CHL Lab

10/27/17-11/01/17: Checking out TACC platforms

Rosalind Xu 18′

login: ssh -l rosjxu login.xsede.org
pw: Hc

see a list of platforms: xsede-gsissh-hosts

to switch into a resource: gsissh stampede2

OR: ssh directly to
rosjxu@stampede2.tacc.utexas.edu

(pw: Hc)

Three Directories:
$HOME
$WORK
$SCRATCH

All computations should be conducted in $SCRATCH (files will be purged if access time is more than 10 days old). Files can then be moved to $HOME or $WORK for permanent storage.

To copy from/to $WORK:
scp ./myfile rosjxu@stampede2.tacc.utexas.edu:\$WORK/

Use rsync to update a large data set (instead of copying and pasting every single small file):
Syncing ./mybigfile with rosjxu@stampede2.tacc.utexas.edu:\$WORK/data/mybigfile:

```
	rsync  -avtr   mybigfile    rosjxu@stampede2.tacc.utexas.edu:\$WORK/data
```

OR:

```
	rsync  -avtr   mybigfile/*    rosjxu@stampede2.tacc.utexas.edu:\$WORK/data/mybigfile
```

DO not move files too frequently between $HOME / $WORK and $SCRATCH, especially if that file is large!

Sanity Check:
module load sanitytool
sanitycheck

Good Citizenship:
Log in nodes: A login node is a good place to *edit and manage files*, *initiate file transfers*, *compile code*, *submit new jobs*, and *track existing jobs*.

1. Don't run research applications (R, MATLAB, etc.)
2. Don't launch too make simultaneous processes: make -j 16 (compile on 16 cores) is rude
3. Don't run anything resource intensive

Shared Lustre file systems:

1. Stripe the receiving directory before transferring large files (a few hundred GB) to that directory
2. Don't run jobs in $HOME. $HOME is NOT for parallel jobs!
3. Run I/O intensive jobs in $SCRATCH rather than $WORK (remember your core load < 100 % problems?)
    1. If you stress $WORK, you affect every user on every TACC system!
4. If you suspect you workflow is I/O intensive, don't submit a bunch of simultaneous jobs.
    1. Writing restart/snapshot files can stress the system; avoid doing so too frequently.
5. Watch your file quotas.
    1. If you are near quota in $WORK and your script keeps trying and failing to write to $WORK, you will stress the system
    2. If you are near quota in $HOME, jobs run on $HOME, $WORK, or $SCRATCH may fail, since all jobs write to $HOME/.slurm
6. Avoid opening and closing files repeatedly in tight loops.
7. Avoid storing many small files in a single directory, and avoid workflows that require many small files.
    1. A few hundred small files is fine, but tens of thousands is too many.
    2. Group small files in separate directories of manageable size.

Internal and external networks:

1. Avoid too many simultaneous file transfers.
    1. Two or three concurrent scp sessions is fine; twenty is not
2. Avoid reclusive file transfers (scp -r), especially one involving many small files!
    1. Create a tar archive before transfers: tar -cvzf [filename].tar.gz [filename]

Submitting jobs:

1. Do not ask for more time than you need.
    1. The scheduler will have an easier time to find you a 2 h slot instead of a 48 h slot; which means shorter waiting time for you and everyone else.
2. Test your submission scripts.
    1. Make sure everything works on 2 nodes before you try 200
    2. Debug the submission code with 5-minute jobs: “hello world” codes; one-liners like “ibrun hostname”.

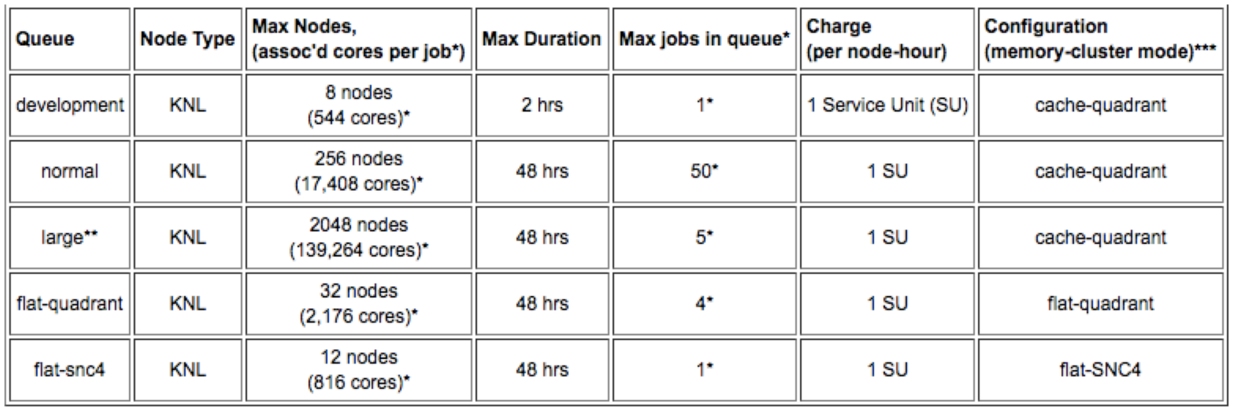
Tar:

```
tar -cvzf   [filename].tar.gz   [filename]     —> compress
tar -xzvf   [filename].tar.gz                  —> decompress
```

Running Jobs:

Job Accounting: SU = #nodes * #wall-clock-hours * job-multiplier

Slurm Partitions:



Queue	Node Type	Max Nodes, (assoc'd cores per job*)	Max Duration	Max jobs in queue*	Charge (per node-hour)	Configuration (memory-cluster mode)***
development	KNL	8 nodes (544 cores)*	2 hrs	1*	1 Service Unit (SU)	cache-quadrant
normal	KNL	256 nodes (17,408 cores)*	48 hrs	50*	1 SU	cache-quadrant
large**	KNL	2048 nodes (139,264 cores)*	48 hrs	5*	1 SU	cache-quadrant
flat-quadrant	KNL	32 nodes (2,176 cores)*	48 hrs	4*	1 SU	flat-quadrant
flat-snc4	KNL	12 nodes (816 cores)*	48 hrs	1*	1 SU	flat-SNC4

Slurm Commands:

```
    sbatch                          —> job submission
    sbatch --dependency=afterok:[JOBID] [new-job-script]
                                    —> start new-job-script after [JOBID]
completes successfully
    sinfo -o "%18P %8a %16F"        —> monitoring node availability
    squeue -u rosjxu                —> monitoring job status
    showq -u                        —>  alternative way to monitor job status
    squeue --start -j [JOBID]       —> estimate start time for a job
    scancel [JOBID]                 —> cancel job
```

scontrol show job=[JOBID]   —> detailed info on job
sacct -u rosjxu --starttime 2017-08-01 --format=JobID,JobName,MaxRSS,Elapsed
—> statistics on completed jobs started on or after 2017-08-01

Environmental variables:
Do not use the Slurm "--export" option to manage your job's environment: doing so can interfere with the way the system propagates the inherited environment. Instead, use export env-var=$val to set environmental variables before submitting a job.

Types of Jobs:

1. Single serial jobs
    1. Serial jobs require 1 task only per job
    2. -N=1   -n=1

2. Single MPI jobs: ibrun [job-script]
    1. If there is little rush in time, it is recommended to run on single node (-N=1)
    2. The max number of mpi processes (-n) per node is 64-68. Start small (say 32), then increase gradually

3. Single OpenMP jobs

4. More than one serial application in the same job: module *launcher*
    1. module load launcher
    2. Read $TACC_LAUNCHER_DIR/README
    3. With launcher, multiple jobs can be submitted to the same node
    4. More processes on same node >> using more nodes
    5. Do scaling to find out optimal # of jobs per node
    6. -N=nNodes   -n=nNodes*optimal#Jobs

5. Interactive sessions: idev or srun
    1. idev -p normal -N 2 -n 8 -m 150 # normal queue, 2 nodes, 8 total tasks, 150 minutes
    2. srun --pty -N 2 -n 8 -t 2:30:00 -p normal /bin/bash -l # same conditions as above

GROMACS

gmx	—> run gmx features
ibrun mdrun_mpi	—> run mdrun with MPI parallelization

SolEFP:

```
python run_biomol.py             —> run biomol
python run_small.py              —> run small molecules
 slv_calc-expres, slv_calc-ftir  —> run slv modules
```