HPS
hpc.gwdg.de

Timon Vogt

Slurm

The Simple Linux Utility for Resource Management

Overview and Timetable

- Slurm concepts, installation, and configuration — 45 min
  - What is Slurm, and how does it work?
  - How to install Slurm?
  - How to configure a Slurm cluster?
- Break & Questions — 15 min
- Lunch break — 60 min
- Exercise: Building a Slurm cluster — 120 min

Table of contents

1 Overview

2 What is Slurm, and how does it work?

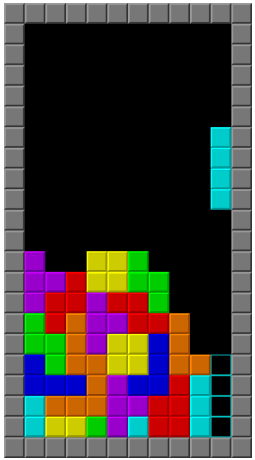
3 How to install Slurm?

4 How to configure a Slurm cluster?

Slurm

- Fault-tolerant resource manager and scheduler
- Interface between the users and the computing nodes
- Matches job requirements with available resources
- Maintains a queue of pending jobs on the cluster
- Prioritizes jobs based on configurable parameters

Slurm plays Tetris ...



Source: https://en.wikipedia.org/wiki/Tetris

Slurm terminology: Hardware

Node

- A physical server in the cluster.
- Has an IP, (usually) a name via DNS, one or multiple CPUs, RAM, maybe GPUs, maybe additional network cards, etc.

Board

- Refers to a mainboard of a node.
- Nodes (usually) only have one board.

Socket

- A phyiscal, **populated**, CPU Socket on a mainboard of a node.
- The number of sockets is the number of physical CPUs in a node.

CPU

- In Slurm, a CPU always refers to a single CPU core
- Physical CPUs however usually have multiple of them.

Slurm terminology: Job

- A Slurm Job is the execution unit that Slurm works on.
  - Wraps a program and its parameters.
- Jobs are submitted by a user and then scheduled by Slurm to be executed on the cluster.
  - Always have resource requests attached to them.
  - Can have an allocation.
- Jobs always have a state (e.g. PENDING, RUNNING or COMPLETED).
  - Also always have a numerical ID.

Slurm terminology: Jobsteps and Tasks

Jobstep

- A Slurm Job might consist of multiple jobsteps.
  - Also always wrap around a program and its parameters.
  - Need to be manually created inside the job's program.
- Jobsteps are executed on a single node or multiple nodes sequentially or in parallel.
- In Slurm's accounting, job steps can be accounted for individually.

Task

- A Jobstep can consist of mulitple tasks
  - A Slurm task can be compared to a process.
  - Tasks always run on a single node, but can occupy multiple CPU cores.

Difference: Tasks have a less overhead, but cannot be started on other nodes

Slurm terminology: Allocations

- In Slurms terms: "A claim on resources".
- Contains the resources *allocated* to a job.
  - Might span multiple nodes, or be just a fraction of a CPU.
  - Can also contain RAM, GPUs, network slices, even software licenses
- A job *might* have an allocation, or not.

Slurm terminology: Partitions

- Are sets of nodes which are (usually) of the same hardware(-type).
  - E.g. All nodes with a certain identical CPU are in one partition.
  - This is not required, partitions can also contain different node types.
- Jobs can be submitted to a certain partition or multiple partitions simultaneously.
  - will then run on the next free node(s) of this partition(s).
  - Usually used to ensure that a job runs on certain hardware.
- Partition can overlap (i.e. one node can be in multiple partitions)
- Partitions are **not** queues.
  - Jobs don't have to run in order of submission

Slurm daemons: `slurmctld`

- The Slurmctld (Slurm-Controller-daemon) is the cluster master of Slurm.
- Has the overview over the entire cluster
  - Knows all pending jobs
  - Knows the state of all running jobs
  - Knows the state of all the clusters resources
- Performs the scheduling, assigning jobs to resource allocations
- Regularly sends job state information to the database
- Multiple additional controllers can act as failovers

Slurm daemons: `slurmdbd`

- The Slurmdbd (Slurm-Database-daemon) maintains the accounting database of Slurm.
- The database holds the job information of all jobs that ran on the cluster.
  - The database itself is an external program (like `mariadb`)
  - The `slurmdbd` acts as an interface to it
- This information can then be used for accounting or debugging purposes.
- If the `slurmdbd` fails, new accounting information is stored inside the `slurmctld` until `slurmdbd` is available again

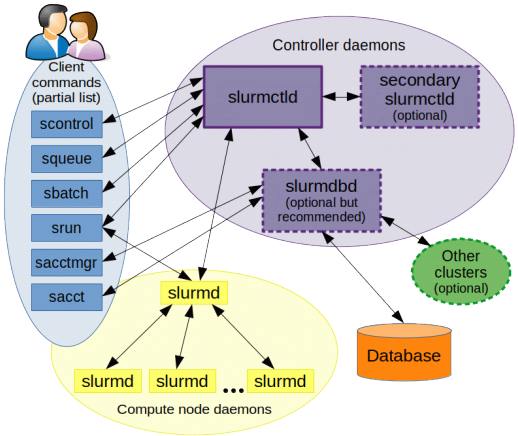
Slurm daemons: `slurmd`

- The Slurmds (Slurm-daemon's) make up the actual cluster
- Run on each node of the cluster
- Advertising themselves to the `slurmctld`
- If a job comes in, they are creating an (isolated) allocation and run the program

Slurm commands

srun	Runs a given command as a Slurm job
sbatch	Submits a shell script as a Slurm job, to be executed on a node.
salloc	Creates a Slurm allocation and gives the user an interactive shell session on it.
squeue	Displays the current Slurm queue, so all currently running and waiting jobs
sinfo	Displays the current state of nodes and partitions of the cluster
sacct	Queries accounting data (e.g. past Jobs of the cluster)
sacctmgr	View and change Slurm accounts and user associations
scontrol	Adminstrator tool to change the state of the cluster

Running a job in Slurm



Source:

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

Program: `srun hostname`

- `srun` is creating a job, sends it to the `slurmctld`
- The `slurmctld` looks for available resources and schedules the job
- The `slurmd` of the selected nodes create the allocation and connect with `srun`
- The `slurmd` run the program, forwarding `stdout`, `stderr`, `stdin` to `srun`

Slurm plugin: MUNGE

MUNGE: `MUNGE Uid 'N' Gid Emporium`

- MUNGE is an authentication service for creating and validating users across the cluster.
- MUNGE runs as a service on each node, each sharing a common *`munge.key`*
- With MUNGE, individual Slurm components can authenticate the communications between each other:
  - MUNGE creates an encrypted, tamper-proof, *credential* object, containing the `UID` and `GID` of the current process
  - This *credential* can then be validated by another munge service running on another node.
- MUNGE is loaded into Slurm as a plugin.

Installation overview

- Slurm as a program is written in C and intended for Linux based clusters.
  - Therefore it is build using the standard Linux build tool combo:
  - `configure`, `make`, `make install`
    - `configure` checks the current environment
    - `make` compiles the software
    - `make install` copies the compiled binaries to their designated places
- As usual, the `--prefix` flag for the configure script can be used to change the install directory.

Dependencies

- Slurm uses a required and optional dependency system
  - The configure script checks which dependencies are available on the system
  - Slurm then builds only those components that are available.
- Some required dependencies:

```
gcc, gcc-c++, make, lbzip2, ...
```

- Some optional dependencies:

```
hwloc, lua, man2html, ...
```

NFS Share

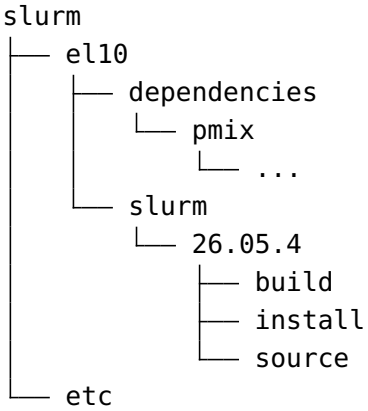
Where to put the Slurm installation:

- Slurm does support building `rpm` files directly.
  - Could be build directly into worker images with warewulf.
  - This would make updates more complex.
  - Also makes it impossible / very complex to have two versions of Slurm installed in parallel.
- Typically, Slurm is installed in a network share (like an NFS share)
  - This makes the `slurmd` binary available on each node immediately
  - Makes in-place updates easier
  - Allows two versions of Slurm to be installed in parallel.

SysConfDir

- The SysConfDir is the directory where the configuration files are located
- Path gets built into the source code
  - However, on run time, the path can be changed via a flag
- Contains the `slurm.conf` file
  - All `slurmd` services need an **identical** `slurm.conf` file
- Some other configuration files are also needed for all `slurmds`
- $\Longrightarrow$ The SysConfDir is also placed on the NFS Share

Installation directory tree



- Allows for multiple versions of Slurm to be installed in parallel
  - Easier for fallbacks in case of update problems
- Source code also remains available
  - Easier to apply patches in case bugs
- SysConfDir is also available on the NFS share
- Allows for multiple Slurm builds in parallel
  - Different OS versions require different Slurm build due to system dependencies

slurm.conf

```
ClusterName=linux-cluster
SlurmctldHost=node1
...
AccountingStorageType=accounting_storage/slurmdbd
AccountingStoreFlags=job_env,job_script
...
PartitionName=linux Nodes=node[2-3] Default=YES
 ↪  MaxTime=INFINITE State=UP
NodeName=node[2-3] NodeAddr=10.0.0.[2-3] CPUs=4 Boards=1
 ↪  SocketsPerBoard=4 CoresPerSocket=1 ThreadsPerCore=1
 ↪  RealMemory=3667
```

slurmdbd.conf

```
DbdAddr=10.0.0.1
DbdHost=localhost
...
StorageType=accounting_storage/mysql
StorageHost=localhost
StorageLoc=slurm_acct_db
```

Further configuration files

`gres.conf` Contains definitions of **G**eneric **Res**ources, like GPUs or network interface cards

`cgroup.conf` Contains configurations for the linux `cgroup` system, used to fence off allocations against one another on the same node

`topology.conf` Contains the network topology (i.e. which node is connected to which switch). Based on it, the `topology` plugin can optimize allocations for low network latency

`plugstack.conf` Contains the paths to self-built Slurm plugins.

Running Slurm

- Slurm serviecs are usually run as a `systemd` service
  - This allows them to run in the background,
  - automatically restart on error and
  - start on node boot automatically
- Systemd service files are placed at

```
/etc/systemd/system
```

- To start the Slurm services, run

```
systemctl start slurmd.service
```