Introduction to HPC @ RCC

January 18, 2017

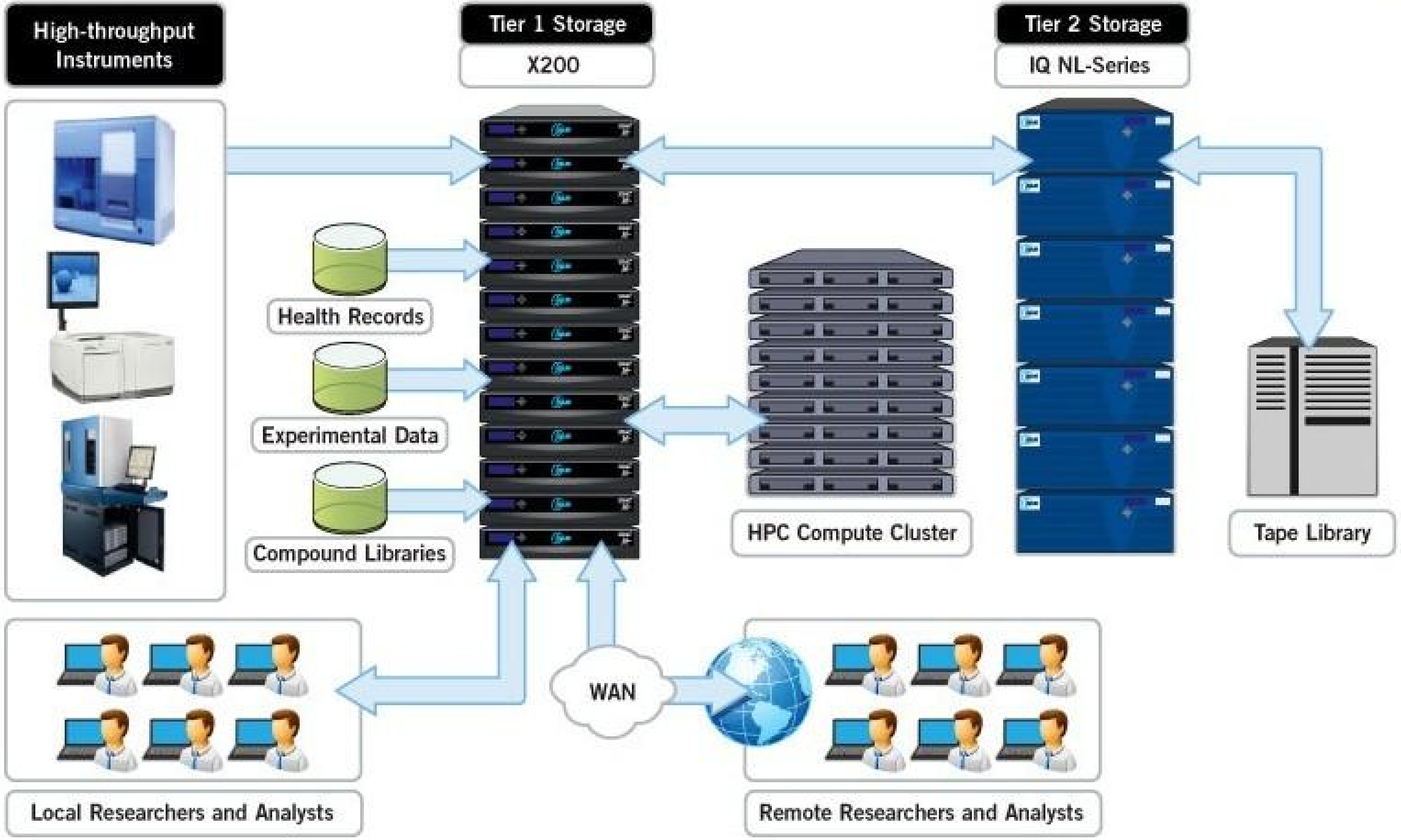
Research Computing Center

FLORIDA STATE UNIVERSITY
RESEARCH COMPUTING CENTER

What is HPC

“High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business”

Typical HPC Workflow



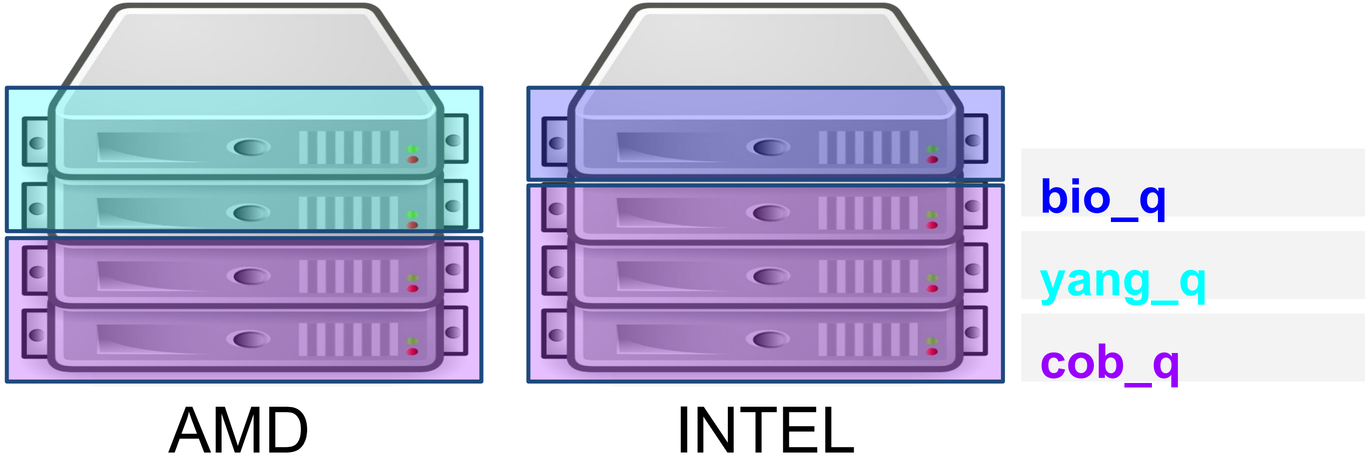
How to allocate resources?

Job scheduler

Partitions?

- Collection of nodes
- Access is granted through an account
- Users can run jobs on “their” account
- Spawn different architectures (eg AMD)
  - Jobs can not spawn different architectures
- Similar to Queues
  - Direct mapping account - partition (RCC)

Nuts and Bolts



- Submit job to a partition.
  - partitions are managed by RCC staff
- Membership to accounts determines who can submit to which partition.
  - accounts are managed by ‘owners’
- Feature (--constraint) determines where the job will run. Default: any.

Commands

SLURM		
sbatch	sbatch -p myqueue myjobscript.sh	Submit a batch script
srun & salloc	srun myprogram.exe salloc myprogram.exe	Submit an interactive program
squeue	squeue -p mypartition	Show jobs in a mypartition
squeue	squeue -j 1251 scontrol show job 1251	Inspect a specific job
squeue	squeue -j 1252 --start	Show start time of job
scancel	scancel 1251	Cancel a job
sinfo	sinfo -p mypartition	Shows nodes in mypartition

https://rcc.fsu.edu/docs/hpc-cheat-sheet

How to submit a job

1. **sbatch**
   non-interactive batch submission
   schedules job in background
2. **srun** & **salloc**

   interactive submission

   srun/salloc run program in foreground

   srun can also be used in batch script!

Submit jobs: sbatch

```
sbatch {flags} myscript
```

- man sbatch
- sbatch -p myqueue -n 10 myscript
  - request 10 cores from the myqueue queue and run *myscript* job script
- sbatch myscript
  - request 1 core from my default queue
- sbatch -D myproject/workdir myscript
  - start job in $HOME/myproject/workdir folder

srun vs salloc for submission

```
srun {flags} program
salloc {flags} program
```

- -n X: both will allocate X cores
  - srun will start program X times
  - salloc will start 1 instance program

srun to submit a job

- man srun
- srun from a submit node will start a new job
  - srun -p myqueue myprogram
- will not run in the background (unless &)
- srun -n *x* myscript.sh will start *x* instances of myscript.sh
  - srun will not “interpret” scripts: ignore #SBATCH flags

srun in job scripts

- slurm enabled replacement of mpirun
- mpirun is no longer supported (mvapich2)
- srun myprogram
  - will run myprogram on requested number of cores (sbatch -n *x*)
- srun -n *y* myprogam
  - will run myprogram on *y* number of cores
  - error if *y*>*x* (sbatch -n *x*)
- be careful when you use srun in a script submitted by srun

Memory

- Slurm takes memory in consideration
- Default is 4GB per core (2GB backfill{2})
- --mem-per-cpu=<MB> or --mem=<MB>
- Under the hood: memory is “mapped” to cores:
  - -n 1 --mem=5GB will reserve 2 cores on a node.
- Memory limit is enforced

s* caveats

- Jobs will start in the current working directory (unless -D flag was used)
  - moab: job will always start in home directory
- Job environment is a copy of your working environment (except for limits)
  - environment variables
  - be careful what modules you autoload in your ~/.bashrc
- sbatch is not for interactive jobs

Common flags for s*

- *-n number* : Request *number* of cores
- *-p partition* : Run a job on this queue
- *-C feature* : Restrict job to nodes with this feature
- *--exclusive* : Do not share nodes with other jobs
- *-J jobname* : job name (not outputfile)
- *-o outputfile* : output file (default slurm)
- *--mail-type=X* : receive this type of notifications
  (ALL, BEGIN, END, FAIL)

Less Common flags

- *--begin=time* : Start a job at time *time*
- *--output=slurm.%N.%j.out* : output log
- *--input=inputfile.txt* : use text from file for std input
- --pty : interactive job, only for srun!

Submit a job

```
sbatch -p bio_q mywrf.sh
srun -p cob_q --constraint=AMD XYZ.exe
sbatch -p yang_q,bio_q job.sh
sbatch -o myjob.%j.out myjob.sh
srun --pty /bin/bash
```

Interactive jobs --pty

srun --pty someprogram

srun --pty /bin/bash

srun --pty R

srun --pty gdb myprogram

- srun -n *x* --pty program will start 1 instance
- srun will start from your submit directory

Job script for parallel program

```
#!/bin/bash

#SBATCH -J MYJOBNAME
#SBATCH -n 10

module load gnu-openmpi
pwd
srun myprogram
```

Run a sequential program

```
#!/bin/bash

#SBATCH -J MY-R-CODE
#SBATCH --input myRinput.txt

pwd
module load R
R --no-save
```

Job Array

- Job arrays are a way to efficiently submit large numbers of jobs.
- Single program with a lot of different datasets
- sbatch **--array=1-10** program.sh
  - $SLURM_ARRAY_TASK_ID

Disclaimer

- There are 2 sites about slurm. One is outdated:
  - ~~computing.llnl.gov~~: original project site
    - refers to version 2.3
  - http://www.schedmd.com: correct website
- Don’t use mpirun, use srun
  - mpirun still available for openmpi
  - both openmpi and mvapich2 support srun

Errors

```
#SBATCH -n 4
srun -n 5 myprogram
```

```
srun: error: Unable to create job step: More
 processors requested than permitted
```

Errors

```
srun --constraint "X&Y"  myprogram
```

```
srun: error: Unable to allocate resources:
  Requested node configuration is not
  available
```