Martinos Center Compute Clusters

Intro
- What are the compute clusters
- How to gain access
- Housekeeping

Usage
- Log In
- Submitting Jobs
- Advanced Options
  - Request CPUs/vmem
  - Queues
    - GPU
  - Email Status
  - Interactive
- Dependencies
  - Daisy Chain
  - Wrapper Script
  - In Progress

Job Status
- Running Jobs
  - Show Job Status
  - See Standard Output
- Completed Jobs
- Failed Jobs
- I/O
- Delete Jobs
- Idle Jobs

Misc Commands
- nodecount
- nodeusage
- usercount
- qselect

Summary
Martinos Center Compute Clusters

Intro - What are the compute clusters?

launchpad
  At Needham Data Center
  127 nodes
  ~115 “normal” nodes
    Two 64 bit Intel Xeon quad cores
    56 GB RAM
  ~12 GPU nodes
    Available exclusively for GPU jobs

tensor
  In CNY “cold room”
  107 nodes
  89 “normal” nodes
    Dual cores with 4GB RAM
  18 “big” nodes
    Dual cores with 8GB RAM each

I will only talk about launchpad today, but the commands for each are the exact same. Use tensor if launchpad is full, if launchpad resources are overkill, or if the data for your jobs live locally and are slowed by the I/O problem.
Intro - How to gain access

Email me: kaiser [at] nmr.mgh.harvard.edu

Let us know who/what/why/how you need access.
Intro - Housekeeping

Questions?:
- Any user-specific questions can be sent to me
  - extend Walltime
  - permission to use MATLAB toolboxes or a high priority queue
- General questions should be sent to the group for advice and help: batch-users [at] nmr.mgh.harvard.edu

Limits:
- We recommend each user use up to 150 job slots during normal usage.
- Evenings/weekends you may use up to 200 slots of CPU/vmem
- While there is a queue, we request you only use ~75 CPU/vmem slots

Do not run anything directly on launchpad. Submit your jobs. Any programs found running on the master node will be killed, no exceptions.
MATLAB:
There is a limited amount of matlab licenses for the entire center (~120). For this reason, we recommend any matlab code submitted for execution should be “compiled” ahead of time. Please follow the URL below for instructions. When the program is compiled, it doesn't use a matlab license and is no longer under a matlab restriction.

Courtesy of coutu:
http://nmr.mgh.harvard.edu/martinos/itgroup/deploytool.html

If you receive a matlab license error, the limited amount of licenses are all occupied. To see the distribution of matlab licenses, run:
   lmstat -f MATLAB
To see the distribution of toolboxes as well, run:
   lmstat -a

Please note, licenses for individual users at their own workstations are given priority, ahead of launchpad users. If users complain we will have to kill a job in order to recover a license.
Usage - Log In

[kaiser@xylo ~]$ ssh launchpad
Last login: Tue Sep 18 10:47:18 2012 from xylo.nmr.mgh.harvard.edu

LAUNCHPAD batch compute cluster

All users of this system must follow the guidelines found in the following Web page:

https://www.nmr.mgh.harvard.edu/martinos/itgroup/launchpad.html

If you need help with launchpad usage, send email to

batch-users@nmr.mgh.harvard.edu

[kaiser@launchpad ~]$
pbsubmit is a wrapper script that:
- formats the command that is executed (/pbs/kaiser/pbsjob_3)
- automatically selects the default settings (unless overridden)
  - number of nodes (nodes=1)
  - number of CPUs (ppn=1)
  - amount of virtual memory (vmem=7gb)
- submits the job using the qsub command

pbsjob_3 is the Job Number
281171.launchpad.nmr.mgh.harvard.edu is the Job ID
Usage – Advanced Options - Request CPUs/vmem

Only request more CPUs or Virtual Memory if you need them.

CPUs
- You should only request extra CPUs if the program you are running is multi-threaded.
- If you aren't sure if the program is multi-threaded, it probably isn't.

Virtual Memory
- Only request as much as you need.
- If you aren't sure how much you'll need, run a single test case. Start with the default of 7GB of vmem. If it fails due to a lack of memory, request 14GB. Then 21GB etc...

So, how much virtual memory did the job use?
Only used 1.9GB of virtual memory. Safely under the default request of 7GB. No need to ask for more.

Limits – Reminder that we prefer each user to only use ~150 job slots during the day. A job that requests 1 CPU and 14GB of vmem counts as two slots worth of resources. Submit the jobs to the max75 queue ('-q max75') to self-regulate.
## Usage – Advanced Options - Queues

<table>
<thead>
<tr>
<th>Queue</th>
<th>Priority</th>
<th>Max CPU/User</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>101000</td>
<td>150</td>
<td>Walltime of 96 hours</td>
</tr>
<tr>
<td>p20</td>
<td>102000</td>
<td>Unlimited</td>
<td></td>
</tr>
<tr>
<td>p30</td>
<td>103000</td>
<td>Unlimited</td>
<td></td>
</tr>
<tr>
<td>GPU</td>
<td>90</td>
<td>Unlimited</td>
<td>GPU nodes</td>
</tr>
<tr>
<td>extended</td>
<td>80000</td>
<td>50</td>
<td>Walltime of 196 hours</td>
</tr>
<tr>
<td>matlab</td>
<td>101000</td>
<td>20</td>
<td>Limit of ~120 matlab licenses for the Center</td>
</tr>
<tr>
<td>max10</td>
<td>101000</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>max20</td>
<td>101000</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>max50</td>
<td>101000</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>max75</td>
<td>101000</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>max100</td>
<td>101000</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>max200</td>
<td>80000</td>
<td>200</td>
<td></td>
</tr>
</tbody>
</table>
For a GPU job, just submit the job to the GPU queue and it will automatically be sent to the GPU nodes.

Since each node only has one GPU (but 8 CPUs) we recommend that users request 5 CPUs for their job. If you choose the default of 1 CPU, then multiple GPU jobs could be sent to the node which will then fail because the GPU is already in use.

The reason for requesting 5 CPUs, and not 8, is if the cluster is overloaded and there are several jobs waiting idle, we want the option to run some regular jobs on the GPU nodes.
Sends email to user (**replace 'kaiser' with your username**) on job start and finish
- To receive email only if job completes with an error, append '-e' to command line
- To receive email only upon job completion (error or no error), append '-f' to command line
Usage – Advanced Options - Email Status

Start Execution:

Identifies the JobID, Job Number, and the node it is running on
Finish Execution:

Shows you the exit status, CPU time, walltime and the virtual memory used
Use the qsub command to start an interactive job using the high priority p30 queue. You will receive an email when the job begins execution. **Replace 'kaiser' with your username!**

Actively wait until the job is slated for execution. Don't immediately leave for lunch.

1. As soon as a slot becomes available, the job is assigned a Job ID and you are ssh'ed to the node where your job will execute.
2. Run your commands...
3. When completed, exit out of the node. Your job will not be completed until you exit.

Please attend to an interactive session. As soon as the job begins and you are ssh'ed into the node, you take up a job slot. Exit out of the node as soon as your commands are done. You will continue to take up a job slot until you exit out of the node.
If you have a series of commands that you want to execute in a row (one after another). The easiest way to do it is to daisy chain the commands together on the command line:

```
[launchpad:~/Subjects] (nmr-stable5.1-env) pbssubmit -c "Command1 ; Command2 ; Command3"
```

Opening pbsjob_7
qsub -V -S /bin/sh -l nodes=1:ppn=1,vmem=7gb -r n /pbs/kaiser/pbsjob_7
283021.launchpad.nmr.mgh.harvard.edu
[launchpad:~/Subjects] (nmr-stable5.1-env) 

The commands are separated on the command line by a semicolon (;).

Each command will run even if the one before it failed.

Replace Command1, Command2, Command3 with the specific commands you want to run
A more elegant way to do it is to write a wrapper script. Use a text editor to create a file called `wrapper.csh` with these contents:

```csh
#!/bin/csh -ef

Command1
Command2
Command3
```

The `-e` flag above instructs the script to exit if any of the individual commands exit with an error. Make the script executable:

```csh
[launchpad:~/.Subjects] (nmr-stable5.1-env) chmod u+x wrapper.csh
```

Submit the script for execution:

```csh
[launchpad:~/.Subjects] (nmr-stable5.1-env) pbsubmit -c "./wrapper.csh"
Opening pbsjob_6
qsub -V -S /bin/sh -l nodes=1:ppn=1,vmem=7gb -r n /pbs/kaiser/pbsjob_6
282987.launchpad.nmr.mgh.harvard.edu
[launchpad:~/.Subjects] (nmr-stable5.1-env)
```
If you already have a job running....

```
[launchpad:~/Subjects] (nmr-stable5.1-env) pbsubmit -c "Command1"
Opening pbsjob_8
qsub -V -S /bin/sh -l nodes=1:ppn=1,vmem=7gb -r n /pbs/kaiser/pbsjob_8
283025.launchpad.nmr.mgh.harvard.edu
[launchpad:~/Subjects] (nmr-stable5.1-env)
```

And you want to start another job that will run immediately after the first job completes without error:

```
[launchpad:~/Subjects] (nmr-stable5.1-env) pbsubmit -o '-W depend=afterok:283025'
-c "Command2"
Opening pbsjob_9
qsub -V -S /bin/sh -W depend=afterok:283025 -l nodes=1:ppn=1,vmem=7gb -r n /pbs/kaiser/pbsjob_9
283027.launchpad.nmr.mgh.harvard.edu
[launchpad:~/Subjects] (nmr-stable5.1-env)
```

This second job will be held until the first one completes without error. If the first job exits with an error, the second job will not run.
## Job Status - Running Jobs – Show Job Status

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Job Name</th>
<th>User</th>
<th>CPUtime</th>
<th>State</th>
<th>Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>283568.launchpad</td>
<td>pbsjob_201</td>
<td>krienen</td>
<td>02:57:33</td>
<td>R</td>
<td>default</td>
</tr>
<tr>
<td>283569.launchpad</td>
<td>pbsjob_252</td>
<td>khoa</td>
<td>02:46:16</td>
<td>R</td>
<td>default</td>
</tr>
<tr>
<td>283604.launchpad</td>
<td>pbsjob_34</td>
<td>spaeth</td>
<td>01:47:00</td>
<td>R</td>
<td>default</td>
</tr>
<tr>
<td>283605.launchpad</td>
<td>pbsjob_35</td>
<td>spaeth</td>
<td>01:45:58</td>
<td>R</td>
<td>default</td>
</tr>
<tr>
<td>283617.launchpad</td>
<td>...0_real011.txt</td>
<td>slbowen</td>
<td>00:34:47</td>
<td>R</td>
<td>default</td>
</tr>
<tr>
<td>283672.launchpad</td>
<td>pbsjob_2</td>
<td>kbickart</td>
<td>00:27:59</td>
<td>R max10</td>
<td></td>
</tr>
<tr>
<td>283673.launchpad</td>
<td>pbsjob_3</td>
<td>kbickart</td>
<td>00:20:10</td>
<td>R max10</td>
<td></td>
</tr>
<tr>
<td>283675.launchpad</td>
<td>pbsjob_4</td>
<td>kbickart</td>
<td>00:20:22</td>
<td>R max10</td>
<td></td>
</tr>
<tr>
<td>283676.launchpad</td>
<td>pbsjob_5</td>
<td>kbickart</td>
<td>00:20:26</td>
<td>R max10</td>
<td></td>
</tr>
<tr>
<td>283677.launchpad</td>
<td>pbsjob_6</td>
<td>kbickart</td>
<td>00:18:20</td>
<td>R max10</td>
<td></td>
</tr>
</tbody>
</table>

**States:**

- [R]unning
- [Q]ueued
- [H]eld

### Additional options:

To see just your jobs:
- `qstat -u <username>`
- `qstat | grep -w <username>`

To get all your running and queued jobs:
- `qstat | grep -w <username> | grep -w R`
- `qstat | grep -w <username> | grep -w Q`
Another Job Status command is 'showq':
It shows the Active (running), Idle (queued) and Blocked (held) jobs

To see just your jobs:
showq -u <username>

To see all your running and idle jobs:
showq -r -u <username>
showq -i -u <username>
Job Status - Running Jobs – See Standard Output

[kaiser@launchpad ~]$ pbsubmit -c "sleep 120 ; echo 'Hello World' ; sleep 120"
Opening pbsjob_13
qsub -V -S /bin/sh -l nodes=1:ppn=1,vmem=7gb -r n /pbs/kaiser/pbsjob_13
3689806.launchpad.nmr.mgh.harvard.edu

[kaiser@launchpad ~]$ pbsubmit -c "sleep 120 ; echo 'Hello World' ; sleep 120"
Opening pbsjob_13
qsub -V -S /bin/sh -l nodes=1:ppn=1,vmem=7gb -r n /pbs/kaiser/pbsjob_13
3689806.launchpad.nmr.mgh.harvard.edu

Job is running:

[kaiser@launchpad ~]$ qstat | grep -w kaiser
3689806.launchpad pbsjob_13 kaiser 00:00:00 R default

[kaiser@launchpad ~]$ qstat | grep -w kaiser
3689806.launchpad pbsjob_13 kaiser 00:00:00 R default

Check on the standard output of the job:

[kaiser@launchpad ~]$ jobinfo -o 3689806.launchpad
Hello World

[kaiser@launchpad ~]$ jobinfo -o 3689806.launchpad
Hello World

To see the standard error of an actively running job; 'jobinfo -e <Job ID>'
Martinos Center Compute Clusters

Job Status - Completed Jobs

The job script, standard output, standard error and the exit status are all saved as separate text files in your pbs directory:

Check the Exit Status of the job. Zero means it completed without errors.

The job script, standard output, standard error and the exit status are all saved as separate text files in your pbs directory:
Ack! My job finished with an Exit Status of 127.

How do I troubleshoot???
Job Status - Failed Jobs

Check the standard error and standard output files for any hints:

Other Possible Hints:

Resource Related
  Check vmem is under the requested amount (default: 7GB)
  Check walltime is under the requested amount (default: 96 hours)

Command Related
  Check standard error and standard output files!!
  Check standard error and standard output files (again)!!
  If the program is home-made, was it compiled for the launchpad architecture?
  Test-run the command locally. If it breaks, the problem is probably not with the cluster.
Usage - I/O

Compare CPUtime and Walltime. If Walltime is larger than CPUtime, time was wasted in I/O.

This job was run using data from my local machine. Over one minute was wasted transferring data back and forth between launchpad (in Needham) to my computer at Martinos.
Martinos Center Compute Clusters

Usage - I/O

How to check the I/O load on the cluster

```
[kaiser@launchpad ~]$ cat /var/log/gpfs_load.log | tail -16

Wed Feb  5 11:40:02 EST 2014
=== gpfs01
  load average: 1.54, 1.64, 1.46
=== gpfs02
  load average: 34.29, 36.26, 37.24
=== gpfs03
  load average: 3.80, 3.16, 2.76
=== gpfs05
  load average: 4.33, 3.49, 2.95
=== gpfs06
  load average: 3.70, 3.64, 3.43
=== gpfs07
  load average: 9.80, 7.42, 6.27
=== gpfs08
  load average: 3.67, 3.28, 3.09
[kaiser@launchpad ~]$ ...
```

If the load average is above 30 (gpfs02), the load is too high. We are automatically notified and have to investigate the reason. If the problem is caused by I/O on launchpad we find the user with the largest I/O need, delete all their jobs and monitor the situation.
Is it your jobs causing the problem?

Look at the %CPU and S (state) columns. If your jobs are using less than 100% of it's CPU and is in the 'D' state, the job is stuck waiting for I/O. In this case, only run 20 jobs at a time. Use the max20 queue.
Tips on dealing with this bottleneck:

- Copy/move local data to /cluster/ directories before running jobs.
- Have scripts/programs write temp data to /cluster/scratch/
- Instead of launchpad, use tensor which lives in CNY.
- Space out submission of jobs so they don't all have large I/O needs at the same time.

For instance, the nu_correct step in FS uses CNY local data. If you have 150 jobs all reach this point at the same time, there are 150 separate requests for CNY data. This load is unmanageable. Since this step is in the beginning of the stream and only takes a minute, space out each job submission by a couple minutes. Or submit ten jobs at a time, wait ten minutes, then submit the next batch.
You submit a job, realize there is a mistake and want to delete it:

```
[kaiser@launchpad ~]$ pbsubmit -c "sleep 120"
Opening pbsjob_12
qsub -V -S /bin/sh -l nodes=1:ppn=1,vmem=7gb -n /pbs/kaiser/pbsjob_12
3689803.launchpad.nmr.mgh.harvard.edu
[kaiser@launchpad ~]$ qstat | grep -w kaiser
3689803.launchpad pbsjob_12 kaiser 00:00:00 R default

[kaiser@launchpad ~]$ qdel 3689803.launchpad
[kaiser@launchpad ~]$ qstat | grep -w kaiser
[kaiser@launchpad ~]$  
```
Shows the Idle Queue. Jobs in this state are waiting for slots to open up. They are sorted by 'Priority'. Priority is determined by the queue (default, max50 etc.) and how long it has been waiting.
Misc Commands - nodecount

```
[kaiser@launchpad ~]$ nodecount
compute-0-0    : 5
compute-0-1    : 3
compute-0-2    : 0
compute-0-3    : 8
...            
...            
compute-0-123  [GPU] : 0
compute-0-124  [GPU] : 0
compute-0-125  [GPU] : 0
compute-0-126  : 8

TOTAL regular:  402 /  792 (99 working nodes)
TOTAL [GPU]    :  0 /  88  (11 working nodes)
[kaiser@launchpad ~]$ 
```

Shows the number of CPUs that are occupied for each compute node.

Helpful in seeing how busy the cluster is and how many job slots are available.
Misc Commands - nodeusage

Condenses the output of nodecount from displaying the number of CPUs in use for each node, into a summary of the number of nodes that have a number of CPUs occupied.

For example, there are zero nodes with zero jobs running (completely free) and there are 35 nodes with only two CPUs in use (can support six job slots).

A similar command is called 'freenodes' (thanks Doug Greve). Instead of counting the nodes with CPUs that are busy, it counts the number of nodes that have free CPUs.
### Misc Commands - usercount

```bash
[kaiser@launchpad ~]$ usercount

<table>
<thead>
<tr>
<th>USER</th>
<th>Running Jobs / CPUS</th>
<th>0/</th>
<th>Queued Jobs / CPUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>adagley</td>
<td>2/5</td>
<td>0/</td>
<td>548/548</td>
</tr>
<tr>
<td>clarsen</td>
<td>51/51</td>
<td>0/</td>
<td></td>
</tr>
<tr>
<td>ganz</td>
<td>10/10</td>
<td>0/</td>
<td></td>
</tr>
<tr>
<td>jonp</td>
<td>1/8</td>
<td>0/</td>
<td></td>
</tr>
<tr>
<td>lzollei</td>
<td>8/8</td>
<td>0/</td>
<td></td>
</tr>
<tr>
<td>martab</td>
<td>1/8</td>
<td>0/</td>
<td></td>
</tr>
<tr>
<td>pwilkens</td>
<td>6/6</td>
<td>0/</td>
<td></td>
</tr>
<tr>
<td>ryu</td>
<td>8/8</td>
<td>0/</td>
<td></td>
</tr>
<tr>
<td>seesaw</td>
<td>2/2</td>
<td>0/</td>
<td></td>
</tr>
<tr>
<td>shou</td>
<td>83/83</td>
<td>0/</td>
<td></td>
</tr>
<tr>
<td>slbowen</td>
<td>13/16</td>
<td>64/64</td>
<td></td>
</tr>
<tr>
<td>ville</td>
<td>2/16</td>
<td>10/80</td>
<td></td>
</tr>
<tr>
<td>vinke</td>
<td>2/2</td>
<td>0/</td>
<td></td>
</tr>
<tr>
<td>wachinge</td>
<td>50/50</td>
<td>149/149</td>
<td></td>
</tr>
<tr>
<td>xnat</td>
<td>192/192</td>
<td>0/</td>
<td></td>
</tr>
</tbody>
</table>
```

Displays the number of jobs and CPUs being used by each user. Also counts the idle (queued) ones as well.
Displays all the JobIDs for a user. This command can accept many parameters to filter out certain job states, resources, queues etc.

It can be very useful if you need access to a set of JobIDs.
Troubleshooting steps:

If a job exits in error, check the standard output and standard error files in your pbs directory.

Search the launchpad page for help and guidelines:
http://www.nmr.mgh.harvard.edu/martinos/userInfo/computer/launchpad.php

Try running the command locally on your own machine. If you receive the same errors, the problem is not with the cluster.

Send email to batch-users mailing list

Any Questions?