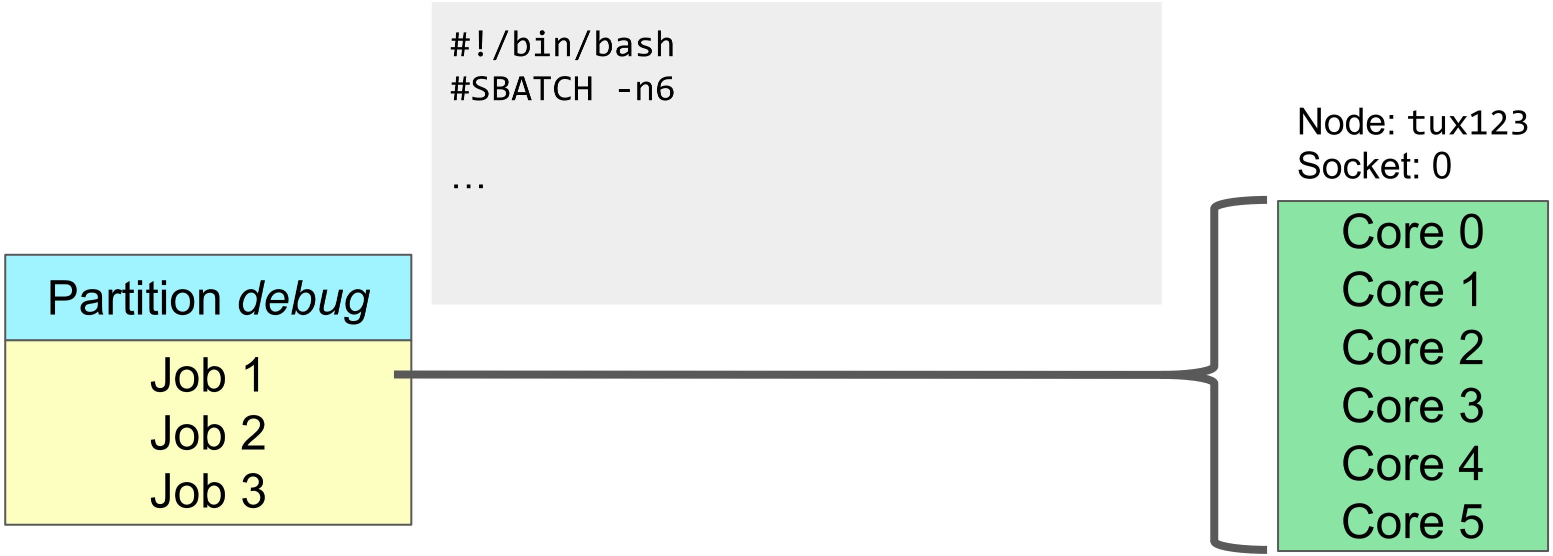
Step Management Enhancements

Felip Moll
Oriol Vilarrubí
Brian Christiansen

Slurm User Group 2024

What is a job and a step?

- Job
    - An allocation of resources (nodes, cpus, memory, gpus, licenses, etc.)



```
#!/bin/bash
#SBATCH -n6

...
```

Partition *debug*

Job 1
Job 2
Job 3

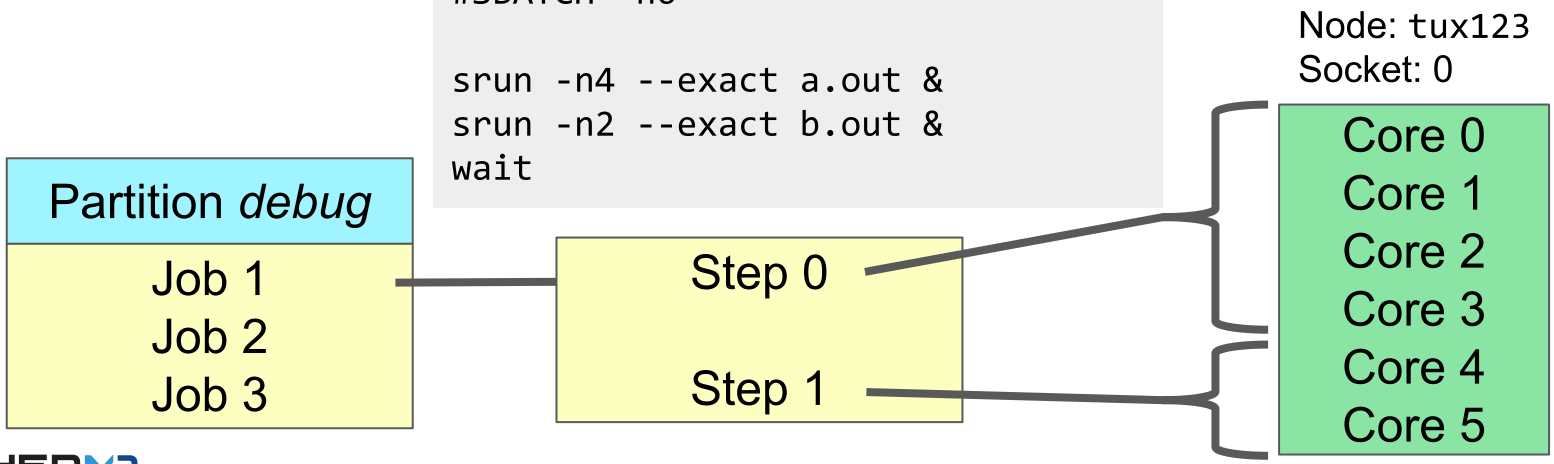
Node: `tux123`
Socket: 0

Core 0
Core 1
Core 2
Core 3
Core 4
Core 5

What is a job and a step?

- Step
  - Set of tasks which use a subset of the job’s resources
  - Steps typically run parallel programs (e.g. MPI)
  - Steps can run serially or concurrently

```
#!/bin/bash
#SBATCH -n6

srun -n4 --exact a.out &
srun -n2 --exact b.out &
wait
```

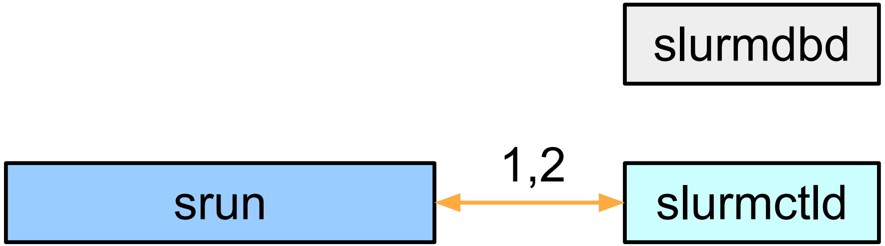


What is a job and a step?

slurmstepd

- batch step
  - Runs the batch script
- interactive step
  - Step created on compute node with a PTY for interactive work inside allocation
- extern step
  - Enabled through PrologFlags=contain
  - Runs on all slurmds
  - Used to:
    - Launch prologs before allocation
    - Setup X11 forwarding, containers
    - Track processes started outside of allocation (e.g. pam_slurm_adopt, ssh, ...)
- job step (.0 , .1 ...)
  - execs and takes care of processes workflow

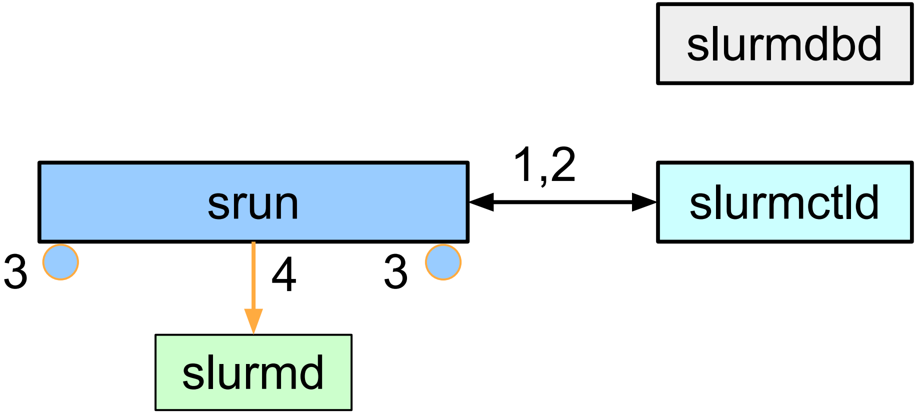
Step Launch Sequence



1. **srun** sends step create request to **slurmctld**
2. **slurmctld** responds with step credential

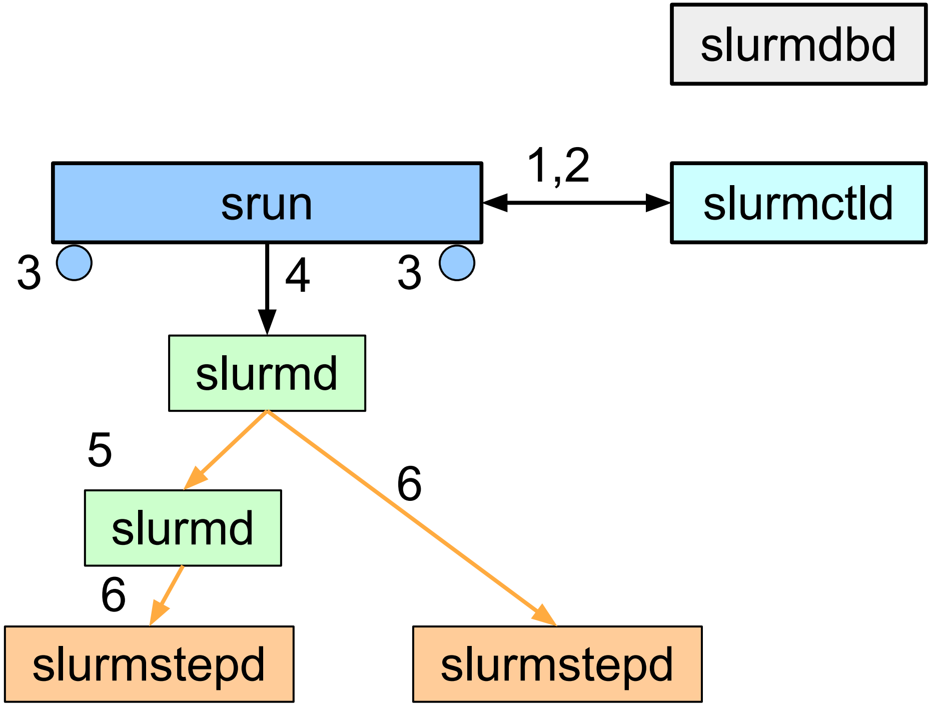
slurm_cred_t

Step Launch Sequence



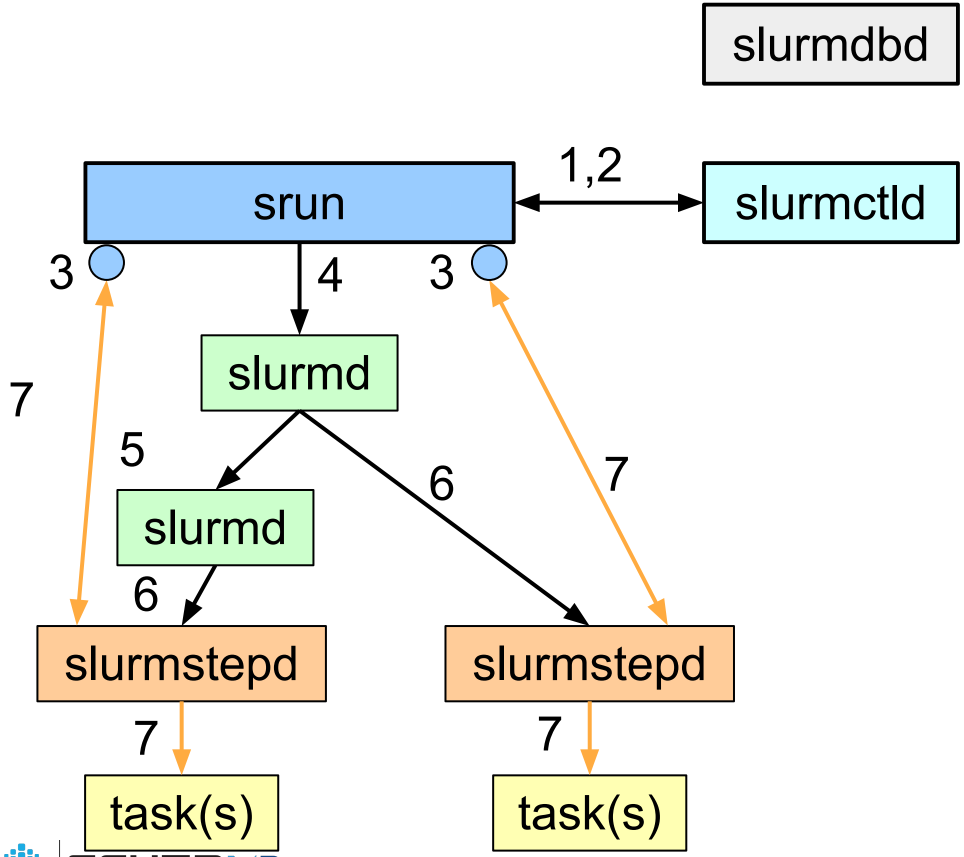
1. **srun** sends step create request to **slurmctld**
2. **slurmctld** responds with step credential
3. **srun** opens sockets for I/O
4. **srun** forwards credential with task info to **slurmd**

Step Launch Sequence



1. **srun** sends step create request to **slurmctld**
2. **slurmctld** responds with step credential
3. **srun** opens sockets for I/O
4. **srun** forwards credential with task info to **slurmd**
5. **slurmd** forwards request as needed
6. **slurmd** forks/execs **slurmstepd**

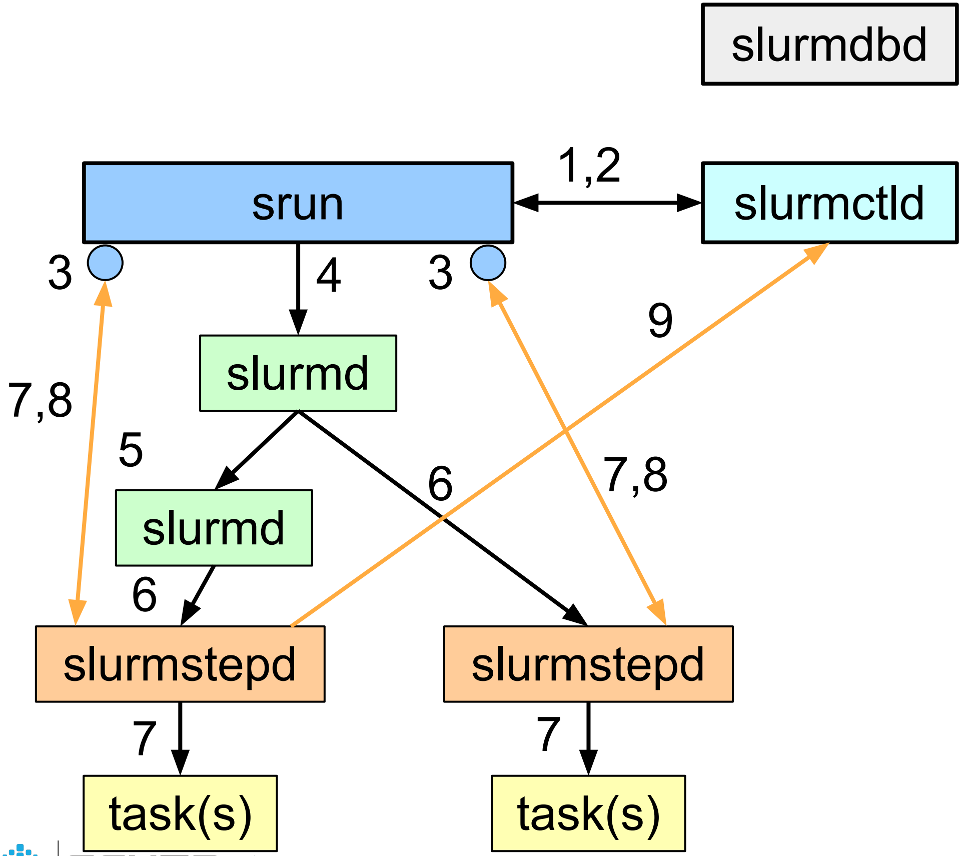
Step Launch Sequence



1. **srun** sends step create request to **slurmctld**
2. **slurmctld** responds with step credential
3. **srun** opens sockets for I/O
4. **srun** forwards credential with task info to **slurmd**
5. **slurmd** forwards request as needed
6. **slurmd** forks/execs **slurmstepd**
7. **slurmstepd** connects I/O and launches **tasks**

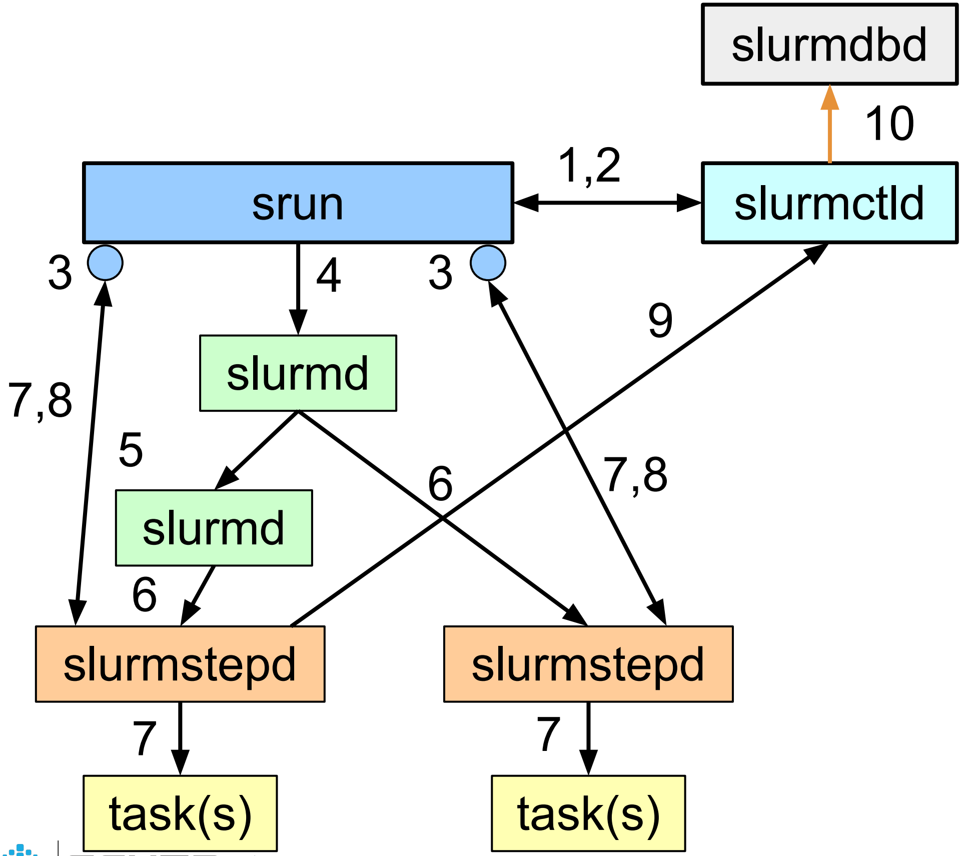
user processes

Step Launch Sequence



1. **srun** sends step create request to **slurmctld**
2. **slurmctld** responds with step credential
3. **srun** opens sockets for I/O
4. **srun** forwards credential with task info to **slurmd**
5. **slurmd** forwards request as needed
6. **slurmd** forks/execs **slurmstepd**
7. **slurmstepd** connects I/O and launches **tasks**
8. On task termination, **slurmstepd** notifies **srun**
9. **slurmstepd** sends step completions to **slurmctld**

Step Launch Sequence



1. **srun** sends step create request to **slurmctld**
2. **slurmctld** responds with step credential
3. **srun** opens sockets for I/O
4. **srun** forwards credential with task info to **slurmd**
5. **slurmd** forwards request as needed
6. **slurmd** forks/execs **slurmstepd**
7. **slurmstepd** connects I/O and launches **tasks**
8. On task termination, **slurmstepd** notifies **srun**
9. **slurmstepd** sends step completions to **slurmctld**
10. **slurmctld** sends step completion to **slurmdbd**

The Step Manager

Problem: Step congestion

- Step management is done by the controller
  - Can be a source of RPC **congestion**

- Step management requires controller **job write lock**
  - Locks up the system
  - Slurm is highly threaded but not highly concurrent

- Bigger issue when creating **many steps within an allocation**
  - User doing own resource management
    - e.g. 1 allocation with 1000's of steps

Solution: Isolated step management

- Move step management out of the controller

- step management done by slurmstepd (stepmgr role)

- Reduce rpc congestion and locking on the controller

The Step Manager

- Designate the extern_step on the first node (aka BatchHost) of the job as the stepmgr for job

- Job allocation/environment tells which slurmd has the stepmgr
    - SLURM_STEPMGR=<host>

- Step Manager creates and manages steps for the job

Step Manager - **accounting**

- Step accounting details are proxied by slurmctld
    - Then forwarded to SlurmDBD
    - Forwarding does not require access to core locks (job write)

- If AccountingStorageType=accounting_storage/slurmdbd
    - The stepmgr will load the accounting_storage/ctld_relay plugin
        - just sends a message to the controller
        - controller receives message and adds work to the slurmdbd *agent queue*

- If a site doesn't require this data, setting AccountingStorageEnforce=nosteps can significantly reduce load on the SlurmDBD

Step Manager - **configuration**

- Globally in slurm.conf
  - SlurmctldParameters=enable_stepmgr

- per-job
  - sbatch --stepmgr ...
  - salloc --stepmgr ...
  - srun --stepmgr ...

- Or can be managed through job_submit.lua
  - Could enable on a partition-by-partition or account-by-account basis
  - Allows for a phased roll-out

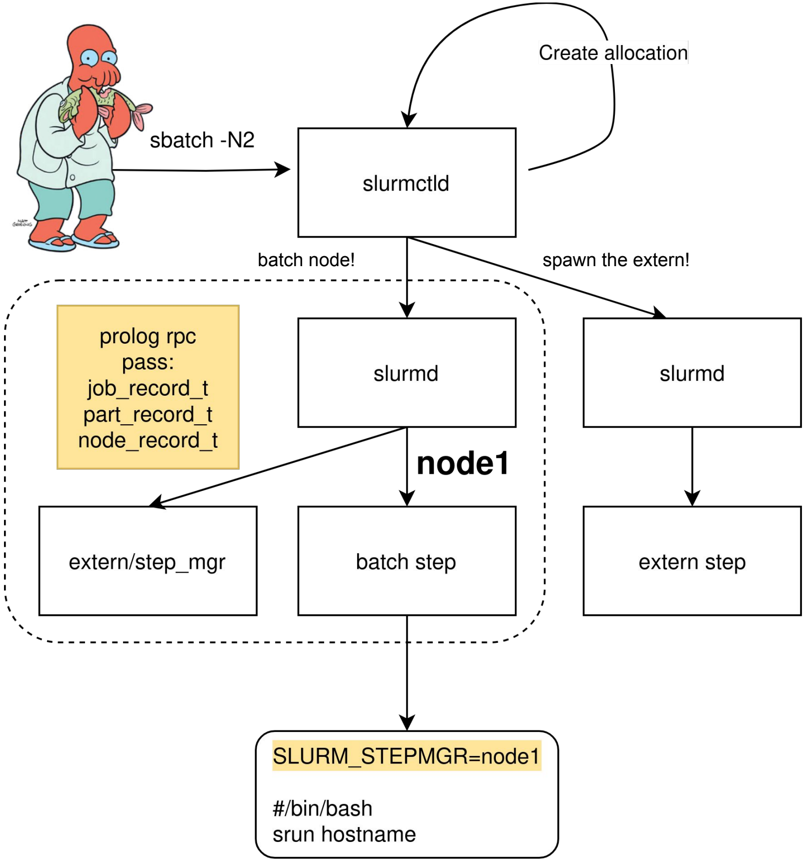
- We expect to make this the default behavior within a few releases
  - And eventually the only supported behavior

Step Manager - **creation**

- The **extern step** on the first node (BatchHost) is designated as the step manager

- The key is a **prolog_launch_msg_t** sent to the stepmgr by slurmd
    - It includes:
        - **job_record_t**
        - **part_record_t** of the job
        - all **node_record_t** of job_ptr->node_bitmap
            - node_record_table_ptr reconstructed on stepmgr with same indexes.

Step Manager - **creation**



sbatch -N2

Create allocation

slurmctld

batch node!

spawn the extern!

prolog rpc
pass:
job_record_t
part_record_t
node_record_t

slurmd

slurmd

node1

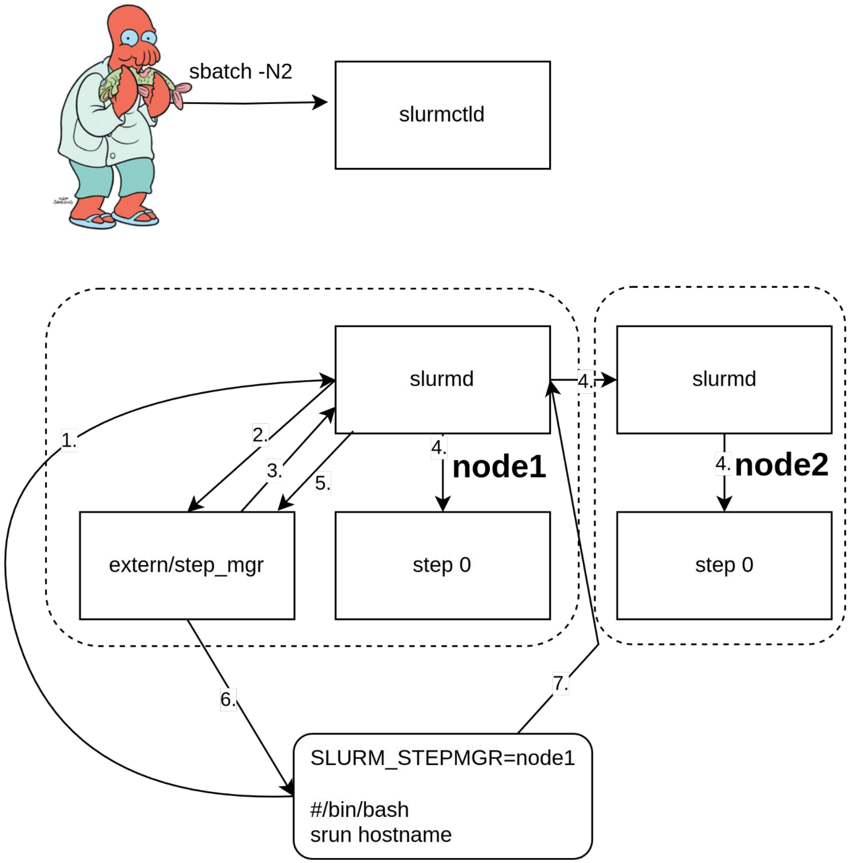
extern/step_mgr

batch step

extern step

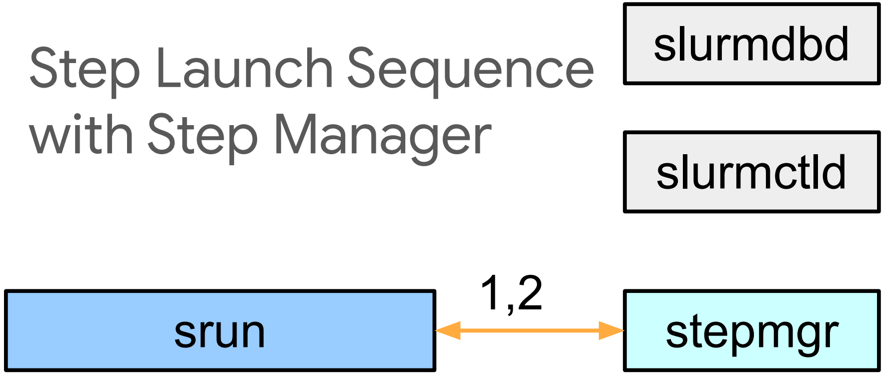
```
SLURM_STEPMGR=node1

#/bin/bash
srun hostname
```

Step Manager - **step creation**



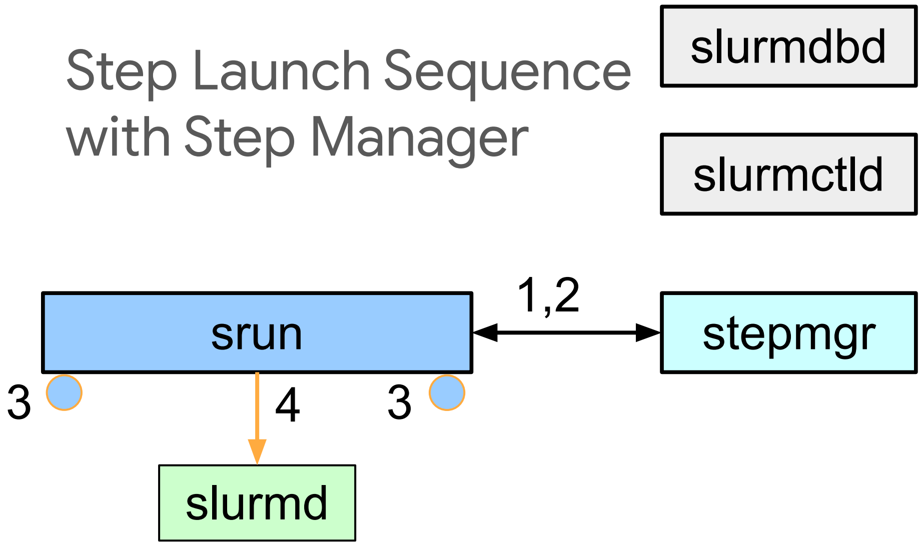
1. create step
2. TCP to UNIX
3. create_step with fanout
4. fanout
5. steps created
6. steps created
7. launch task

Step Launch Sequence with Step Manager



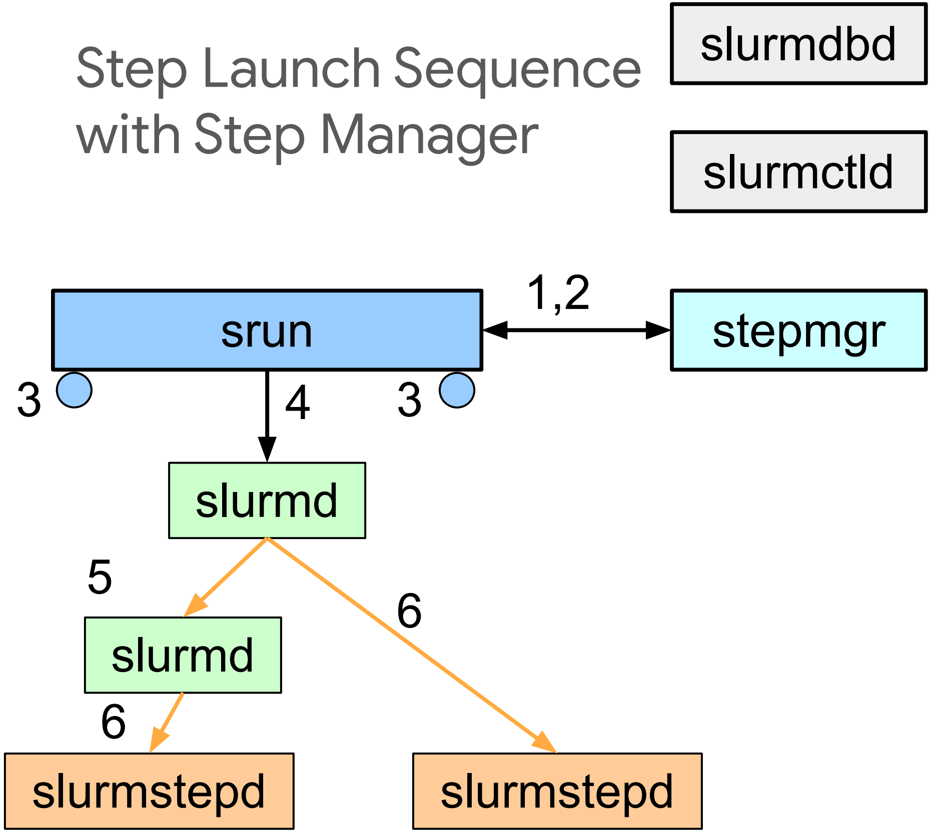
1. **srun** sends step create request to **stepmgr**
2. **stepmgr** responds with step credential

Step Launch Sequence with Step Manager



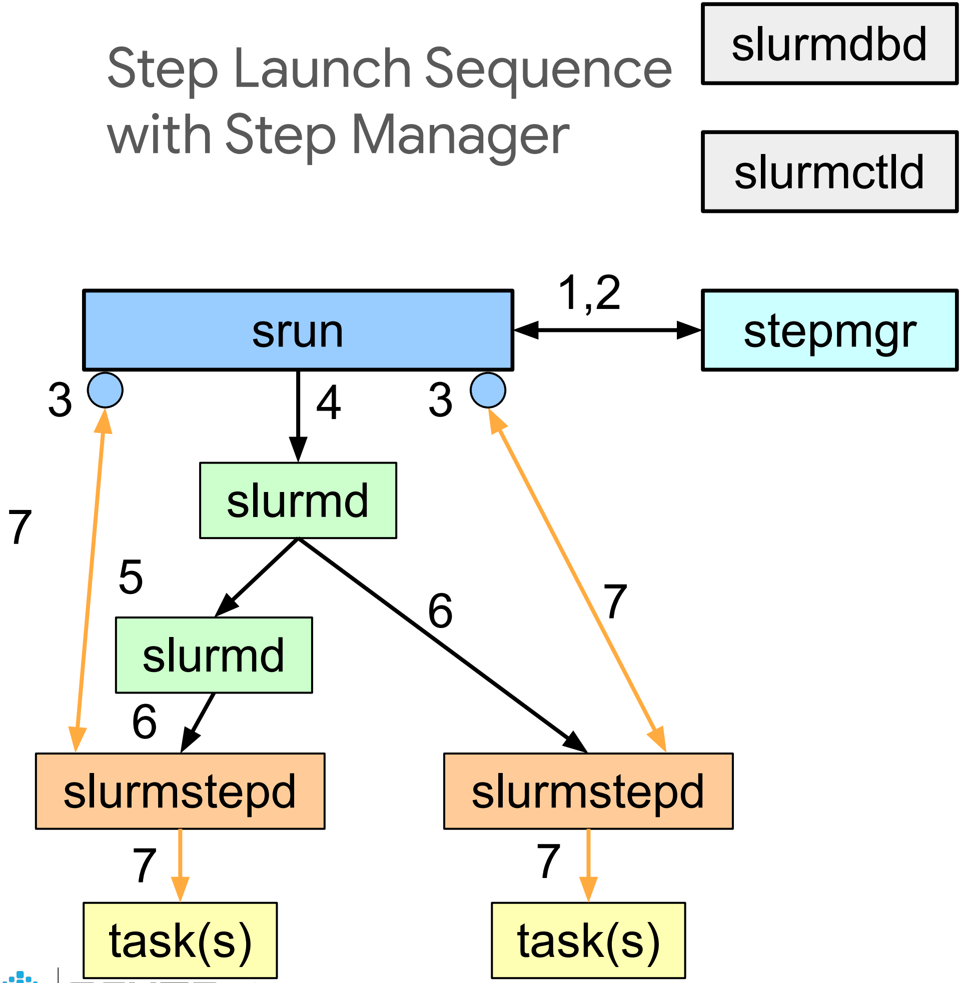
1. **srun** sends step create request to **stepmgr**
2. **stepmgr** responds with step credential
3. **srun** opens sockets for I/O
4. **srun** forwards credential with task info to **slurmd**

Step Launch Sequence with Step Manager



1. **srun** sends step create request to **stepmgr**
2. **stepmgr** responds with step credential
3. **srun** opens sockets for I/O
4. **srun** forwards credential with task info to **slurmd**
5. **slurmd** forwards request as needed
6. **slurmd** forks/execs **slurmstepd**

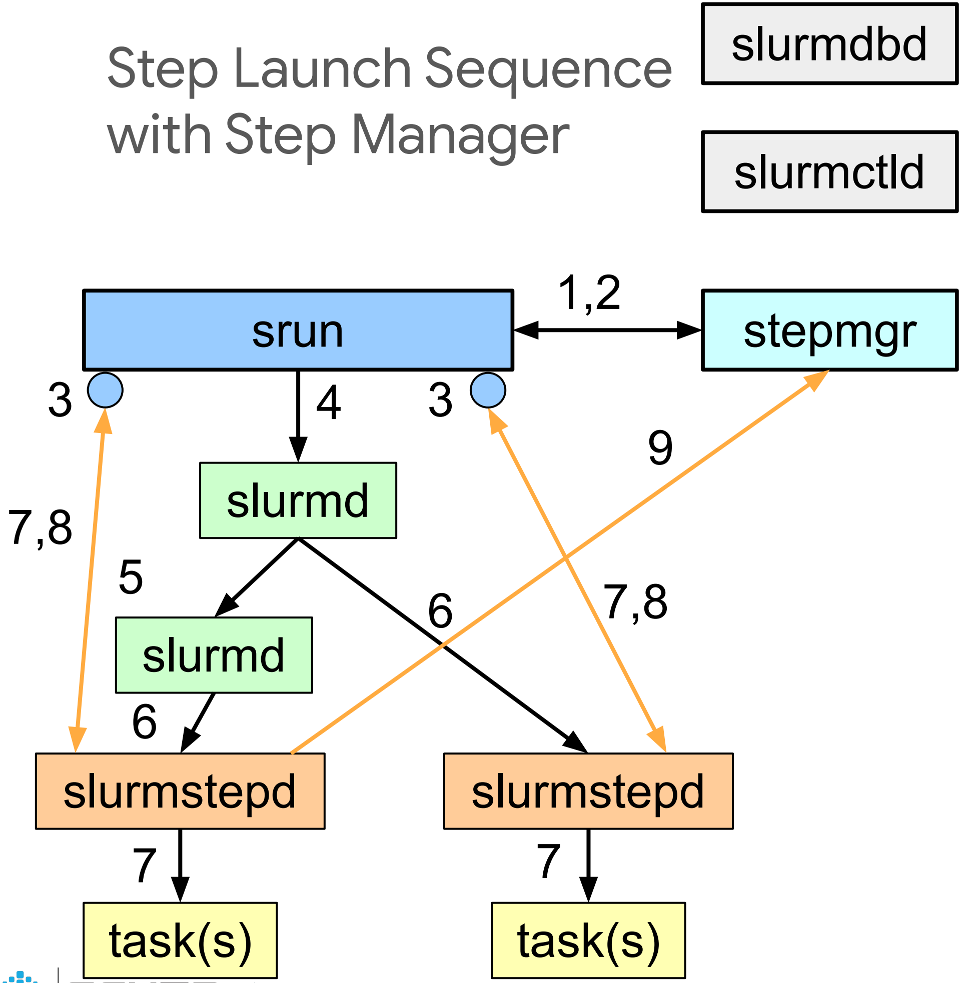
Step Launch Sequence with Step Manager



1. **srun** sends step create request to **stepmgr**
2. **stepmgr** responds with step credential
3. **srun** opens sockets for I/O
4. **srun** forwards credential with task info to **slurmd**
5. **slurmd** forwards request as needed
6. **slurmd** forks/execs **slurmstepd**
7. **slurmstepd** connects I/O and launches **tasks**

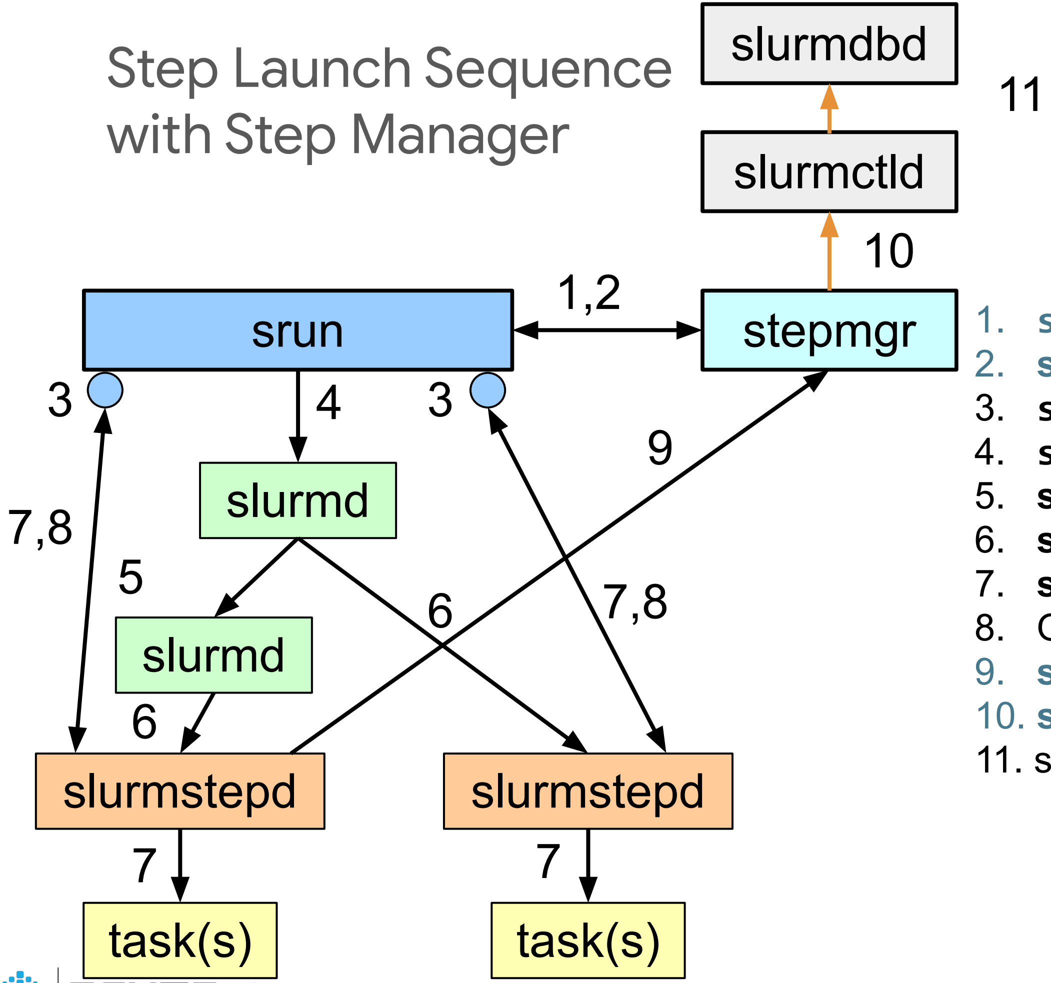
user processes

Step Launch Sequence with Step Manager



1. **srun** sends step create request to **stepmgr**
2. **stepmgr** responds with step credential
3. **srun** opens sockets for I/O
4. **srun** forwards credential with task info to **slurmd**
5. **slurmd** forwards request as needed
6. **slurmd** forks/execs **slurmstepd**
7. **slurmstepd** connects I/O and launches **tasks**
8. On task termination, **slurmstepd** notifies **srun**
9. **slurmstepd** sends step completions to **stepmgr**

Step Launch Sequence with Step Manager

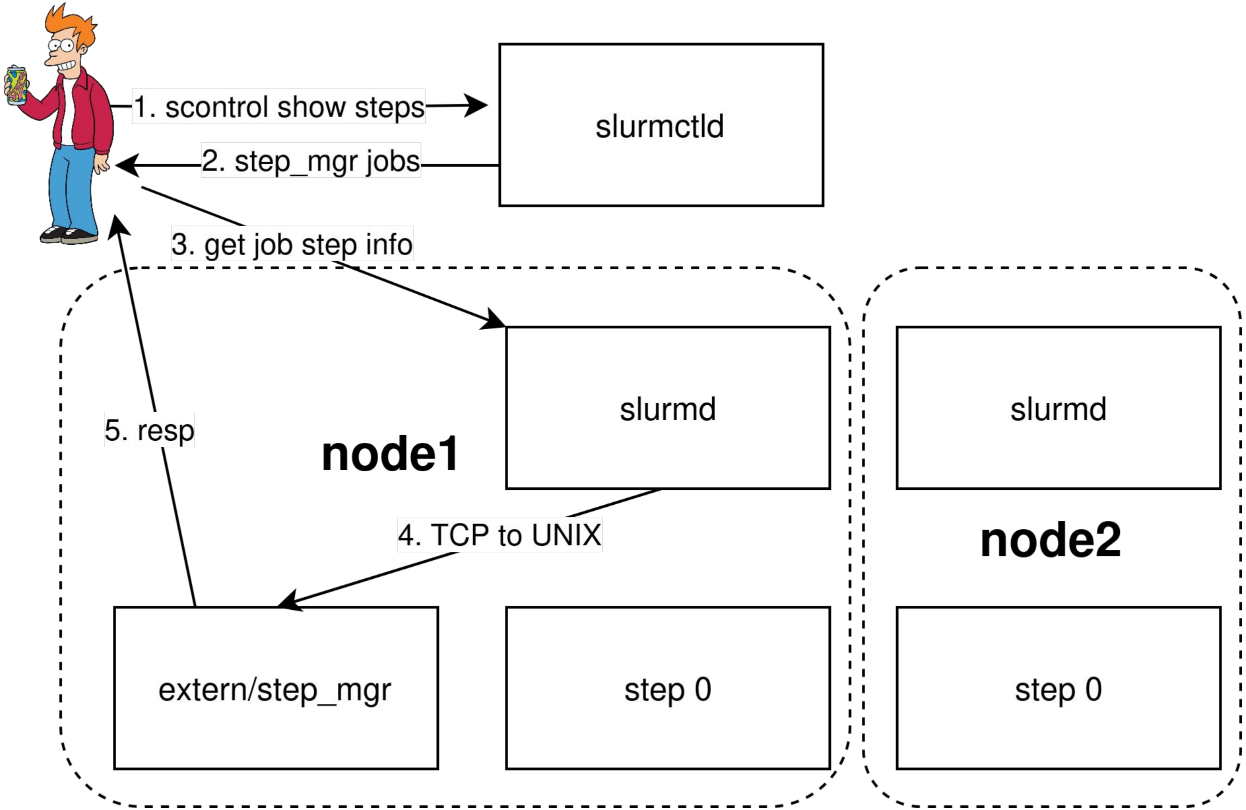


1. **srun** sends step create request to **stepmgr**
2. **stepmgr** responds with step credential
3. **srun** opens sockets for I/O
4. **srun** forwards credential with task info to **slurmd**
5. **slurmd** forwards request as needed
6. **slurmd** forks/execs **slurmstepd**
7. **slurmstepd** connects I/O and launches **tasks**
8. On task termination, **slurmstepd** notifies **srun**
9. **slurmstepd** sends step completions to **stepmgr**
10. **stepmgr** sends completion to slurmctld
11. slurmctld sends completion to **slurmdbd**

User commands

- **scontrol show steps**
  - Will show the steps of a job only when requesting a specific job
  - If no job specified, it will return the list of steps managed by the controller
    - Including the extern (stepmgr), batch, interactive steps
  - If a job is specified, the client contacts the stepmgr to get all the steps of the job

Step Manager - scontrol show steps

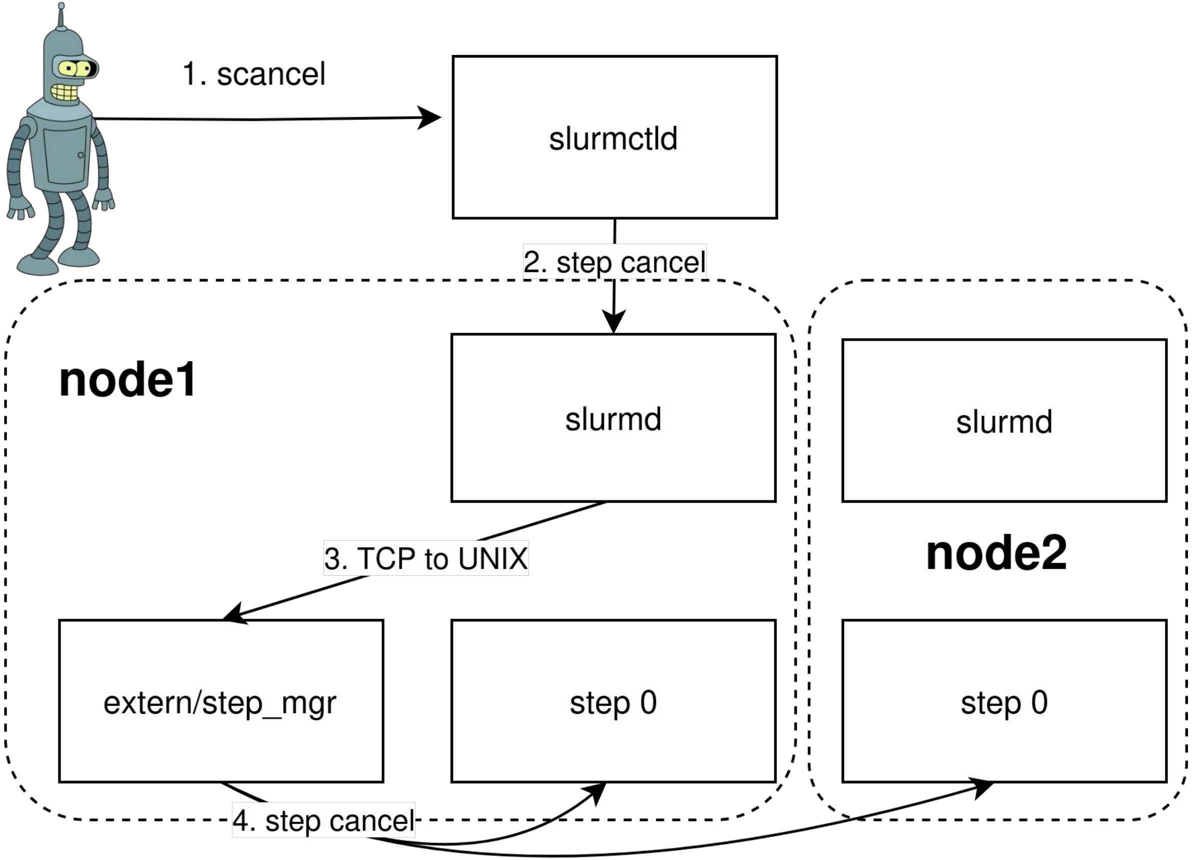


User commands

- **scontrol show steps**
  - Will show the steps of a job only when requesting a specific job
  - If no job specified, it will return the list of steps managed by the controller
    - Including the extern (stepmgr), batch, interactive steps
  - If a job is specified, the client contacts the stepmgr to get all the steps of the job

- **scontrol update step & scancel**
  - You send a message to the controller which talks to the stepmgr

Step Manager - scancel



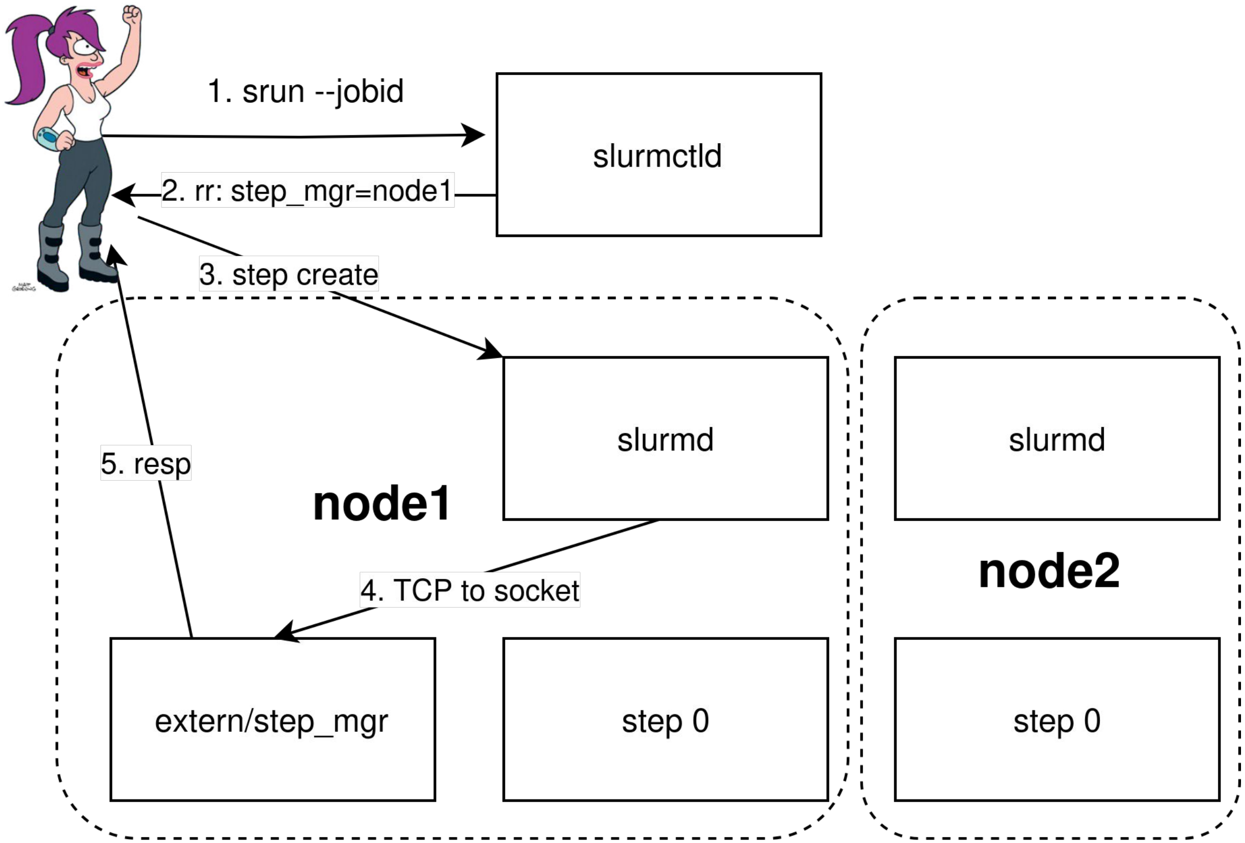
User commands

- **scontrol show steps**
  - Will show the steps of a job only when requesting a specific job
  - If no job specified, it will return the list of steps managed by the controller
    - Including the extern (stepmgr), batch, interactive steps
  - If a job is specified, the client contacts the stepmgr to get all the steps of the job

- **scontrol update step & scancel**
  - You send a message to the controller which talks to the stepmgr

- **Message re-routing**
  - The controller tells you which host you need to talk to
  - srun --jobid, sbcast, sstat

Step Manager - srun –jobid=#



User commands

- **scontrol show steps**
  - Will show the steps of a job only when requesting a specific job
  - If no job specified, it will return the list of steps managed by the controller
    - Including the extern (stepmgr), batch, interactive steps
  - If a job is specified, the client contacts the stepmgr to get all the steps of the job

- **scontrol update step & scancel**
  - You send a message to the controller which talks to the stepmgr

- **Message re-routing**
  - The controller tells you which host you need to talk to
  - srun --jobid, sbcast, sstat

- **squeue -s**
  - Currently not supported
  - squeue doesn't request specific steps, gets all of them and then it filters
  - If a job is specified, it queries the controller which returns its stepmgr step

Heterogenous jobs

- Do not use Step Manager
- The controller still keeps track of the different steps in an heterogeneous job
- The controller does all the step management for het jobs

Summary

- slurmctld still tracks batch, extern, interactive steps and heterogeneous jobs
- There’s one extern step designated as the Step Manager
  - it will communicate with slurmd, slurmctld and client commands
- The stepmgr tracks job_steps and relieves the controller

- Debugging
  - stepmgr logs
    - **[**504284**.extern stepmgr]** debug: Message thread exited
  - process name

```
■   root     3901556  0.0  0.0 218344 10592 ?        Sl   23:20   0:00 slurmstepd: [504286.extern stepmgr]
■   root     3901564  0.0  0.0 217260 10224 ?        Sl   23:20   0:00 slurmstepd: [504286.batch]
■   root     3901583  0.0  0.0 282476  9900 ?        Sl   23:20   0:00 slurmstepd: [504286.0]
```

Step Manager benefits

Benefits of isolated Job Step Management

- Allows for **vastly** improved concurrency for systems with heavy step usage
  - Current reality
    - Step launch throughput is directly related with job launch throughput
    - Most Slurm installations peak at around 300 (jobs and steps) per second
  - Long-term goal is to be able to sustain 500 jobs per second
    - With each job - independently - being able to launch 1000 steps per second

Benefits of isolated Job Step Management

- Change in architecture provides flexibility to adopt other workflow tooling within the job
    - Currently, slurmctld is very performance sensitive
        - Adopting "heavier" tooling, such as workload management stacks, is risky
    - Moving the step management to the compute node isolates any performance or stability issues to the individual job
        - Looking at support for CWL, or similar workflow tooling, within the step management layer

Isolated Job Step Management - Cray Specifics

- *mpi/cray_shasta plugin*
  - Cray MPI requires applications to use the same TCP port across all nodes
  - Uses "MpiParams=ports=xxxx-yyyy" to centrally schedule TCP ports in slurmctld
  - For single-node jobs this isn't an issue
  - But needs to happen on a step-by-step basis for multi-node jobs
    - Each stepmgr process is delegated a subset of the overall range

- *switch/hpe_slingshot*
  - Per-job VNI assignment works
  - Per-step VNI assignment does not work with --stepmgr

Questions?

SCHEDMD

The Slurm Company