Slurm Community Birds-of-a-Feather

Danny Auble
Tim Wickberg

SC24

Welcome

Welcome

- The BoF is being broadcast on the SC24 Digital Experience
    - Please use the microphones to ask questions so everyone in the room, and everyone watching remotely, can hear
    - Danny will be monitoring online questions
        - Note that there's a broadcast delay online
            - If it's pertinent to a specific slide, please mention the slide number
    - Feel free to ask questions throughout
        - Although we may defer, or ask to discuss offline

Survey

Community Survey

https://schedmd.com/survey

Slurm 24.05, 24.11, and Beyond

Tim Wickberg
Chief Technology Officer

Development Cycle

Release Cycle

- Major releases are now made every ~~nine~~ six months
- Version is the two digit year, two digit month:
    - 24.05 - May 2024
    - 24.11 - November 2024
    - 25.05 - May 2025
- Major releases are supported for 18 months
    - Currently: 24.11, 24.05, and 23.11
- Maintenance releases are made roughly monthly
    - Usually only for the most recent major release
        - One main exception: security releases are made for all supported major releases

Revised Release Cycle

- Direct upgrades from 3 prior major releases will be supported starting with 24.11
  - Previously upgrades were only supported from the 2 prior major releases

Development Process

- Most larger work is handled through sponsored projects
    - SchedMD support only covers maintenance
- Some projects - those of wider community interest - may be handled internally on a best-effort basis

Slurm 24.05 - May 2024

topology/block

- Additional optimization for "block" based topologies
- "Exclusive" block access
  - Allow a job to indicate that it should be the only occupant of the associated blocks
  - Avoid contention between performance-sensitive workloads
- Allow "Segment" size specification
  - E.g., if a 40-node job naturally decomposes into 4x 10-node sections, allow a specification of "--segment 10" to alter the topology allocation strategy to avoid straddling internal block boundaries.

Node Features

- Allow node features to be flagged as not requiring a node reboot to change
  - E.g., allow for GPU mode changes without taking the entire node offline

MaxTRESRunMinsPerUser / PerAccount

- New QOS limits reduce configuration complexity
  - Automatically group and limit utilization within the QOS by User or Account

Adjustments to the "Coordinators" status

- Adjust the Coordinator to only permit accounting changes that fit within constraints applied to the account
  - E.g., do not allow the coordinator to set MaxJobs=10000 on an individual user if the account has a lower limit of MaxJobs=10 already in effect
  - The high-level view is that the coordinator is permitted to tweak settings within the scope of the existing account, but should not be able to override the size/shape of the restrictions at the account level
- Options to disable the coordinator status in the Slurm Controller or the SlurmDBD
  - E.g., if a site wants coordinators to handle job workflow changes (hold/suspend/requeue) but not making accounting changes, they can limit the permission to the Slurm Controller-only

Prolog/Epilog

- New PrologFlags=RunInJob option to run the Prolog/Epilog within cgroups corresponding to the job itself
    - Implies PrologFlags=contain
        - Scripts will be run/managed by the "extern" slurmstepd process, instead of directly invoked by slurmd
    - Avoid, e.g., having the script accidentally make GPU mode changes to cards that aren't allocated to the job

ReservedCoresPerGPU

- Dedicate cores on node to GPU work
    - Cores only assigned if the corresponding GPU has been allocated to the job
    - Allows for CPU-based workloads to better overlap into GPU nodes, without threatening to starve the GPU workloads and risk idling the (expensive) GPUs
- Currently, the same use case can be partially covered by using the MaxCPUsPerNode setting on a Partition
    - But that doesn't easily scale with a heterogeneous mix of nodes, and requires splitting work across multiple partitions

Job State Monitoring API

- New API call / squeue option / REST mode that only returns the job state
- Designed for external workflow tools, and avoids performance issues when returning the entire job state
  - Will use state tracking that is not tied to the "job lock" in slurmctld, which also greatly improves performance in the face of heavy RPC load

auth/slurm Improvements

- Allow for non-disruptive auth/slurm key rotation
- Add hash/sha3 plugin as an alternative to hash/k12 for network traffic validation

Step Management Performance

- Decouple job step management from the Slurm Controller
- Manage per-job on the "batch" host assigned to the compute job
- Allows for massively improved job step launch scalability
  - And significantly reduces load on the Slurm Controller, allowing it to focus on job scheduling

Slurm 24.11 - November 2024

New gpu/nvidia plugin

- New plugin that does not use NVIDIA libraries
    - Unlike gpu/nvml plugin, which has suffered from various CUDA packaging issues
        - And requires CUDA to be installed at build time
- Builds everywhere
- Uses standard kernel interfaces for GPU enumeration
    - /proc/driver/nvidia/gpus/%s/information
    - /sys/bus/pci/drivers/nvidia/%s/local_cpulist
- Adds Autodetect=nvidia for gres.conf
- Significant limitations
    - Can't detect MIGs
    - Can't detect NVlink topology
    - Can't provide energy statistics
- Hoping to encourage NVIDIA to provide more interfaces under sysfs

GPU detection in 'slurmd -C'

```
$ slurmd -C
NodeName=nuclear CPUs=12 Boards=1 SocketsPerBoard=1 CoresPerSocket=6
ThreadsPerCore=2 RealMemory=31840 Gres=gpu:nvidia_geforce_gtx_1650_ti:1
Found gpu:nvidia_geforce_gtx_1650_ti:1 with Autodetect=nvml (Substring
of gpu name may be used instead)
UpTime=3-18:23:04
```

Job submission against multiple QOSes

- Similar to submissions against multiple Partitions, --qos now supports a comma-separated list of QOSes to test against
  - Expectation is that these should have different prioritization
  - But will schedule in whichever is available ASAP
- Additional lookup calls now available in job_submit.lua to help sites automatically set or filter these submissions
  - slurm.get_qos_priority() returns the priority for a given QOS name
  - job_desc["assoc_qos"] field shows all QOSes the user has access to

QOS-based accounting reports

- New AccountUtilizationByQOS option in sreport

Topology + Backfill Work

- New experimental option - "bf_topopt_enable"
- Allows an "oracle" function to evaluate the fragmentation level of a block topology network
  - And decide whether that job launch should be delayed to a future backfill interval in the interest of reducing system fragmentation

"scontrol listjobs" and "scontrol liststeps"

- Complement existing "scontrol listpids" command
  - Works directly on the local node
  - Output in --json/--yaml available for all three

sbcast --no-allocation

- Use sbcast outside of a job allocation
- Only available to root or SlurmUser
- Requires an explicit nodelist (--nodelist)

Hostlist Functions

- Hostlist Functions extend some of the concepts that the NodeList introduced previously
- In most locations, allows for use of:
  - "feature{foo}", which substitutes all nodes with Feature=foo
  - "switch{switch1}" which substitutes all nodes attached directly to switch1
    - "block{block1}" for topology/block
  - "switchwith{node0001}" which substitutes all nodes on the leaf switch that node0001 is connected to
    - "blockwith{node0001}" for topology/block

Performance work

- Improvements to:
    - Scheduling interfaces, cutting down redundant placement checks
    - Bitstring handling
        - bit_test() replaced with a macro within the bitstring code
        - Internal cache for node_record_count length bitstrings to avoid constant malloc()/free() churn
    - List construction
        - Significantly reduce malloc()/free() pressure
        - Internalize list node structures within larger blocks to take advantage of cpu caches
    - Database handling
        - Generate db_index within slurmctld, rather than slurmdbd
            - Significant improvement in performance for workloads relying on frequent requeues

New TaskPluginParam=OOMKillStep option

- If any tasks within the step are killed, kill the entire step

DataParserParameters

- New DataParserParameters option controls --json / --yaml output formats
    - Allows sites to specify default data_parser plugins, and default options
    - E.g., default to the v41 format, with fast parsing enabled:
      DataParserParameters=v0.0.41+fast

Added "sacctmgr ping"

- Pings the slurmdbd
- Complements long-standing "scontrol ping" command

Ephemeral cluster startup quality-of-life improvements

- On first start, slurmctld will retry the connection to slurmdbd indefinitely
- On first start, slurmd and sackd will retry the connection to fetch "configless" config files from slurmctld indefinitely
- Helpful when deploying ephemeral Slurm clusters
  - Avoids needing to explicitly sequence the components

HPE Slingshot

- Removed "Instant On" support
- Added "Collectives" support
    - Requires support in the fabric manager, due out in an HPE update soon

conmgr

- Not intended to be directly visible, but considerable work in 24.11 went into refactoring slurmctld RPC handling mechanisms into a centralized thread-pool model
  - As well as unifying the signal handling for each daemon
- Replaces prior ephemeral thread-per-connection model
- Introduces a number of new tunable settings, see SlurmctldParameters and SlurmdParameters for further details

Slurm 25.05 - May 2025

Network Traffic Encryption

- Encrypt all Slurm traffic
  - Optional
- New "certmgr" plugin interface to help with certificate management on the compute nodes

Further conmgr work

- Send all RPC responses asynchronously from slurmctld
- Directly manage TLS connection state

"minibatch" operation

- Extend the new stepmgr code to allow steps to queue
  - Stepmgr would dispatch the command or script to launch directly
    - Rather than the srun command

… and Beyond

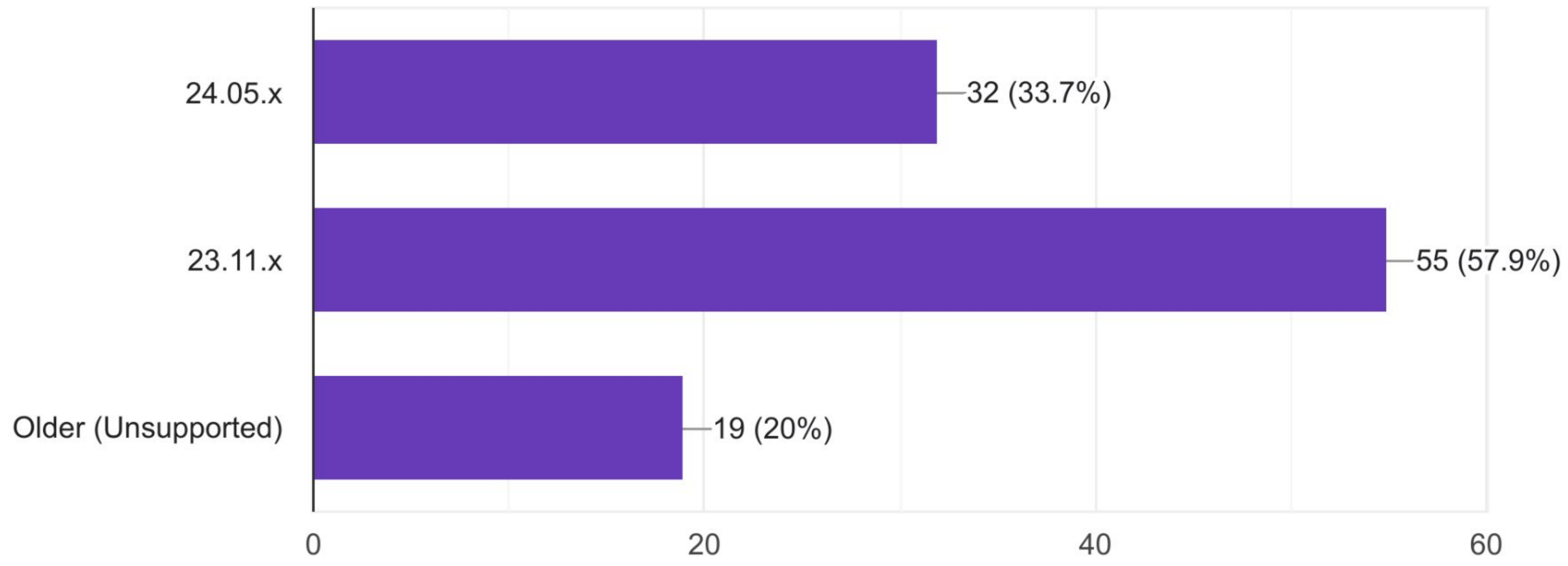
SLUID - Slurm Lexicographically-sortable Unique ID

- 64-bit identifier for each job
  - Replaces db_index
    - Available on systems without SlurmDBD
  - Changes on each requeue
    - JobID does not - this is why the tuple of (JobID, Start Time) is what needs to be tracked externally
- Example: s8FXJCCS3F9Z00
  - Leads "s", then 13 base-32 characters (0-9, A-Z excluding ILOU)
  - "sacct --format=sluid" is the one place you can see these directly in 24.11

Survey Results

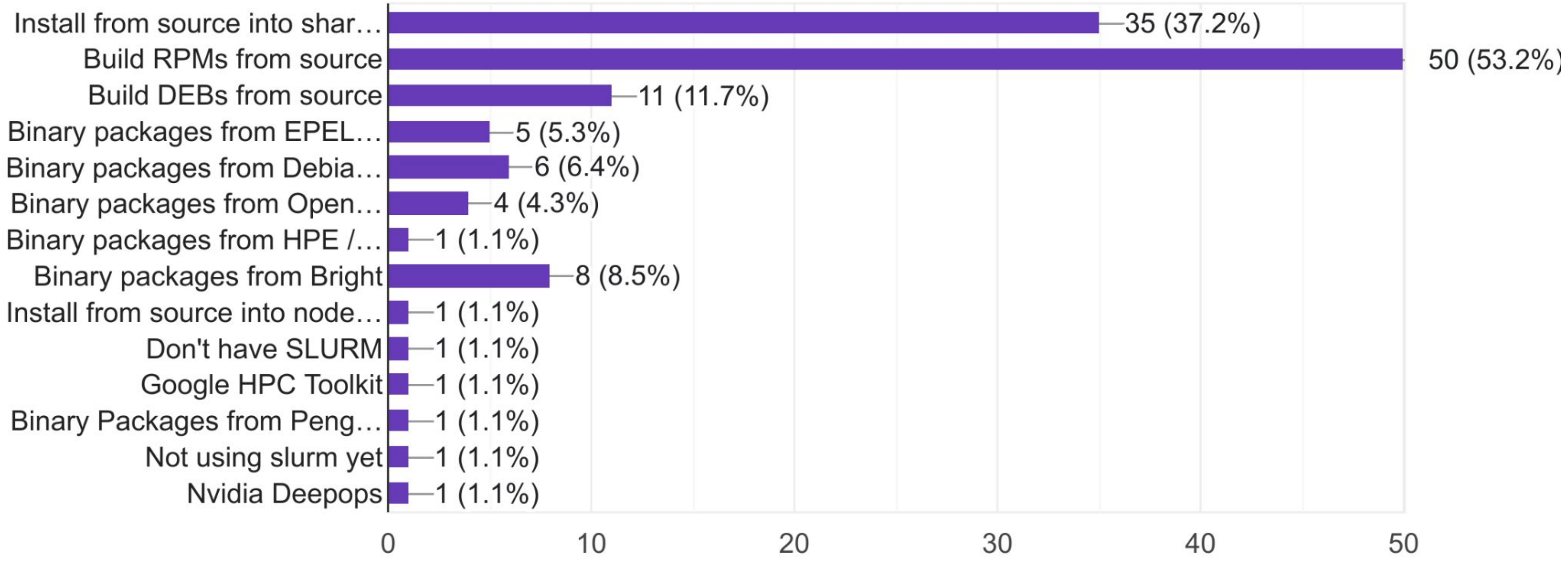
Which Slurm releases are you currently running in production?

95 responses



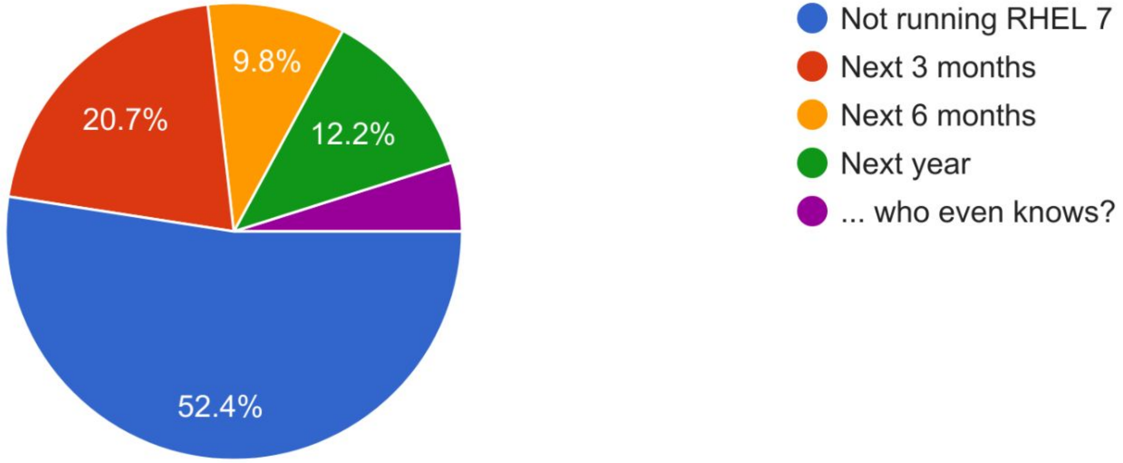
How do you manage your Slurm installation?

94 responses



If you're still running RHEL 7 (or derivatives), how long until you expect to migrate to something modern?

82 responses



Questions?

Slinky: The Missing Link Between Slurm and Kubernetes

What is Slinky?

What is Slinky?

What is Slinky?

- These slides are a subset of those presented as the keynote at CANOPIE-HPC earlier this week
- That full deck is available:
  - https://slurm.schedmd.com/SC24/Slinky-CANOPIE.pdf

What is Slinky?

- A toolkit of components to enable Slurm integration with Kubernetes
    - Open-source, Apache 2.0 licensed
    - Initial components were released on November 8th
    - SlinkyProject on GitHub
    - Direct link to overview page - slinky.ai
- Uses Slurm's REST API for all core interaction
    - Wrapped into a client Golang library

What is Slinky?

- Three main components:
    - The Slurm Operator
        - Managing Slurm running within Kubernetes
    - Assorted Tooling
        - Helm charts, Dockerfiles, Container Images, Slurm REST Client Library
    - The Slurm Bridge (Future)
        - Integration with Kubernete's scheduling API
            - Use Slurm's scheduling wherewithal to manage a converged pool of computing resources
                - Run K8s workloads through the Kubelet
                - Slurm workloads through slurmd

What is Slinky **not?**

What is Slinky **not**?

- Slinky is not a direct part of Slurm
  - Although Slinky's design has had - and will continue to have - influence on Slurm
  - Separate development team within SchedMD
  - Separate license - Apache 2.0
    - Slurm's license is "GPL v2 or later, with an OpenSSL exception"
- Slinky is not included in SchedMD's support for Slurm
- Slinky is not currently intended as an out-of-the-box solution
  - Instead intended to provide flexibility in how it is adapted into an environment
  - Assumes willingness to alter the various components as part of this adaptation
    - Changing the Dockerfiles, Helm charts, and even Golang code

Slinky Components - Today

Slinky Components

- Slurm Operator
  - Kubernetes Operator for Slurm
- Slurm Exporter
  - Prometheus collector and exporter for metrics extracted from Slurm
- Slurm Client
  - Slurm versioned REST API endpoints are multiplexed for seamless request/response
- Helm charts
  - Slurm Cluster
  - Slurm Operator
  - Slurm Exporter
- Container images
  - Slurm Daemons
  - Slurm Operator
  - Slurm Exporter

Slinky Roadmap

Future Directions

- Next major development goal is the Slurm Bridge
  - Slinky's Kubernetes Scheduler plugin
    - Schedule both Slurm jobs as well as Kubernetes jobs on the same hardware
  - Target is Spring '25, ahead of KubeCon Europe
  - Ideally made in conjunction with a new DRA CPU Core management plugin
    - Can operate without, but requires limiting nodes to running either K8s pods or Slurm jobs, not both simultaneously

DRA for Cores

- "Dynamic Resource Allocation" (DRA) is an API to request and reserve specific resources within a Kubernetes node
  - Used to manage access to GPUs on the node
  - Plugins can be added to control additional resources
    - Intent is to add Core management through this interface
      - Giving the Slurm Bridge a way to communicate core allocations for the Kubernetes jobs
        - And avoid contention between Slurm vs Kubernetes jobs sharing a compute node
    - SchedMD working with partners to get this built as an out-of-tree driver
      - Want to get this added as a central capability in a future K8s release

Slurm Bridge Design

- Translate K8s pods into "placeholder" Slurm jobs
  - Automatically translate resource requests
    - Core count, memory amount, GPUs
  - Support custom annotations for Slurm-specific settings
    - Such as the partition, account, QoS, time limit
- When the placeholder job is scheduled, inform the Kubernetes scheduler API of the node placement
  - Inject resource claims for DRA
    - For GPUs
    - And - once developed - for Cores

Questions?

SCHEDMD

The Slurm Company