Intermediate Usage of
Shared Computing Cluster (SCC)

Charles Jahnke
Research Computing Services
Information Services & Technology

BOSTON UNIVERSITY

Topics for Today

- Introduction
- The Cluster
- Running Jobs
- Monitoring Jobs
- Resource Bottlenecks
- Workflow and Job Management
- Bonus Material

- Not Hands-on
- Present concepts
- Discuss common challenges
- Please ask questions
- Time at the end for personal help

Research Computing Services

This will be quick.

Research Computing Services (RCS)

A group within Information Services & Technology at Boston University provides computing, storage, and visualization resources and services to support research that has specialized or highly intensive computation, storage, bandwidth, or graphics requirements.

Three Primary Services:

- Research Computation
- Research Visualization
- Research Consulting and Training

Me

- Systems Programmer and Administrator
- Background in biomedical engineering, technology, and bioinformatics
- Office on the Boston University Medical Campus
  - We also have staff on the Charles River Campus
- Contact:
  - Email: cjahnke@bu.edu
  - Phone: 617-638-7727
  - Office: 801 Mass Ave, Boston, Suite 485

Our whole team: help@scc.bu.edu

You

- Who just came from the Intro to SCC tutorial?
- Who has an account on SCC?
- Who has used SCC for more than 1 month?
- 6 months?
- 1 year?
- Who has used other compute clusters?

The Shared Computing Cluster

Shared Computing Cluster

- **Shared** - Transparent multi-user and multi-tasking environment
  - Also, buy-in

- **Computing** - Heterogeneous environment for
  - Interactive jobs
  - Single processor and parallel jobs
  - Graphics job

- **Cluster** - Many connected computers
  - Connected via fast local area network
  - Jobs scheduler coordinates work loads.

Massachusetts Green High Performance Computing Center

Boston University

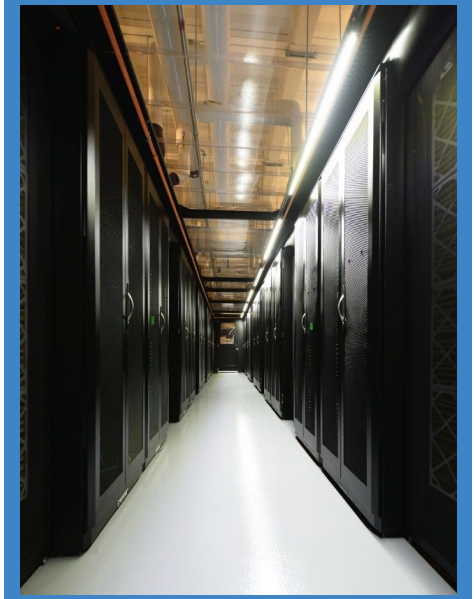
Northeastern University

MIT

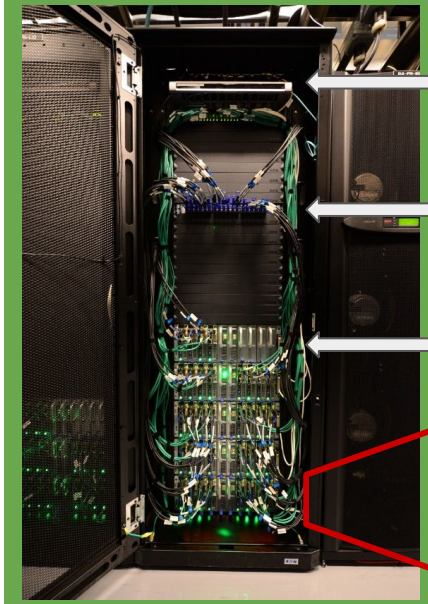
UMASS

HARVARD

Shared Computing Cluster

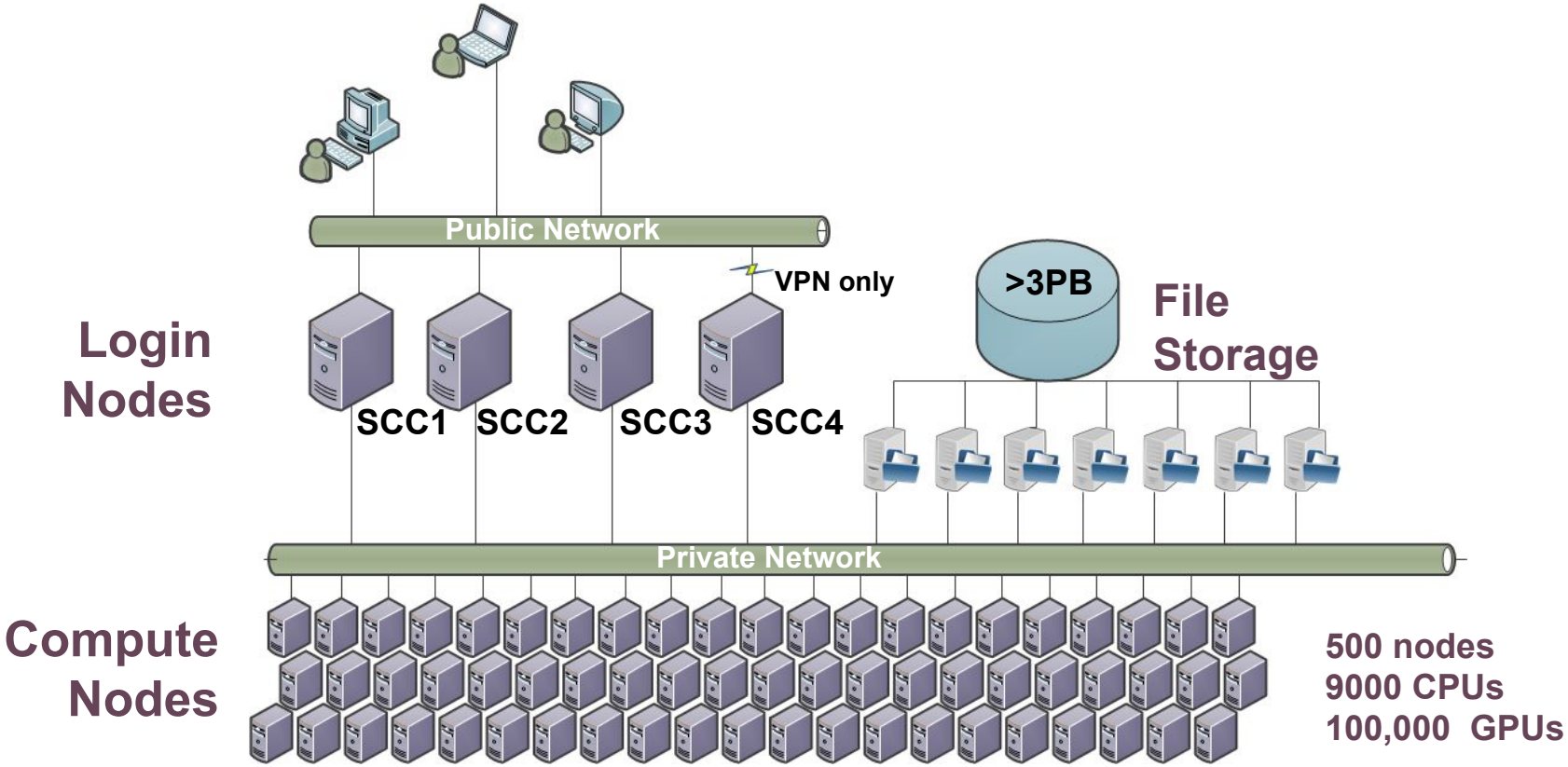


Server Cabinets



Rear View

SCC Architecture



SCC Resources

- Processors: Intel and AMD
- CPU Architecture: bulldozer, sandybridge, ivybridge, haswell, broadwell
- Ethernet connection: 1 or 10 Gbps
- Infiniband: FDR, QDR (or none)
- GPUs: NVIDIA Tesla K40m, M2070 and M2050
- Number of cores: 8, 12, 16, 20, 64 / node
- Memory: 24GB – 1TB / node
- Scratch Disk: 244GB – 886GB /node

http://www.bu.edu/tech/support/research/computing-resources/tech-summary/

Running Jobs

Interactive, Interactive Graphics, Batch

The Login Nodes

Login nodes are designed for light work:

- Text editing
- Light debugging
- Program compilation
- File transfer

Anything else should be done on a compute node with a “Job”

Running Jobs: Types of Jobs

Interactive job

- Interactive shell, run GUI applications, code debugging, benchmarking of serial and parallel code performance.

Interactive Graphics job

- Interactive shell with GPU and hardware acceleration for software with advanced graphics.

Non-Interactive “Batch” Job

- Controlled script or binary execution.

Interactive Jobs (qrsh)

"qrsh" - Request from the queue (q) a remote (r) shell (sh)

- Interactive shell
- GUI applications
- Code debugging
- Benchmarking

```
[cjahnke@scc1 ~]$ qrsh -P project
********************************************************************************
        This machine is governed by the University policy on ethics.
          http://www.bu.edu/tech/about/policies/computing-ethics/

                This machine is owned and administered by
                           Boston University.

 See the Research Computing web site for more information about our facilities.
                http://www.bu.edu/tech/support/research/

       Please send questions and report problems to "help@scc.bu.edu".

********************************************************************************

[cjahnke@scc-pi4 ~]$
```

Interactive Jobs

(`qrsh`)

Multiple Interactive Modes!

- Mostly do the same thing.
- We usually teach `qrsh`
  - Doesn't require X11
- You have options.

	qsh	qlogin / qrsh
X-forwarding is required	✓	—
Session is opened in a separate window	✓	—
Allows X-Forwarding	✓	✓
Current environment variables passed to session	✓	—
Batch-system environment variables ($NSLOTS, etc.)	✓	✓

Interactive Jobs

(`qrsh`)

Request appropriate resources for the interactive job:

- Some software (like MATLAB, STATA-MP) might use multiple cores.
- Make sure to request enough time if you need more than 12 hours.
- Make sure to request enough RAM (4GB min) is often not enough.

More on this later.

Interactive Graphics (VirtualGL) Jobs (`qvgl`)

Preface: The majority of graphical applications perform well using VNC.

- Some require OpenGL for full 3D hardware acceleration
  - fMRI Applications
  - Molecular Modeling
- This job type combines dedicated GPU resources with VNC.

- VirtualGL offering is a very limited resource.
- Most applications (MATLAB, RStudio, QGIS, etc) do **NOT** need this.

http://www.bu.edu/tech/support/research/system-usage/running-jobs/virtual-gl/

Non-Interactive “Batch” Jobs (qsub)

- **Using a Script**

Submit script to scheduler

After completion, we get an output file.

```
[cjahnke@scc1 ~]$ qsub test.qsub
Your job 9253374 ("test") has been submitted
[cjahnke@scc1 ~]$
[cjahnke@scc1 ~]$ ls
test.qsub    test.o9253374  results
```

- **Using a Binary**

Submit binary “cal”

```
[cjahnke@scc1 ~]$ qsub -b y cal -y
Your job 542359 ("cal") has been submitted
[cjahnke@scc1 ~]$
[cjahnke@scc1 ~]$ ls
cal.e542359  cal.o542359
```

Non-Interactive “Batch” Scripts (qsub)

Script Interpreter

Scheduler Directives

Task Commands

```
#!/bin/bash -l

#$ -P rcs
#$ -N test
#$ -j y
#$ -m bae


echo "=============================================="
echo "Starting on : $(date)"
echo "Running on node : $(hostname)"
echo "Current directory : $(pwd)"
echo "Current job ID : $JOB_ID"
echo "Current job name : $JOB_NAME"
echo "=============================================="

module load R
sleep 10

echo "=============================================="
echo "Finished on : $(date)"
echo "========= ==========================================="
```

Scheduler Options - General Directives

General Directives	
Directive	Description
-P *project_name*	Project to which this jobs is to be assigned. **Mandatory** for all users associated with any BUMC project.
-N *job_name*	Specifies the job name. The default is the script or command name.
-o *outputfile*	File name for the stdout output of the job.
-e *errfile*	File name for the stderr output of the job.
-j y	Merge the error and output stream files into a single file.
-m *b|e|a|s|n*	Controls when the batch system sends email to you. The possible values are – when the job begins (b), ends (e), is aborted (a), is suspended (s), or never (n) – default.
-M *user_email*	Overwrites the default email address used to send the job report.
-V	All current environment variables should be exported to the batch job.
-v *env=value*	Set the runtime environment variable *env* to *value*.
-hold_jid *job_list*	Setup job dependency list. *job_list* is a comma separated list of job ids and/or job names which must complete before this job can run. See Advanced Batch System Usage for more information.

Scheduler Options - Resource Directives

Directives to request SCC resources	
Directive	Description
-l h_rt=*hh:mm:ss*	Hard runtime limit in *hh:mm:ss* format. The default is 12 hours.
-l mem_total =#*G*	Request a node that has at least this amount of memory. Current possible choices include 94G, 125G, 252G 504G.
-l mem_per_core =#*G*	Request a node that has at least these amount of memory per core.
-l cpu_arch=*ARCH*	Select a processor architecture (sandybridge, nehalem, etc). See Technical Summary for all available choices.
-l cpu_type=*TYPE*	Select a processor type (E5-2670, E5-2680, X5570, X5650, X5670, X5675). See Technical Summary for all available choices.
-l gpus=*G/C*	Requests a node with GPU. *G/C* specifies the number of GPUs per each CPU requested and should be expressed as a decimal number. See Advanced Batch System Usage for more information.
-l gpu_type=*GPUMODEL*	Current choices for *GPUMODEL* are M2050, M2070 and K40m.
-l eth_speed=*N*	Ethernet speed (1 or 10 Gbps).
-l scratch_free=#G	Request a node that has at least this amount of available disc space in scratch. Note that the amount changes!
-pe omp *N*	Request multiple slots for Shared Memory applications (OpenMP, pthread). This option can also be used to reserve larger amount of memory for the application. *N* can vary from 1 to 16.
-pe mpi_#_tasks_per_node *N*	Select multiple nodes for MPI job. Number of tasks can be 4, 8, 12 or 16 and *N* must be a multiple of this value. See Advanced Batch System Usage for more information.

SCC General limits

Default:

- Login nodes are limited to 15min. of CPU time
- Default Compute Job: 1 CPU core, 4GB RAM, 12 hours.

Upper Limits:

1 processor job (batch or interactive)	720 hours
omp job (16 processors or less)	720 hours
mpi job (multi-node job)	120 hours
gpu job	48 hours
Interactive Graphics job (virtual GL)	48 hours

These will come up again.

Delete Jobs

- “`qdel`” - Delete a job from the queue

```
[cjahnke@scc1 ~]$ qdel -j 12345
cjahnke has deleted job 12345
```

If you have a lot of jobs -- delete them all by user

```
[cjahnke@scc1 ~]$ qdel -u cjahnke
cjahnke has deleted job 12345
cjahnke has deleted job 12346
cjahnke has deleted job 12347
cjahnke has deleted job 12348
cjahnke has deleted job 12349
```

Monitoring a Job

Monitoring: Running Jobs (`qstat`)

- "`qstat`" - Show the status of Grid Engine jobs and queues

```
[cjahnke@scc1 ~]$ qstat -u cjahnke
job-ID  prior   name    user      state submit/start at      queue                  slots ja-task-ID
-----------------------------------------------------------------------------------------------
336431 0.10059 phy.24 cjahnke   r     09/03/2016 18:02:32  l@scc-ka4.scc.bu.edu   16
336432 0.10059 phy.25 cjahnke   qw    09/03/2016 18:00:00                         16
```

Monitoring: Running Jobs In-Depth (`qstat`)

- "`-s {p|r|s|...}`" -- Prints only jobs in the specified state

```
[cjahnke@scc1 ~]$ qstat -s r -u cjahnke
job-ID   prior   name    user     state  submit/start at      queue                slots ja-task-ID
-----------------------------------------------------------------------------------------------
336431  0.10059  phy.24  cjahnke  r      09/03/2016 18:02:32  l@scc-ka4.scc.bu.edu  16
```

- "`-r`" -- Prints extended information about the resource requirements

```
[cjahnke@scc1 ~]$ qstat -r -u cjahnke

job-ID   prior    name    user     state  submit/start at      queue                slots ja-task-ID
-----------------------------------------------------------------------------------------------
336431   0.10059  phy.24  cjahnke  r      09/03/2016 18:02:32  l@scc-ka4.scc.bu.edu  16
       Full jobname:     phy.24.ortho1.pbs
       Master Queue:     linga@scc-ka4.scc.bu.edu
       Requested PE:     omp16 16
       Granted PE:       omp16 16
       Hard Resources:   h_rt=2588400 (0.000000)
                         mem_free=8g (0.000000)
       job=1 (default)
       Soft Resources:   buyin=TRUE
```

```
[cjahnke@scc4 ~]$ qstat -j 336431
==============================================================
job_number:                 336431
exec_file:                  job_scripts/336431
submission_time:            Sat Sep  3 18:02:22 2016
owner:                      cjahnke
uid:                        157672
group:                      scv
gid:                        2630
sge_o_home:                 /usr3/bustaff/cjahnke
sge_o_log_name:             cjahnke
sge_o_path:                 /share/apps/6.0/cufflinks/2.2.0/bin:/share/pkg/r/3.1.1/install/bin:/.../:/.../
sge_o_shell:                /bin/bash
sge_o_workdir:              /projectnb/scv/test
sge_o_host:                 scc4
account:                    sge
cwd:                        /projectnb/scv/test
merge:                      y
hard resource_list:         h_rt=2588400,mem_free=8g
soft resource_list:         buyin=TRUE
mail_options:               be
mail_list:                  cjahnke@bu.edu
notify:                     FALSE
job_name:                   phy.24.ortho1.pbs
stdout_path_list:           NONE:NONE:phy.24.ortho1.pbs.qlog
jobshare:                   0
shell_list:                 NONE:/bin/bash
env_list:                   PATH=/share/apps/6.0/cufflinks/2.2.0/bin:/share/pkg/r/3.1.1/install/bin:/.../:/.../
script_file:                phy.24.ortho1.pbs
parallel environment:       omp16 range: 16
project:                    scv
usage    1:                 cpu=174:11:55:06, mem=67369810.71167 GBs, io=0.05496, vmem=4.584G, maxvmem=4.584G
scheduling info:            (Collecting of scheduler job information is turned off)
```

Monitoring: Running Jobs In-Depth

Can look at your processes directly on the compute-node

1. Login to the compute node

```
scc1 % ssh scc-ca1
```

2. Run top command

```
scc-ca1 % top -u <userID>
```

3. Exit from the compute node

```
scc-ca1 % exit
```

```
top - 9:50:04 up 18 days, 6:38, 3 users, load average: 7.77, 8.24, 7.98
Tasks: 422 total,   6 running, 416 sleeping,   0 stopped,   0 zombie
Cpu(s): 56.2%us,  0.1%sy,  0.0%ni, 43.6%id,  0.0%wa,  0.0%hi ...
Mem: 132064100k total, 126879140k used, 5184960k free, 402380k buffers
Swap: 8388604k total, 21320k used, 8367284k free, 110327144k cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
38680 cjahnke   20   0 2024m 391m  26m R 399.8  0.3  1:06.27  pscf
38681 cjahnke   20   0 12.4g  12g 1708 R 100.0  9.8  1:06.26  p0.out
46777 cjahnke   20   0 13404 1528  948 R   0.3  0.0  0:00.03  top
46696 cjahnke   20   0 88256 1812  896 S   0.0  0.0  0:00.00  sshd
46697 cjahnke   20   0  9680 1820 1360 S   0.0  0.0  0:00.00  bash
```

Monitoring: Completed Jobs

- `qacct` - query the accounting system
  - Usually, job specific
  - Can summarize information

```
[cjahnke@scc1 ~]$ qacct -j 9253374
==============================================
qname        linga
hostname     scc-ka4.scc.bu.edu
group        sibs
owner        cjahnke
project      sibs
department   defaultdepartment
jobname      test
jobnumber    9253374
taskid       undefined
account      sge
priority     0
qsub_time    Wed Jun 29 12:35:21 2016
start_time   Wed Jun 29 12:35:37 2016
end_time     Wed Jun 29 12:35:47 2016
granted_pe   NONE
slots        1
failed       0
exit_status  0
ru_wallclock 10
...
cpu          0.126
mem          0.000
io           0.000
iow          0.000
maxvmem      13.953M
arid         undefined
```

Monitoring: Completed Jobs In-depth (qacct)

- User Summary

```
[cjahnke@scc1 ~]$ qacct -o cjahnke
OWNER    WALLCLOCK     UTIME    STIME       CPU     MEMORY       IO   IOW
================================================================================
cjahnke    7292588  11181866   640582  12466059   78805572  4763667     0
```

- Project Summary

```
[cjahnke@scc1 ~]$ qacct -o cjahnke -P
OWNER    PROJECT    WALLCLOCK     UTIME    STIME       CPU      MEMORY       IO    IOW
================================================================================
cjahnke adsp          819153    527036   20364    548526      607738    64984      0
cjahnke fhspl          42707     64479    3364     67844       80656     7401      0
cjahnke scv          5490737   8019616  528385   9124199    44585143  4661583      0
cjahnke sprnaseq      678025   1401724   59668   1527681     4754390    16354      0
```

- Time/Date

```
[cjahnke@scc1 ~]$ qacct -d 30
Total System Usage
    WALLCLOCK      UTIME      STIME           CPU             MEMORY             IO             IOW
================================================================================================
   2618582489 6901132453 222669927 8346277146.776   147078894593.431    76658298.739          0.000
```

Monitoring: Accounting Tool (`acctool`)

- A “Service Unit” (SU) is a normalized measure of CPU time usage.
  - Used for project management, allocation, and accounting.
  - Some processors are faster than others.
  - There is no monetary charge.
- Use `acctool` to get the information about SU (service units) usage:

```
[cjahnke@scc4 ~]$ acctool -b 9/01/16 y
Hostname                Time          Jobs
shared                3190.72           235
linga                  896.07           203
jcvi                   538.73           122
TOTAL                 5866.51           607

Project       SU Balance
scv             874.0952
charges      37538.8022
linga_admin    2398.3860
fhspl        41304.6475
```

```
[cjahnke@scc4 ~]$ acctool -p scv -b 9/01/16 y
Hostname                Time          Jobs
shared                2596.14            42
jcvi                   337.75            55
TOTAL                 3562.13           128

Project       SU Balance
scv             874.0952
```

```
[cjahnke@scc1 ~]$ acctool -host shared -b 9/01/16 y
Hostname                Time          Jobs
shared                3190.72           235
```

Monitoring: Email Notifications

The system can send email notifications

- Use "`-m`" qsub option
  - b = when job begins,
  - a = if job aborts
  - e = when job ends

For example:

```
qsub -P project -m bae spline.qsub
```

```
To:       cjahnke@bu.edu
From:     nobody@scc.bu.edu
Subject:  Job 7883980 (spline) Complete

------------------------------------------

User              = cjahnke
Queue             = p@scc-pi2.scc.bu.edu
Host              = scc-pi2.scc.bu.edu
Start Time        = 08/29/2016 13:18:02
End Time          = 08/29/2016 13:58:59
User Time         = 01:05:07
System Time       = 00:03:24
Wallclock Time    = 00:40:57
CPU               = 01:08:31
Max vmem          = 6.692G
Exit Status       = 0
```

Bottlenecks

Bottleneck: Time

SCC General limits

Default:

- Login nodes: 15 min
- Compute Job: 12 hours

Upper Limits:

- 1 Processor job (batch or interactive) 720 hours
- OMP job (16 processors or less) 720 hours
- MPI job (multi-node job) 120 hours
- GPU job 48 hours
- Interactive Graphics job (virtual GL) 48 hours

Bottleneck: Time

Policies regarding time usage on SCC protect the system (and you).

- Keep high occupancy nodes running
  - Compute time limited to 15 minutes on login nodes.
  - Prevents a single user (or several users combined) from bogging down a system that many other people are using for administrative tasks.
- Prevent runaway processes or endless loops
  - Default runtime of jobs limited to 12 hours.

Bottleneck: Time - Login Node

Example: You need to run a compute process for >15 min

Solution: Interactive Job

```
[cjahnke@scc4 ~]$ qrsh -P proj
[cjahnke@scc-pi2 ~]$
```

To: cjahnke@bu.edu
From: nobody@scc.bu.edu
Subject: Message from the process reaper on SCC4

The following process, running on SCC4, has been terminated because it exceeded the limits for interactive use. **An interactive process is killed if its total CPU time is greater than 15 minutes and greater than 25% of its lifetime.** Processes which may exceed these limits should be submitted through the batch system.

See http://www.bu.edu/tech/support/research/system-usage/running-jobs for more information.

COMMAND	STATE	PID	PPID	TIME	RATE(%)	SIZE	RSS	START TIME
MATLAB	S	127687	127592	17	101	8427	5562	09/14 12:27:34

Please email help@scc.bu.edu for assistance.

Bottleneck: Time - Compute Job

Example: You need to run a job for >12 hours.

Solution:
Increase the "Hard Runtime Limit" with the "`-l h_rt=HH:MM:SS`" qsub option.

```
$ qsub -P proj -l h_rt=24:00:00 script.qsub
```

```
To:       cjahnke@bu.edu
From:     nobody@scc.bu.edu
Subject:  Job 9022506 (myJob) Aborted

_____________________________________________
Job 3828407 (sRNA_intersection) Aborted
 Exit Status     = 137
 Signal          = KILL
 User            = cjahnke
 Queue           = b@scc-he1.scc.bu.edu
 Host            = scc-he1.scc.bu.edu
 Start Time      = 03/06/2016 00:32:16
 End Time        = 03/06/2016 12:32:17
 CPU             = 11:56:32
 Max vmem        = 379.809M
failed assumedly after job because:
job 3828407 died through signal KILL(9)
```

Bottleneck: Time - Compute Job

Dear Admins,

I submitted a job and it takes longer than I expected.
Is it possible to extend the time limit?

-- Advanced User

Unfortunately, not. Once running, the job parameters cannot be modified.

Bottleneck: CPU

Bottleneck: CPU Optimization

Writing code from scratch?

→ **Optimize it!**

- There are a best-practices and techniques for every language.
- There are also some specifics in running the code on the cluster.
- Do this before parallelizing your code!
  - Parallelized bad code is still bad code.
- Discuss with us at help@scc.bu.edu

Are you compiling your code?

Modern CPUs can handle complex instructions, but you need to use non-default compilers.

Compiler Options and Versions:

- GCC
  - 4.8.1, 4.9.2, 5.1.0, 5.3.0
- PGI
  - 13.5, 16.5
- Intel
  - 2015, 2016

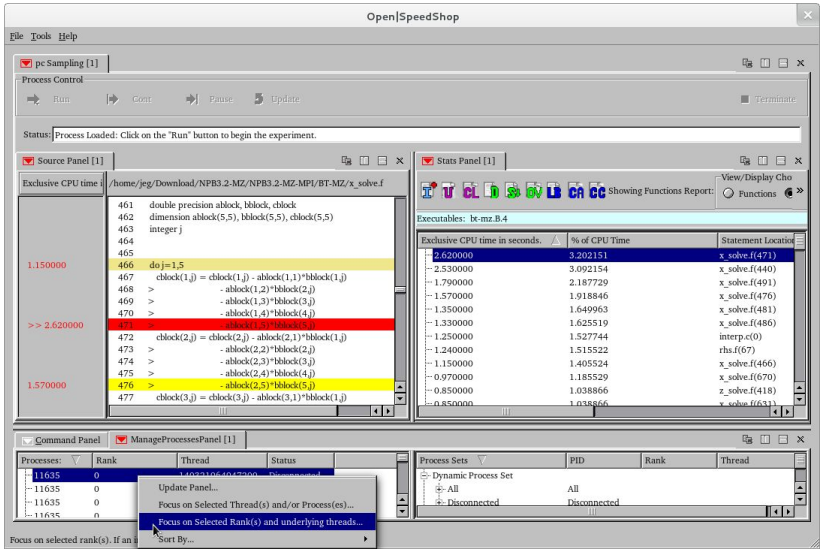
Bottleneck: CPU Optimization (program debug)

Integrated Development Environments (IDE)

- codeblocks
- geany
- Eclipse

Debuggers:

- gdb
- ddd
- TotalView
- OpenSpeedShop



Bottleneck: CPU Parallelization

Parallelization

- OpenMP: Single node using multiple processes
  - Common with scripts when the user only wants a single job.
- OpenMP: Single node threading a single process
  - Commonly built into applications.
- OpenMPI: Multi-node, many CPU, shared memory processing
  - Very powerful computation, not used much on BUMC.
- Tasks/Arrays
  - We will discuss this later

Bottleneck: CPU Parallelization

OpenMP: Single node with multiple processes run simultaneously.

Example: User only wants to manage one job, but wants to run a script on 4 files.

```
#!/bin/bash -l

#$ -pe omp 4

module load python
python script.py file1 &
python script.py file2 &
python script.py file3 &
python script.py file4 &
wait
```

Background process

On compute node: 4 processes running simultaneously

```
top - 9:50:04 up 18 days, 6:38, 3 users, load average: 7.77, 8.24, 7.98
Tasks: 422 total,   6 running, 416 sleeping,   0 stopped,   0 zombie
Cpu(s): 56.2%us,  0.1%sy,  0.0%ni, 43.6%id,  0.0%wa,  0.0%hi ...
Mem: 132064100k total, 126879140k used, 5184960k free, 402380k buffers
Swap: 8388604k total, 21320k used, 8367284k free, 110327144k cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
38681 cjahnke   20   0 12.4g  12g 1708 R 100.0  9.8  1:06.26 python
38682 cjahnke   20   0 12.3g  12g 1708 R 100.0  9.7  1:06.26 python
38683 cjahnke   20   0 12.2g  12g 1708 R 100.0  9.6  1:06.27 python
38684 cjahnke   20   0 11.6g  12g 1708 R 100.0  8.4  1:06.27 python
46777 cjahnke   20   0 13404 1528  948 R   0.3  0.0  0:00.03 top
46696 cjahnke   20   0 88256 1812  896 S   0.0  0.0  0:00.00 sshd
46697 cjahnke   20   0  9680 1820 1360 S   0.0  0.0  0:00.00 bash
```

Bottleneck: CPU Parallelization

Background processes with distinct processes aren't the best way to do this.

- Could also be done in python itself
  - "`import subprocess`" - spawn new processes
  - "`from joblib import parallel`" - runs functions in parallel
  - "`from multiprocessing import Pool`" -- pools processing
- Also in R
  - "`library(parallel)`" - Basic parallel package
  - "`library(snowfall)`" - Easier cluster computing (based on snow)
  - Go to R tutorials next week.
- And most languages

http://scv.bu.edu/examples/SCC/par/python/par.py

Bottleneck: CPU Parallelization

OpenMP: Single Process Threading

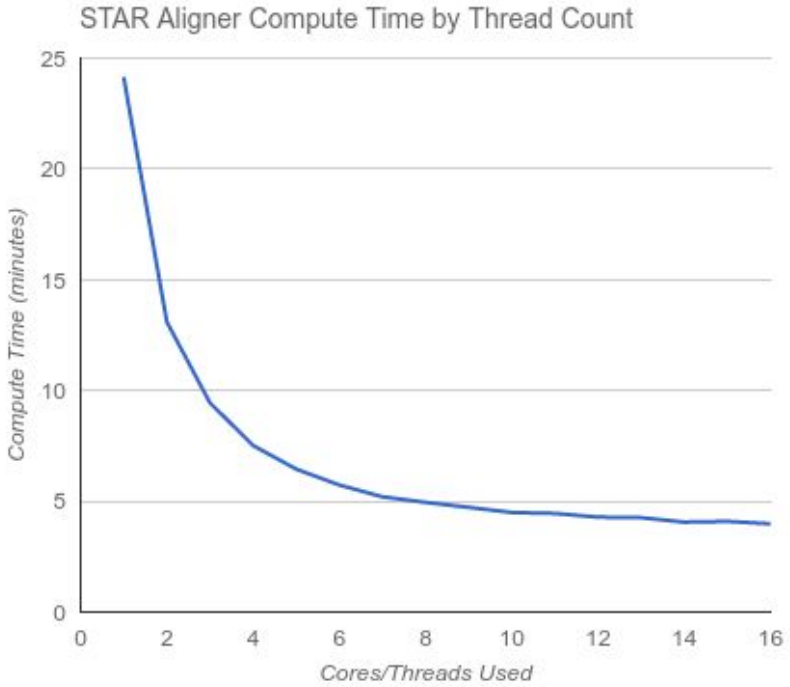
- Many applications have this built in -- Look for a “threads” option

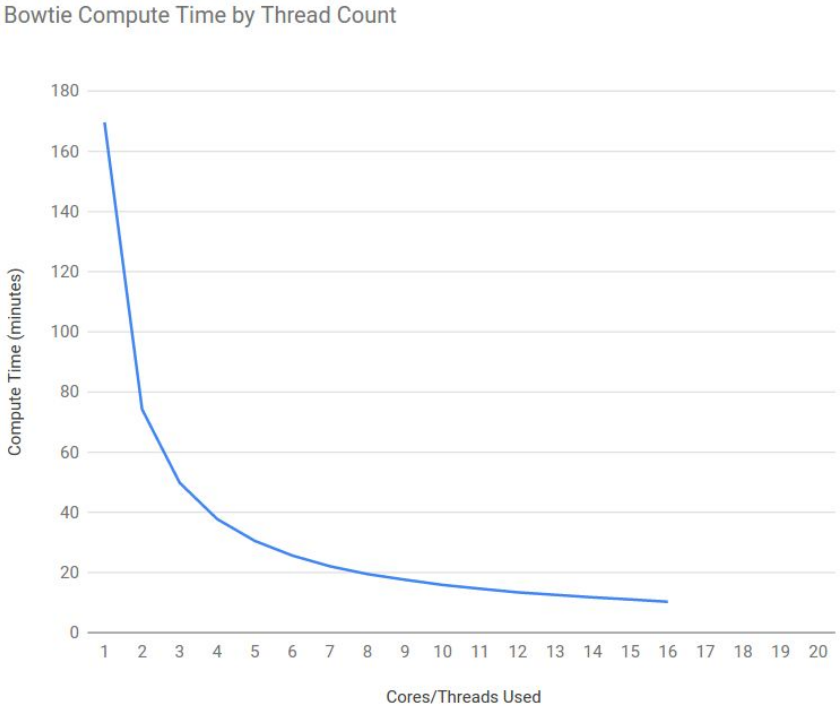
```
#!/bin/bash -l

#$ -pe omp 2

module load bowtie2
bowtie2 --threads 2 -x <bt2-idx>
```

On compute node: A single process using 200% CPU

```
top - 9:50:04 up 18 days, 6:38, 3 users, load average: 7.77, 8.24, 7.98
Tasks: 422 total,   6 running, 416 sleeping,   0 stopped,   0 zombie
Cpu(s): 56.2%us,  0.1%sy,  0.0%ni, 43.6%id,  0.0%wa,  0.0%hi ...
Mem: 132064100k total, 126879140k used, 5184960k free, 402380k buffers
Swap: 8388604k total, 21320k used, 8367284k free, 110327144k cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
38681 cjahnke   20   0 12.4g  12g 1708 R 200.0  9.8  1:06.26  bowtie2
46777 cjahnke   20   0 13404 1528  948 R   0.3  0.0  0:00.03  top
46696 cjahnke   20   0 88256 1812  896 S   0.0  0.0  0:00.00  sshd
46697 cjahnke   20   0  9680 1820 1360 S   0.0  0.0  0:00.00  bash
```

Bottleneck: CPU Parallelization





Bottleneck: CPU Parallelization

Some applications parallelize automatically (use all cores on node).

- Examples: Matlab, stata-mp
- This is bad behavior and your job will be killed
  - Instruct them not to (if possible) or request the whole node.

```
matlab -nodisplay -singleCompThread -r "commands"
```

The $NSLOTS Variable

- When using "-pe omp #" the $NSLOTS variable is set equal the "#".
- This allows you to change the number of threads and not edit the script

```
bowtie2 --threads $NSLOTS -x <bt2-idx>
```

Bottleneck: CPU -- Process Reaper

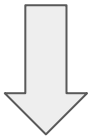
What happens if you use more slots than requested?

- We kill it to preserve other jobs running on that node.

If you have email notifications enabled, you will receive a notice that the job was aborted.

- Note that it ran for 9 minutes and the CPU ran for 22.

You will also receive an explanation email.



```
To:      cjahnke@bu.edu
From:    nobody@scc.bu.edu
Subject: Job 9022506 (myJob) Aborted

_____________________________________________
Job 2885976 (rnaseq.ngs) Aborted
 Exit Status      = 137
 Signal           = KILL
 User             = cjahnke
 Queue            = b@scc-hc2.scc.bu.edu
 Host             = scc-hc2.scc.bu.edu
 Start Time       = 02/01/2016 15:51:07
 End Time         = 02/01/2016 16:00:01
 CPU              = 00:22:03
 Max vmem         = 1.026G
failed assumedly after job because:
job 2885976 died through signal KILL(9)
```

Bottleneck: CPU -- Process Reaper

To: cjahnke@bu.edu
From: nobody@scc.bu.edu
Subject: Message from the process reaper on scc-gb11

The following batch job, running on SCC-GB11, has been **terminated because it was using 5.5 processors** but was **allocated only 1**. Please resubmit the job using an appropriate PE specification.

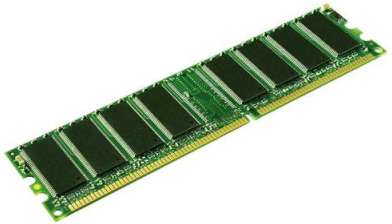
See http://www.bu.edu/tech/support/research/system-usage/running-jobs for more information.

job 461082.1: owner: cjahnke pe: none type: "Qsh interactive" slots: 1
sge_gid: 1000791 job_pid: 8447
cputime: 42 min. rate: 548.39% starttime: 09/14 11:57:17

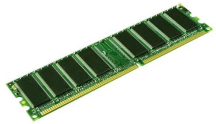
COMMAND	STATE	PID	PPID	TIME(min.)	**RATE(%)**	SIZE	RSS	START TIME
TSF_process.x64	R	8483	8473	4	**268**	68	13	09/14 11:58:54
TSF_process.x64	R	8482	8473	4	**174**	68	13	09/14 11:58:54
TSF_process.x64	R	8481	8473	4	**68**	68	13	09/14 11:58:54
xterm	S	8447	8446	0	0	53	3	09/14 11:57:17

Please email help@scc.bu.edu for assistance.

Bottleneck: Memory



Memory Optimization



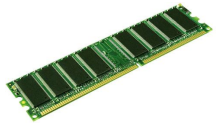
First things first, optimize memory usage in your code

- Many languages allow operations on vectors/matrices
- Pre-allocate arrays before accessing or writing
  - Especially within loops.
- Reuse variables when possible
- Delete variables that are not needed.
- Access elements within your code according to the storage pattern in this language (FORTRAN, MATLAB, R – in columns; C, C++ - rows)

Huge in R

Some of this can be tricky, our applications team is happy to assist. Email help@scc.bu.edu

Memory Limits



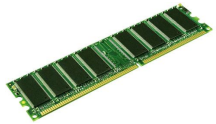
The memory on each node in the SCC is shared by all the jobs on that node.

$$\texttt{Memory}_{\texttt{slot}} = \texttt{Memory}_{\texttt{total}} \ / \ \texttt{N}_{\texttt{cores}}$$

Using too much memory

- Will slow down your job
- Could cause your job to fail (process is killed automatically)
- Could bring down the node (causing yours and others job to fail)

Memory Availability



SCC is Heterogenous!

(Which is good, but makes this complicated)

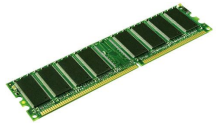
Shared Nodes		
8 cores	24 GB RAM	**3 GB/slot**
12 cores	48 GB RAM	**4 GB/slot**
20 cores	128 GB RAM	**6 GB/slot**
16 cores	128 GB RAM	**8 GB/slot**
8 cores	96 GB RAM	**12 GB/slot**
16 cores	256 GB RAM	**16 GB/slot**

Medical Campus Only		
64 cores	256 GB RAM	**4 GB/slot**
64 cores	512 GB RAM	**8 GB/slot**

Buy-In Nodes		
16 cores	64 GB RAM	**4 GB/slot**
12 cores	96 GB RAM	**8 GB/slot**
20 cores	256 GB RAM	**12 GB/slot**
16 cores	1024 GB RAM	**64 GB/slot**

http://www.bu.edu/tech/support/research/computing-resources/tech-summary/

Memory Requests



Example: You've profiled your job and expect it to use 10 GB

Solution:

1. Request single slot on node with >10GB/slot
   (e.g. an 8 core/96GB node = 12GB/slot)

   ```
   #$ -l mem_per_core=10G
   ```

2. Request multiple slots on a more common node
   (e.g. 2 slots on a 16 core/128GB node = 16GB/2slots)

   ```
   #$ -pe omp 2
   #$ -l mem_per_core=5G
   ```

There are >100 of these nodes lower memory nodes. Less time in queue.

Bottleneck: Disk

Bottleneck: Disk Space

The most common disk bottleneck is a full directory.

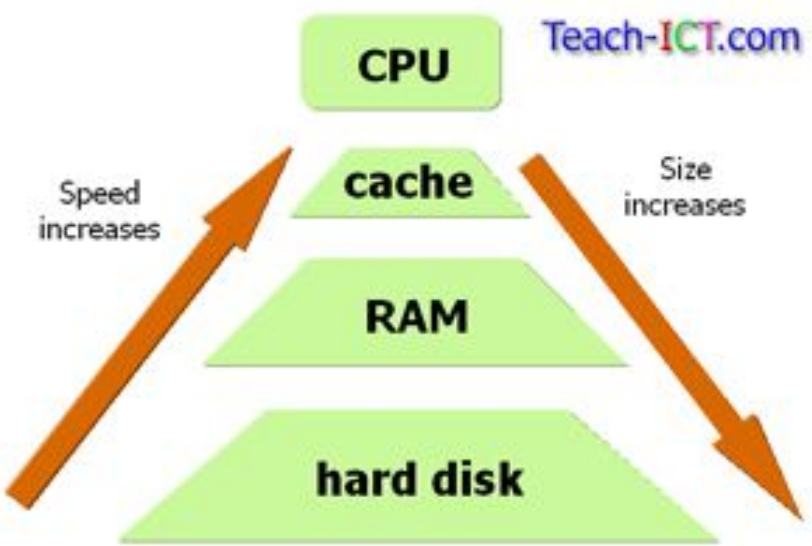
- Depending on the program, the error can be relatively cryptic error message
- Check both your **home directory** and **project space**.

```
[cjahnke@scc1 ~]$ quota -s
Home Directory Usage and Quota:
Name        GB   quota   limit   grace  |  files  quota  limit in_doubt   grace
Cjahnke   7.37    10.0    11.0    none  |  39144 200000 200000       40    none

[cjahnke@scc1 ~]$ pquota rcs
                                    quota    quota       usage     usage
project space                        (GB)   (files)       (GB)   (files)
---------------------------------  ------  --------  ---------  --------
/project/rcs                         50   1638400      21.00       687
/projectnb/rcs                      1050  33554432       2.01      1454
```

Bottleneck: Disk Optimization

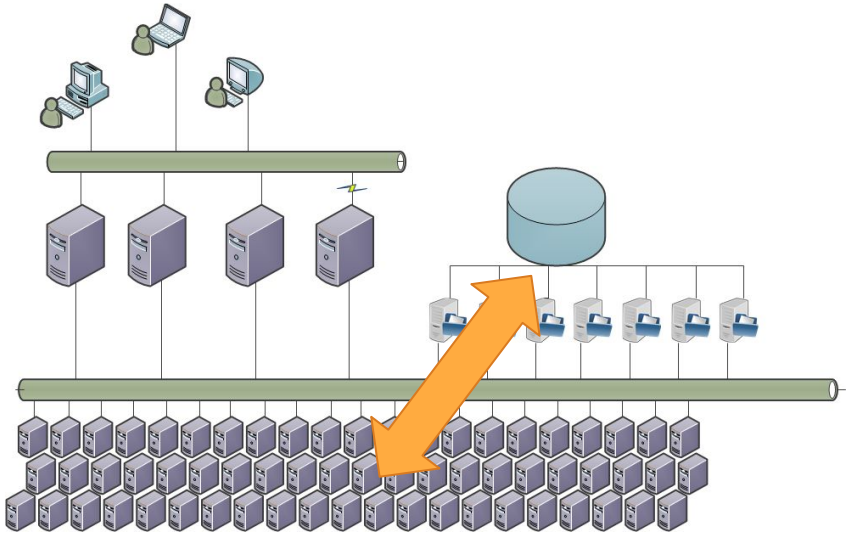
SCC has a large distributed file system shared to compute nodes on a high speed network, but transactions on disk can be slow (compared to other transactions).

- Reduce transactions
- Use a local disk (/scratch)



Bottleneck: Disk -- Optimize Read/Write

- Reduce the number of I/O to the home directory/project space (if possible)
- Group small I/O transactions into blocks.
  - **Don't:** open file, read line, close file, process, open file, write line, close file.
  - **Do:** open, read whole file, process, write.
- Optimize the seek pattern to reduce the amount of time waiting for disk seeks.
  - Sequential vs Random
- If possible, read and write numerical data in a binary format.



Bottleneck: Disk -- Use /scratch

If you can’t do that or it’s not enough

- Avoid the network entirely!
- Utilize local /scratch space
- The $TMPDIR environment variable refers a job specific directory in scratch space. This directory is deleted at the end of the job.
- Scratch files are kept for 30 days, with no guarantees.

```
#!/bin/bash -l

# copy data to scratch
cp /project/proj/file $TMPDIR/

cd $TMPDIR

module load bowtie2
bowtie2 -x $TMPDIR/file fastq1 fastq2

# Copy results back to project space
cp $TMPDIR/resultsfile /project/proj/dir
```

https://www.bu.edu/tech/support/research/system-usage/running-jobs/resources-jobs/#scratch

Bottleneck: Network

Bottleneck: Network

Request a node with 10Gbps network connectivity

```
[cjahnke@scc1 ~]$ qsub -l eth_speed=10 script.qsub
```

- Not a common bottleneck
- Useful if you know that you will be moving a lot of data
  - Great for moving lots of big files within the cluster
  - I'll talk about a better way to do this if you are downloading data from external sources

Job Management and Workflows

Decide where to run jobs

Typically, job requirements dictate the resources needed, but you have options.

Example: Your job needs 4 cores, 40 GB RAM and will take 20 hours.

Literal needs

```
#$ -pe omp 4
#$ -mem_per_core=10
#$ -l h_rt=20:00:00
```

Pro:
- Simple

Con:
- 256 GB nodes are rare
- Buy-in nodes limit 12 hr

Run on 128GB nodes

```
#$ -pe omp 8
#$ -mem_per_core=5
#$ -l h_rt=20:00:00
```

Pro:
- Use common value node

Con:
- Wastes some CPU
- Still exceeds 12 hour limit

Make it 2 Jobs

```
#$ -pe omp 4
#$ -mem_per_core=5
#$ -l h_rt=10:00:00
```

x2

Pro:
- 128 GB = common node
- <12 hr = use shared node

Con
- Not always possible

How to decide what to request

Information about our resources

- Technical Summary: http://www.bu.edu/tech/support/research/computing-resources/tech-summary/
- "`qconf -sc`" - Show the resources that can be requested
- "`qhost`" - Show the status of hosts, queues and jobs
- "`qselect`" - Show the nodes that support specified options
- See cheat sheet.
- If your job is complicated → email help@scc.bu.edu.

Very technical. For advanced users.

Command Line Arguments

Submit a job with extra arguments.

```
scc1 % qsub -P scv script.qsub 1 two "3 word string" file.txt
```

Your ***script.qsub*** could contain:

```
#!/bin/bash -l

echo $1
echo $2
echo $3
Rscript my_R_program.R $4
```

The output would be:

```
1
two
3 word string
# R would have run a script on file.txt
```

Very useful for using a generic script on multiple files or parameters -- but wait until you see “tasks”!

Job Dependency

Pipeline: Some jobs may be required to run in a specific order

Example: or this application, the job dependency can be controlled using "**`-hold_jid`**" option:

```
scc1 % qsub -N job1 script1
scc1 % qsub -N job2 -hold_jid job1 script2
scc1 % qsub -N job3 -hold_jid job2 script3
```

Post-Processing: A job might need to wait until a group of jobs have completed.

In this example, “lastJob” won’t start until job1, job2, and job3 have completed.

```
scc1% qsub -N job1 script1
scc1% qsub -N job2 script2
scc1% qsub -N job3 script3
scc1 % qsub -N lastJob -hold_jid "job*" script4
```

Job Arrays / Tasks

Let’s say you have some “embarrassingly parallel” code

- Simulations - want 1000 runs, using different seed
- Chromosome Analysis - same analysis, different chromosome files
- Large File - divide and conquer / scatter-gather / map-reduce

Array Jobs (qsub option “-t”)

- One “array job” has many related “tasks”.
- Each task runs the same job script, but is has a unique ID to work with.
- Task is placed on the cluster independently (different nodes).

Job Arrays / Task Variables

Submit a 10 task array job, numbered 1 to 10:

```
scc1 % qsub -t 1-10 <my_script>
```

Your ***<my_script>*** could contain:

```
#!/bin/bash -l

Rscript my_R_program.R $SGE_TASK_ID
```

Which would run ***my_R_program.R*** 10 times, each one using a number from 1 to 10.

Batch environment variables:

- SGE_TASK_FIRST=1
- SGE_TASK_STEPSIZE=1
- SGE_TASK_LAST=3
- SGE_TASK_ID=2

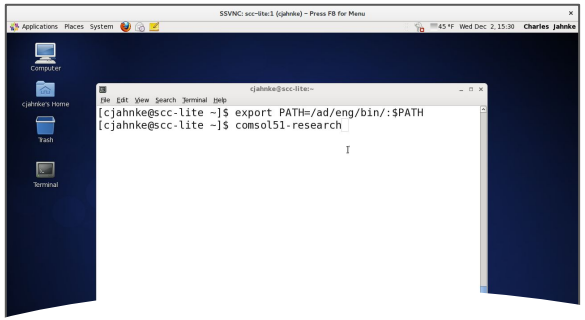
Bonus Material

VNC, Data transfer, Buy-In, Hadoop

VNC - Remote Desktop

VNC (Virtual Network Computing) is a graphical desktop sharing system that allows users to remotely work on another computer. It works by transmitting the keyboard and mouse events from your local machine to the remote machine

- Graphics compression — Allows responsive graphics interaction
- Persistent Sessions — Disconnect and reconnect later



http://www.bu.edu/tech/support/research/system-usage/getting-started/remote-desktop-vnc/

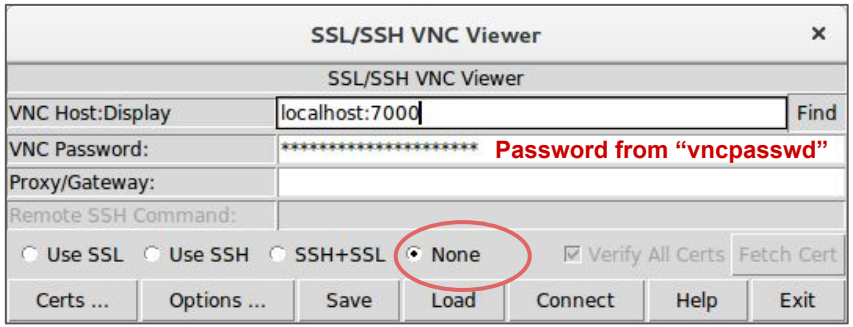
```
[local_prompt ~]$ ssh user@scc1.bu.edu
[cjahnke@scc4 ~]$ vncpasswd
Password:
Verify:
```

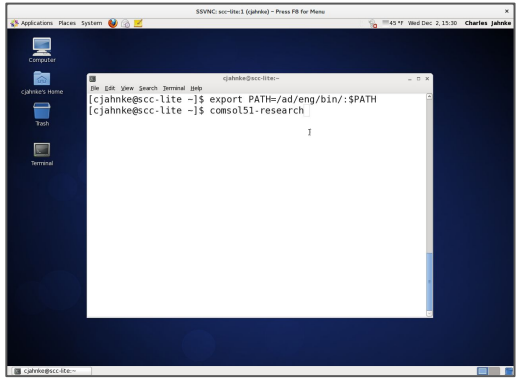
```
[cjahnke@scc4 ~]$ vncstart
================================================
     *** Your VNC server is now running! ***
         VNC desktop number: 1
         VNC Port number: 5901
================================================
To connect via VNC client:
 1. On your local machine execute the following:

    ssh cjahnke@scc4.bu.edu -L XXXX:localhost:5901

    where XXXX - some number greater than 1023.
    You will be prompted to enter your SCC password.

 2. Start your local VNC Client application and
    enter the following address in VNC server field:

    localhost:XXXX

    where XXXX is the number you selected in step 1.
    When prompted, use your VNC password.

To terminate VNC server, execute command (in your scc1
terminal window):

    vncserver -kill :1
================================================
[cjahnke@scc4 ~]$
```

From your local system, forward the port

```
[local ~]$ ssh user@scc4.bu.edu -L 7000:localhost:5901
```

From your local system, open the VNC session





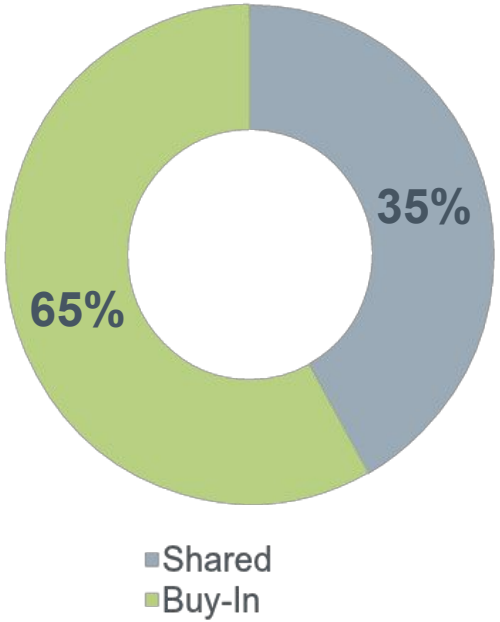
Transfer Node: scc-globus.bu.edu

- High-bandwidth node for data transfer to and from the SCC
- Has 10 Gbps Ethernet connection to internet
- Designed for Globus Connect Service
  - Can setup endpoints on project spaces.
- Supports other protocols
  - Aspera,Globus, GridFTP, AWS, you name it

https://www.bu.edu/tech/support/research/system-usage/getting-started/globus-online/

Service Models – Shared and Buy-In

Buy-In: purchased by individual faculty or research groups through the Buy-In program with priority access for the purchaser.



Shared: Centrally funded by BU and university-wide grants. Resources are free to the entire BU Research Computing community.

SCC Compute Nodes

Buy-in nodes:

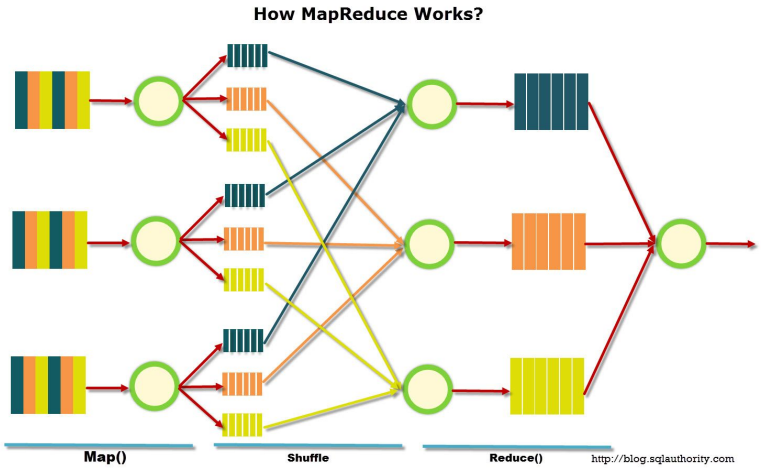
- All buy-in nodes have a hard limit of 12 hours for non-member jobs. The time limit for group member jobs is set by the PI of the group

- About 60% of all nodes are buy-in nodes. Setting time limit for a job larger than 12 hours automatically excludes all buy-in nodes from the available resources;

- All nodes in a buy-in queue do not accept new non-member jobs if a project member submitted a job or running a job anywhere on the cluster.

Hadoop

- Pilot Cluster
- Must request access
- Limited support
- Bioinformatics software soon



If you know what Hadoop is, you might be a good test user.

Support Links and Email

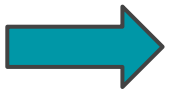
- RCS Website: http://rcs.bu.edu
- RCS Software: http://rcs.bu.edu/software/
- RCS Examples: http://rcs.bu.edu/examples/
- RCS Tutorials: http://rcs.bu.edu/tutorials/

Please contact us at help@scc.bu.edu if you have any problem or question

Questions?

Research Computing Services Website

http://rcs.bu.edu



RCS Tutorial Evaluation

http://rcs.bu.edu/survey/tutorial_evaluation.html