HPC

SLURM Resource Manager and job scheduler

LEARNING

DR@B. DIOP

Role of resource manager

SLURM

Execute parallele jobs

Role of resource manager

SLURM

- **Allocate** resources within a cluster
- **Launches** and **manages** jobs
- **Schedule** works by managing queues using complex scheduling algorithms

What is SLURM

SLURM

- Simple Linux Utility for Resource Management
- Started in 2002 as a simple resource management for Linux clusters
- Used on many of the world largest computers
- +500 l000 lines of code today

What is SLURM

SLURM

- Small and simple
- Open source v2 GPL
- Fault tolerant - Secure
- Portable
- System admin friendly
- Highly scalable

What is SLURM

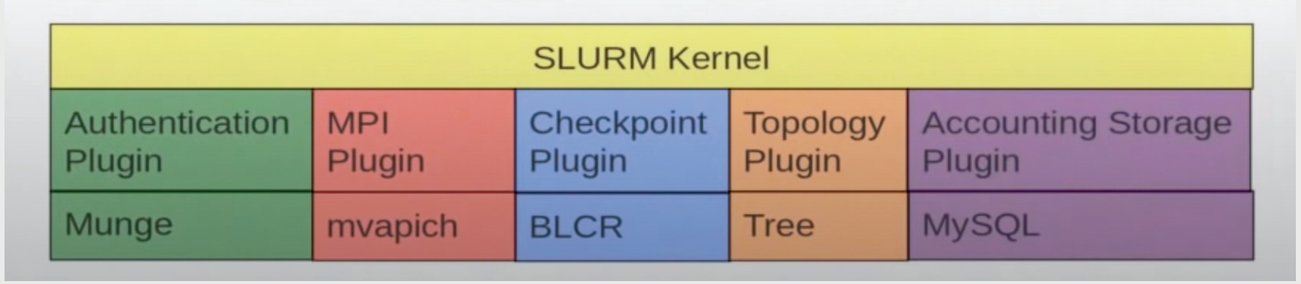
SLURM

- No kernel modifications
- C language
- Skeleton functionality can be extended using plugin
- Various system specific plugins available

Plugins

SLURM

- 70 plugins
  - **Storage** : MySQL, PostgreSQL
  - **Network topology** : 3D-torus, tree
  - **MPI** : OPenMPI, MPICH1, MVAPICH, MPICH2,



Plugins developement

SLURM

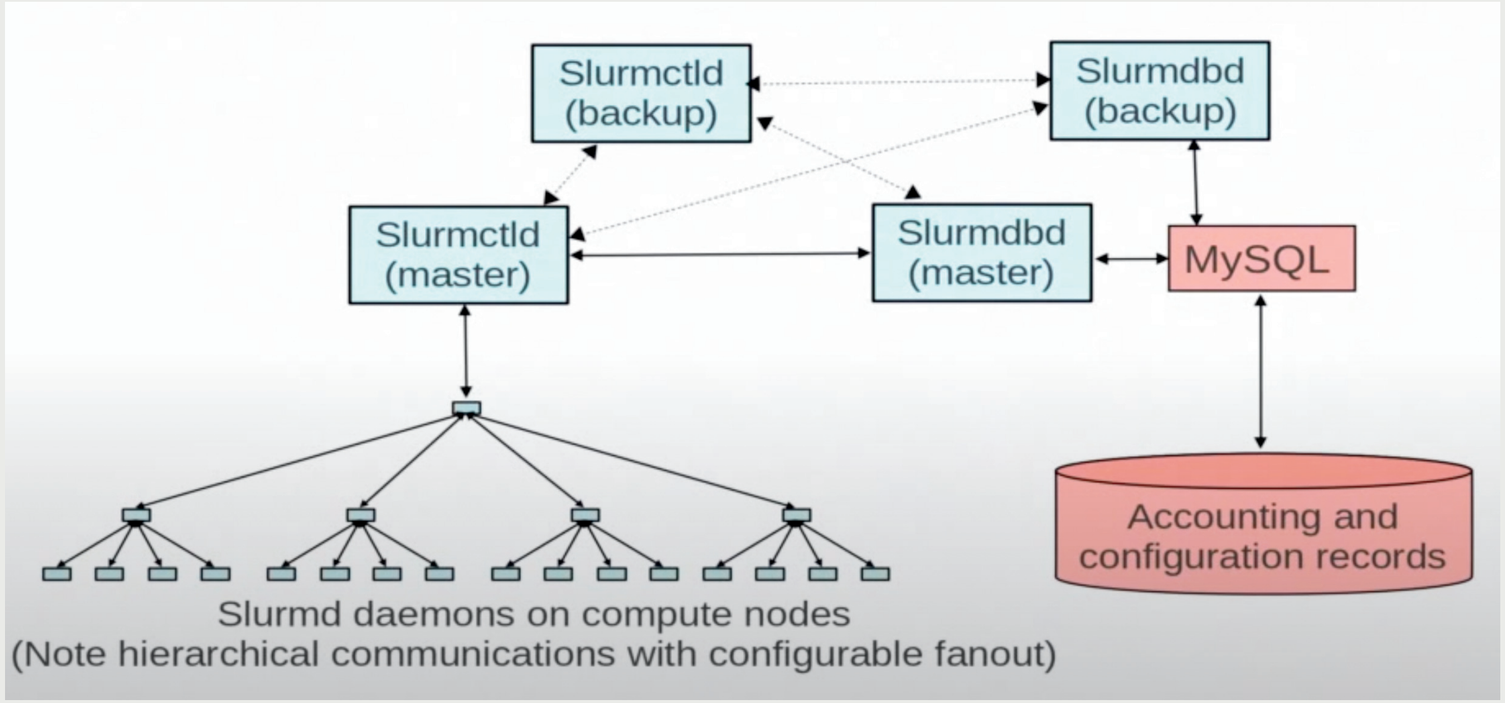
- Job submit plugin
- Call for each job submission or modification
- Can be used to set default values
- 2 functions
  - **job_submit()**
  - **job_modify()**

SLURM design and architecture

HPC

CLuster architecture

SLURM



Daemons

SLURM

- **slurmcltd**: central controller
- **slurmd**: compute node daemon
- **slurmdbd**: database daemon

Exercise: describe in details the use of such daemons

Daemon command line options

SLURM

- **-c**: clear previous stte
- **-D**: run in foreground
- **-v**: verbose
- **Example**:
  - slurmctld -Dcvvvv
  - slurmd -Dcvvv

Compute node config

SLURM

- Execute **slurmd** with **-C** option to print node's current config and exit
- Can be used as input to the **SLURM** config file

```
> slurmd -C
NodeName=jette CPUs=6 Sockets=1 CoresPerSocket=6 ThreadsPerCore=1
     RealMemory=8000 TmpDisk=930837
```

Shepherd a job step

SLURM

- One **slurmstepd** per job step
- Spawned by slurmd at job step initiation
- Manages job steps and processes I/O
- Only performs while the job step is active

SLURM build and configuration

HPC

SLURM commands : job/step allocation

SLURM

- **sbatch** - submit script for later execution
- **salloc** - create job allocation and start a shell
- **srun** - Create a job allocation and launch job
- **sattach** - connect stdin/out/err for an existing job or job step

SLURM commands : job/step allocation

SLURM

- **sbatch** - submit script for later execution
- **salloc** - create job allocation and start a shell
- **srun** - Create a job allocation and launch job
- **sattach** - connect stdin/out/err for an existing job or job step

Job/step allocation examples

Submit a sequence of three batch jobs

Submit sequence of three batch jobs

> sbatch –ntasks=1 –time=10 pre_process.bash
Submitted batch job 45001
> sbatch –ntasks=128 –time=60 --depend=45001 do_work.bash
Submitted batch job 45002
> sbatch –ntasks=1 –time=30 --depend=45002 post_process.bash
Submitted batch job 45003

Job/step allocation examples

Create allocation for 2 tasks then launch "hostname" on the allocation

Create allocation for 2 tasks then launch “hostname” on the allocation, label output with the task ID

> srun –ntasks=2 –label hostname
0: tux123
1: tux123

As above, but allocate the job two whole nodes

> srun –nnodes=2 --exclusive –label hostname
0: tux123
1: tux124

Job/step allocation examples

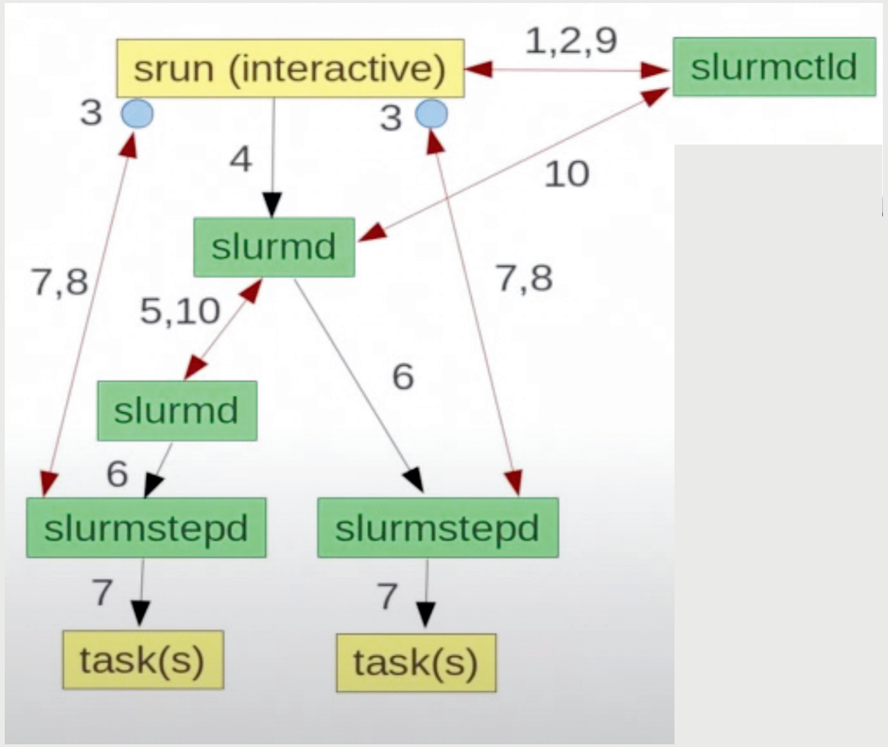
Create allocation for 8 tasks and 10 min for bash shell

Create allocation for 8 tasks and 10 minutes for bash shell, then launch some tasks

```
> salloc –ntasks=8 –time=10 bash
salloc: Granted job allocation 45000
> env | grep SLURM
SLURM_JOBID=45000
SLURM_NPROCS=4
SLURM_JOB_NODELIST=tux[123-124]
…
> hostname
tux_login
> srun –label hostname
0: tux123
1: tux123
2: tux124
3: tux124
> exit  (terminate bash shell)
```

Job execution sequence

About ?



- 1a- **srun** send job allocation request to **slurmctld**
- 1b- **slurmctld** grant allocation and returns details
- 2a- **srun** send step create request to slurmctld
- 2b- **slurmctld** responds with step credential
- 3- **srun** opens socket for I/O
- 4- **srun** forwards credential with task info to **slurmd**
- 5- **slurmd** forward request as needed
- 6- **slurmd** forks/execs **slurmstepd**
- 7- **slurmstepd** connects I/O to run and launches tasks
- 8- on task termination, **slurmstepd** notifies **srun**
- 9- **srun** notifies **slurmcltd** of job termination
- 10- **slurmctld** verifies termination of all processes via **slurmd** and releases resources for next job

SLURM commands : system information

example

- **sinfo** - report system status of nodes
- **squeue** - report job and job step status
- **smap** - report system, job or step status with topology
- **sview** - report and/or update system, job step partition or reservation status with topology
- **scontrol** - admin tool to view/update system, job, step, partition or reservation

sinfo commands

example

sinfo - report system status of nodes or partitions

```
> sinfo --Node (report status in node-oriented form)
NODELIST   NODES  PARTITION  STATE
tux[000-099]      100   batch       idle
tux[100-127]       28   debug       idle

> sinfo -p debug (report status of nodes in partition “debug”)
PARTITION AVAIL TIMELIMIT NODES NODELIST
debug              up       60:00        28 tux[100-127]

>squeue -i60   (report status every 60 seconds)
```

squeue commands

example

squeue - report status of jobs/steps in **slurmctld** daemons records

```
> squeue -u alec -t all (report jobs for user “alec” in any state)
JOBID PARTITION  NAME  USER  ST  TIME  NODES NODELIST(REASON)
45124  debug          a.out    alec     CD   0:12           1 tux123

> squeue -s -p debug  (report steps in partition “debug”);
STEPID PARTITION NAME USER TIME NODELIST
45144.0 debug          a.out    moe   12:18 tux[100-115]

>squeue -i60   (report currently active jobs every 60 seconds)
```

scontrol commands

example

scontrol - designed for system administrator use
Many fields can be modified

```
> scontrol show partition
PartitionName=debug
   AllocNodes=ALL AllowGroups=ALL Default=YES
   DefaultTime=NONE DisableRootJobs=NO GraceTime=0 Hidden=NO
   MaxNodes=UNLIMITED MaxTime=UNLIMITED MinNodes=1
   Nodes=tux[000-031]
   Priority=1 RootOnly=NO Shared=NO PreemptMode=OFF State=UP
   TotalCPUs=64 TotalNodes=32 DefMemPerNode=512 MaxMemPerNode=1024
> scontrol update PartitionName=debug MaxTime=60
```

SLURM commands : accounting

example

- **sacct** - report accounting information by individual job and job step
- **sstat** - more details than sacct
- **sreport** - report resources usage by cluster, partition, user, account, etc.

Scheduling

example

- **sacctmgr** - database management tool
  - add/delete clusters, accounts, users
  - get/set resource limits
- **sprio** - view factors comprising a job's priority
- **sshare** - view current hierarch. fair-share info
- **sdiag** - view stats about scheduling module operations

Documentation

MORE

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