Field Notes 8

How to make the most of Slurm, and avoid common issues

Alejandro Sánchez
Europe Team Lead

Slurm User Group 2024

slurm | SCHEDMD

Field notes - tradition

Continuing to build on last year's field notes

https://slurm.schedmd.com/publications.html

Purpose

To show you how to get the most out of your support experience and point you in a better direction while working with Slurm

Agenda

Agenda

- System Requirements
- Cluster Architecture
- Release Cycle and Support
- Build, Install and Upgrade
- Dynamic Nodes
- Internal Authentication
- Monitoring Insight
- Cgroup Insight
- Commercial Support

System Requirements

System Requirements - slurmctld

- Hardware recommendations for the hosts where slurmctld / slurmdbd run and fast access to StateSaveLocation has been discussed in previous editions of Field Notes
  - https://www.schedmd.com/publications/

System Requirements - slurmctld

- Some incremental performance considerations when choosing hardware:
  - slurmctld memory usage
    - slurm.conf MinJobAge is relevant for slurmctld memory usage
      - Time after finished jobs are cleared from slurmctld memory
    - Jobs can still be queried after MinJobAge through accounting
      - sacct / slurmrestd
    - If slurmdbd not reachable, slurmctld caches accounting messages

System Requirements - slurmdbd

- Some incremental performance considerations when choosing hardware:
  - slurmdbd performance
    - Number of jobs/steps saved/queried to/from the database
    - slurm.conf AccountingStoreFlags
      - job_comment, job_env, job_extra, job_script, no_stdio
    - slurmdbd.conf Archive*/Purge* options are highly recommended
    - slurmdbd.conf MaxQueryTimeRange as well

System Requirements - Management network

- slurmctld, slurmdbd and slurmd traffic should be on a dedicated network
- Should not share the same “twisted pair” network or bandwidth as user storage/data
- For most sites, a simple flat network is recommended
    - e.g. 192.168.0.1/24

System Requirements - Continued network

- Support for IPv4 and IPv6 available
- Knowing what to expect from the Slurm network layer can avoid many firewall (and other) networking issues
    - https://slurm.schedmd.com/network.html

System Requirements - Continued

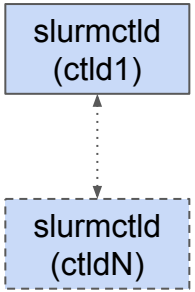
- Widely known in Slurm community, but worth emphasizing:
  - There must be a uniform user and group name space (including UIDs and GIDs) across the cluster
  - Clocks must be in sync across cluster nodes
  - We still see instances of issues because this unmet requirement

Cluster Architecture

High-Availability (HA) options - slurmctld

- Multiple SlurmctldHost configured
  - SlurmctldHost=ctld1(name1/addr1)
  - ...
  - SlurmctldHost=ctldN(nameN/addrN)
  - First parsed interpreted as primary and all other backups
  - If one fails, the next available will request control to act as primary

...

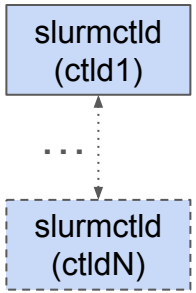


High-Availability (HA) options - slurmctld

- Host where slurmctld runs modified by another process (i.e. pacemaker)
  - SlurmctldHost=ctld1,ctld2(ctld/addr)
- SlurmctldPrimary[Off|On]Prog
  - On: runs when backup becomes primary. Use-cases:
    - Set ip address to interface
    - Kill unresponsive controller
    - Flush ARP caches ...
  - Off: runs when primary becomes backup

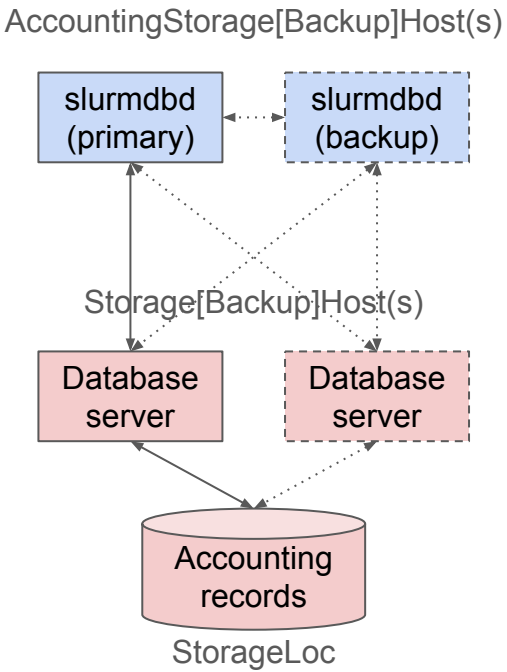
High-Availability (HA) options - slurmctld

- Our preference is multiple SlurmctldHost
- The previous two setups are mutually exclusive
- The StateSaveLocation must be accessible **from all configured SlurmctldHost** controllers
- The StateSaveLocation should not be local storage on either of the slurmctld hosts or local to one slurmctld and serviced out to the other slurmctld



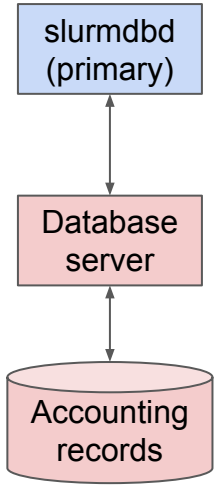
High-Availability (HA) options - slurmdbd

- ... While Slurm allows configuring a primary and a backup slurmdbd(s)
  - AccountingStorage[Backup]Host
- And while it is also possible to setup HA for the database server itself (MySQL/MariaDB) ...



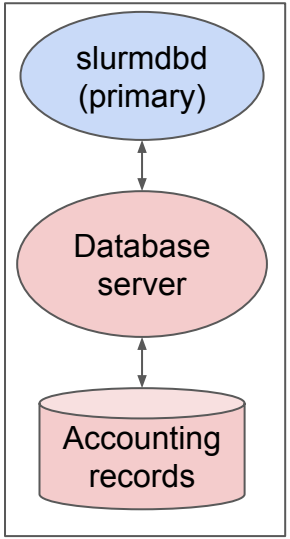
High-Availability (HA) options - slurmdbd

- … We highly recommend sticking with a single slurmdbd instance, against a non-HA MySQL/MariaDB server.
- All slurmctld messages intended to slurmdbd are cached in the event that slurmdbd is unavailable. Relevant slurm.conf options:
  - MaxDBDMsgs
    - MAX(value, MaxJobCount * 2 + Node Count * 4)
  - SlurmctldParameters=max_dbd_msg_action=[discard|exit]
    - Discard order: step start/complete ⟶ job start



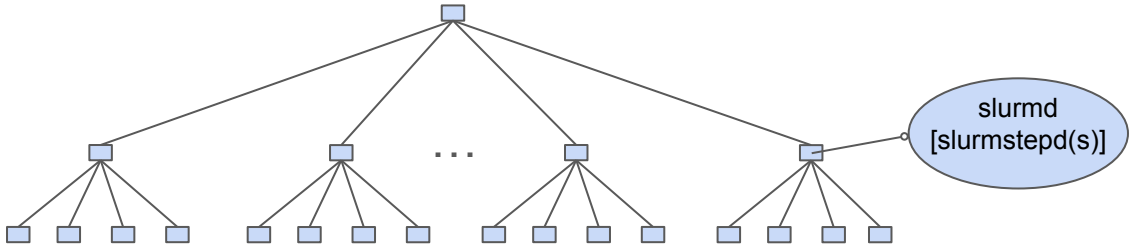
High-Availability (HA) options - slurmdbd

- From our experience, this setup is less complex than introducing database server replication into the mix.
  - Simpler is better from our perspective - the fewer technologies in the mix, the less there is to troubleshoot.
- Co-locating slurmdbd with its own MySQL/MariaDB instance is preferred
  - AccountingStorageHost == DbdHost == StorageHost
  - Having them in separate hosts it's also okay (depends on resources), but less preferred



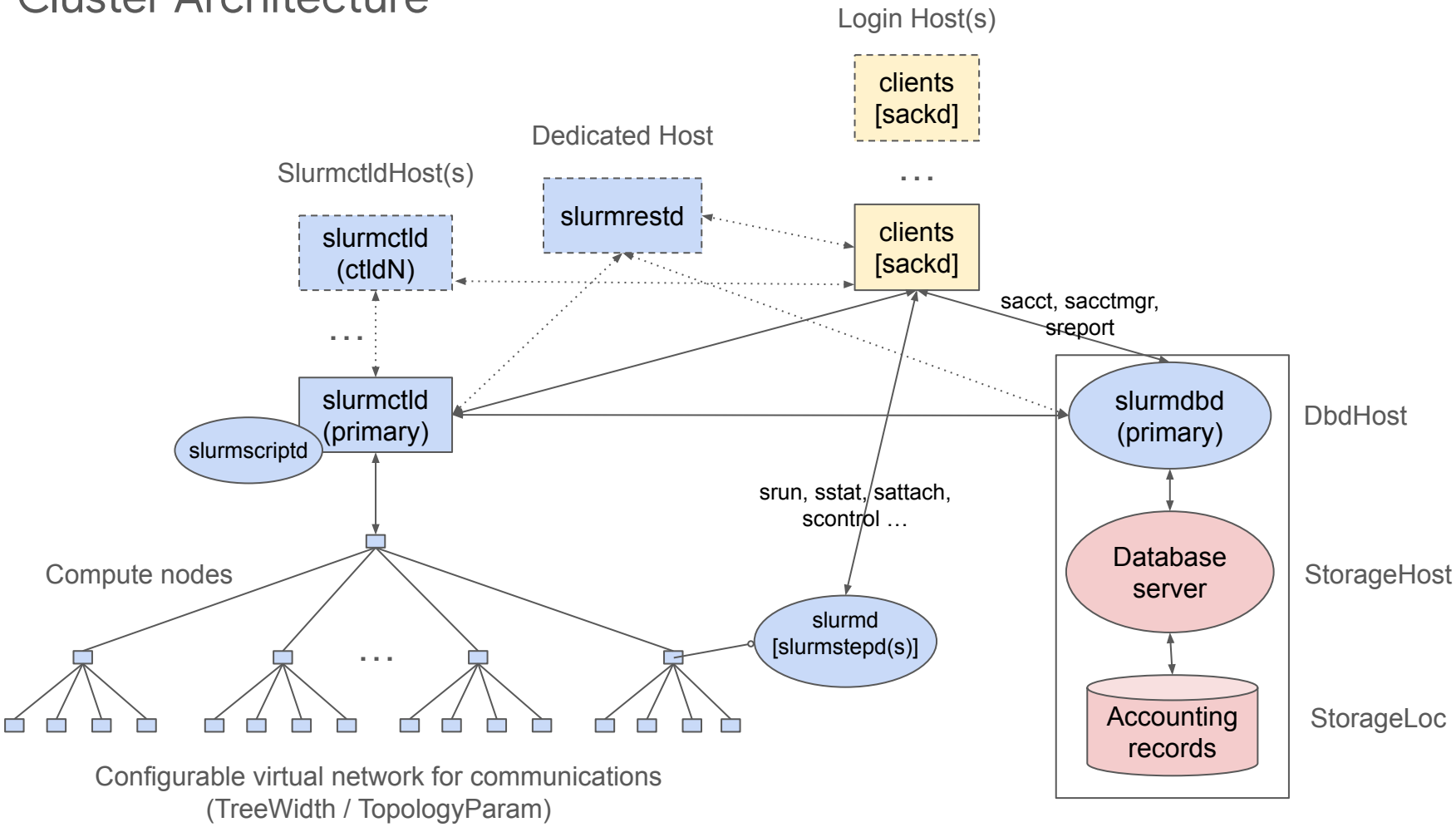
Compute node possibilities

- On-prem
- Cloud
  - https://slurm.schedmd.com/power_save.html
- Mixed On-prem and Cloud
- Dynamic Nodes
  - https://slurm.schedmd.com/dynamic_nodes.html

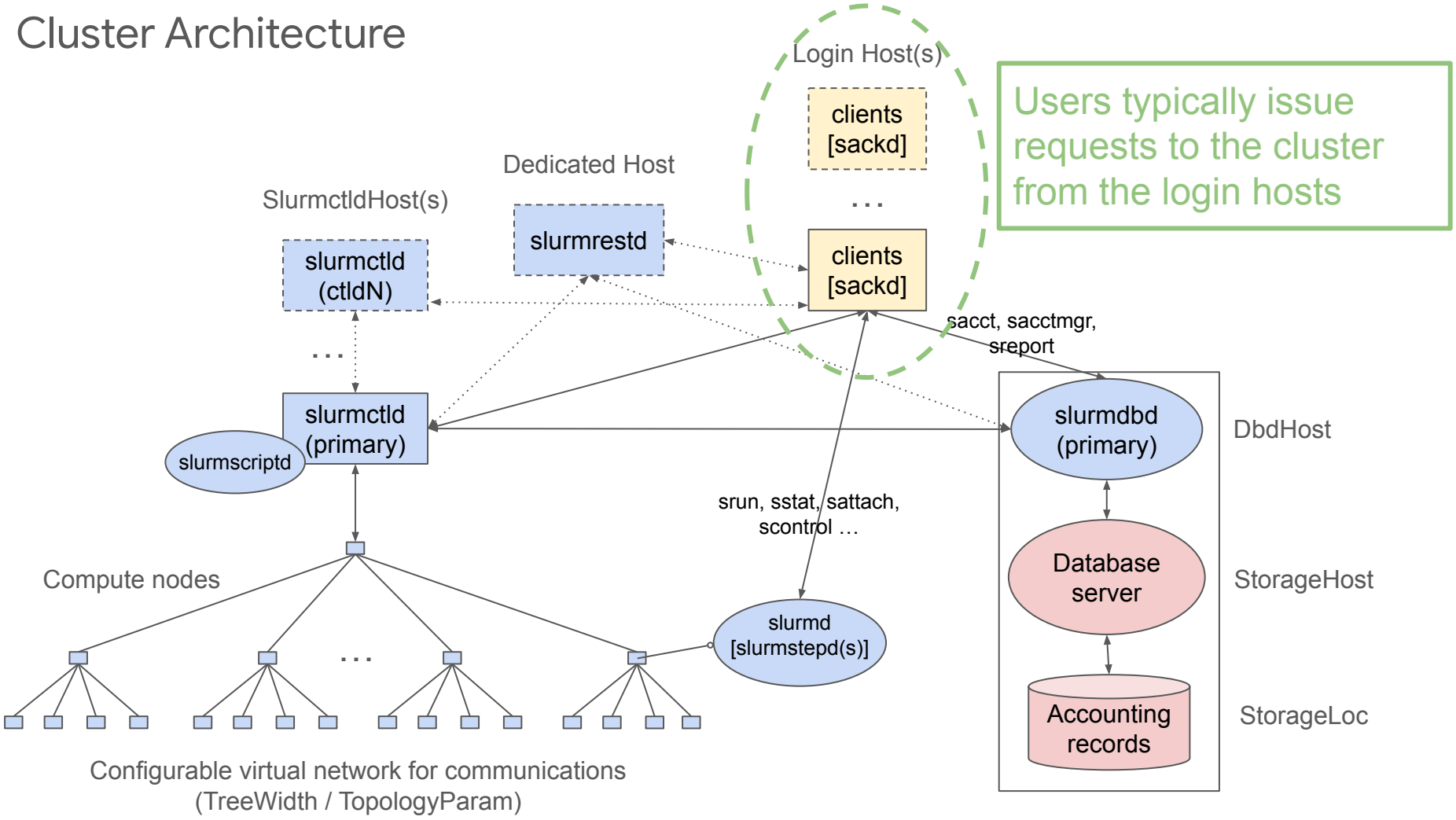


Configurable virtual network for communications
(TreeWidth / TopologyParam)

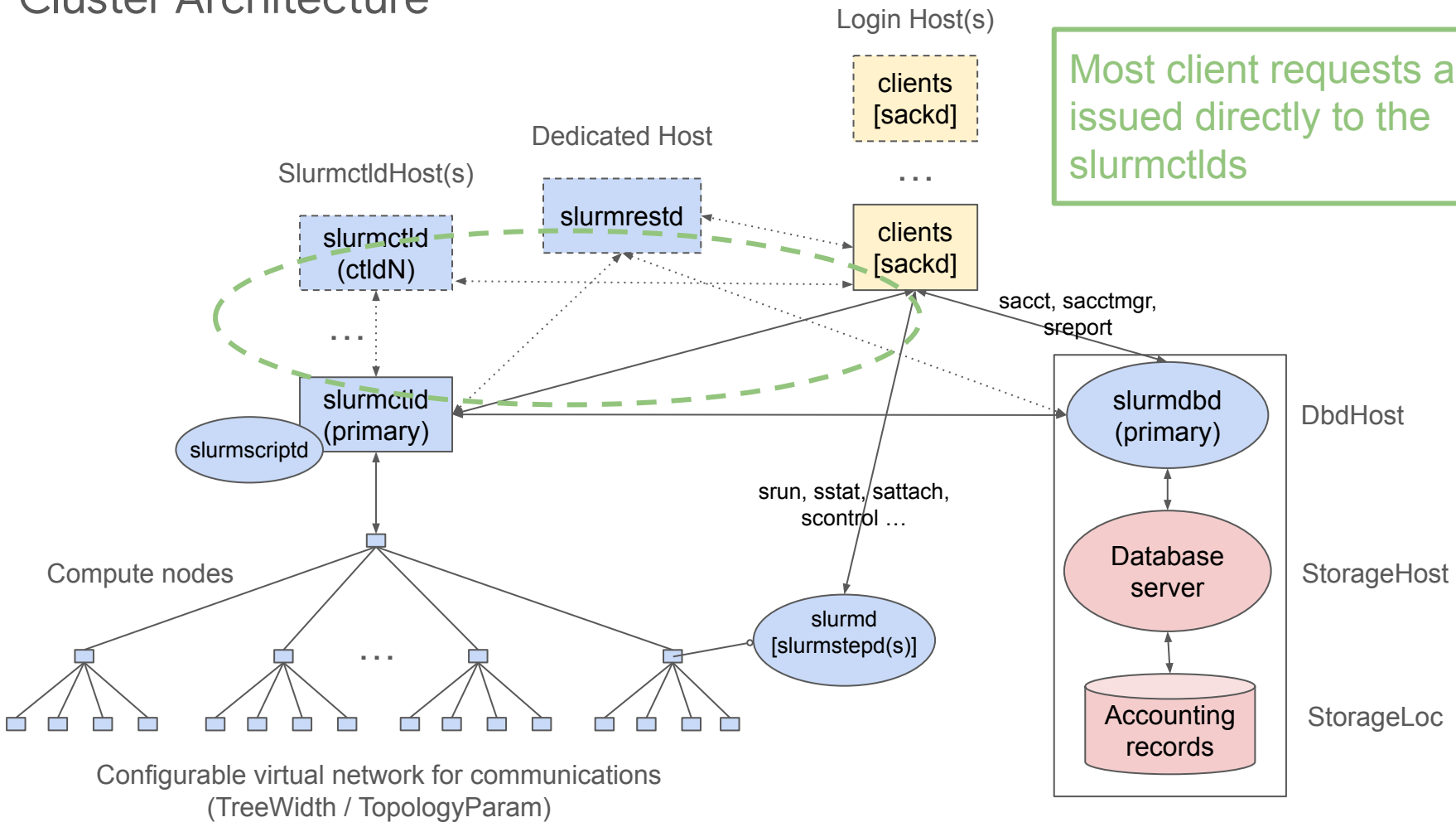
Cluster Architecture



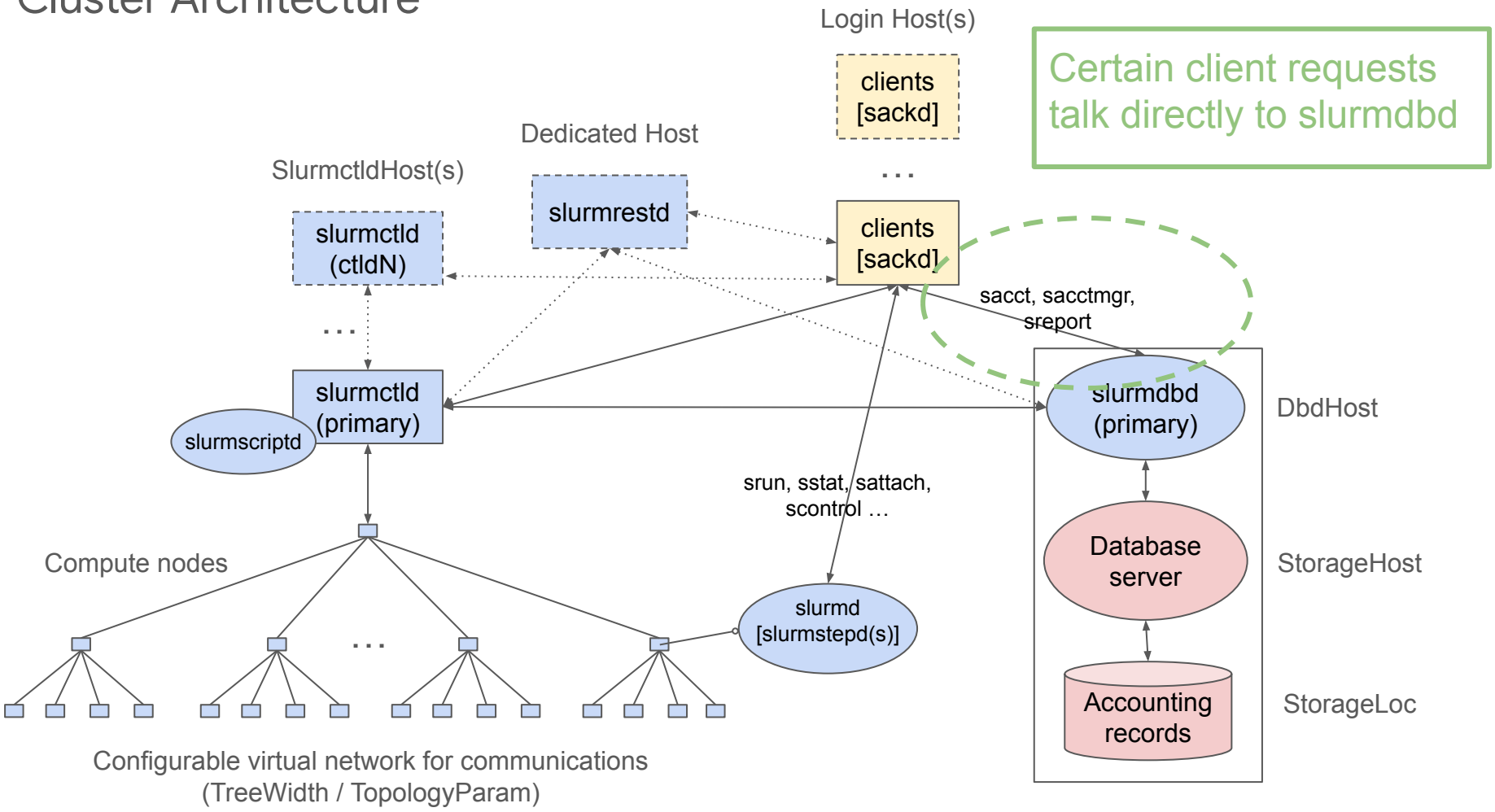
Cluster Architecture



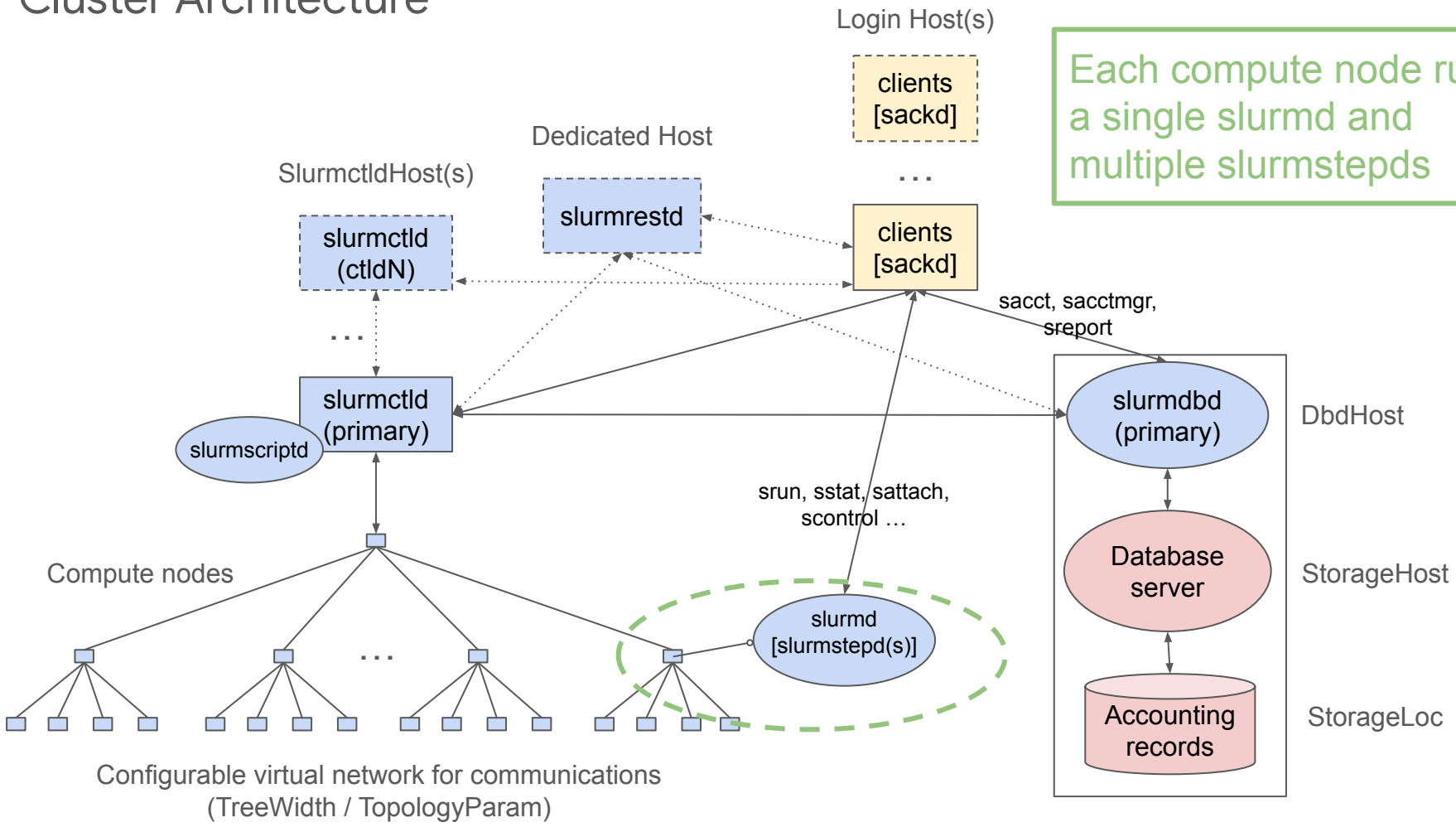
Cluster Architecture



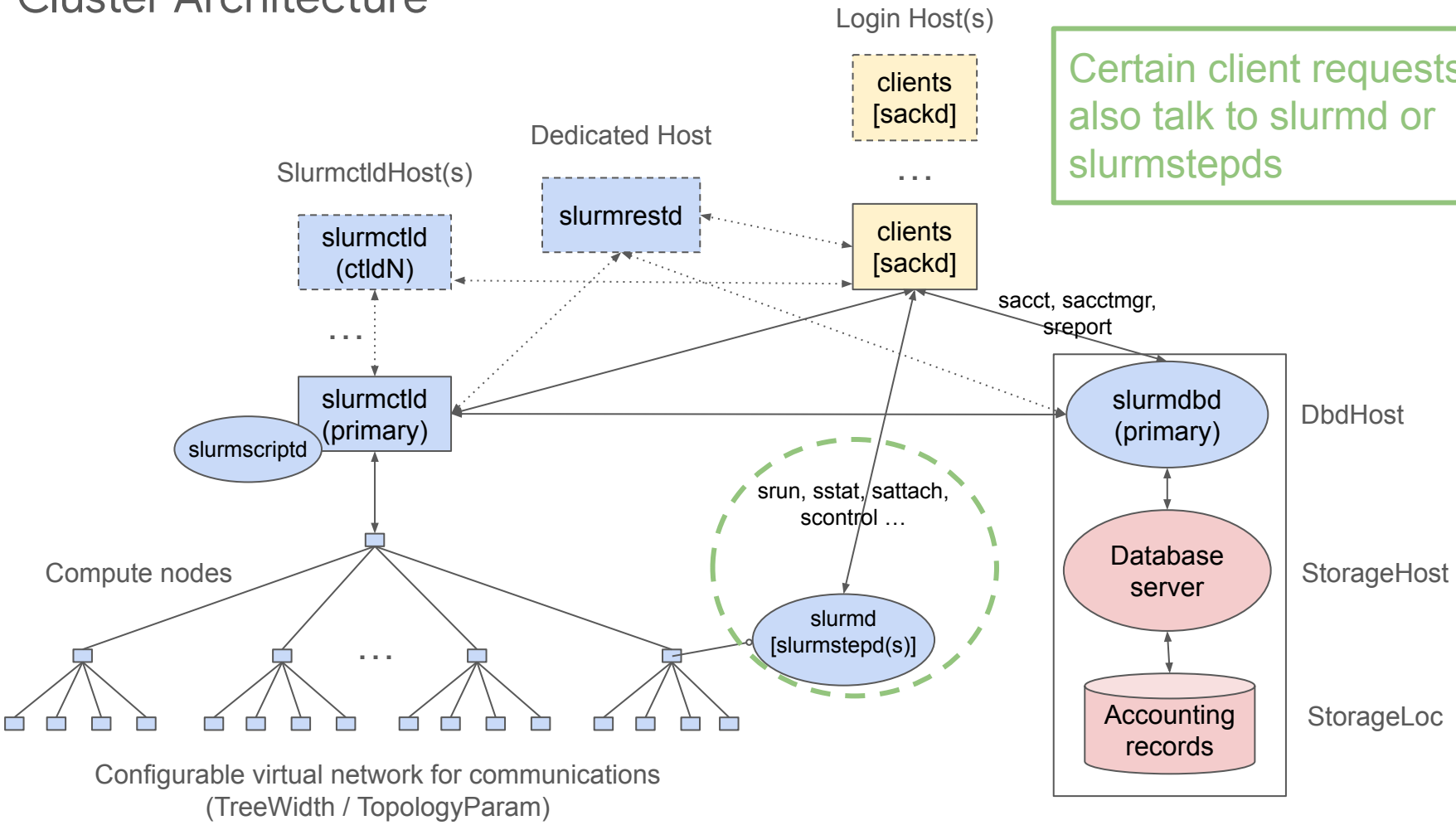
Cluster Architecture



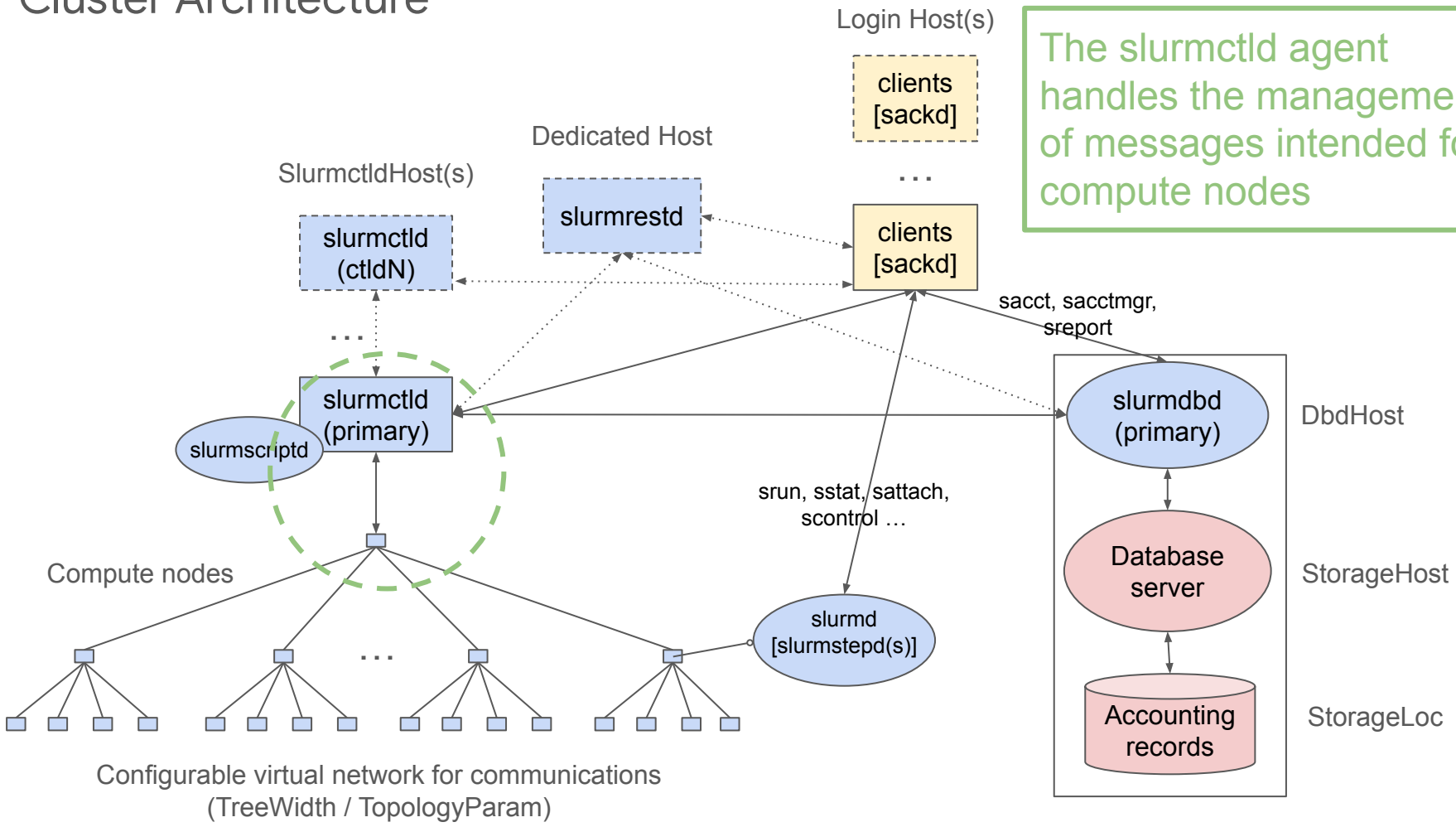
Cluster Architecture



Cluster Architecture

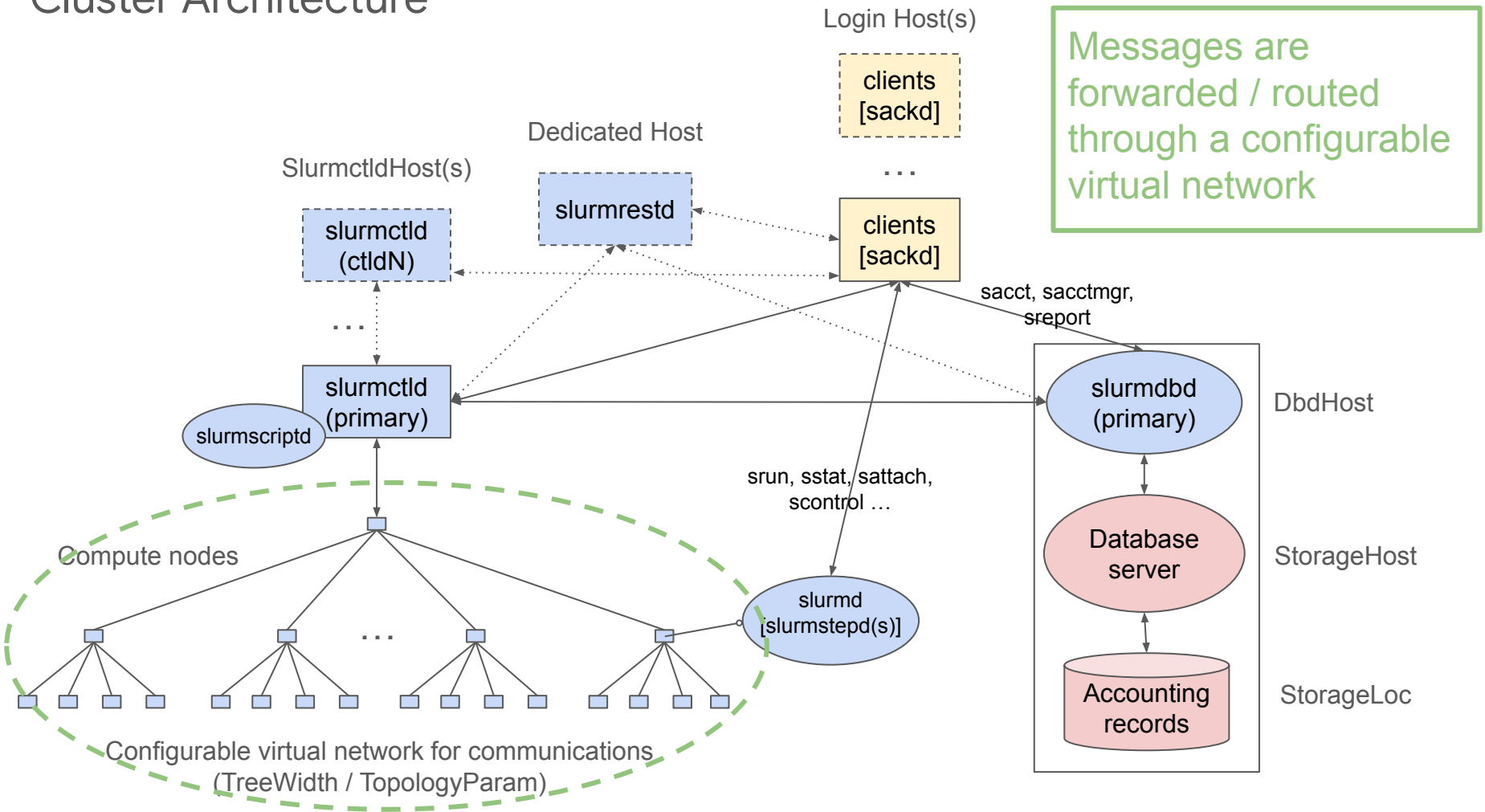


Cluster Architecture



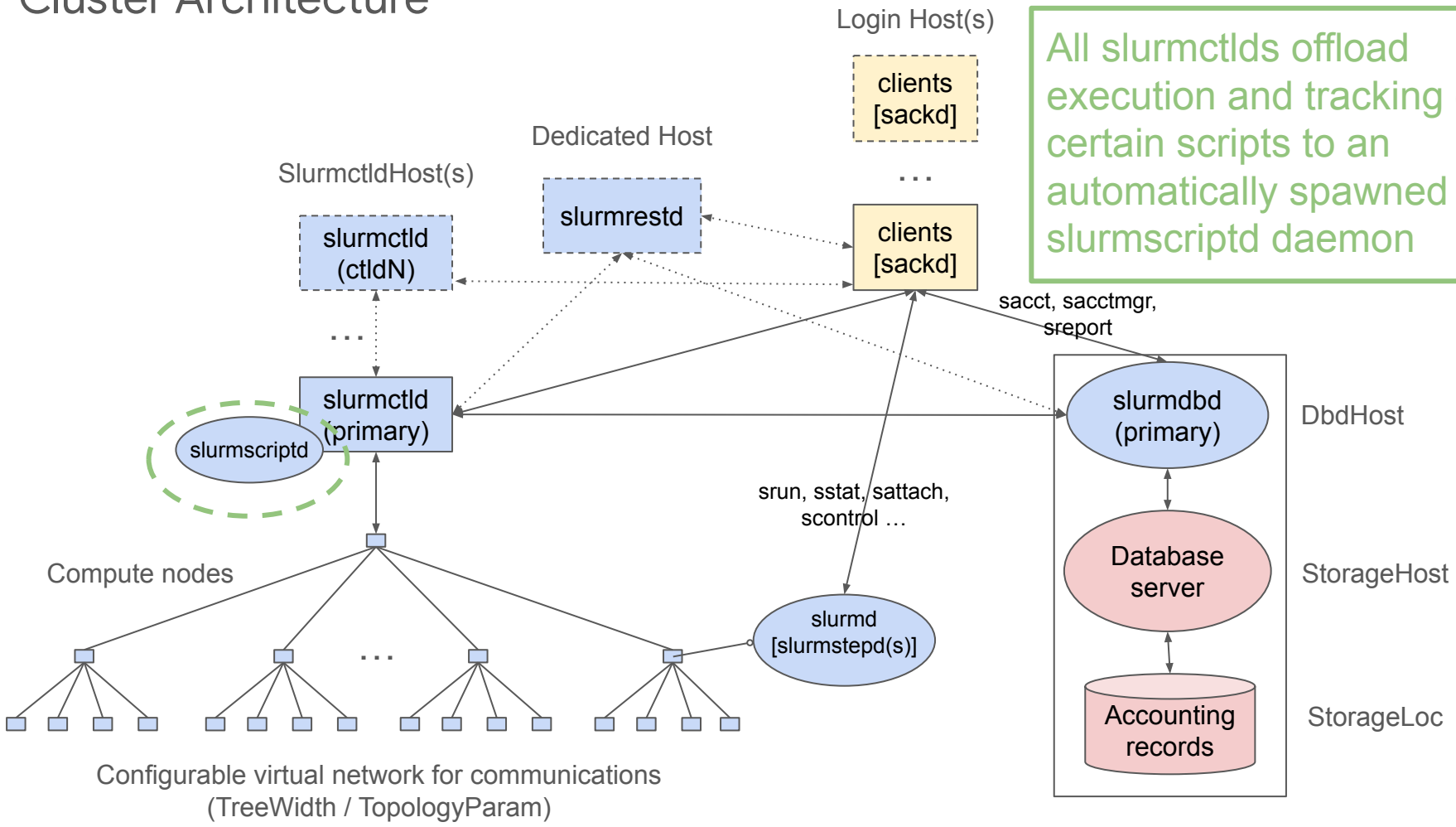
The slurmctld agent handles the management of messages intended for compute nodes

Cluster Architecture



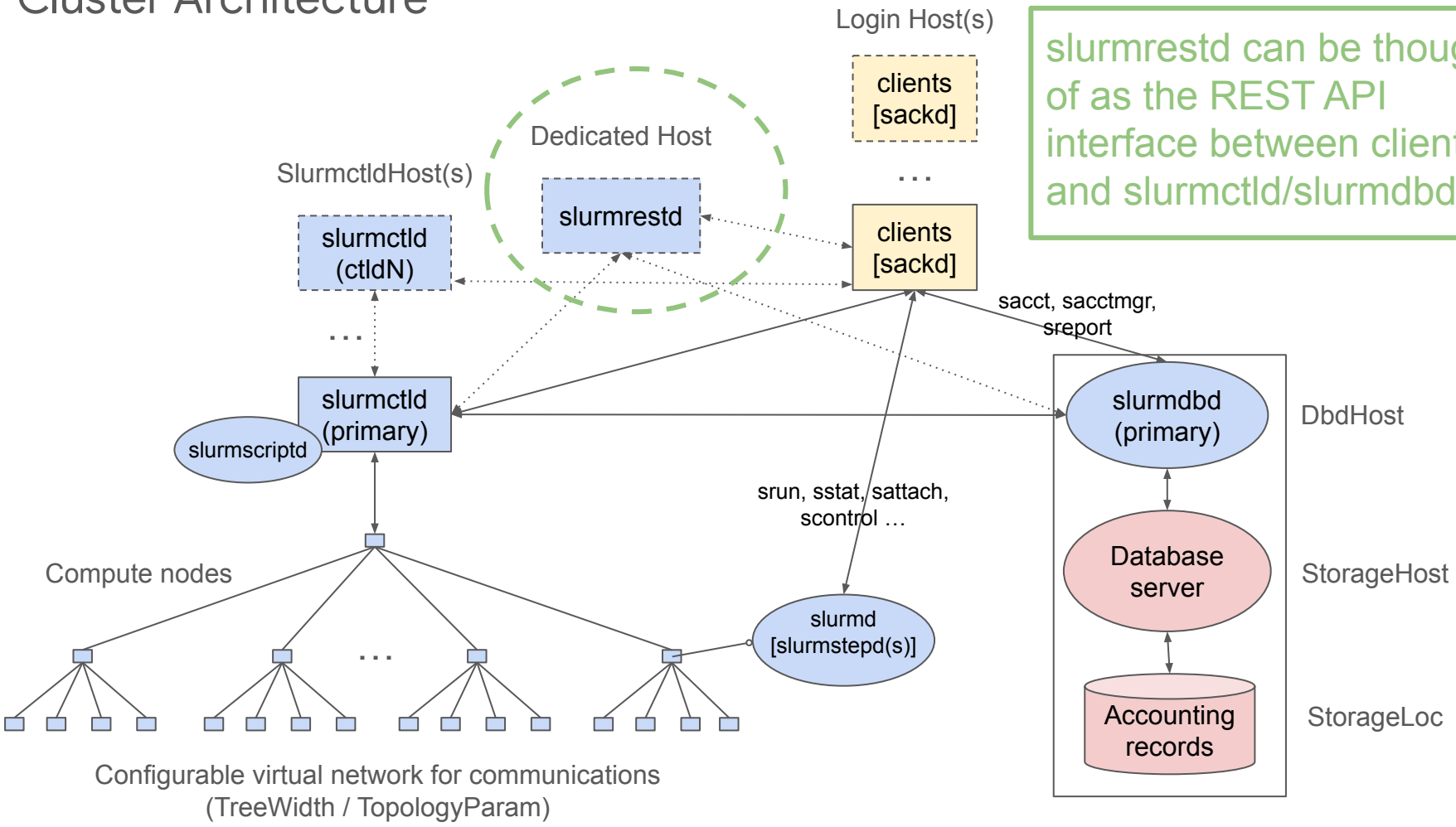
Messages are forwarded / routed through a configurable virtual network

Cluster Architecture



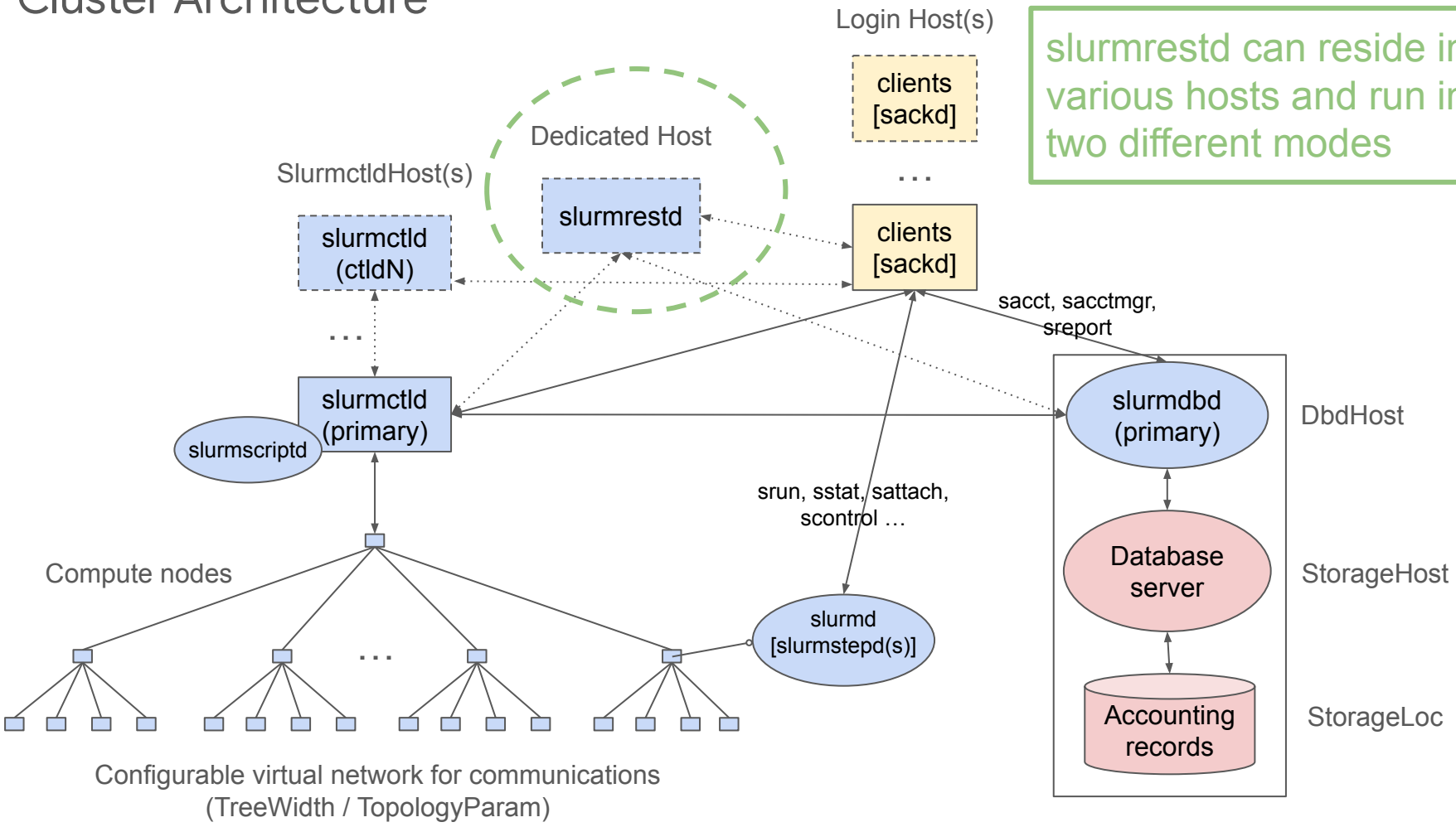
All slurmctlds offload execution and tracking of certain scripts to an automatically spawned slurmscriptd daemon

Cluster Architecture

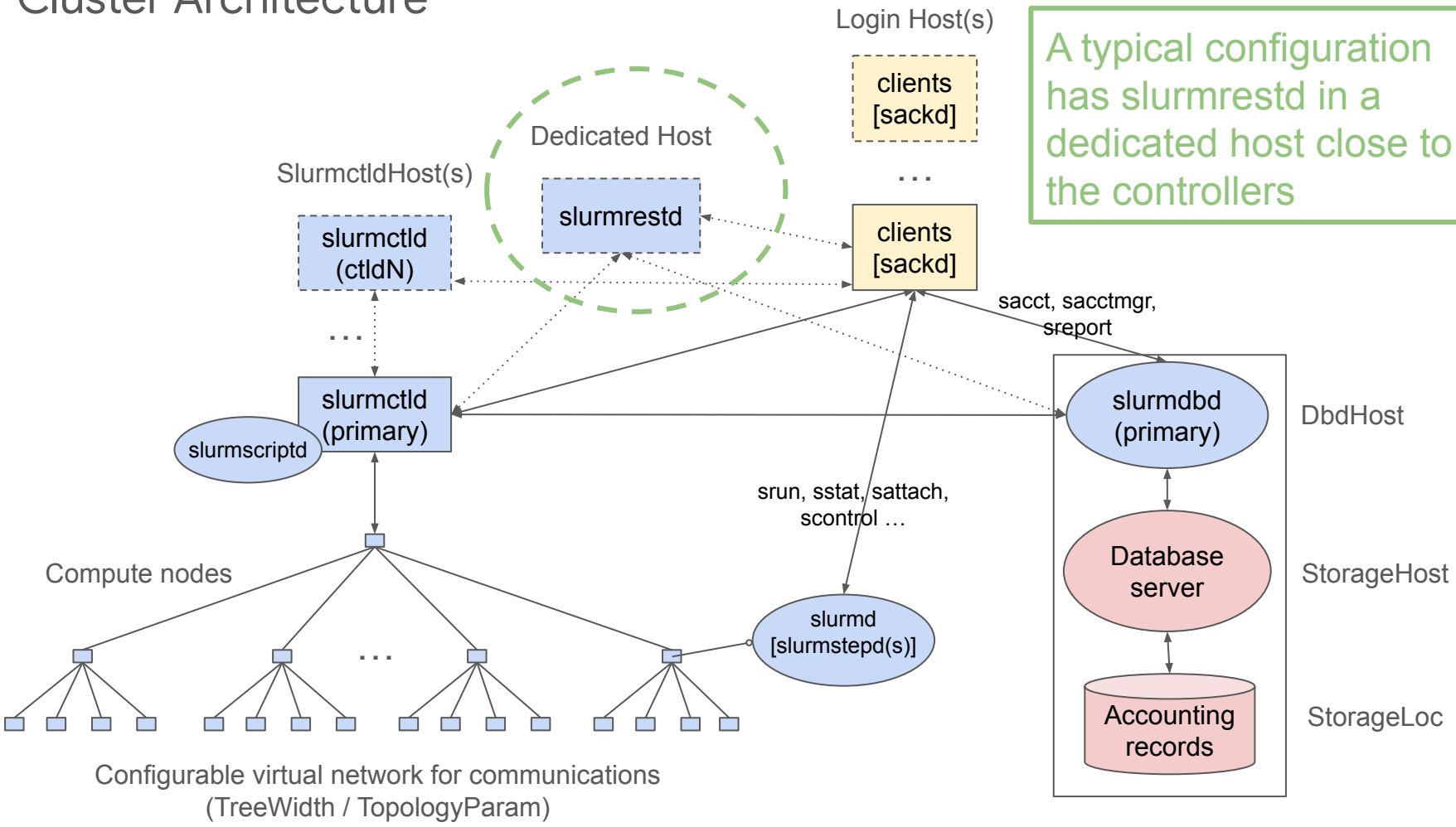


slurmrestd can be thought of as the REST API interface between clients and slurmctld/slurmdbd

Cluster Architecture



Cluster Architecture

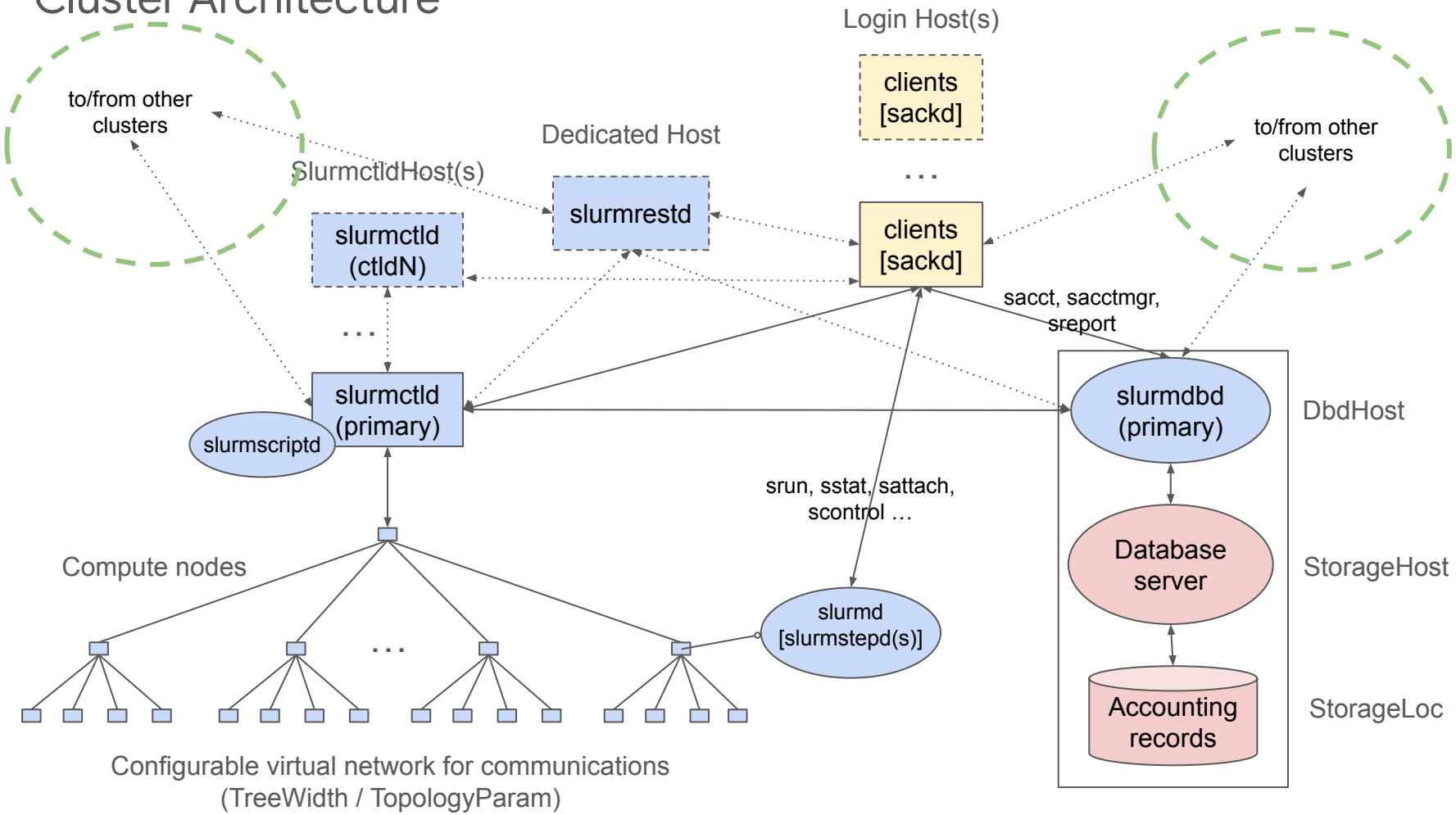


A typical configuration has slurmrestd in a dedicated host close to the controllers

Clustering options

- Single-cluster
- Multi-cluster
  - https://slurm.schedmd.com/multi_cluster.html
  - Federation (not standard workflow, only useful in a limited use case)

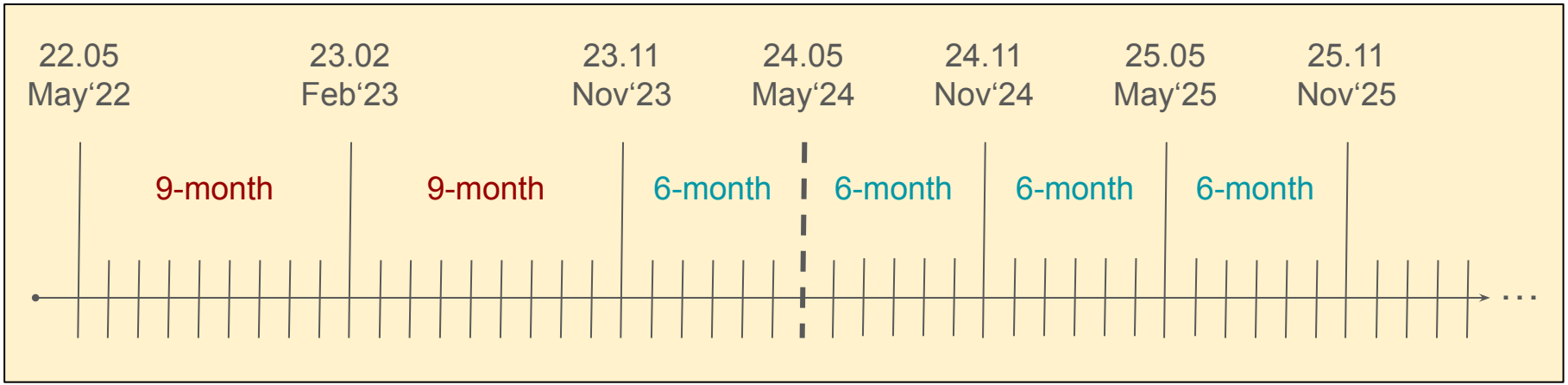
Cluster Architecture



Release Cycle and Support

Release cycle and support - continued

- Slurm has moved from an historical 9-month to a **6-month major release cycle**
- The change started with the Slurm version 24.05 (released in May 2024)
- Slurm 24.11 will follow (in November 2024)
- Major releases then continue every May and November in 2025 and beyond
  - Rather than… November, August, May, February, then November again…



Release cycle and support - continued

- There are two main goals of this change:
  - Faster feature delivery for bleeding-edge systems
  - Alignment with annual maintenance windows for stable systems
    - “Predictable” timeline for major releases
    - We expect a lot of sites will stick with either May or November releases, and will skip every-other major release going forward

Release cycle and support - continued

- Maintenance releases are made roughly monthly
  - Usually only made for the most recent major release
  - Security issues are fixed in all currently supported major releases

Release cycle and support - continued

- SchedMD's Slurm Support has been built around an 18-month cycle
- This 18-month cycle has traditionally covered the current stable release, plus one prior major releases
  - 2 x 9-month
- With the increase in release frequency this support window will now cover to the current stable release, plus two prior major releases.
  - 3 x 6-month
- Slurm 23.02 in particular will be supported until 24.11 is released
  - Window extended to 21 months, instead of 18 months, to help with transition period

Release cycle and support - continued

- SchedMD will be adjusting our handling of backwards-compatibility within Slurm itself
- For the 24.05 release, Slurm will still only support upgrading from (and mixed-version operations with) the prior two releases (23.11, 23.02)
- Starting with 24.11, Slurm will start supporting upgrades from the prior three releases (24.05, 23.11, 23.02)
    - This provides a similar support duration with the more frequent 6-month release cycle

Release cycle and support - continued

Slurm release	Revised End of Support	Supports Direct Upgrades From
23.02	November 2024 (21 months)	22.05, 21.08
23.11	May 2025 (18 months)	23.02, 22.05
24.05	November 2025 (18 months)	23.11, 23.02
24.11	May 2026 (18 months)	**24.05, 23.11, 23.02**
25.05	November 2026 (18 months)	24.11, 24.05, 23.11
25.11	May 2027 (18 months)	25.05, 24.11, 24.05

Release cycle and support - continued

- Slurm release eligibility for patches
  - Each Slurm major release is supported for 18 months
  - As each major release matures, there is an increasingly restrictive threshold for what is eligible for inclusion in that branch

Release cycle and support - Patches Eligibility Matrix

- Green cells (✓) indicate a patch is eligible for that branch
- Yellow cells (~) indicate a patch will be considered on a case-by-case basis
- Red cells (×) indicate a patch is not eligible

T = Major release month	T+0 month	T+3 month	T+6 month	T+9 month	T+12 month	T+15 month
Security fixes	✓	✓	✓	✓	✓	✓
Daemon inoperable	✓	✓	✓	~	x	x
Major regressions	✓	✓	~	x	x	x
Minor regressions	✓	~	x	x	x	x
Documentation	✓	✓	~	x	x	x
Dev project rework	~	x	x	x	x	x

Release cycle and support - continued

- Green cells (✓) indicate a patch is eligible for that branch
- Yellow cells (~) indicate a patch will be considered on a case-by-case basis
- Red cells (×) indicate a patch is not eligible

T = Major release month	T+0 month	T+3 month	T+6 month	T+9 month	T+12 month	T+15 month
Security fixes	✓	✓	✓	✓	✓	✓
Daemon inoperable	✓	✓	✓	~	x	x
Major regressions	✓	✓	~	x	x	x
Minor regressions	✓	~	x	x	x	x
Documentation	✓	✓	~	x	x	x
Dev project rework	~	x	x	x	x	x

Release cycle and support - continued

- Green cells (✓) indicate a patch is eligible for that branch
- Yellow cells (~) indicate a patch will be considered on a case-by-case basis
- Red cells (×) indicate a patch is not eligible

T = Major release month	T+0 month	T+3 month	T+6 month	T+9 month	T+12 month	T+15 month
Security fixes	✓	✓	✓	✓	✓	✓
Daemon inoperable	✓	✓	✓	~	x	x
Major regressions	✓	✓	~	x	x	x
Minor regressions	✓	~	x	x	x	x
Documentation	✓	✓	~	x	x	x
Dev project rework	~	x	x	x	x	x

Release cycle and support - continued

- Green cells (✓) indicate a patch is eligible for that branch
- Yellow cells (~) indicate a patch will be considered on a case-by-case basis
- Red cells (×) indicate a patch is not eligible

T = Major release month	T+0 month	T+3 month	T+6 month	T+9 month	T+12 month	T+15 month
Security fixes	✓	✓	✓	✓	✓	✓
Daemon inoperable	✓	✓	✓	~	x	x
Major regressions	✓	✓	~	x	x	x
Minor regressions	✓	~	x	x	x	x
Documentation	✓	✓	~	x	x	x
Dev project rework	~	x	x	x	x	x

Release cycle and support - continued

- Green cells (✓) indicate a patch is eligible for that branch
- Yellow cells (~) indicate a patch will be considered on a case-by-case basis
- Red cells (x) indicate a patch is not eligible

T = Major release month	T+0 month	T+3 month	T+6 month	T+9 month	T+12 month	T+15 month
Security fixes	✓	✓	✓	✓	✓	✓
Daemon inoperable	✓	✓	✓	~	x	x
Major regressions	✓	✓	~	x	x	x
Minor regressions	✓	~	x	x	x	x
Documentation	✓	✓	~	x	x	x
Dev project rework	~	x	x	x	x	x

Release cycle and support - continued

- Green cells (✓) indicate a patch is eligible for that branch
- Yellow cells (~) indicate a patch will be considered on a case-by-case basis
- Red cells (×) indicate a patch is not eligible

T = Major release month	T+0 month	T+3 month	T+6 month	T+9 month	T+12 month	T+15 month
Security fixes	✓	✓	✓	✓	✓	✓
Daemon inoperable	✓	✓	✓	~	x	x
Major regressions	✓	✓	~	x	x	x
Minor regressions	✓	~	x	x	x	x
Documentation	✓	✓	~	x	x	x
Dev project rework	~	x	x	x	x	x

Release cycle and support - continued

- Green cells (✓) indicate a patch is eligible for that branch
- Yellow cells (~) indicate a patch will be considered on a case-by-case basis
- Red cells (×) indicate a patch is not eligible

T = Major release month	T+0 month	T+3 month	T+6 month	T+9 month	T+12 month	T+15 month
Security fixes	✓	✓	✓	✓	✓	✓
Daemon inoperable	✓	✓	✓	~	x	x
Major regressions	✓	✓	~	x	x	x
Minor regressions	✓	~	x	x	x	x
Documentation	✓	✓	~	x	x	x
Dev project rework	~	x	x	x	x	x

Build, Install and Upgrade

Build, Install and Upgrade

- A list of supported platforms can be found here:
  - https://slurm.schedmd.com/platforms.html

Build, Install and Upgrade

- Different Slurm functionalities and plugins depend on the availability of external libraries and system components versions
  - https://slurm.schedmd.com/quickstart_admin.html#build_install
- The configure script can be passed options for:
  - Fine-tune build/config/install directories
  - Soft-requirements (warnings logged to stderr)
  - Hard-requirements (errors logged to stderr) and configure exits
    - Basic link test against external libraries
  - Non-default paths to search for libraries / components
  - Enable/disable functionalities or modes of operation
- The result of the configuration can be inspected in config.log located in the BUILD dir
- Effort has been put in standardizing the configuration scripts and macros

Build, Install and Upgrade

- Configuration vs Runtime considerations
  - While pre-requisites might be met in the host where Slurm is configured, some other hosts in the cluster might need a subset or all of those pre-requisites also installed
  - While we try to keep adding checks at runtime before accessing external libraries initialization, it is highly recommended to double-check pre-requisites are installed in all the hosts that are needed
    - Slurm has various dlopen(), asserts and mechanisms for runtime checks
  - Divergence of installed libraries/components versions between the host where configuration is performed and runtime hosts is usually a source of pain
    - A recurrent example is GPU API libraries / runtimes

Build, Install and Upgrade

- Building/installing possibilities
  - Building RPMs
    - Pass configure switches via RPM macros
      - Command line
      - .rpmmacros file
  - Building Debian packages (Slurm >= 23.11.0)
    - Install dependencies
      - mk-build-deps -i debian/control
    - Packages conflict (and are named differently) than those shipped with Debian
  - Installing resulting .rpm/.deb packages
    - Generated packages oriented for different host purposes
      - Slurmdbd host(s)
      - Slurmctld host(s)
      - Slurmd host(s)
      - Login/client host(s)

Build, Install and Upgrade

- Building/installing possibilities
  - Building manually (preferred/recommended)
    - Download / clone Slurm
    - configure, make, make install...

Build, Install and Upgrade

- A personal preference is to have separate directories for:
  - Slurm downloaded / cloned "**project** files / bundle"
    - Optionally also version-specific
  - A directory where **build is performed** (i.e. from which configure, make, etc. are called)
    - This avoids mixing the project files with the result of the build process
    - For example, the "config.log" result of running configure script
  - As commented in previous Field Notes editions, a directory where files are **version-specific installed** (i.e. --prefix)
    - Create symlinks to current version components
    - Helps with rolling upgrades
  - Optionally create separate directory for configuration files (i.e. --sysconfdir)

Upgrade

- Please, refer to the (recently updated) Upgrade Guide and previous Field Notes editions for general information and recommendations for Upgrading
  - https://slurm.schedmd.com/upgrades.html
  - https://www.schedmd.com/publications

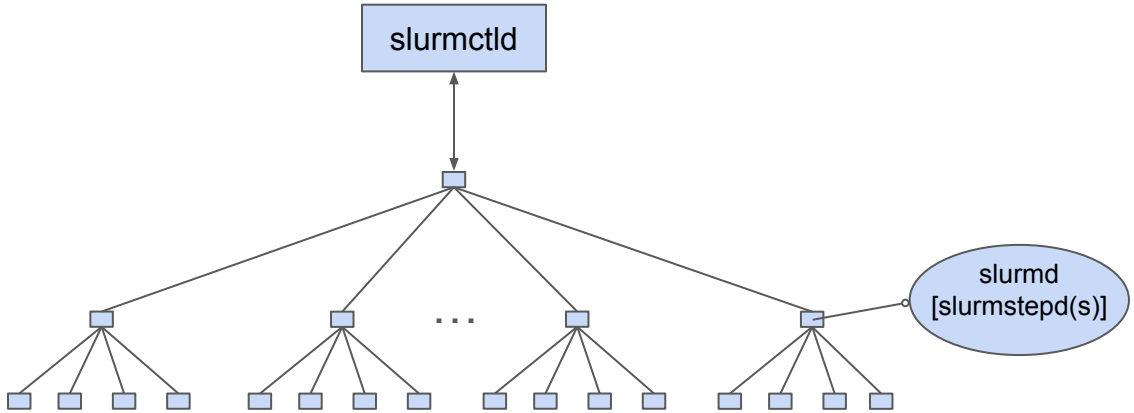
Upgrade

- In summary:
  - Comprehend the Release Cycle
  - Prepare cluster for the upgrade
    - Know changes in new versions
    - Plan downtime
  - Create backups
  - Slurm component versions must satisfy:
    - An upgrade order
    - A compatibility matrix defined by the Release Cycle stage
  - Downgrading is not supported

Dynamic Nodes

Node Communications

- Hierarchical communication with configurable fanout
  - Better efficiency for large clusters and large MPI jobs
  - Less overhead on the controller
    - Ping, reconfigure, reboot, step launch, etc.



Configurable virtual network for communications
(TreeWidth / TopologyParam)

Node Communications

- Clients/daemons rely on slurm.conf for how to communicate with slurmds
- This is one reason why the slurm.conf needs to be synced across all nodes

Static Nodes

- Traditional
  - Known system size
  - Define all nodes in slurm.conf
  - Even for cloud nodes

```
NodeName=node[1-100] CPUs=16 Sockets=1 CoresPerSocket=8 ThreadsPerCore=2 RealMemory=31848

PartitionName=debug Nodes=node[1-100] Default=Yes
```

Adding Static Nodes

- Avoid issues communicating to nodes that didn't exist in the slurm.conf
  - e.g. Add node to controller but don't restart slurmds
- Recommended process for adding a node
  - Stop slurmctld(s)
  - Update the slurm.conf on all nodes in the cluster
  - Restart slurmd on all nodes in the cluster
  - Start slurmctld(s)

Dynamic Nodes

- Functionality that permits nodes to be added/deleted from system without adding them into slurm.conf, restarting slurmctld or slurmd
- Use Cases
  - Multiple dynamic clusters, where nodes are added/removed frequently
  - Temporary addition of a new node(s)
  - Cloud services adding/removing nodes

Dynamic Nodes Communications

- Prior to 23.11, the fanout had to be disabled
  - Direct communication from controller to clients to each node
- 23.11, fanout works with dynamic and non-addressable nodes
  - On dynamic node registration, node addresses are grabbed by and passed from the controller to the clients/daemons and forwarded on.

Dynamic Nodes

- Full Dynamic Nodes guidelines:
  - https://slurm.schedmd.com/dynamic_nodes.html
  - https://www.schedmd.com/publications/

Internal Authentication

Authentication - auth/slurm + cred/slurm

- Available since Slurm 23.11
- Internal authentication and job credential plugins
  - Alternative to MUNGE
  - Builds off existing capabilities - unix socket authentication through SO_PEERCRED (used by slurmstepd to authenticate RPCs), plus auth/jwt authentication plugin
- Simple HMAC scheme (SHA-256) built off JWT
  - Separate from existing auth/jwt plugin
  - Will require a shared (locked down) key throughout the cluster
    - `/etc/slurm/slurm.key`
    - Similar security posture to MUNGE
- Client commands use a local socket, automatically managed by slurmctld / slurmdbd / slurmd, or new sackd daemon on the login node
- Will allow for future extension and flexibility...

Authentication - auth/slurm + cred/slurm

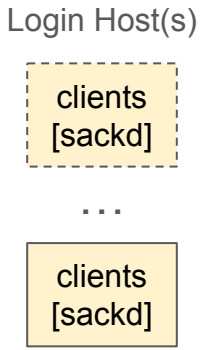
- Beginning with version 24.05, you may alternatively create a slurm.jwks file with multiple keys defined
- The slurm.jwks file aids with key rotation, as the cluster does not need to be restarted at once when a key is rotated. Instead, an scontrol reconfigure is sufficient

LDAP-less control plane

- Support running the slurmctld without LDAP
  - Optional capability enabled through auth/slurm's credential format extensibility
  - Username, uid, gid, groups will be captured alongside the job submission
  - auth/slurm permits the login node to securely provide these details, which auth/munge cannot due to protocol limitations
  - Set AuthInfo=use_client_ids in slurm.conf and slurmdbd.conf to enable

New login node process - sackd

- For sites running auth/slurm, a new daemon - sackd - provides authentication for client commands
- This daemon can also integrate into a "configless" environment, and manage the locally cached set of configuration files for the login node
  - Updates will be received automatically through "scontrol reconfigure"
    - Similar mechanism already exists to update slurmd processes



auth/slurm

- auth/slurm plugin actually behaves as both an auth and cred plugin
- CredType=slurm, CredType=cred/slurm, CredType=auth/slurm all work fine

auth/slurm

- Daemons internally manage authentication
  - Don't connect out externally any longer
  - Faster for slurmctld than MUNGE by removing the socket overhead

SACK

- [S]lurm's [a]uth and [c]red [k]iosk
- Listens on a UNIX socket for connections from client commands
  - Authenticates them using SO_PEERCRED
  - Two API calls supported:
    - Create and sends back a token
    - Validate an existing token

auth/slurm

- Only need one daemon on each host:
  - slurmctld -> /run/slurmctld/sack.socket
  - slurmdbd -> /run/slurmdbd/sack.socket
  - slurmd -> /run/slurm/sack.socket
  - sackd -> /run/slurm/sack.socket
- Runtime directories expected to be created/handled by systemd
  - RuntimeDirectory[Mode] option for each daemon service file

Monitoring Insight

Monitoring

- Monitoring services is in general highly recommended
  - Triggers can help catching and notifying some events within slurmctld
- Monitoring the health of your compute nodes is also recommended
  - HealthCheckInterval
  - HealthCheckNodeState
  - HealthCheckProgram
- As well as configuring diagnostics on unkillable steps
  - UnkillableStepTimeout
  - UnkillableStepProgram

Monitoring - Continued

- One possible useful approach for UnkillableStepProgram diagnostics:
  - Retrieve information on cgroup hierarchy and pids being tracked
  - Retrieve information on stuck processes:
    - Pid
    - Parent pid
    - State
    - User
    - WCHAN (name of the kernel function in which the process is sleeping)
    - Process open files
    - Process strace
  - Retrieve information on the freezer cgroup contents and hierarchy

Monitoring - UnkillableStepProgram example

- Perform a find over Slurm cgroup hierarchy to retrieve pids for this job:
  - find /sys/fs/cgroup/cpuset/slurm*/uid_*/job_${SLURM_JOB_ID}
  - cgroup.procs cgroup interface
- And for each pid, log to a file the result of:
  - ps -o pid,ppid,stat,user:20,wchan:30,command $pid
  - lsof -p $pid
  - timeout -s 9 10 strace -p $pid
- Also log tree contents of freezer cgroup:
  - find /sys/fs/cgroup/freezer/slurm/ -iname "cgroup.procs"
  - Log file find hits and contents
- Recursive cgroup hierarchy list contents
  - ls -R /sys/fs/cgroup/freezer/slurm/ -latrh

Cgroup Insight

Cgroup Insight

- Cgroup (“Control Group”) is a Linux kernel feature that limits, accounts for, and isolates the resource usage (CPU, memory, devices, etc.) of a collection of processes
- Slurm provides support for:
  - cgroup/v1 (aka Legacy mode)
  - cgroup/v2 (aka Unified mode, since 22.05)
    - See the web documentation for requirements
  - Hybrid mode where controllers from both v1 and v2 are mixed in the same compute node is not supported

Cgroup Insight

- Slurm mainly uses cgroup for:
  - Process tracking and management
    - Via proctrack/cgroup plugin
  - Constraining/monitoring resources at step and task level
    - Via task/cgroup plugin
  - Gather statistics
    - Via jobacct_gather/cgroup plugin
  - Resource specialization
    - Reserve CPUs / memory for daemons/system use (thus not available as allocatable for user jobs)

Cgroup Insight

- As of today, Slurm uses these cgroup controllers:
  - cpuacct, cpuset, devices, freezer, memory

Cgroup Insight

- To ensure correct interaction with cgroup functionality, **Slurm must be the sole writer on its managed cgroup hierarchy**

Cgroup Insight

- Problem 1:
  - systemd has a default *single-writer* design on the whole system cgroup hierarchy (typically /sys/fs/cgroup)
  - This is can be a source of pain and Slurm cgroup interaction disruption
    - I.e. systemd moving pids around or enabling/disabling cgroup controllers
- Solution:
  - Ensure slurmd processes are started via systemd with the option Delegate=yes set in the service file

Cgroup Insight

- While we continuously fix Slurm bugs and work on hardening the code ...
  - ... A decent amount of times unexpected Slurm-cgroup behavior is also caused by privileged external software / processes also modifying the Slurm cgroup hierarchy
- We've proof of recurrent issues due to the external software doing things like:
  - Having a per-job metrics component that:
    - Creates/removes per-jobs cgroups
    - Putting processes into them
  - Or setting the cpuset.cpu_exclusive to exclusively dedicate cores outside Slurm awareness
- Solution:
  - It's mainly admins responsibility to ensure that this doesn't happen

Cgroup Insight

- We vastly prefer identifying and potentially consider new RFEs than having other software do so at the cost of messing with Slurm's cgroup sub-hierarchy
- Proving to our customers that the external software is the actual cause of problems is sometimes not that straightforward …
    - … To the point that we've even been internally developing and testing a pilot tool that could help diagnosing who is modifying the cgroup hierarchy

Commercial Support

Commercial Support

- SchedMD offers “Level-3” support only
- Customers must be comfortable with day-to-day operations, basic troubleshooting, and initial setup
- On-Site training is available to assist with initial onboarding and software configuration
    - Email sales@schedmd.com

Commercial Support

- Our workflow is built around Bugzilla, allowing transparent hand off between support engineers, and thorough tracking
  - Tickets are public by default, however they can be marked private when required
  - https://support.schedmd.com/

Commercial Support

- Specific contractual timelines for each Severity level
- Support hours are 2 AM to 5 PM Mountain Time / 8:00AM - 11:00PM GMT, Monday - Friday
- We strive to always significantly exceed the SLA, regardless of system size or complexity

Dinner

Dinner, 6:30pm at Olympen Bar & Restaurant

- Possible transport option from University of Oslo
  - Get into Metro Blindern stop
    - Take T4 (Blue) / T5 (Green)
  - Get out at Grønland stop
    - Olympen at 240m distance

Questions


SCHEDMD
The Slurm Company