Service Level and Availability Refactoring

HPC Multi-cluster Data Base Deployment

Steven Senator

Approved for release
Nov 2022

Outline

- Faults, Availability, Services and Scheduler Data
  - Vocabulary: Faults, Errors and Failures
  - Services Availability Mitigations
    - Scheduler services
    - Planned, Unplanned and Scheduler service availability, 2017-2022
    - Mitigations
  - LANL HPC and Scheduler Component Architecture
    - LANL HPC
    - TLCC, CTS, Cray scheduler components
    - Failure mode
  - Combined Multicluster Slurm DB project
    - Background
    - Implementation
    - Failure modes
- Learnings

Context

- Background
  - Formative portion of my career at Tandem Computers

  - Select customers: critical availability requirements

NYSE

  - Service Unavailability
    - A single system or service, must be multiplied by user community size and its impact

      Reputation, Visibility, Monetary, Risk

  - Reliable services can and must be constructed from unreliable components, but may constructed of combinations of
    - Unreliable components
    - Algorithms in code
    - Human processes (algorithms of people's procedures)

HIGH PERFORMANCE COMPUTING

Vocabulary

- **State**
  - **Fault** The underlying system **is** erroneous.
  - **Error** The system **deviates** from its specification with respect to a service.
  - **Failure** The delivered service is **observed** to deviate from the service specification.

- Action
  - Activation A fault transitions to an error. The service is no longer within specification.
  - Propagation An error transitions to a failure. It becomes observable.

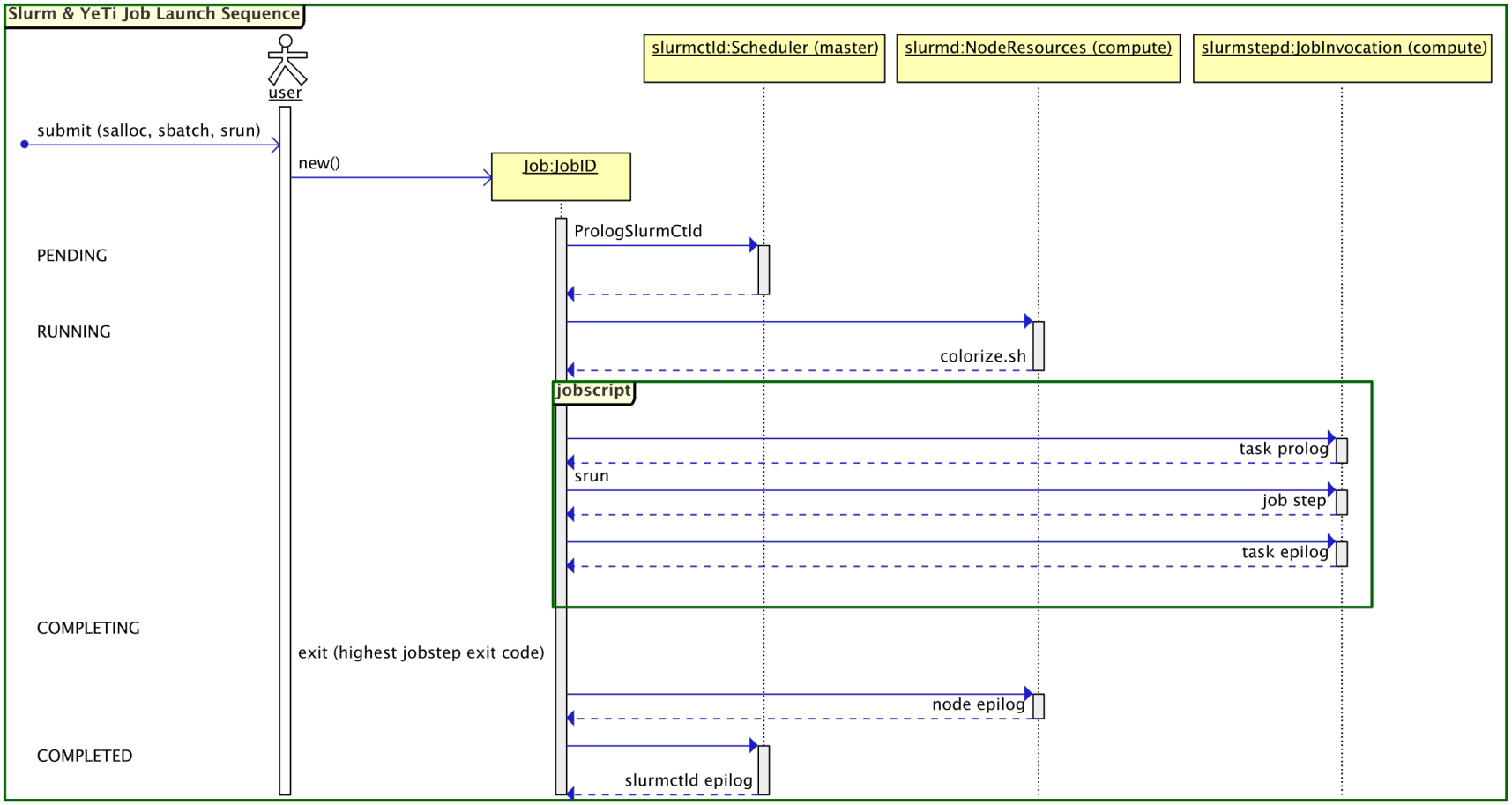
- *Duration*
  - *Fault Latency* The time between a fault's manifestation and its activation.
  - *Error Latency* The time between a fault's activation and its detection.

From: Debardeleben, Daly et. al., LANL Resilience Workshop, 2009

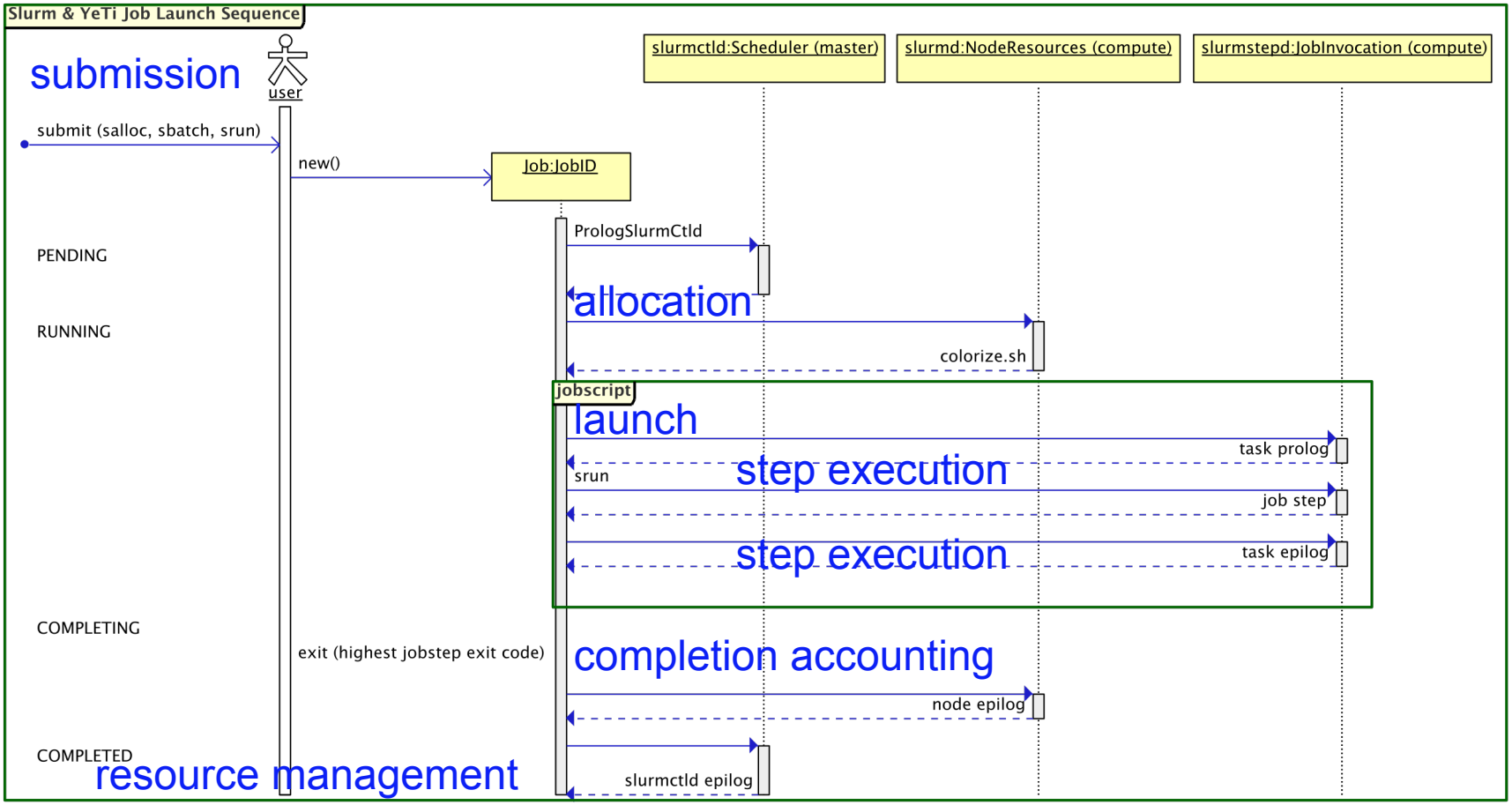
Scheduler Service definition

- Slurm Scheduler and Controller services
  - Job Submission → Job Enqueue
  - Resource Allocation
  - Job Launch
  - Job Step Execution
  - Job Completion Accounting
  - Resource Management
    - Nodes, Licenses, GPUs, Storage, Credentials
  - Querying and Reporting
    - Job, Queues, Job History, Accounts, Usage
- Slurm Data Base Daemon service specifications
  - Account management
    - C reate
    - R ead
    - U pdate
    - D elete
- Each use case relies on a different service subset.

Job Timeline



Job Timeline with Slurm Scheduler Services



Not shown: CRUD entity state queries ie. node, partition, reservation, job

Tasks in blue generate data base records.

Service Availability: 2017-2022

- TLCC and CTS Service Failure definition ➔ whole cluster unavailable
- Service Outages:
  - Planned Unavailability:
    - `# clusters x 12-24 planned outages/year x 1-5 days/DST`
  - Unplanned Unavailability (power failures or environment issues):
    - `# clusters x # power failures x recovery duration`
  - Unplanned unavailability (scheduler outages): **8** incidents / **5+** years

Count	Date	Type	Notes
1	2019	ENOSPC	gr-master
3	2017-2019	ENOSPC	tr-drm
1	2019	slurm upgrade	
1	2019	RPM dependency	
1	2020	accounting roll-up	slurmctld
1	2020	lengthy schema update	slurmdbd

Mitigations

1. Resources
   - more storage, more and redundant RAM, redundant network connectivity
2. Process improvement
   - Automated build and upgrade process
   - Vendor & TriLab peer coordination
   - Active and anticipatory delta tracking
   - Vendor support contracts
   - Active and frequent monitoring and alerting
   - DB replication and backups
   - Automated pre-production testing
3. Software
   - Software vendor fixed the bug which triggered roll-up error
   - Enhancement, generated directly by summer student project results

     max_dbd_msg_action=exit ➔ fail-fast, to surface the failure
4. Configuration
   - Increased storage for in-cluster DBD message spool, tuned to cluster role

Combined Multicluster Slurm Data Base Project

Enhanced Features Prerequisite

- Multicluster Accessibility
  - Users may view another cluster's queue
- Multicluster Management Queries, Comparison, Reporting, Analysis
- Future capabilities:
  - User job control from any front-end to any set of back-end nodes
  - Project Leaders, Program Management on-demand queries
  - System Administrators, HPC Management dynamic analysis, alerts, comparisons
  - Users and Project Leaders self-tuning job workflows

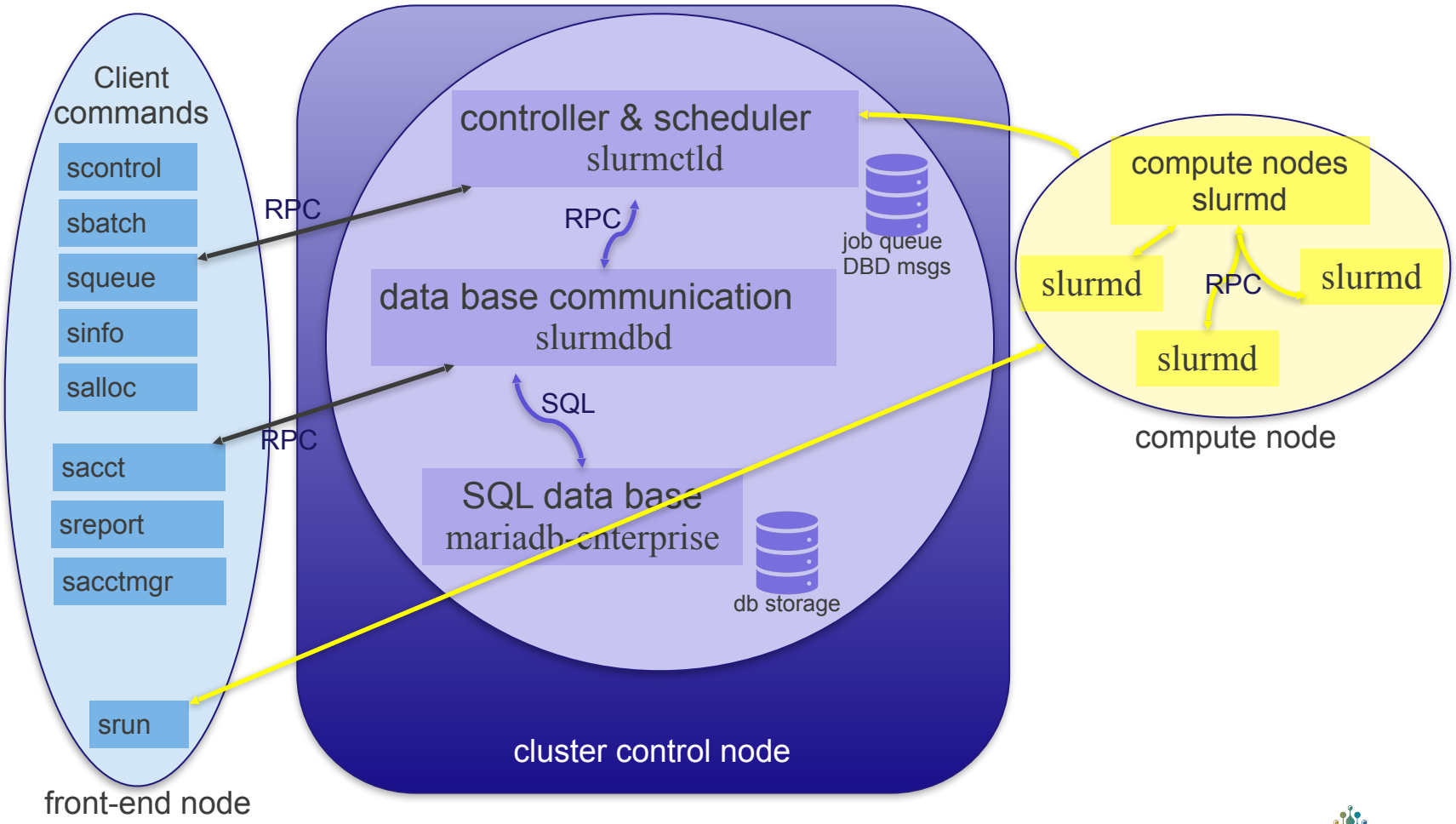
Service Separation, Factorization and Signatures

- Data base service is independent of compute service
- Enable HPC Administrators, Monitoring, Cybersecurity, Acceptance to focus on service-specific improvements:
  - Resilience, Instrumentation, Oversight, Tuning, Scalability
  - Signatures, Analysis, Visualization

Slurm Component Architecture

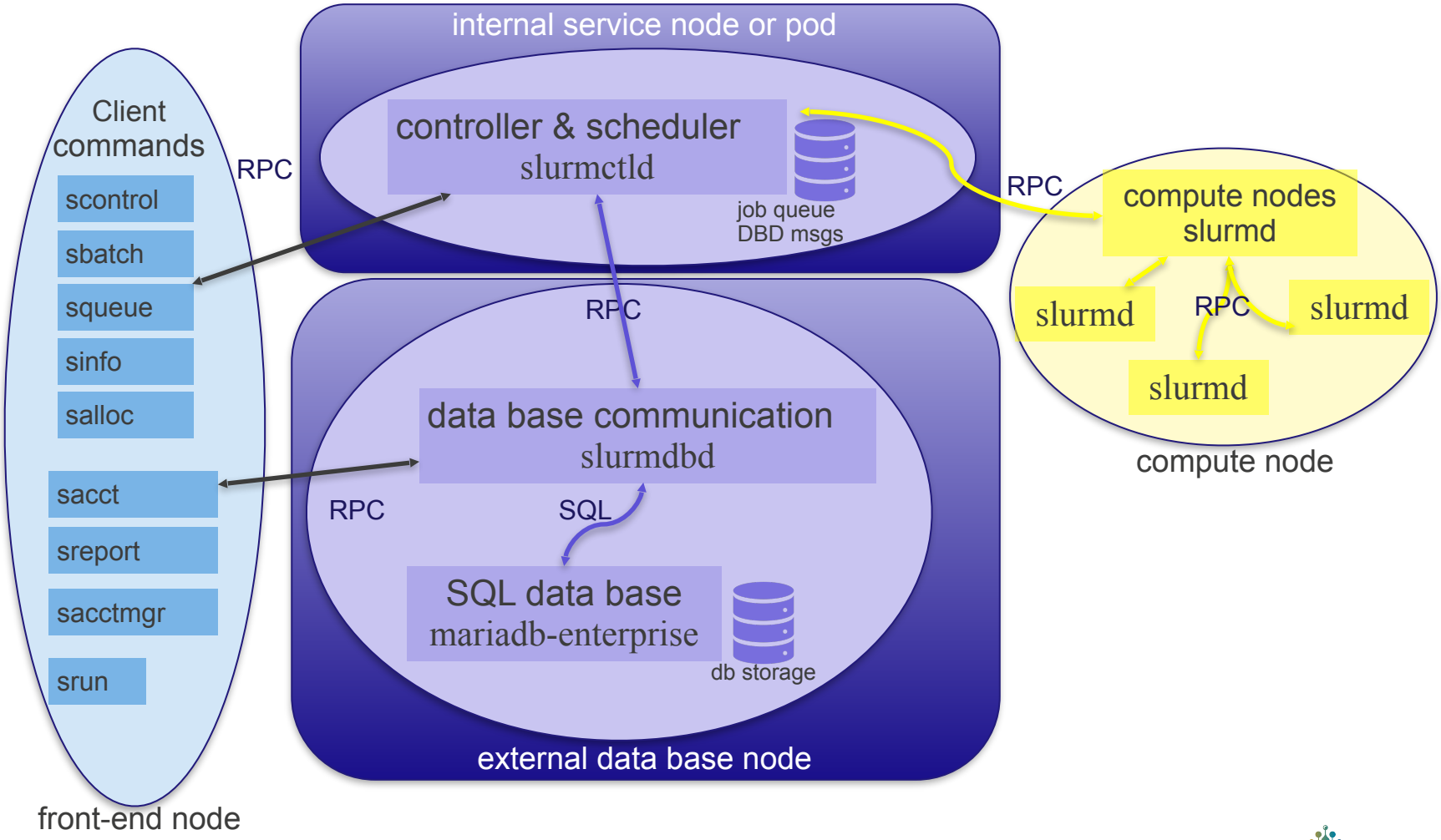
LANL 2017-present

Slurm Component Architecture TLCC CTS/Capacity



Based on: https://docs.schedmd.com/quickstart.html

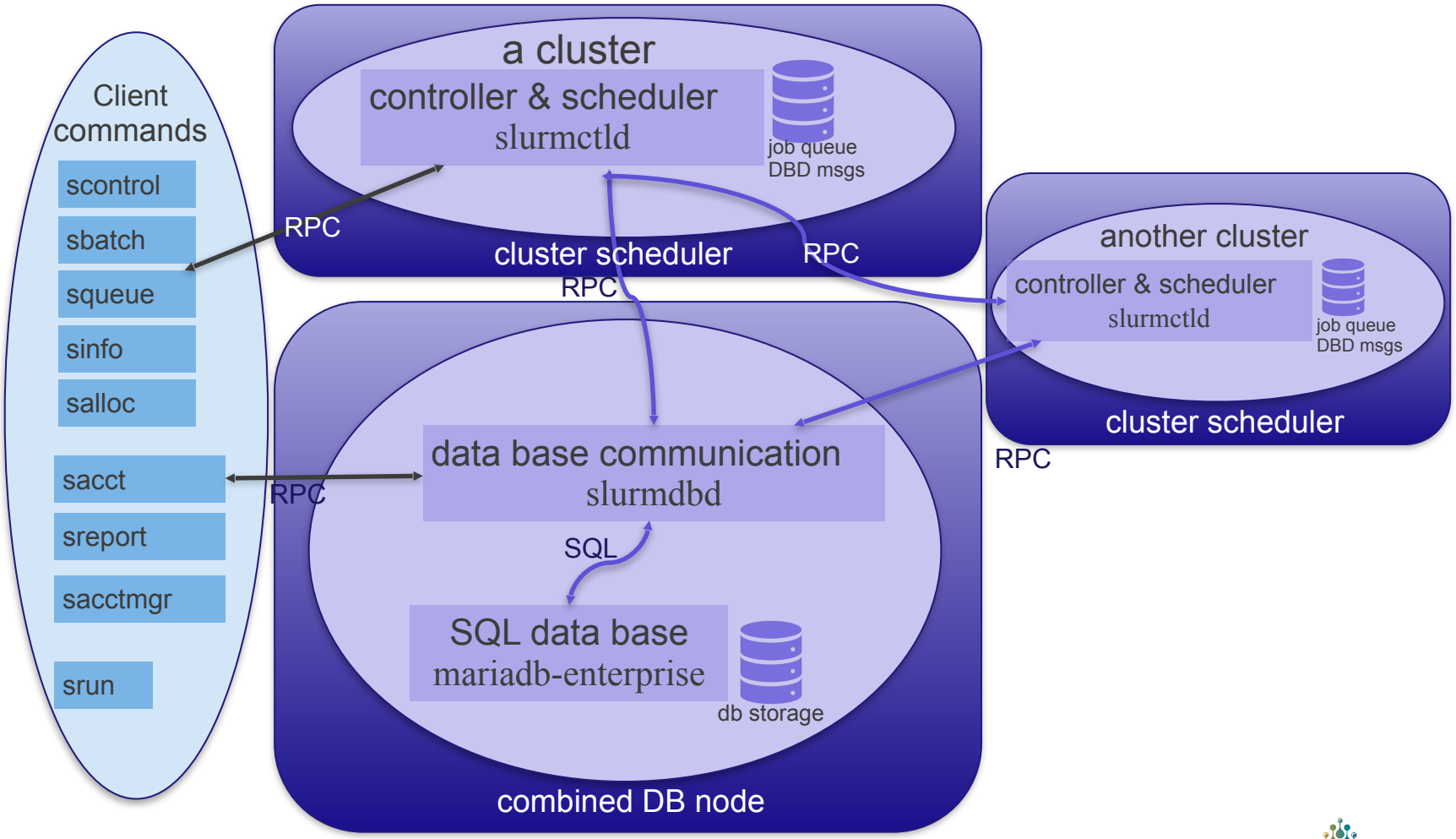
Slurm Component Architecture Cray/Capability



Based on: https://docs.schedmd.com/quickstart.html

Component Architecture

Multi-cluster combined DB



Based on: https://docs.schedmd.com/quickstart.html

Combined Slurm Data Base Project Learnings

1. Service refactoring has demonstrable, immediate and multiple benefits
   - Removal of single points of failures
   - Severable Implementation: independent upgrades especially
   - Single points of failure are now defined by quantified services.
   - Failure footprints removed, mitigated or reduced.

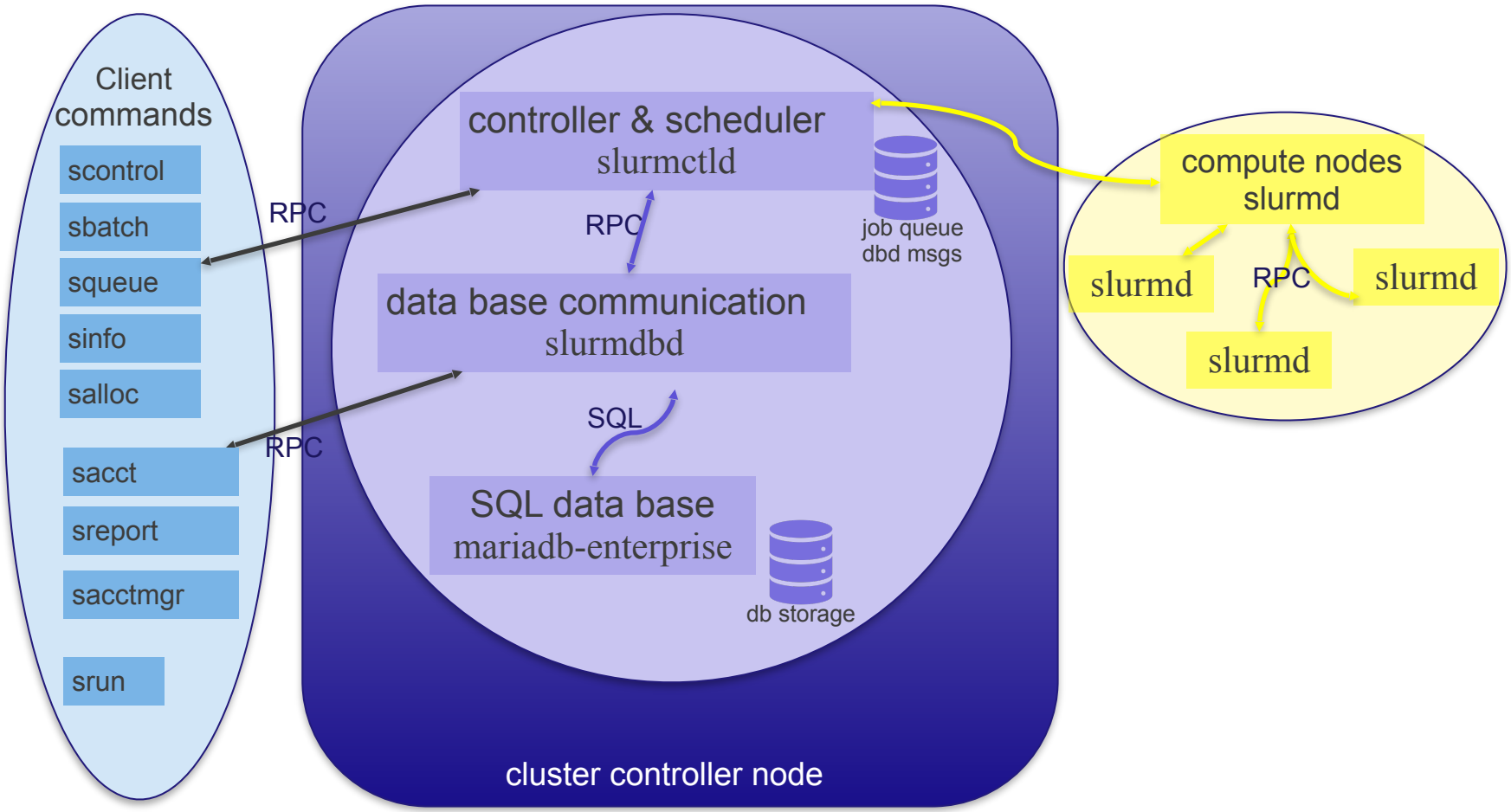
2. Validation testing ensures service level agreement compliance.

3. Alerting ensures service level agreement process compliance.

4. Availability and scheduling data identify HPC opportunities:
   - Reductions in DST outages, in frequency and size,
     yield significant increase in delivered cpu hours, on the order of a component refresh.
   - Power reliability

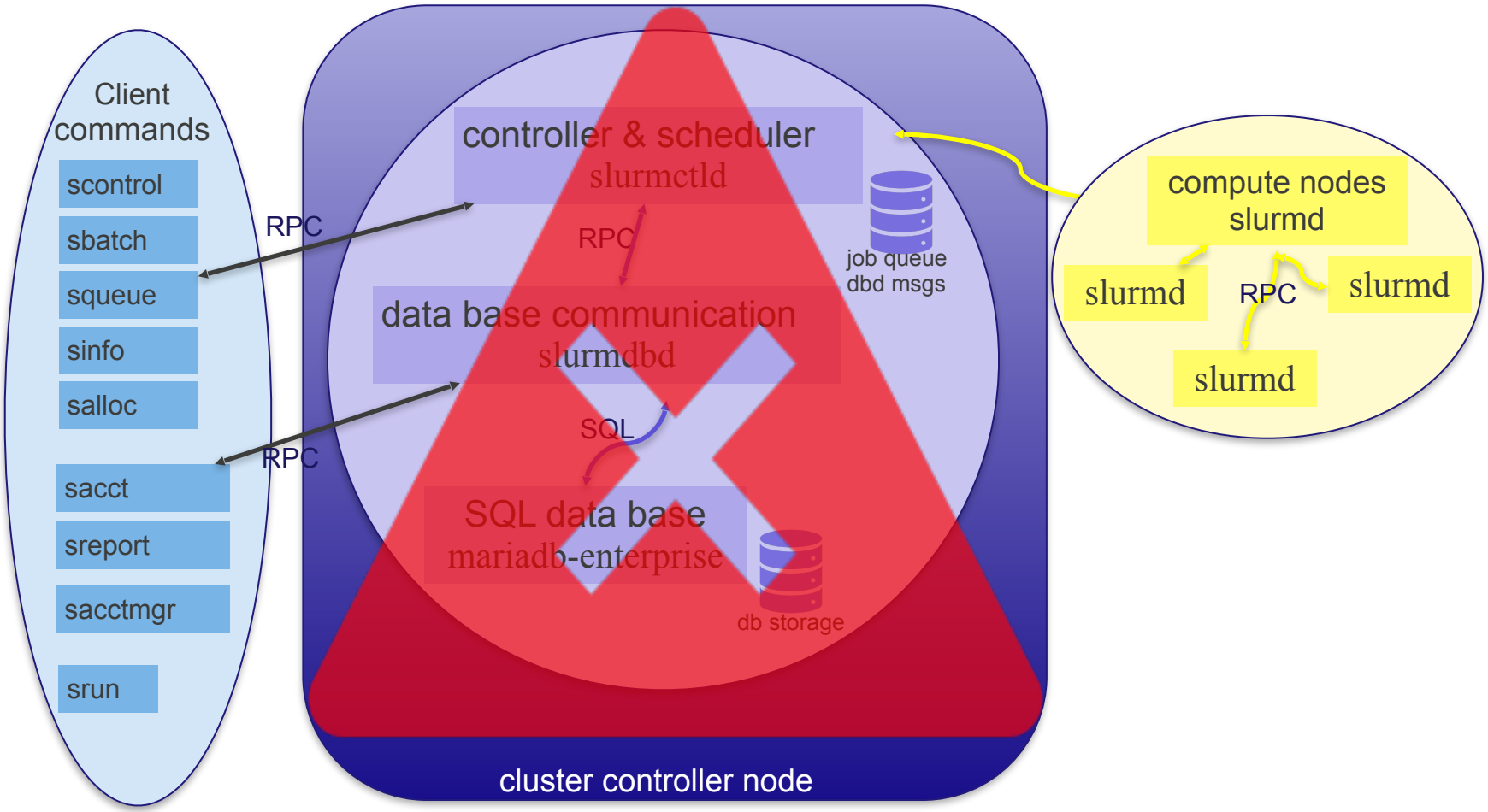
5. Analysis of data does and will show additional learnings.

Combined Slurm Data Base Project TLCC CTS



Based on: https://docs.schedmd.com/quickstart.html

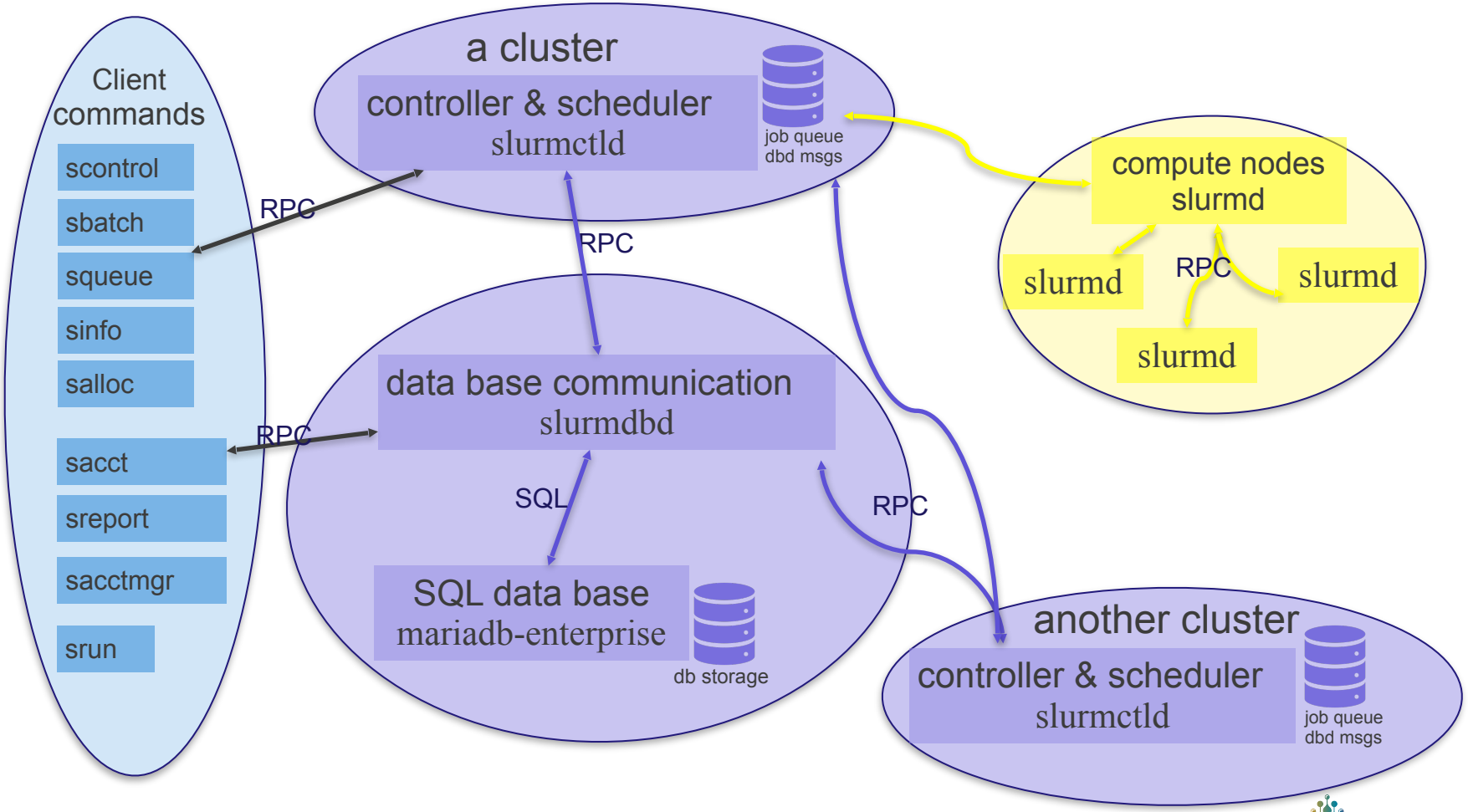
Combined Slurm Data Base Project TLCC CTS



Based on: https://docs.schedmd.com/quickstart.html

Combined Slurm Data Base Project Differences: 1

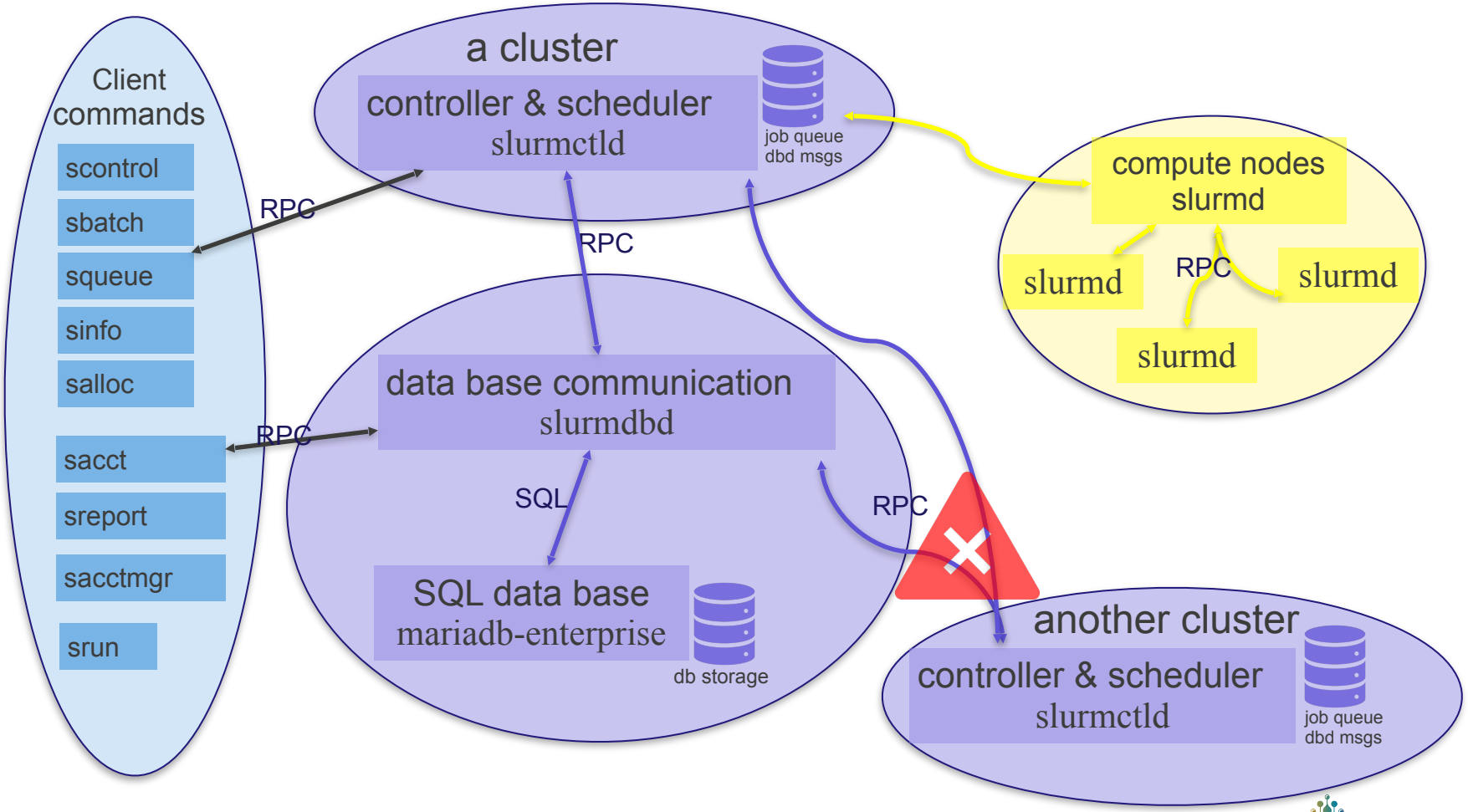
Slurm Components: multicluster (failure: service unreachable)



Based on: https://docs.schedmd.com/quickstart.html

Combined Slurm Data Base Project Differences: 2

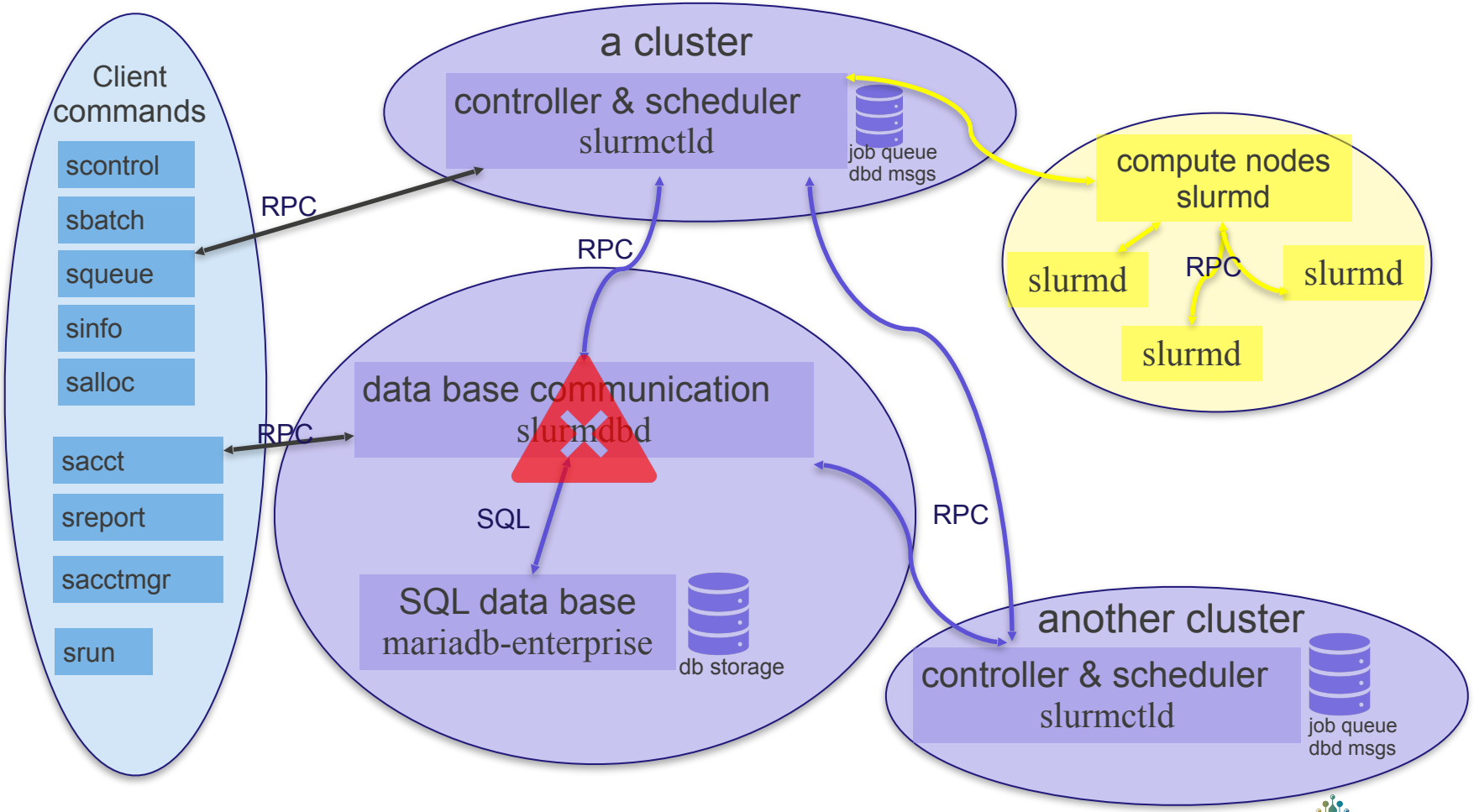
Slurm Components: multicluster (failure: service unreachable)



Based on: https://docs.schedmd.com/quickstart.html

Combined Slurm Data Base Project Differences: 3

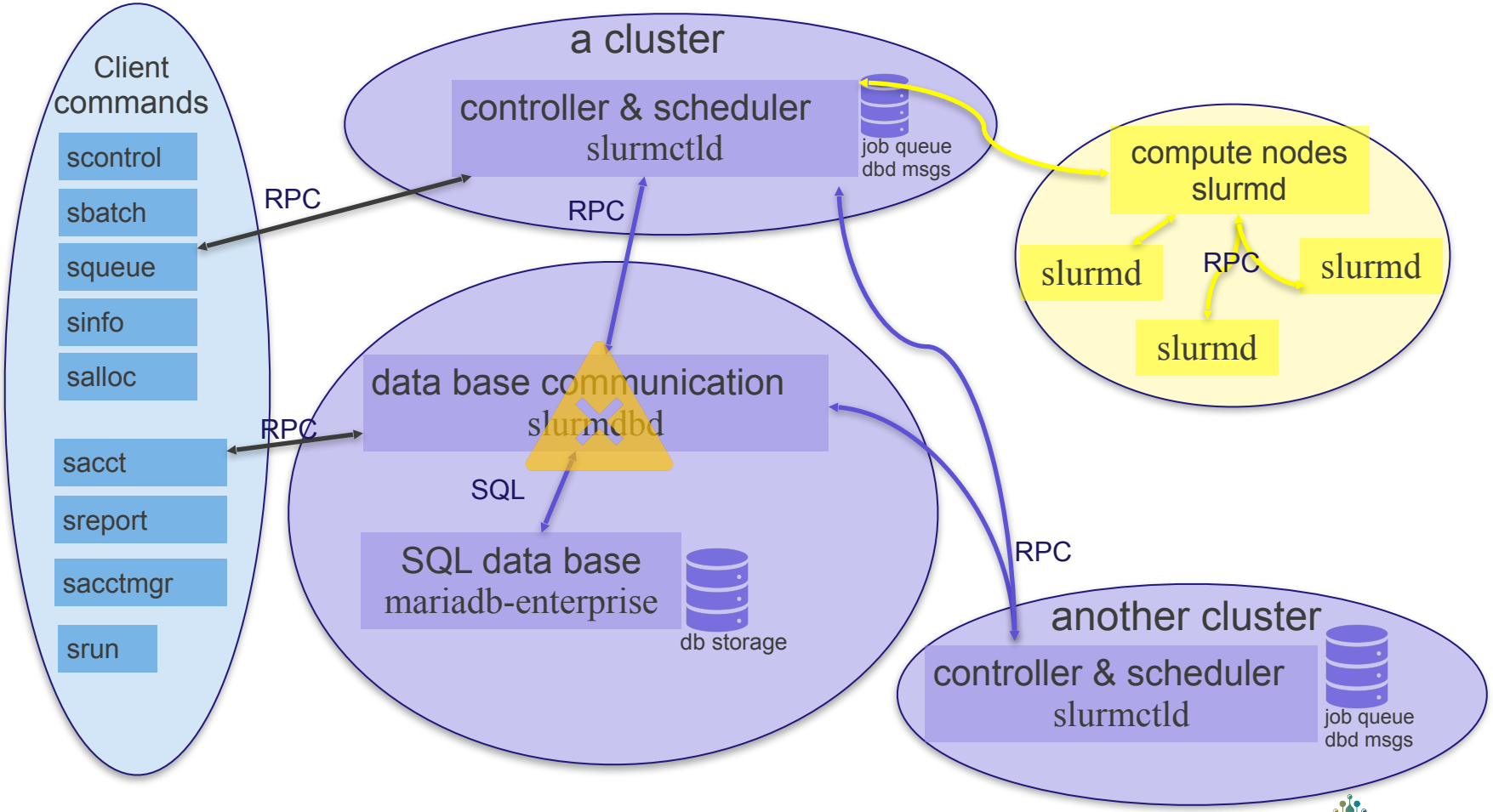
Slurm Components: multicluster (failure: service outage)



Based on: https://docs.schedmd.com/quickstart.html

Combined Slurm Data Base Project Differences: 4

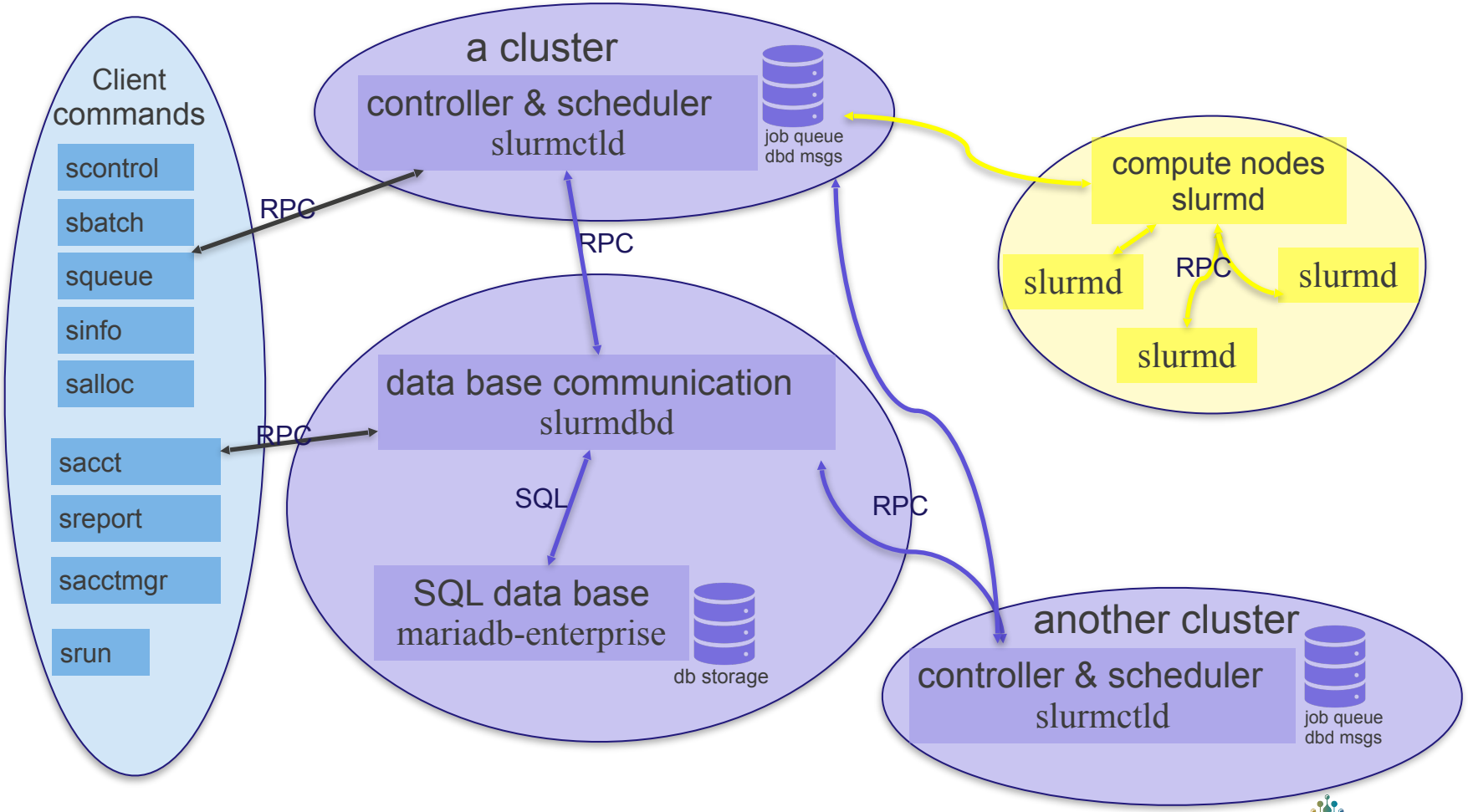
Slurm Components: multicluster (failure: service degradation)



Based on: https://docs.schedmd.com/quickstart.html

Combined Slurm Data Base Project Services++

Slurm Components: multicluster (failure: service unreachable)

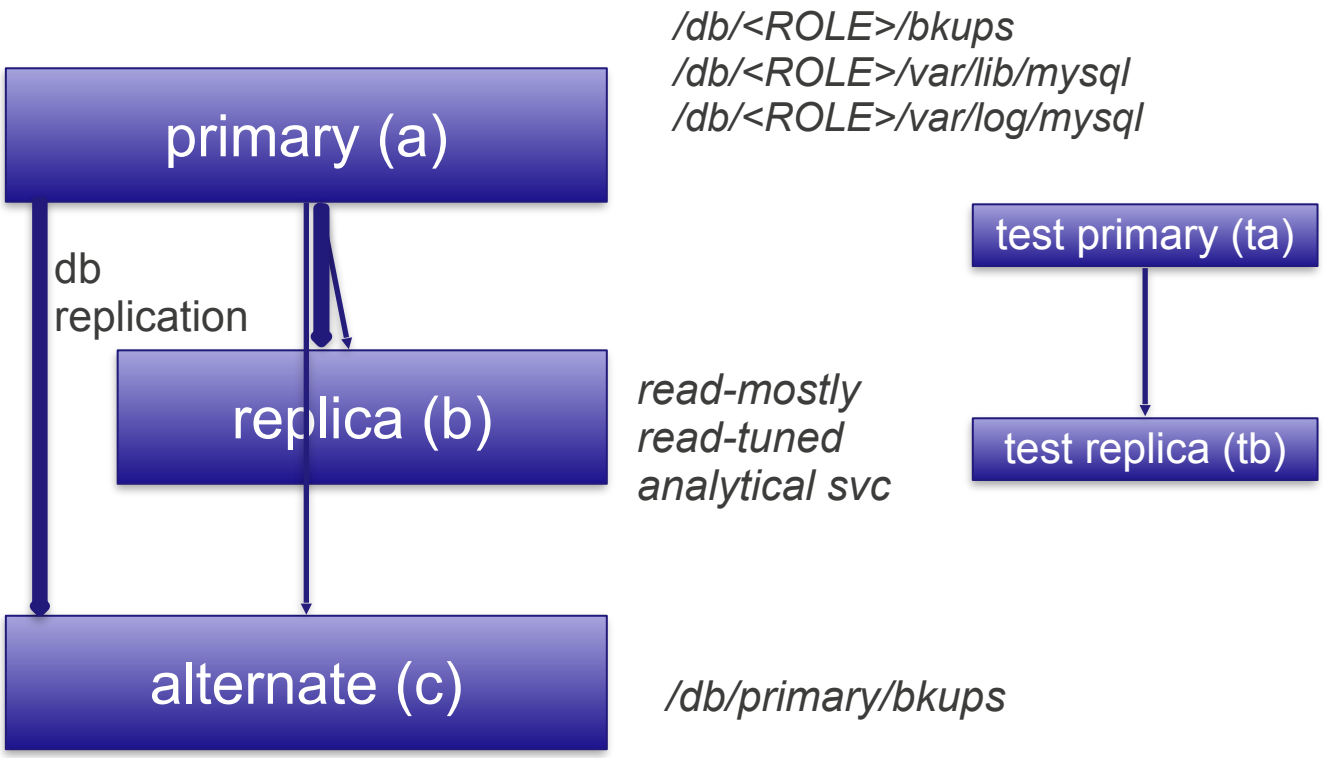


Service Level Specification: maximum 5 minute response

Based on: https://docs.schedmd.com/quickstart.html

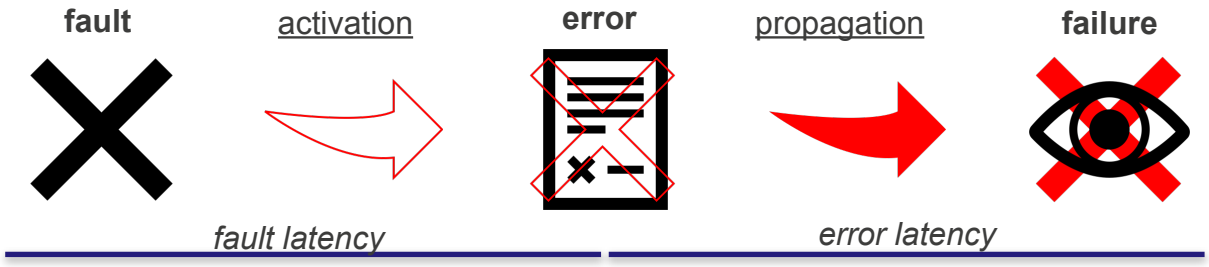
Combined Slurm Data Base Project How (servers)?

DB server nodes: replication, backups and preproduction testing



Combined Slurm Data Base Project Surfacing Errors

- Errors are surfaced and detectable when they propagate to a failure.



- Active and automated *testing, monitoring and alerting* surface the errors.
  - Validation tests run every 5 minutes on all three slurmdb production servers {a,b,c}
    - Slurmdb's role-specific tests verify all of the links between the prior list of services.
    - A variation of these tests will be used in tech-ops dashboards.
    - Quick lightweight tests are used.

Human attention to, analysis of, and remediation of meaningful alerts

- Pre-production {ta, tb} changes are verified with an automated dual-cluster test verification environment which use the identical set of tests.

From: Debardeleben, Daly et. al., LANL Resilience Workshop, 2009

Combined Slurm Data Base Project Errors & Failures

- Scheduling Service Faults, Errors, Failures and Mitigations

Component	Manifest Failure	Repair	Repair Time (min.)	Human	Type
NIC	interface errors	replace	0	no	error, ! failure
slurmdbd	spool growth	diagnose	0? < 5	yes	error, ! failure
data base	accounting CRUD, new acct job submit	diagnose, restore	~5	yes	failure
node	*as above*	alternate	~5	yes	failure
data center	*as above*	kickstart	~45	yes	failure

Combined Slurm Data Base Project Learnings

- Service refactoring has demonstrable, immediate and multiple benefits
  - Removal of single points of failures
  - Severable Implementation: independent upgrades especially
- Validation testing ensures service level agreement compliance.
  - *Lack ensures non-compliance*.
- Reliability of data log stream is critical.
- Alerting ensures service level agreement process compliance.
- Availability and scheduling data identify HPC opportunities:
  - Reductions in DST outages, themselves possible due to service refactoring, yield significant increase in delivered cpu hours, on the order of a component refresh.
  - Power reliability
- Analysis of data does and will show additional learnings.

Combined Slurm Data Base Project MariaDB Learnings

- **Benefits**
  - **Demonstrable, Immediate, Quantifiable**
  - **Refactor Services**
    - Fractally useful. Relevant at all scales.

      physical node, software service, application or intra-application
  - **Open-Source Community Good Citizenship**
- This project would not have been justifiable without vendor-supported open-source vetted code.
- This project would not have been quantifiable without incorporation of community-contributed features.
  - Vendor-supported => oversight compliance possible
  - Open-source => ensures adherence to our long-term responsibilities; incorporates best practices
  - Vetted => risk assessment possible

MariaDB

HIGH PERFORMANCE COMPUTING

Combined Slurm Data Base Project Thank you

Thank you to those who took the time to provide feedback.

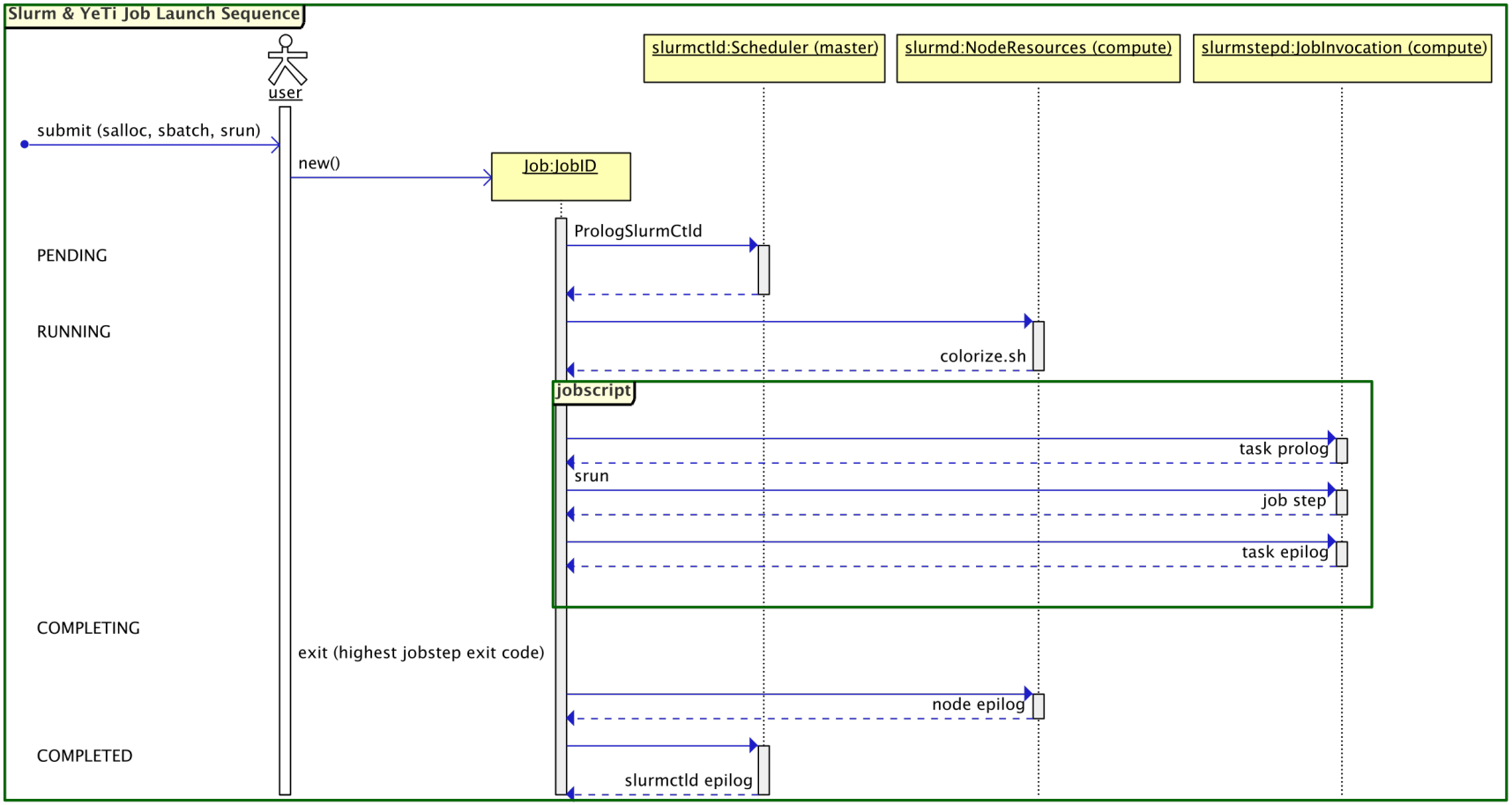
You have improved this work.

Over 70 years at the forefront of supercomputing

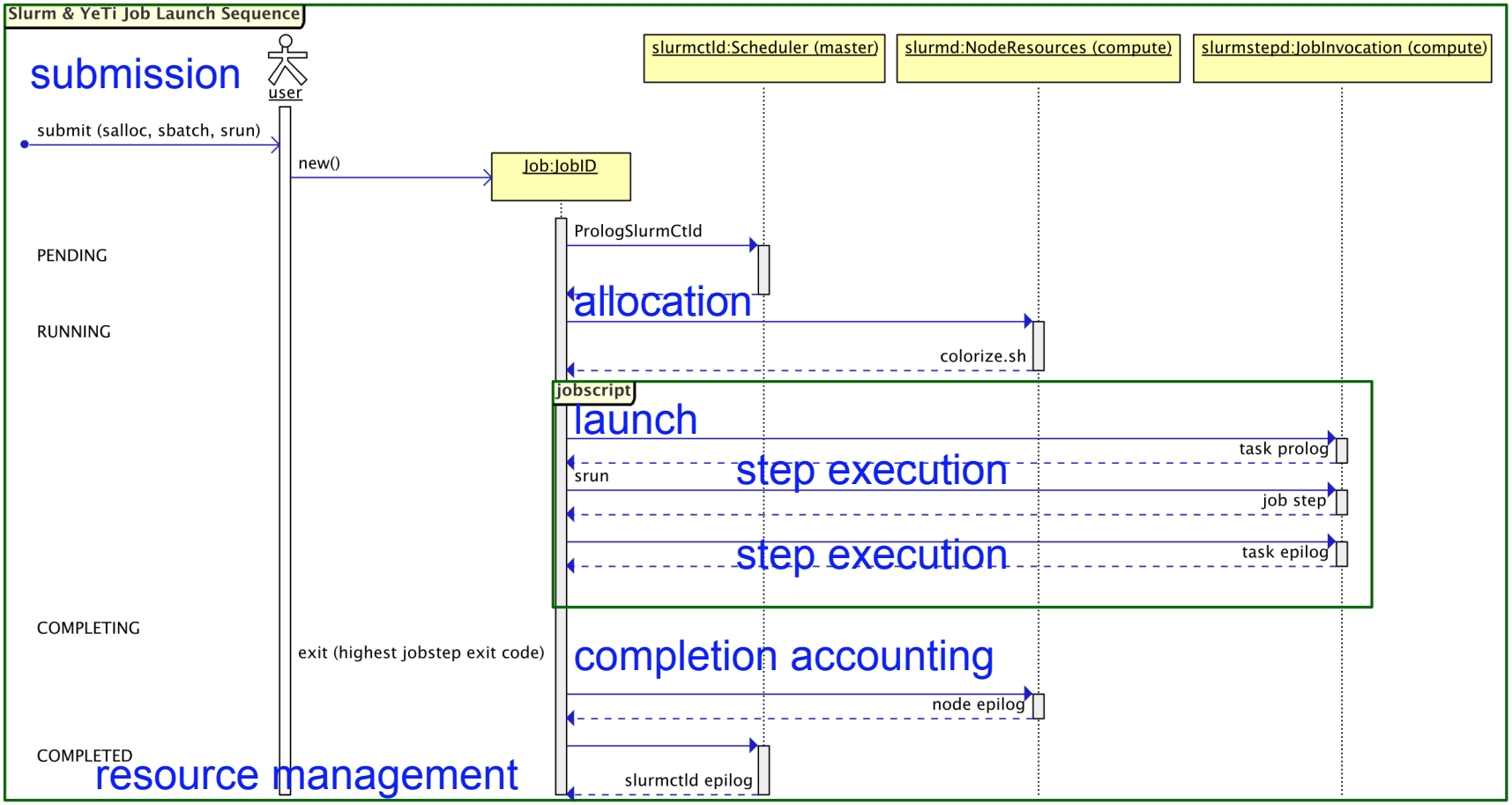
addenda

Job Lifecycle

Job Timeline



Job Timeline with Slurm Scheduler Services



Not shown: state queries ex. “sinfo”, “scontrol show [node | partition | reservation | job]”

User view and use cases

User view and use cases

Enhanced Features Prerequisite

- Multicluster Accessibility

```
badger$  squeue  -M grizzly   -t R
ice$  sbatch       -M ice,fire   myjob.sh
fire$  salloc      -M all  -time=30:00  -reservation=debug
```

- Mullticluster Management Queries, Comparison, Reporting, Analysis

```
grizzly$ sreport   -all_clusters  -tres=cpu,gpu
ice$   sreport       -M all   cluster AccountUtilizationByUser
fire$  sreport       -M all   reservation Utilization
```

- Containerized personal front-end image (notional)

```
pn12345-hpc-frontend$  sbatch  -M all  myjobscript.sh
```

Service Separation

- Service-specific configuration: storage, connectivity, tuning, resilience

User view and benefits

Enhanced Features Prerequisite

- Multicluster Accessibility

```
badger$ squeue -M grizzly -t R
ice$ sbatch -M ice,fire myjob.sh
fire$ salloc -M all -time=30:00 -reservation=debug
```

Benefit => Users

- Mullticluster Management Queries, Comparison, Reporting, Analysis

```
grizzly$ sreport -all_clusters -tres=cpu,gpu
ice$ sreport -M all cluster AccountUtilizationByUser
fire$ sreport -M all reservation Utilization
```

Benefit => Principal Investigators, Project Leaders, Program Management

- Containerized personal front-end image

```
pn12345-hpc-frontend$ sbatch -M all myjobscript.sh
```

Benefit => Users

Service Separation & Factorization

- Service-specific configuration: storage, connectivity, tuning, resilience

Benefit => System Administrators, Tech. Ops., Monitoring, HPC Acquisition, Security

Combined Slurm Data Base Project User view

Enhanced Features Prerequisite

- Multicluster Accessibility

```
badger$  squeue  -M grizzly   -t R
ice$     sbatch  -M ice,fire   myjob.sh
fire$    salloc  -M all        --time=30:00 --reservation=debug
```

- Mullticluster Management Queries, Comparison, Reporting, Analysis

```
grizzly$ sreport -all_clusters --tres=cpu,gpu
ice$     sreport -M all        cluster AccountUtilizationByUser
fire$    sreport -M all        reservation Utilization
```

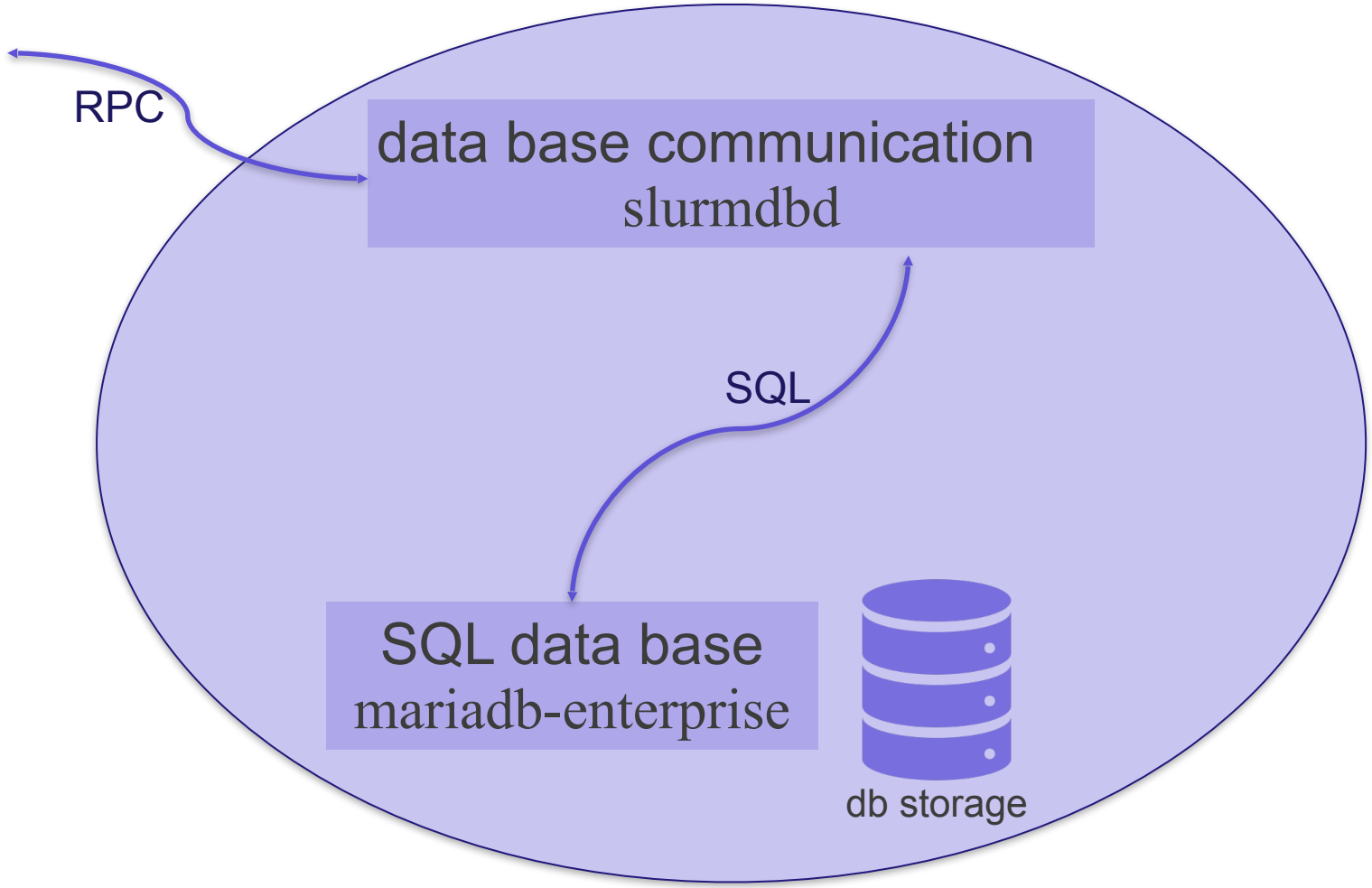
- Containerized personal front-end image (notional)

```
pn12345-hpc-frontend$  sbatch  -M all  myjobscript.sh
```

Server component implementation

Combined Slurm Data Base Project DB Servers

Conceptual Zoom in: service components



Combined Slurm Data Base Project Server hardening

Physical Zoom in: DB server components

Dell R640 DB server

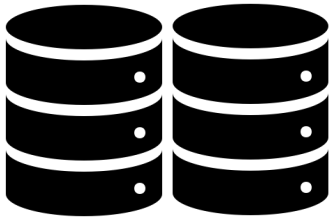
RAM in-memory db + pad

dual NIC ←→ back-end
dual NIC ←→ private

/db

RAID 10 + hot spare
mixed-use SSD

system
mostly read-only



RAID 1
NVME

Availability Equations and Calculations

Availability Equations

- $P_{n-1} \sim= P^{n-1} = (MTTR/MTTF)^{n-1}$

  probability that a given module has failed in the interval from 1 to n-1

- $P_f = 1/MTTF$

  probability that a given module fails

- $P_f * P_{n-1} \sim= (1/MTTF) * (MTTR/MTTF)^{n-1}$

  combining

- $P_{n\text{-}plex} \sim= (n/MTTF)*(MTTR/MTTF)^{n-1}$

  probability that a module N causes a failure of an *n* component complex of modules, an “n-plex”

- This depends on how the n-plex is constructed. A 2-module duplex:

  $P = MTTF^2/(2*MTTR)$

  The combined multicluster DB server nodes consist of a primary and alternate, a 2-node duplex.

  Assume:

  MTTF = 1 year

  MTTR = 4 hours

  $MTTF_{primary+alternate} = MTTF^2 / 2\ MTTR = 1095$ years

It is possible to construct a highly reliable service from unreliable components

after: Transaction Processing, Gray & Reuter, c. 1993, Morgan Kaufmann, 978-1558601901

Over 70 years at the forefront of supercomputing