Courteous Job Scheduling on ACCRE: How Not to be “That Guy”

SDT

May 10, 2016

Background and Terminology

ACCRE is a shared resource and every account has a limit on the number of those resources it may access. The higher the limit, the more an account is allowed to "borrow" from other accounts when their resources are not in use.

Fairshare: Your "share" in ACCRE. Determines group priority and burst limits

Burst Limits: The maximum resources available (concurrently) to a group

Queue: All jobs waiting to run on ACCRE, in order of priority

Priority: Determined by fairshare, resource requests, recent resource usage, and time in queue. Determines when a job will be scheduled to run.

Scheduler: An algorithm that tracks job priority and resource requests and attempts to optimize job scheduling

Gateway: An ACCRE server that is not used for cluster computation, only for local computing (chgr2)

DORS: A filesystem that is independent from ACCRE, but available to all ACCRE gateways and compute nodes

Know Your Limits

How to find account limits

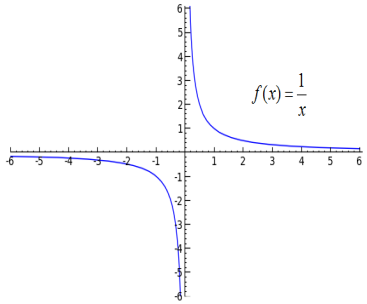
```
$ showLimits -g capra_lab
ACCOUNT          GROUP    FAIRSHARE     MAXCPUS    MAXMEM(GB)  MAXCPUTIME(HRS)
-------------------------------------------------------------------------------
capra_lab_account            16          272        2720           26112
            capra_lab        1            -           -               -
-------------------------------------------------------------------------------

$ showLimits -g chgr
ACCOUNT          GROUP    FAIRSHARE     MAXCPUS    MAXMEM(GB)  MAXCPUTIME(HRS)
-------------------------------------------------------------------------------
chgr_account                 10          200        2000           19200
            chgr                        1           -           -               -
-------------------------------------------------------------------------------
```

How to submit jobs to specific accounts

```
$ sbatch --account=capra_lab <job_name>.slurm

$ sbatch --account=chgr <job_name>.slurm
```



Know Your Limits

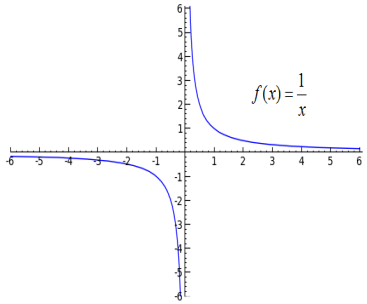
The VAMPIRE Cluster (Feb 2016)

# Processor Cores	Processor Type	Processor Speed (GHz)	Memory* (GB)	Processor Model
2184	Dual Quad	2.3	**24.0-128.0**	Intel Xeon Westmere
1056	Dual Hex	2.4-3.0	**48.0-64.0**	Intel Xeon Westmere
2352	Dual Hex	1.9	**128.0-256.0**	Intel Xeon Sandy Bridge
504	Dual Hex	2.4	**128.0**	Intel Xeon Haswell

- Request resources with wide availability when possible
- Don't under-request.
- Run a test job to ensure resources are adequate (spy on a node!)

How to watch a running ACCRE job

```
ssh vmp722
top -Mu <user>
```



Know Your Limits

Getting info on a node

Can come in handy for seeing how many CPUs or how much memory is available on a given node.

Here we can see that vmp620 has 8 CPUs and 20.5Gb of memory, and that node is pretty much fully loaded, with 7.02/8 CPUs working and 19.7/20.5Gb allocated.

```
$ scontrol show node vmp620

NodeName=vmp620 Arch=x86_64 CoresPerSocket=4
   CPUAlloc=8 CPUErr=0 CPUTot=8 CPULoad=7.02 Features=intel
   Gres=(null)
   NodeAddr=vmp620 NodeHostName=vmp620 Version=14.11
   OS=Linux RealMemory=20500 AllocMem=19700 Sockets=2 Boards=1
   State=ALLOCATED ThreadsPerCore=2 TmpDisk=0 Weight=1
   BootTime=2014-04-22T14:27:10 SlurmdStartTime=2015-01-20T15:09:59
   CurrentWatts=0 LowestJoules=0 ConsumedJoules=0
   ExtSensorsJoules=n/s ExtSensorsWatts=0 ExtSensorsTemp=n/s
```

Simple Jobs

Most scripts and programs can be run with minimal resource requests as a single job

```
$ cat simple.slurm
### Requests one CPU and 5GB of memory
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --mem=5GB
```

High memory jobs
e.g. BLAST searches against large databases, de novo assembly

Don't forget to leave some memory for the OS—for most nodes this means:
Available memory = Base memory – 4.8 GB

```
### Requesting up to 19.2G of memory can run on any node.
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --mem=19.2GB

### A job requesting 120G can only run on select nodes.
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --mem=120GB
```

Multi-Threaded Jobs

e.g., BLAST, TopHat, RAxML

```
### Be sure to specify the number of threads in the program you are
running.
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --mem=19200
```

e.g.: blastp -num_threads 8 [rest of blast command]

Warning: Some programs will allow you to request all available CPUs (mafft --thread -1).

If you use this flag, make sure your request the full node. The program won't know that you were allocated only 5 out of 8 nodes. It will try to use them all, and if another person is also using trying to use those CPUs, you will both experience a significant slow down.

Strategies for Launching Large Jobs

Beyond multithreading supported by software, there are several user initiated strategies to minimize impacts of larger jobs:

- Split input files into smaller files and run multiple jobs
- Run more jobs with shorter wall times *(works up to a point)*
- SLURM batch arrays allow for easy parallelization `#SBATCH --array=0-99%20`
- SLURM jobs may be submitted in succession to allow for easy handling of split outputs `sbatch --dependency=afterok:$FIRST`
- Ask your neighbors (and be willing to help your neighbors)

- Finally--check cluster load to know when you should get worried/frustrated/ticked-off

Batch Array Jobs

Embarrassingly parallel tasks can be easily performed in parallel
Requires only one SLURM script

```
SUBMITDATE=`scontrol show job
            ${SLURM_ARRAY_JOB_ID} |
            grep SubmitTime |
            head -n1 |
            cut -f2 -d"=" |
            cut -f1 -d" " |
            cut -c1-10`
```

```
$ cat script.py
import sys
array_id  = int(sys.argv[0])
total_ids = int(sys.argv[1])
all_files = [< list of files >]
### Reduce to files assigned to this job
files4me  = [f for i,f in enumerate(all_files) if i%total_ids==array_id]
```

```
$ cat batch.slurm
### Resource allocations for each job in the array
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --mem=5GB
#SBATCH -time=1-0
### Call script.py, passing this task ID and the total number of tasks
./script.py ${SLURM_ARRAY_TASK_ID} $1

### Submit 100 jobs via batch.slurm with IDs 0-99
$ sbatch --array=0-99 batch.slurm 100

### Submit 100 jobs via batch.slurm, but only schedule 20 at a time
$ sbatch --array=0-99%20 batch.slurm 100
```

Checking the cluster load

Checking the cluster utilization can give you a sense of how long your job could wait in the queue. In this example, 5,755,386 CPU minutes (95,923 CPU hours) are allocated, and 1,346,531 minutes (22,442 hours) are idle. This suggests that there probably resources available for your job.

```
$ sreport cluster utilization
--------------------------------------------------------------------------------
Cluster Utilization 2015-01-26T00:00:00 - 2015-01-26T23:59:59 (86400*cpus secs)
Time reported in CPU Minutes
--------------------------------------------------------------------------------
  Cluster    Allocated       Down  PLND Down         Idle  Reserved     Reported
--------- ------------ ---------- ---------- ------------ --------- ------------
    accre      5755386      56254          0      1346531    426980      7585152
```

Checking group usage

```
$squeue --format=%C --account capra_lab -h | paste -sd+ | bc
84
```

Here we are using 84, which is under our limit. Assuming you don't need hundreds of CPUs or gigabytes of memory, your job should start running very quickly. If it doesn't, double check your SBATCH commands to make sure you are requesting the appropriate resources.

Monitoring and Altering SLURM Jobs

PENDING and RUNNING jobs may be altered:

- Put a job on hold `: $ scontrol hold <JOBID>`
- Restart a held job `$ scontrol release <JOBID>`
- Jobs (theoretically) can even be suspended (`scontrol suspend`) and resumed but this is not recommended and on ACCRE you need to submit a ticket to initiate

Leverage .bashrc for a Happier Life

```
# two commands I use to see what jobs I'm running personally, and what jobs our lab is running
alias myq="squeue --format='%.18i %.8j %.10u %.10a %.8T %.10M %.15l %.5D %.5C %.10m %R' --user
\"$USER\""
alias grpq="squeue --format='%.18i %.8j %.10u %.10a %.8T %.10M %.15l %.5D %.5C %.10m %R' --account
capra_lab"
alias grpcpu='squeue --format=%C --account capra_lab -h | paste -sd+ | bc'

# Checking completed jobs
alias sacct="sacct --allusers --
format=User,JobID,JobName,account,Timelimit,elapsed,TotalCPU,CPUTime,ReqMem,MaxRss,MaxVMSize,nnodes,ncp
us,nodelist,Start,End,ExitCode,state"
alias mycd="sacct  --user \"$USER\" -s cd"     #list just my jobs that complete successfully
#alias myfl="sacct  --user \"$USER\" -s f"     #list just my jobs that fail
alias myrn="sacct  --user \"$USER\" -s r"     #list just my jobs that are currently running.
alias myto="sacct  --user \"$USER\" -s to"     #list just my jobs that timeout

# Getting info on a job or node
alias jinfo='scontrol show job'
alias ninfo='scontrol show node'

# Checking the cluster load
alias load='sreport cluster utilization'
alias idle='sinfo --states=IDLE'
alias jobhog='sreport user top'
alias usage='sreport cluster AccountUtilizationByUser'

# Checking user and group quota
alias myquota='mmlsquota -u \"$USER\" --block-size auto'
alias mydu='du -h --max-depth=1'
alias grpfair='sacctmgr show account name=capra_lab_account,chgr_account WithAssoc
format=account%15,user,fairshare,grpcpus,maxcpus,GrpMem,GrpWall,GrpCPUMins'
```

Questions/Discussion