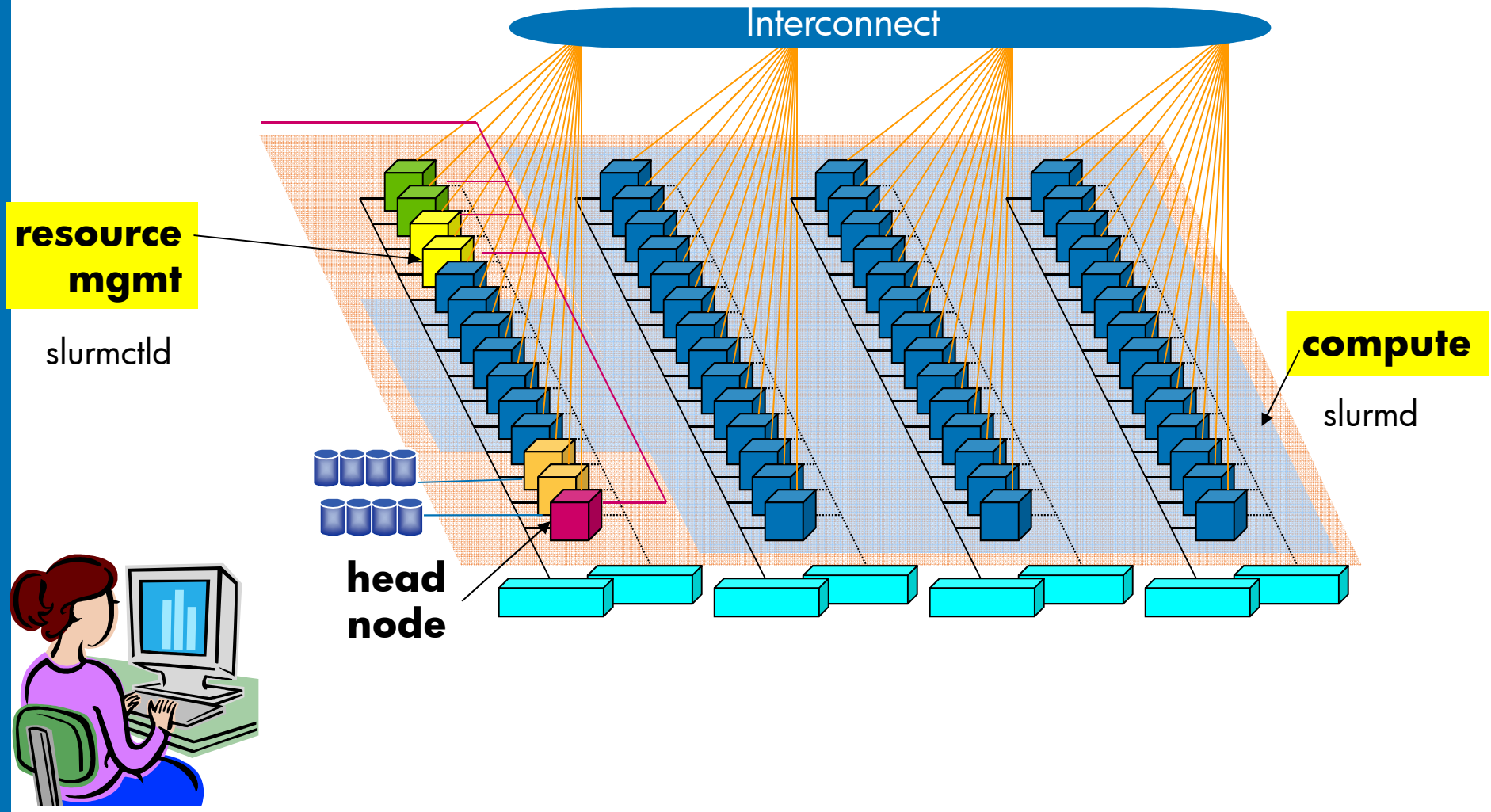
SLURM

Exercises

SLURM Architecture

- Two daemons
  - slurmctld - controller, optional backup
  - slurmd - per node worker daemon
- Some user commands
  - sacct - display job accounting information
  - scancel - signal or cancel a job or job step
  - scontrol - administration tool, get/set configuration
  - sinfo - reports general system information
  - squeue - reports job and job step information
  - smap - graphical information viewer
  - srun - submit/initiate job or job step

Slurm daemons on XC



Preparation for Slurm

- Log on as your user
- Get the examples
  - YOU already got them yesterday. To get there type
    - cd
    - cd slurm
- Make sure that the mpi module is loaded
  - module load mpi
  - module avail
- Check the your Slurm is running
  - srun hostname

scontrol

- Check the status of the partition and daemon using scontrol

[user@n16 root]# ***scontrol ping***

[user@n16 root]# ***scontrol show partition***

sinfo

- Check the status of all the nodes

[user@n260 ~]$ **sinfo**

[user@n260 ~]$ **sinfo –all**

[user@n260 ~]$ **sinfo --long --Node --exact**

srun resource request options

Short form	Long form	Description
-n	--ntasks	processors count
-N	--nodes	Node count
-c	--cpus-per-task	count of CPUs required per task
-r	--relative	start allocation on specified node within allocation
-w	--nodelist	include (at least) the listed nodes
-A	--allocate	Allocate a number of processors
-x	--exclude	exclude listed node(s)
-C	--constraint	Constrain allocation based on given feature list

Running Simple Jobs

[user@n260 ~]$ **srun hostname**

[user@n260 ~]$ **srun –N 2 hostname**

[user@n260 ~]$ **srun –c 2 hostname**

[user@n260 ~]$ **srun –c 4 hostname**

[user@n260 ~]$ **srun -N 3 –n 6 hostname**

[user@n260 ~]$ **srun -N 2 –n 8 hostname**

More Simple Jobs

[user@n260 ~]$ **srun –N2 –-label hostname**

[user@n260 ~]$ **srun –n10 –N1 –-label hostname**

[user@n260 ~]$ **srun –N2 bash**
Type “ls”
Type “ls –lart”
Type “touch `hostname`_yourname
Type “ls –lart”
double “CTRL-C” to kill

Selection/Allocation/Distribution

[user@n260 ~]$ srun -N 2 -p pappa --nodelist=n[300-301] hostname

[user@n260 ~]$ srun -N 2 -n 4 -p pappa --nodelist=n[300-301] hostname

[user@n260 ~]$ srun -N 1 -n 4 -p pappa --nodelist=n[300-302] hostname

Selection/Allocation/Distribution

[user@n260 ~]$ srun -N 3 -n 5 -p pappa hostname

[user@n260 ~]$ srun -N 3 -n 5 –ntasks-per-node=2 -p pappa hostname

Selection/Allocation/Distribution

[user@n260 ~]$ srun --nodes=3 --ntasks=4 --label --distribution=block hostname

- [user@n260 ~]$ srun --nodes=3 --ntasks=4 --label --distribution=cyclic hostname

Selection/Allocation/Distribution

[user@n260 ~]$ srun --nodes=3 --ntasks=4 --label --distribution=block hostname

- [user@n260 ~]$ srun --nodes=3 --ntasks=4 --label --distribution=cyclic hostname

Monitor Jobs

- Open another ssh connection to the cluster (lets call it T2)
- Type
  - watch “squeue –t all”
- In the initial terminal run some jobs

  $ srun –N2 sleep 60 &

  $ srun hostname

  $ srun –N 4 uptime
- In terminal T2 watch the jobs come and go….

Cancel Jobs

- In the initial terminal run again some jobs

  $ srun –N2 sleep 100 &

- Type the following commands

  $squeue (check the job ID)

  $sacct –j <Job_Id> (just to check that the job is Running)

  $sacct --long -j <Job_Id>

  $scancel <Job_id> (that you obtain in the previous command)

  $sacct –j <Job_Id> (just to check that the job was cancelled)

Details about the jobs

[user@n260 ~]$ srun --nodes=2 --ntasks=4 --ntasks-per-node=2 sleep 30&

[user@n260 ~]$ squeue → TO get the JOB ID

[user@n260 ~]$ scontrol --detail show job JOB_ID

Somewhere it is written the CPUs allocation.
NumNodes=2 NumCPUs=4 CPUs/Task=1 ReqS:C:T=*:*:*
    Nodes=n300 CPU_IDs=5-6 Mem=1024
    Nodes=n301 CPU_IDs=0-1 Mem=1024

Repet the same for

[user@n260 ~]$ srun --nodes=2 --ntasks=4 sleep 30 &

How are the processes distributed?

Details about CPUs

[user@n260 ~]$ srun --nodes=2 --ntasks=4 --tasks-per-node=2 sleep 30&

[user@n260 ~]$ squeue → TO get the JOB ID

[user@n260 ~]$ sstat -j JOB_ID –pidformat

[user@n260 ~]$ sstat --format=AveCPU,AvePages,AveRSS,AveVMSize,JobID - j JOB_ID

Running a Serial Job using SLURM

- emacs hw_hostname.c

```
#include  <unistd.h>
#include  <stdio.h>
  int main()
  {
     char  name[100];
     gethostname(name, sizeof(name));
     printf("%s says Hello!\n", name);
     return 0;
}
```

Compile the program and run it

- cc hw_hostname.c -o hw_hostname
- ./hw_hostname
  - What do you see?
  - Run the program now with srun
- srun ./hw_hostname
  - What do you see?

Run the same SERIAL program on mode nodes using srun

- srun –n10 ./hw_hostname
- srun -N2 -n10 --ntasks-per-node=5 hw_hostname

- What do the command output?

Batch Jobs

*emacs script

```
##!/bin/bash

#SBATCH --job-name=<YOUR_USERNAME>
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
#SBATCH --get-user-env
#SBATCH -D  /home/<YOUR_USERNAME>

hw_hostname
```

Batch Example without srun

$ **sbatch script**

$ squeue –t all

$ **ls**

script slurm-XXXX.out

$ cat slurm-XXXX.out

Output goes to slurm-`job id`.out

Modify script as follows:

*emacs script

```
##!/bin/bash

#SBATCH --job-name=<YOUR_USERNAME>
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
#SBATCH --get-user-env
#SBATCH -D  /home/<YOUR_USERNAME>
#SBATCH  -o /home/<YOUR_USERNAME>/JOB.%j.%N.out


srun -n 2 hw_hostname
srun  -n 2 hostname
```

Batch Example without srun

$ **sbatch script**

Output goes to slurm-`job id`.out

$ squeue –t all

$ **ls**

script slurm-XXXX.out JOB.%j.%N.out

$ cat JOB.%j.%N.out

BATCH JOB VARIABLES

- run a “man sbatch” command but some of them are:

- SBATCH_JOBID Same as --jobid
- SBATCH_JOB_NAME Same as -J, --job-name
- SBATCH_MEM_BIND Same as --mem_bind
- SBATCH_CPU_BIND Same as --cpu_bind
- SBATCH_DEBUG Same as -v, --verbose
- SBATCH_DISTRIBUTION Same as -m, --distribution
- SBATCH_EXCLUSIVE Same as --exclusive
- SBATCH_OVERCOMMIT Same as -O, --overcommit
- SBATCH_PARTITION Same as -p, --partition

Run parallel jobs

- cp –r /home/hpcscrmt/apps .
- cd apps
- Running Hello_world

- srun --nodes=2 --ntasks=4 --label hello

- srun -N2 -n2 --ntasks-per-node=1 hello

- srun --nodes=2 --ntasks=4 --ntasks-per-node=2 --label hello

Run parallel jobs

- cd apps
- srun -n2 ping_pong_ring
- srun -n2 -N1 ping_pong_ring
- srun -n2 -N1 ping_pong_ring 100000

- srun -n2 -N2 ping_pong_ring
- srun -n2 -N2 ping_pong_ring 1000000

- srun -n4 -N2 ping_pong_ring
- srun -n4 -N2 ping_pong_ring 1000000

- srun -n4 -N2 --ntasks-per-node=2 ping_pong_ring
- srun -n4 -N2 --ntasks-per-node=2 ping_pong_ring 1000000

Run parallel jobs

- default : export I_MPI_FABRICS=dapl
- srun -n4 -N2 --ntasks-per-node=2 ping_pong_ring
- srun -n4 -N2 --ntasks-per-node=2 ping_pong_ring 1000000

- export I_MPI_FABRICS=tcp
- srun -n4 -N2 --ntasks-per-node=2 ping_pong_ring
- srun -n4 -N2 --ntasks-per-node=2 ping_pong_ring 100000

- export I_MPI_FABRICS=shm
- srun -n4 -N2 --ntasks-per-node=2 ping_pong_ring

- export I_MPI_FABRICS=shm:dapl
- srun -n4 -N2 --ntasks-per-node=2 ping_pong_ring
- srun -n4 -N2 --ntasks-per-node=2 ping_pong_ring 1000000

Monitor Parallel Programs

- Run again the pingpong but with

- $sbatch -n 8 script_ping.sh 8
- cat slurm-JOBID.out

- In T2 type the commands

```
$ squeue  --format "%i  %u  %C"     (change format of squeue)
$
```


Questions?