Job Step Management in User Space

Morris Jette
jette@schedmd.com

SchedMD LLC

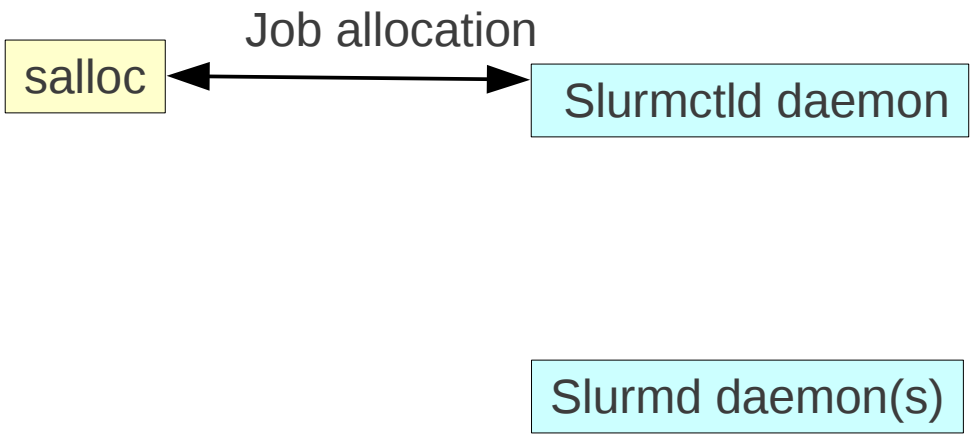
Outline

- Current design
- The problem
- Proposed design
- Status

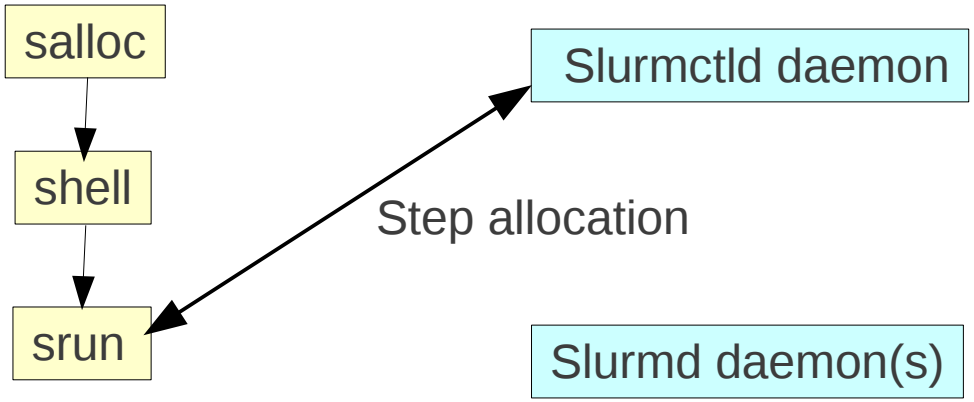
Current Design

- Job allocation (salloc, sbatch or srun) is performed as a single RPC and sets various environment variables for the job
- Each job step allocation is performed as a separate RPC that makes use of command line arguments and the environment variables

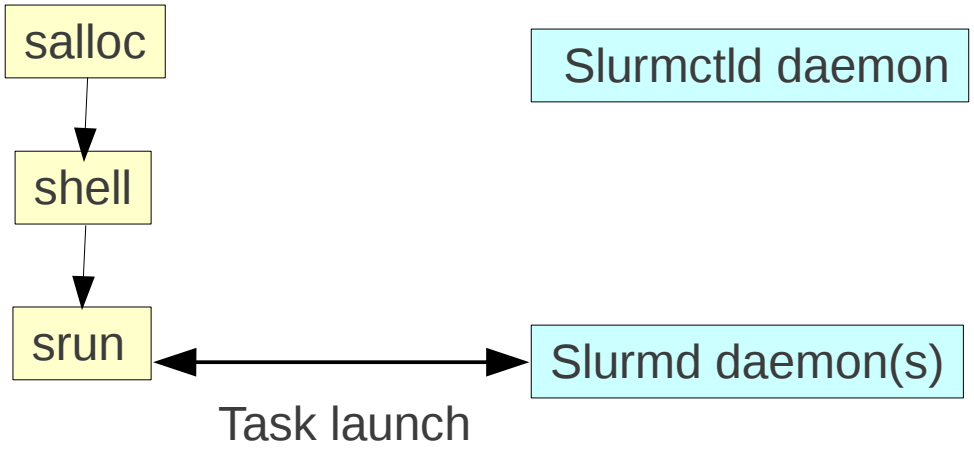
Current Architecture



Current Architecture



Current Architecture



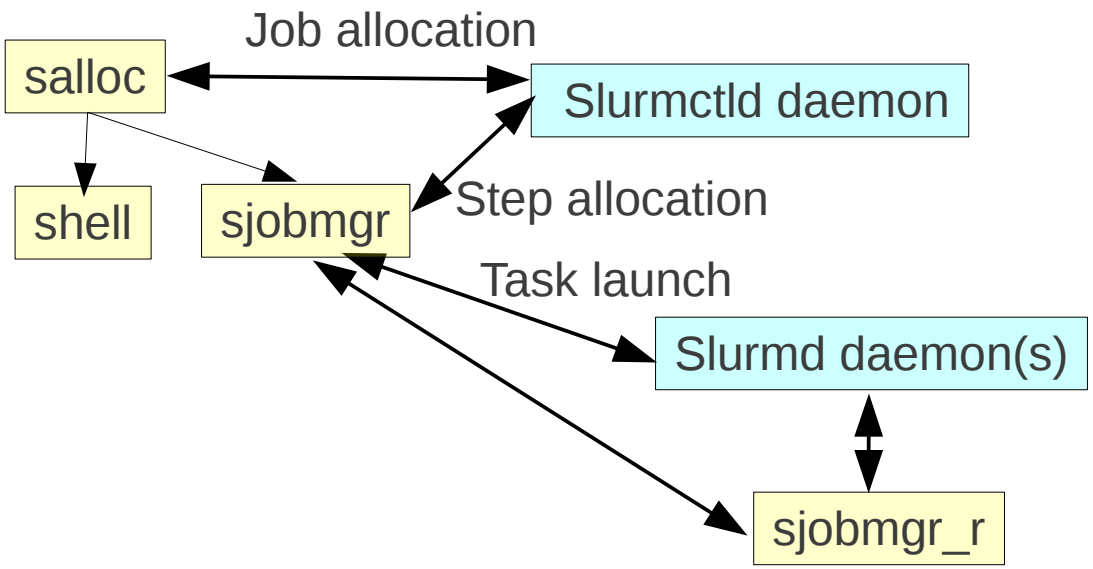
The Problem

- Scalability and performance
  - Jobs with many job steps place a heavy burden upon the *slurmctld* daemon
- Flexibility
  - Jobs lack a good mechanism to manage size changes
- Fault-tolerance
  - Some environment variables become invalid when allocated nodes fail
- Job step management tools relatively simple
  - No mechanism to prioritize or have dependencies

Proposed Solution

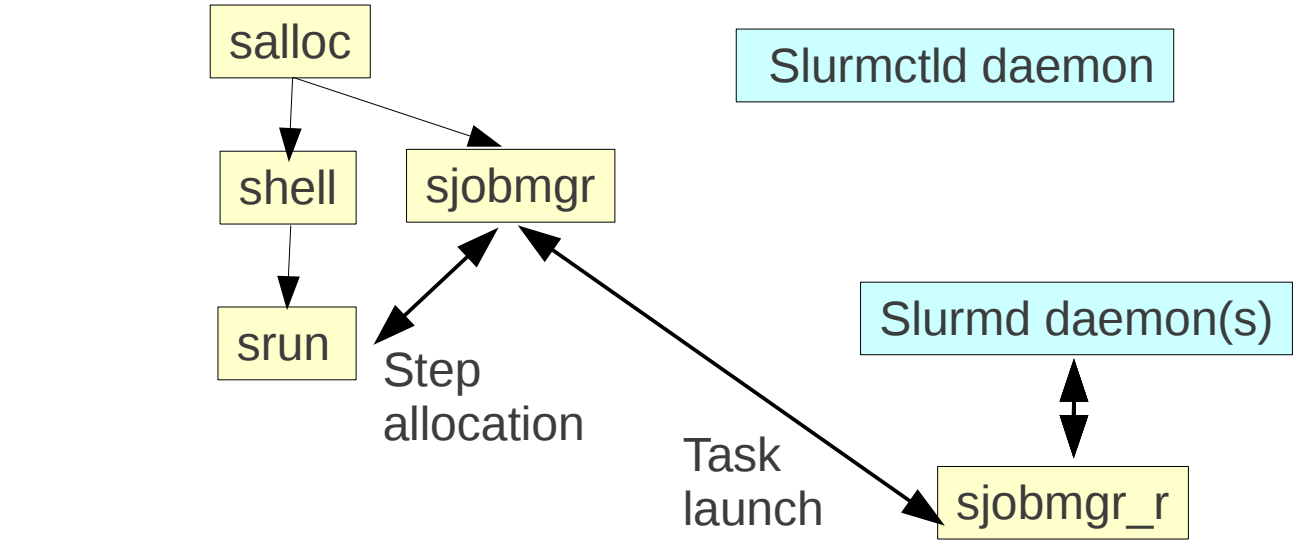
- Develop a program that runs in user space to manage the job's resources
  - Allocate resources for job steps
  - Launch tasks for job steps
  - Monitor resource failures for the job
  - Manage resources for the job to grow or shrink through time
  - Simple language and API for user

Proposed Architecture



Optionally at job startup, launch a local job manager plus one remote job manager on each compute node

Proposed Architecture
(Step allocations)



Subsequent step allocations take place entirely in user space

Changing Job Sizes

- SLURM version 2.3 has the ability to grow a job
  - Submitting a new job with “dependency=expand:<jobid>”
  - After the job is scheduled, explicitly transfer its resources
  - Multiple jobs can be submitted to grow any job
  - Jobs can also shrink at will
- Develop new language to manage these resources
  - Submit requests to resize a job
  - Query what expansions are pending
  - Query when expansions are expected to occur
  - Automatically claim resources when available

Fault Tolerance

- Note when failures are expected or actually occur
  - Stop using failing nodes as soon as possible and relinquish those resources
  - Trigger checkpoint and/or migration of job steps
  - Secure additional resources to replace failing or failed resources

Improved Job Step Management

- Improved job step management could go directly into the *slurmctld* daemon and/or the new tools
  - Current logic is very simple and not well suited to managing large numbers of job steps within a job
  - Job step dependencies
  - Prioritization of job steps

Lost Functionality

- Since the job step execution happens entirely in user space, the *slurmctld* lacks information about it
  - No accounting or log records about the job steps

Status

- Seeking feedback from user community about design
- No immediate plans for this development to occur