter, from the PDU to rack, aisle, and DC level

Observability

How to tell what's running on your systems, and correlate with other systems

- Expanded Slurm's Kafka integration to provide job details at start as well as completion
- Internalized support for exporting OpenMetrics (Prometheus) metrics directly from slurmctld
- Initial support landed in Slurm 25.11
  - Will expand available metrics in future release
- New identifier introduced for better traceability – SLUID
  - SLUID is an acronym – "Slurm Lexicographically-sortable Unique ID"
  - Unique to each run of a job
  - Never reused
    - JobId values wrap after 67,043,328
    - And repeat for requeued jobs – SLUID will change on requeue

Example SLUID: s8FXJCCS3F9Z00

Slinky

Slinky

Slinky is not an acronym

Because the best we could come up with was "**Sl**urm **in K**ubernetes, **Y**es"

- Slinky is the toolkit of components to enable Slurm operation in Kubernetes environments
  - Started with the Slurm Operator to manage Slurm cluster deployments on Kubernetes clusters
  - Expanded to the Slurm Bridge to plug Slurm directly into Kubernete's scheduling API
  - Includes additional tooling
    - Slurm Client for golang access to Slurm's REST API
    - As well as various container images

Slinky References

- Documentation:
  - https://slinky.schedmd.com/
- Repositories:
  - https://github.com/SlinkyProject

Slinky - Slurm Operator

Slurm Operator

Reference architectures for deploying Slurm in Kubernetes

- Deploy and manage Slurm clusters within a Kubernetes environment
  - Each Slurm compute node maps to a Kubernetes pod running the slurmd process
- Support autoscaling based on cluster utilization metrics
- Runs Slurm jobs natively within Slurm’s own resource management layer
  - Allows for fast deployment of multi-node workloads
- Users interact with Slurm through traditional CLI tools
  - Through one or more "login node" pods they can SSH into
  - Kubernetes is not involved in scheduling or managing compute jobs
- Slurm runs Slurm workloads directly
  - Allows for fine-grained resource limits
  - Backfill scheduling
  - Advanced topology planning

Slurm Operator - Updates for v1.1.0

- New daemonset scaling mode
  - Consistent mapping between Kubernetes node name and Slurm name
- Support for pod topology
  - Node annotation injected into the slurmd pod to trigger Slurm's automatic topology mode
  - `topology.slinky.slurm.net/spec: topo-switch:s0,topo-block:b0`
- Added PodDisruptionBudget support for slurm-operator and slurm-operator-webhook pods

Slinky - Slurm Bridge

Slurm Bridge

- Slurm Bridge enables Slurm as a first-class Kubernetes scheduler
  - Enable Slurm's scheduling capabilities – efficient multi-node planning, backfill, fair share, … – in Kubernetes environments
- For compute nodes, the Slurm Bridge assumes responsibility for workload planning
  - Initial focus has been on multi-node workloads
    - Work continues to provide more granular mechanisms for sub-node workloads as well

Slurm Bridge

Technical overview

- Slurm as a Kubernetes scheduler using the Scheduling Framework
  - Uses PreEnqueue, PreFilter, Filter, and PostFilter for placement decision
  - Uses PreBind to generate DRA ResourceClaims
- Slurm Bridge translates pod resource requirements into a Slurm “external job”
  - External jobs – compared to traditional batch jobs – are only a request for Slurm to allocate resources
    - No batch script, no job environment, no processes launched on the compute node
  - Scheduled by Slurm – launched by kubelet
- Slurm schedules to nodes running slurmd, or to “external nodes” without slurmd
  - Allows for “converged” system designs

Bridge Demo

Slurm Bridge - Updates for v1.1.0

v1.1, a.k.a. DRAapalooza

- Support for automatically creating Slurm “external” nodes from labelled Kubernetes nodes
  - `scheduler.slinky.slurm.net/external-node: true`
  - `scheduler.slinky.slurm.net/external-node-partitions: slurm-bridge`
- Support for emitting resource claims against the DRA GPU driver
- Support for emitting resource claims against the DRA CPU driver
- Beta support for NVLink Compute Domain setup
- New annotation to allow for sub-node resource allocation for a Pod:
  - `slurmjob.slinky.slurm.net/exclusive: false`

What’s Next

What’s Next?

Roadmap

Upcoming releases

Slurm

- Slurm 26.05 in May
- Slurm 26.11 in November

Slinky

- Slinky v1.1.0rc1 available now
  - New daemonset scaling mode for Slurm Operator
  - DRA support in Slurm Bridge
- Slinky v1.2.0 this summer

Thank you


NVIDIA