


# Slurm 26.11 and Beyond

Tim Wickberg, Doug Jacobsen

SC'26

# Agenda

- Slurm Development Overview
- Slurm 26.05 Release
- Slurm 26.11 Release



# Slurm Development Overview



# Slurm Release Cycle

- Slurm major releases are made every six months
- Version is the two digit year, two digit month:
  - 26.05 - May 2026
  - 26.11 - November 2026
  - 27.05 - May 2027
- Major releases are supported for 18 months
  - Currently: 26.05, 25.11, 25.05
  - Maintenance releases made *at least* monthly for the most recent major release
  - Security issues fixed in all supported major releases
    - For *confidentiality* and *integrity* issues only
    - *Availability* issues - daemon crashes - are not always backported

NVIDIA

# Slurm 26.05 Release



# Dynamic Memory Limits

Compute Nodes Are Evolving

- Job launches with the initial user-set limit on memory
- Later, the job can release unneeded memory which can then be used by other jobs
  - Can now update a running job to *lower* memory usage
    - `scontrol update job=1234 MinMemoryNode=1024m`
  - New option to sbatch command to automatically check memory usage at after a given time, and automatically update the memory limit to the current usage plus some configurable buffer
    - `sbatch --mem-update=10@00:15:00`

# New Topology Plugins

## Clusters Are Growing

- `topology/ring` – single-dimensional torus
- `topology/torus3d` - three-dimensional torus

NVIDIA

# New slurmrestd Endpoints

Clusters No Longer Live In Isolation

- Job Requeue:
  - GET /slurm/{version}/job/{job_id}/requeue[?{flags}]
    - Enables scontrol requeue and scontrol requeuehold functionality (depending on flags)
- Slurm Configuration
  - GET /slurm/{version}/conf
  - GET /slurmdb/{version}/conf

# Asynchronous Step Management

## Clusters No Longer Live In Isolation

Building on stepmgr capability, create queue of steps within a job allocation

- `srun --async` … to queue a step
- Queue many steps and offload task management to stepmgr

NVIDIA

# Exclusive Job Updates

- Clarified `srun --exclusive` behavior when srun is used to both allocate a job and run a step at the same time
  - Added new srun option `--exclusive=allocation` to get a node-exclusive allocation without implying `--exact`
- New OverSubscribe field in `sacct` output to show which options for `--exclusive/--oversubscribe` were used in job allocation

# Expanded metrics support

- Add gpu statistics to the new built-in openmetrics (Prometheus) support
- Support optional jwt authentication for the openmetrics interfaces

# Changes to remote resources (licenses)

- Option to preserve the case for licenses (remote resources) managed through SlurmDBD
  - Parameters=PreserveCaseResource
- Allow remote resources to be requested without the “@server” suffix
- Allow a remote resource to be 100% accessed by multiple clusters, rather than requiring subdivision
  - Set `Flags=SharedPool` on the resource
  - Expected to be used alongside a cron job updating the “`LastConsumed`” value

# Slurm 26.11 Release



# SlurmDBD Upgrade Warning

- Minimum database version has been updated to MySQL 8.0.21 or MariaDB 10.2.7
  - slurmdbd logs the server version and exits if it is older
- The SlurmDBD 26.11 upgrade requires rewriting the job and step tables on first start
  - This can be slow depending on your underlying filesystem
  - Sites are encouraged to test the SlurmDBD upgrade process offline first

| Phase | MySQL 8.0, NVMe | MariaDB 11.8, NVMe | MariaDB 11.8, HDD |
|---|---:|---:|---:|
| Job table conversion (full rebuild) | 65.7 s | 50.2 s | 216.7 s |
| Step table conversion | 21.0 s | 21.4 s | 84.1 s |
| Jobs: uid to user name (1M rows) | 12.3 s | 7.1 s | 42.1 s |
| Steps: uid to user name | 11.3 s | 5.2 s | 43.4 s |
| **slurmdbd start to ready** | **110.5 s** | **84.4 s** | **389.2 s** |

NVIDIA

# HRES Plugins – Interface with Cluster-level Devices

Clusters No Longer Live In Isolation

External devices deployed alongside conventional clusters - such as quantum processors - are becoming more common. Job scheduling, health monitoring, and user access all need interfaces for Slurm to manage these devices.

The HRES (Hierarchical Resources) subsystem models cluster-adjacent devices, and is an evolution of Slurm's traditional support for licenses.

**Scope**

- Create a new plugin infrastructure for HRES permitting each HRES to have it's own optional lua-based plugin implementing:
  - controller-level job pre-run, job revoke, job epilog
  - node-level prolog/epilogs
  - task prolog/epilogs
  - controller maintenance thread to track device availability
- User interface can be specified per-plugin:

```
sbatch --resources=qpu:1[qubits=30] ...
```

- Controller pre-run can generate a payload that is distributed to node-level epilogs/prologs, useful for generating access credentials from the controller
- The maintenance callback can be used to drain an HRES and optionally terminate jobs using that HRES
- Add ability to set DisableHRES or DisableLayer to drain an HRES from the plugin

NVIDIA

# HRES Plugins – Interface with Cluster-level Devices

Clusters No Longer Live In Isolation

- Example configuration of HRES Plugins

```
- resource: qpu
  mode: MODE_1
  plugin:
    type: hres/lua
    config:
      - name: path
        value: /etc/slurm/hres_qpu.lua
    maintenance_cycle: 60
  layers:
    - layer_name: qpu1
      nodes: ["node[01-16]"]
      count: 1
```

NVIDIA Technical Blog on Quantum Computing Integration with HRES Plugins



# HRES – Dynamic Changes for Dynamic Nodes

Clusters No Longer Live In Isolation

Enable support for defining which layer a dynamically created node should be added to upon creation or update to the layer.

**Scope**

- Add `layer_name` to the HRES resources.yaml definition to refer to the layer.
- Add `parent_name` to enforce structure for Mode 3 HRES (only root has no parent_name)
- Enable dynamic add to the HRES and layer:

```
scontrol create node NodeName=node1 HRES='[{"hres_name": "power", "layer_name": "rack2"}]' ...
```

- Enable dynamic updates to the HRES:

```
scontrol update HRESname=... LayerName=... Nodes=[+|-]NodeList
```

# HRES Mode 3 Preemption

## Clusters No Longer Live In Isolation

Preemption is needed for HRES to enable complex workloads to leverage HRES effectively.

**Scope**

- For 26.11, only Mode 3 HRES support preemption

**Restrictions**

- Main scheduler only, not backfill
- Only works for cancel and requeue methods for preemption, suspend keeps HRES (job still running)
- `PreemptParameters=reclaim_licenses` does not apply to HRES

# task/resctrl Plugin – Limit/Share CPU Resources

Compute Nodes Are Evolving

Sharing a physical node between large scale GPU jobs and smaller scale CPU high throughput jobs is increasingly requested as node resources get larger. Controlling memory bandwidth or L2/L3 cache access via resctrl enables prioritized sharing.

**Scope**

- task/resctrl plugin creates a fixed set of resctrl partitions using verbatim configuration strings
  - Identifies which partition a job belongs in and adds job PIDs to that partition
- The initial set of groups are GPU-focused, specifically "gpu_one_per_socket", "gpu_two_per_socket", "cpu_only".

**Impact**

- Enable site-directed memory bandwidth policy enforcement

**Risks/Restrictions**

- Feature still in development for Linux kernel
- Every CPU vendor exposes different capabilities, thus the verbatim application of resctrl schemata
- No per-job partitions are created, not enough hardware resources to enable this

# Job Log Aggregation

## Clusters No Longer Live In Isolation

Not every cluster has a shared filesystem. Cloud vendors offer proprietary log aggregation services. Enterprise users want to centralize job analysis.

**Scope**

- Create an *io_relay* plugin infrastructure to efficiently push logs to aggregation stores for asynchronous analysis
- Create initial plugin to produce newline delimited JSON records per logline, rely on local scraper to read logs
- Site provided scraper sends logs with job/cluster metadata to enable flexible data analysis workflows

**Impact**

- Allow for Slurm cluster operation in environments lacking a centralized shared filesystem
- Enable organizations to centralize job analysis - especially valuable for large deployments with specific well-known workloads (e.g., AI training)
- Initial version is cloud vendor agnostic - any can implement their own relay service on the node

**Future Development Options**

- Specific vendor implementations (e.g. io_relay/kafka)
- Integrate with enhanced job metadata KV store to allow site-directed labeling

# Topolology – Setting Segment Size

Compute Clusters Are Growing

Making setting and changing segment size more convenient, can change without job cancelation.

**Scope**

- Allow scontrol update job segment options

```
scontrol update job=X SegmentSize=4
scontrol update job=X SpreadSegments=yes
scontrol update job=X ConsolidateSegments=no
```

- Updating SegmentSize to 0 clears the field
- Add DefSegmentSize (default) on Partitions

```
PartitionName=gpu Topology=block1 Nodes=node[001-064] DefSegmentSize=8
scontrol update PartitionName=gpu JobDefaults=DefSegmentSize=8
scontrol update job=1234 SegmentSize=0   # opt out
```

**Restrictions**

- Can only update pending jobs
- The node count request must be evenly divisible by the new requested segment size
- These changes are only supported by topology/block, topology/ring, and topology/torus

NVIDIA

# Dynamic Nodes – Task Distribution and Ordering

Compute Clusters Are Growing

Many users rely on the ordering of output nodelists to identify where a given task ran for doing post-job analysis.

**Scope**

- `SLURM_JOB_NODELIST`, `SLURM_NODELIST`, `SLURM_STEP_NODELIST` and the accounting database `NodeList` field are all ordered in the task distribution order now
  - regardless of dynamic nodes or topology
  - `SLURM_JOB_CPUS_PER_NODE` and `SLURM_TASKS_PER_NODE` are positional with respect to the nodelist
- Task distribution for block and cyclic now follows block ordering and then alphanumeric ordering within each block by default (can disable with `SelectTypeParameters=CR_NO_DIST_TOPO`)
- The batch host is now the lowest topology-ranked node within the allocation

**Restrictions**

- `--distribution=arbitrary` keeps the hostfile order
- Arbitrary distribution order jobs are killed on node failure even with `--no-kill`, cannot `scontrol update NumTasks=X`

NVIDIA

# scontrol tasknode

Compute Clusters Are Growing

Dynamic nodes and topology-optimized node selection can make it non-obvious where specific tasks are running based on the requested task distribution strategy.

**Scope**

- `scontrol tasknode` within a job shows where a specific task is running

```
$ scontrol tasknode 123.0 task=2
StepId=123.0
  TaskId=2 Node=node1235
```

- slurmstepd local to the compute node answers this query - does not generate an RPC to slurmctld

**Impact**

- Useful to find a particular task - for example if a workflow coordinator is running on task 0 of step 0, can automatically discover it scalably from all nodes without guessing or other tricks.

**Restrictions**

- If the job/step is not specified, relies on SLURM_JOB_SLUID to get the job identifier
- Heterogeneous steps - layout and task ids span all components, no array/het job id formats allowed

# Key/Value Store per Job/Node/Reservation/Partition

## Clusters No Longer Live in Isolation

JobName/Comment/Admin_Comment/System_Comment/Extra/WCKey are the various ways job metadata has been tracked, none support structured data well.  Add a new Key/Value store to enable arbitrary metadata to be tracked and updated for many key Slurm objects.

**Scope**

- Create a JSON-backed key/value store for managing user and site definable metadata that can be attached to most of the controller data structures (jobs, nodes, reservations, partitions)
- Store these data in the Slurm database
- Make objects viewable via sacct, sinfo, squeue
- Ability to create KV store upon job submission or modification (while pending):

```
sbatch --kv='{"project": "abc", "attempt": 2}' ...
scontrol update job=X KV='{"project": "def", "attempt": 2}'
scontrol update job=X KVPatch='[{"op": "add", "path": "/tags", "value":
  ["a", "b", "c"]}]'
```

**Restrictions**

- Limited to 1MB per object
- Outermost layer must be a dictionary
- KV is at most 16 layers deep

NVIDIA

# Async Steps – Step Dependencies

Compute Clusters Are Growing

Building on async steps from 26.05, step dependencies enable complex workflows to be run directly by the stepmgr within a job, enabling native horizontal scalability for Slurm.

**Scope**

- Add srun `--step-dependency=[after, afterany, afterok, afternotok]`
  - AND with ',', OR with ':' (like `--dependency`)
- Generate step ids immediately upon async submission to reference in dependency (no more TBD)

**Impact**

- Encourage adoption of alternative workflows that reduce stress on the main scheduler level

**Restrictions**

- Requires the stepmgr to be enabled (`--stepmgr` or `SlurmctldParameters=enable_stepmgr`)
- Does not work for batch steps, interactive steps, --external-launcher steps, or hetjobs/hetsteps

```
A=$(srun --async --parsable -n1 ./prep)
srun --async --step-dependency=afterok:$A ./run
srun --async --step-dependency=afterany:$A ./cleanup
swait
```

NVIDIA

# Async Steps – Per-step waits and --follow



swait blocks until asynchronous steps complete.  It has been enhanced to support more complex workflows and be more informative.

**Scope**

- Add new option to allow swait to block on just one specific step
- Add new option to allow swait --follow to print a list of steps being completed as they finish and a final line indicating all steps completed (drained).

```
$ step=$(srun --async --parsable -n 8 ./solve_a)
$ swait "$step"   # StepId=1234.0 code=0:0
```

```
$ srun --async --parsable -n 8 ./solve_a
$ srun --async -n 8 ./solve_b
$ swait --follow
JobId=XXX.0 code=0.0 reason=COMPLETED
JobId=XXX.1 code=0.0 reason=COMPLETED
JobId=XXX steps drained
```

# Container Runtime Plugins Infrastructure

Clusters No Longer Live in Isolation

Centralizes Slurm container support to improve user interface, container functionality, and integration with Slurm.

**Scope**

- Create a new `runtime` plugin infrastructure exposing all the integration points needed to develop a container plugin
- Migrate Slurm OCI support to `runtime/oci` plugin
- Add `runtime/none` for basic slurmstepd execution in base environment
- Enable slurmrestd support for container plugins
- Enable layering of Slurm namespaces (namespace/linux) and container runtime namespaces

**Future Work**

- Create `runtime/lua` as an integration point for all current SPANK container runtime plugins

# switch/lua

Compute Clusters Are Growing

Provide a mechanism to enable adhoc integration with new, and diverse networking hardware integrated on the same system.  Explicitly designed to work with RESTful services managing interconnect.

**Scope**

- Create new switch/lua plugin providing lua callbacks:
  - job lifecycle in slurmctld:
    - job start -- `slurm_switch_job_start()`
    - job completion -- `slurm_switch_job_complete()`
  - per node, per task in slurmstepd:
    - task initialization -- `slurm_switch_task_preinit()`
    - environment setup -- `slurm_switch_task_init()`
    - task teardown -- `slurm_switch_task_postfini()`
- Enhance switch to have slurmctld switch functions able to run asynchronously in addition to previous synchronous behavior
  - Has explicit retry logic for failure or for slurmctld restart
- Enhance switch job completion to drain nodes if switch teardown fails

# Expansion of the jobcomp interface

Clusters No Longer Live in Isolation

jobcomp/lua and jobcomp/kafka enhanced to add more job state change events

**Scope**

- Existing job event types:
  - Started
  - Ended
- New job events supported:
  - Queued
  - Requeued
  - Held
  - Released
- jobcomp/kafka only sends job identifier and state fields for new events
- Requires configuration in JobCompParams to enable the new event types
  - JobCompParams=enable_job_hold,enable_job_queue,enable_job_release, enable_job_requeue,enable_job_start

# GPU Plugins

## Compute Nodes Are Evolving




Continue to ensure first-class support for all major accelerators on the market.

**Scope**

- gpu/amd – Replacement for gpu/rsmi
  - RSMI library is deprecated by AMD
  - gpu/amd does not currently support frequency management, usage accounting
  - Work is planned for 27.05 to bring the plugin into parity with gpu/rsmi
- gpu/nvidia – Replacement for gpu/nvml
  - Extended to support dmem cgroup
  - Work is planned for 27.05 to bring the plugin into parity with gpu/nvml

# Drain TRES

Compute Nodes Are Evolving

Ability to drain and resume TRES (GRES, CPU, Memory) on a node without draining the whole node.

Drained resources are excluded from new job allocations.

- Allow an administrator to mark a specific device (by index or UUID) as drained
- Add DEGRADED node state to make nodes with drained TRES easy to find
- Node parameter `MaxDrainedTres` sets a threshold to automatically drain an entire node once too many components have been drained
- `SlurmctldParameter=drain_absent_gres` to automatically drain a gres that is missing instead of marking the node invalid

```
$ scontrol update TresName=gres/gpu Nodename=node0 Index=3 Action=Drain Reason=broken
$ scontrol show node node0 -d | grep -E "DrainedTRES|GresUsed"
  GresUsed=gpu:xyz:0(IDX:N/A),shard:xyz:0(0/10,d/10,0/10,0/10)
  DrainedTRES=gpu(IDX:1)(broken),cpu(IDX:6-7)(bad),mem(CNT:4096)(unavail)
$ scontrol show drainedtres
NodeName=node0 TresName=gres/gpu:xyz Index=3 Uuid=gpu-ddd Reason=broken
NodeName=node0 TresName=cpu Index=6-7 Reason=bad
NodeName=node0 TresName=mem Count=4096 Reason=unavail
$ scontrol update TresName=gres/gpu Nodename=node0 Index=3 Action=Resume
```

NVIDIA

# GRES Topology Enforcement

Compute Nodes Are Evolving

New `--gres-flags=enforce-links` ensures GRES devices are aligned according to the GRES topology

**Scope**

- Extend GRES model to represent cross-device relationships within a single GRES type
  - E.g., NVLINK topology between GPUs
- Ensure allocations respect the relevant topology for each device
  - E.g., always assign pairs of GPUs that are directly connected

**Impact**

- Fix long-standing issue with multi-GPU allocations not always selecting the optimal devices

**Restrictions**

- Device plugins will need to be updated to detect topology
- This may be blocked for certain plugins – such as gpu/nvidia (works for gpu/nvml)

# GRES Topology – Chunks

Compute Nodes Are Evolving

Open-Source.
Vendor-Neutral.

NVIDIA

Allows GRES devices to be allocated in arbitrarily aligned groups.

**Scope**

- Allow chunk sizes to be specified: `gres.conf ChunkSizes=1,2,4,8`
- Force resource selection to be aligned to chunk boundaries
  - A job requesting **N** devices only starts on a device positioned at a multiple of **N** so free devices stay in available chunks as much as possible

**Impact**

- Enables devices to be allocated according to administrator specified ordering and aligned in progressively sized blocks
- Particularly relevant for TPU/Trainium devices arranged in a torus to keep allocations aligned

**Restrictions**

- `ChunkSizes` must be ascending with each value a multiple of the previous value

# GRES Autodetection with Cloud Nodes

Compute Nodes Are Evolving

Allows AutoDetect option to be used in gres.conf for cloud nodes.

**Scope**

- AutoDetect for cloud nodes now sets up the GRES upon node registration
- Any job scheduled to a "powered down" cloud node will get a temporary incomplete assignment of GRES
- The allocator will re-run prior to clearing the powering-up state to keep the job bound to the nodes
- If the node turns out to have incorrect resources the node will be drained and the job requeued or canceled.

```
# slurm.conf
NodeName=cloud[1-8] State=CLOUD Gres=gpu:h100:8 ...
# gres.conf
AutoDetect=nvml
```

# TPU Support

Compute Nodes Are Evolving

Open-Source.
Vendor-Neutral.

NVIDIA

Added native TPU support to Slurm

**Scope**

- User interface provided by new `--layout salloc/sbatch/srun` option
- `job_submit/tpu` translates `--layout` into the detailed `gres/tpu` and node count request
- New `tpu.yaml` configuration describing valid layouts for TPUs

```
sbatch --layout=tpu7x=4x4x4 ...
# job_submit/tpu turns it into: -N 4, gres/tpu:tpu7x:4 per node
```

```
# slurm.conf
GresTypes=tpu
JobSubmitPlugins=tpu
TopologyPlugin=topology/block
NodeName=tpu[001-064] Gres=tpu:tpu7x:4
# gres.conf
Autodetect=tpu
```

```
# tpu.yaml
tpu_types:
  - type: tpu7x
    shape_per_node: "2x2x1"
    shape_per_segment: "4x4x4"
    nodes_per_pod: 2304
    pci_device_id: "0x0076"
```

# dmem Linux Cgroup Support

Compute Nodes Are Evolving

Enforcement of memory limits when sharing a GPU between multiple jobs.

**Scope**

- Optionally, redefine a shard as a proportion of device memory (e.g., 1 shard of 10 would be 1/10th the memory).
- At job start setup dmem cgroup limiting the job to use the requested amount of memory for the specific device(s) assigned to the job.
- Enabled with new `ConstrainDeviceMemory=yes` parameter in `cgroup.conf`

**Impact**

- Memory allocations to the device will fail if the requested allocation would exceed the allowed limit, no OOM-killer.

**Restrictions**

- There is no way to restrict compute time/proportion when sharing a GPU – this only covers memory.

# Slurm Control Plane Without UIDs

Clusters No Longer Live In Isolation / Slurm As A “Pure” Scheduler

Slurm has always required users to map to a specific uid/gid.

This was architecturally critical for authentication with MUNGE, but is no longer strictly desired, and alternative authentication methods like `auth/slurm` have started to remove this dependency.

**Scope**

- Allow authentication of users through auth/jwt with only a username
  - For External jobs (used by Slurm Bridge), no resolution is ever required
  - For normal jobs, the slurmd will need to provide the resolution before the job launches
    - Requires new `SlurmctldParameters=enable_deferred_user_lookup` option
- For auth/slurm, remove requirement that SlurmUser’s UID match throughout the cluster

# HetStep Management by stepmgr

Compute Clusters Are Growing

Offload launch and management of HetSteps (within a HetJob) to stepmgr

**Scope**

- Enable stepmgr to start and then manage HetSteps
- Let clients and component stepmgrs find and contact each component's stepmgr directly
- Coordinate unique HetStep IDs within the job (no slurmctld)

**Impact**

- Reduce slurmctld load for starting HetSteps and recording their completion
  - Particularly important if a HetJob runs many HetSteps

**Restrictions**

- stepmgr must be used (`--stepmgr` or `SlurmctldParameters=enable_stepmgr`)
- `srun --async` is not supported for HetSteps

# Show Topology – --json/--yaml and REST

Compute Clusters Are Growing

## Scope

- `scontrol --json|--yaml show topology [{name}]` returns topology information as structured data
- New REST endpoints:
  - GET /slurm/v0.0.46/default/topology
  - GET /slurm/v0.0.46/topology/{topology_name}
- Same schema covers all topology plugins: block, flat, ring, torus3d, tree

## Restrictions

- Only available with data_parser v0.0.46+ (not in older versions)
- An unknown topology returns HTTP 404 (ESLURM_REST_TOPO_NOT_FOUND)

# SlurmDBD – HTTP health endpoints

Compute Clusters Are Growing

## Scope

- `slurmdbd` can now respond to HTTP sent to its `DbdPort` (default 6819) like slurmcltd and slurmd
- Common health status probes: /livez, /healthz, /readyz
- Versioned JSON endpoints:
  - /{version}/conf (slurmdbd config)
  - /{version}/diag (RPC and rollup statistics)
- New slurmdbd.conf settings: `HttpAuthParameters` (JWT/JWKS) and `CommunicationParameters=disable_http`

## Impact

- Automation tooling can check DBD status without sacctmgr
- /readyz succeeds only on the active slurmdbd, can be used to identify which is active

## Restrictions

- Probe details (/readyz?verbose) requires root or SlurmUser. Other callers get status code only
- Versioned endpoints need JWT through HttpAuthParameters, otherwise all callers get 401
- A standby backup doesn't listen until first takeover

NVIDIA

# Per-Step PID Limits – `--max-pids`

Compute Clusters Are Growing

**Scope**

- New `--max-pids=<n>` for `salloc/sbatch/srun`. Limits the total count of processes and threads that can exist on the each step per node. The Linux kernel enforces through the cgroup v2 pids controller
  - The job level value sets a default for every step, including batch and extern steps. srun --max-pids can lower the limit, but cannot raise it
- Limit can be viewed:
  - `MaxPids` field for `scontrol show job/step`, `squeue`, and `slurmrestd`
  - sacct: `AvePids`, `MaxPids`, `MaxPidsNode`, `MaxPidsTask`
- Accessible in `job_submit/lua` as `max_npids`

**Impact**

- Provides enforcement about process/thread creation in steps, preventing one job from over-consuming resources on a shared node

**Restrictions**

- When the limit is hit `fork()/clone()` fail with `EAGAIN` and Slurm does not detect or report it. The application needs to handle the error.
- Limit is set by user, site policy can be applied via `job_submit/lua`

NVIDIA

# namespace/linux – IPC Namespace Isolation

Compute Clusters Are Growing

**Scope**

- New `CLONE_NEWIPC` value for `clone_ns_flags` in `namespace.yaml`
- Each job gets its own IPC namespace, isolating:
  - System V IPC: shared memory segments, semaphores, message queues
  - POSIX message queues
- `/dev/mqueue` is remounted inside the job, listed only the job's queues

**Impact**

- Jobs sharing a node can't see or change each other's IPCs or message queues
- Leftover IPC objects are automatically cleaned up at job end: leaked shared memory segments and semaphores don't build up on the node

**Restrictions**

- Requires `NamespaceType=namespace/linux` and `PrologFlags=contain` plus configuration in namespace.yaml
- POSIX shared memory (`/dev/shm`) is not covered via the IPC namespace. `/dev/shm` is already isolated by default by `namespace/linux`.
- IPC namespace is per job per node, not per step. All steps on the same share share the namespace.

NVIDIA

# HealthCheckProgram healthy/unhealthy split

Compute Clusters Are Growing

**Scope**

- Split healthy nodes (idle, running jobs, mixed, planned) from unhealthy nodes (drained, down) and separately launch the health check for these two populations
- Set `SLURM_NODE_IS_HEALTHY` depending on which population the node is in:
  - yes - node is in service
  - no - down, draining or drained, failing, maintenance reservation
  - unset - not known (the run at slurmd start)

**Impact**

- Health check can now use `SLURM_NODE_IS_HEALTHY` to determine if a success should resume or a failure should drain – without running sinfo.
  - This enables the health check program to only generate slurmctld RPC traffic on changes, can substantially reduce slurmctld traffic

**Restrictions**

- The health check script needs to be updated to use this feature.

# Declarative Load of SlurmDB Data

Compute Clusters Are Growing

**Scope**

- New `sacctmgr load <file> declarative`. Each account, user, and qos listed in the file is updated to match the file
  - Any settings omitted from the file will return to defaults (e.g. association limits)
  - Entities that don't exist yet are created
- `sacctmgr dump` followed by `load declarative` would make no changes, so a dump file can be used to regenerate state

**Impact**

- Accounting configuration as code: keep the account/user/QOS layout file in version control and apply repeatedly, only the changes will be made, no re-creates
  - Edits are in place, association ids and history are maintained

**Restrictions**

- Can't be combined with `clean`
- Nothing is ever deleted. Accounts, users, associations, and QOS that aren't in the file are left alone, so removals must be done by some other process.
- Not all-or-nothing, if it fails part way through the partial changes remain

NVIDIA

# Prevent lower priority job starts with backfill and node failures

Compute Clusters Are Growing

**Scope**

- Add `SchedulerParameters=bf_node_unavail_exit`
- Ends the backfill cycle if a node fails during a yield in the backfill cycle

**Impact**

- Prevents lower priority jobs from starting by forcing the backfill cycle to restart upon node failure

**Restrictions**

- Only useful with `SchedulerParameters=bf_continue`

# HetJobs – Mixed slurm and external components

Compute Clusters Are Growing

## Scope

- HetJobs can now include `--external` components, three kinds of HetJobs are now possible:
  - Regular: every component runs through slurmd/slurmstepd (unchanged)
  - External: every component is on an external node (typically mangaged by Slinky slurm-bridge)
  - Mixed: a regular leader component and any number of regular or external components
- Fixed both `sbatch --external : ...` and slurmrestd submission of external hetjobs

## Impact

- Using slurm-bridge one can now orchestrate complex workflows in kubernetes that have both Slurm and native kubernetes components (or only kubernetes components)
  - e.g., Database or web services can easily be co-scheduled with a slurm job depending on those services

## Restrictions

- Unless all components are external, then the leader must be a regular component, the batch script runs in the leader component.
- An external leader cannot have a batch script or `--wrap` argument


NVIDIA