Neon Overview and Quick Start Guide

Neon Quick Start

This document is intended for users who have experience with using cluster computing resources and want to get started on Neon quickly. It describes the overall architecture of the system and the tools that are used to interact with the system and run jobs. More detailed information can be found in other documentation pages on this site. To apply for an account on this system please visit this page: http://hpc.uiowa.edu/user-services/apply-account and click on Neon HPC Cluster which takes you to the Workflow page. Please sign in using your HawkID and password. Fill in the form appropriately (Sponsor name, if you are an undergraduate, graduate or post-doctoral scholar, faculty and staff don't need the Sponsorship and Investor queue, if you belong to an Investor's group) and your account will be created.

System Overview

The Neon Cluster is a High Performance Computing System consisting of 4256 Standard Processor Cores and 2280 Xeon Phi Cores. The system runs CentOS 6.4, a Linux operating system based on Red Hat Linux.

There are 2 Login nodes (8 cores each), 259 compute nodes (most of which have QDR Inifiniband connections), and 85TB of shared nfsscratch storage.

The Compute nodes comprise of:

- 188 64GB Nodes, 2.6GHz 16 Core (Standard Nodes)
- 59 256GB Nodes, 2.6GHz 16 Core (Mid-Memory Nodes)
- 12 512GB Nodes, 2.9GHz 24 Core (High-Memory Nodes)
- 38 Xeon Phi 5110P Accelerator Cards
- 11 Nvidia Kepler K20 Accelerator Cards

Accessing Neon

The Neon cluster is accessed via SSH, either through a command line client or using NoMachine to get a graphical environment. Your HawkID credentials (username and password) are used to gain access to Neon once an account has been set up. You may set up your ssh keys such that you can gain access to Neon from your workstation or laptop without having to enter your account credentials. See the Set up Password-less Login documentation for a general process, but the process for generating the keys and setting them up varies for different clients so see the documentation of your ssh client for more in depth information.

> **DUO**
> Duo Two-Facter authentication has been implemented on Neon for all logins. Please refer to this documentation for more info: Two-factor authentication

SSH

If you are **on-campus**, you may access the Neon cluster via command line SSH by opening a terminal and entering:

```
ssh hawkid@neon.hpc.uiowa.edu
```

This will bring you (round-robin style) to one of two 16 core 256gb login nodes.

> If you are on campus and your workstation/laptop account also uses your HawkID, then you do not need to specify the username as above

If you are **off-campus**, you must specify the alternative port of 40, as the default port of 22 has been blocked at the campus border:

```
ssh -p 40 hawkid@neon.hpc.uiowa.edu
```

An alternative to using port 40 is to use the campus VPN. This will give you an "on-campus" IP address, and you may then access Neon via the default port 22. More information on using the campus VPN, UI Anywhere, is available from the ITS website.

Note that **only IPv4 SSH connections are allowed from off-campus machines.** Most connections will attempt to connect via IPv4 so there should be nothing to do. However, if your machine is attempting an IPv6 connection it will fall back to IPv4 after some period of time. To avoid this time delay, specify that the connection use IPv4 explicitly with the following:

```
ssh -4 neon.hpc.uiowa.edu
```

FastX

FastX is a program that allows you to remotely run graphical X11 sessions on Linux and Solaris servers from any client (Mac, Windows, or Linux). For instructions on connecting to Neon with FastX, click here.

Queues and Policies

Neon			
Queue	**Investor**	**Node Description**	**Slots**
ANTH	Andrew Kitchen	(2) 16-core, 64 GB	32
AS	Ashley Spies	(7) 16-core, 64-GB	112
BA	Bruce Ayati	(2) 16-core, 64-GB	32
CGRER	Center for Global and Regional Environmental Research	(4) 16-core, 64-GB	64
CLL	Ching-Long Lin	(10) 16-core, 64-GB (1) 16-core, 256-GB	176
COB	College of Business (Brian Heil)	(1) 16-core, 64-GB	16
COE	College of Engineering	(10) 16-core, 64-GB (10) 16-core 256-GB	320
DAWSON	Jeff Dawson	(1) 16-core, 256-GB	16
DR	Dan Reed	(4) 16-core, 64-GB (1) Xeon Phi each	64

FISH	Larry Weber	(10) 16-core, 64-GB	160
FISHER	Public Health (Patrick Breheny)	(1) 24-core, 512-GB	24
GW	Ginny Willour	(2) 16-core, 256-GB	32
HJ	Hans Johnson	(11) 16-core, 64-GB (4) 16-core, 256-GB (1) 24-core, 512-GB	264
IIHG	Iowa Initiative for Human Genetics	(7) 16-core, 256-GB	112
IVR	Institue for Vision Research	(4) 16-core, 64-GB (1) 16-core, 256-GB	80
JM	Jake Michaelson	(1) 24-core, 512-GB (2) 32-core, 512-GB	88
LT	Luke Tierney	(1) 16-core, 256-GB (1) Xeon Phi each (1) Nvidia K20 each (1) 24-core, 512-GB	40
MF	Michael Flatte	(1) 16-core, 64-GB	16
MP	Miles Pufall	(1) 16-core, 256-GB	16
MS	Mike Schnieders	(16) 16-core, 64-GB (1) Xeon Phi each	256
MORL	Ann Black	(4) 16-core, 64-GB (1) Xeon Phi each (Nodes moved from MS per Mike Schnieders)	64
MANSCI	Management Science	(1) 16-core, 64-GB	16
SB	Scott Baalrud	(2) 16-core, 64-GB	32
SH	Shizhong Han	(5) 16-core, 256-GB	80
SHIP	Fred Stern	(10) 16-core, 64-GB	160
SS	Scott Spak	(2) 16-core, 64-GB	32
TB	Terry Braun	(1) 16-core, 64-GB	16
TUKEY	Public Health (Kathryn Chaloner)	(2) 16-core, 256-GB	32
KA	Kin Fai Au	(1) 16-core, 256-GB	16
INFORMATICS	Greg Carmichael	(13) 16-core, 256-GB Note - the INFORMATICS queue will contain 13 nodes until the end of the fall 2017 semester.	208
AL	Amaury Lendasse	(1) 16-core, 256-GB (1) Xeon Phi	16
PABLO	Pablo Carrica	(7) 16-core, 64-GB (1) 16-core, 256-GB	128

The University of Iowa (UI) queue

Centrally funded nodes	Neon
Description	UI: (73) 16-core, 64-GB 11 with Xeon Phi cards 8 with Nvidia K20 cards (Note: the UI queue will have only (8) Kepler nodes until the end of the Fall 2017 semester.) UI-HM: (6) 24-core, 512-GB
UI queue limit	10 running jobs per user
UI-HM queue limit	2 running jobs per user
UI slot limit/user	320

Performing Computations

Submitting Jobs

After logging in to Neon, you will be in a typical Linux environment. Unless you have requested a different shell, the default shell is Bash. While at this shell, you can create your job input files and job scripts, compile your code, etc. Ultimately, you will want to submit a job to the cluster to run on the compute nodes.

> Please do not run your production jobs on the login nodes. The login nodes are limited in number and shared among all of the users. The compute nodes on the other hand are dedicated to running a specific number of job processes at any given time. They are what your jobs should run on. Any long running jobs found running on a login node will be terminated, with the exception of localized job control scripts that are sometimes needed.

The job queueing system used on Neon is Sun Grid Engine (SGE). SGE uses the typical commands that most other queueing systems use although some of the syntax may be a bit different. It is typically best to create a job script that specifies desired resources and to run the application needed for the computation. Assuming that one writes a job script called, "myscript.sh", it can be submitted to the queue system with

```
qsub myscript.job
```

The default queue on Neon is the UI queue. For most jobs that is appropriate but there is a **10 running jobs per user limit** in that queue. If you need to run more than 10 jobs at a time on the system, then you should specify the **all.q** queue. While the all.q queue has no running job limit, the jobs running in it are subject to eviction if the hosts are needed by higher priority jobs.

To specify that the job will use the all.q queue, the myscript.job file could specify the queue

```
#$ -q all.q
some_program my_job.inp
```

or it can be specified at the command line.

```
qsub -q all.q myscript.job
```

There is a second UI queue (UI high memory), which is for Jobs with higher memory requirements. This queue has a Job Limit of 4.

```
qsub -q UI-HM myscript.job
```

Selecting nodes based on resources

The machines in the Neon cluster are not all identical. Some have Xeon Phi accelerator cards and some have NVidia Kepler GPU accelerator cards. To request a node with special resources use the '-l' option of qsub. For instance, if your job can make use of a Phi accelerator submit your job as

```
qsub -l phi myscript.job
```

> Note that if your code is linked against the Intel Math Kernel Libraries (MKL) then you can make use of a Phi accelerator in offload mode by specifying `MKL_MIC_ENABLE=1`.

To select a node with a Kepler GPU accelerator

```
qsub -l kepler myscript.job
```

There are also 3 different types of machines that are defined by the amount of memory in them. There are memory limits set per job slot so be aware of this as you are creating your job files. If your job exceeds the amount of memory defined by the limit below multiplied by the number of slots requested then the job will be aborted.

machine type	amount of memory	SGE request	Memory limit per slot
standard memory	64G	std_mem	4G
mid memory	256G	mid_mem	16G
high memory	512G	high_mem	20G

The UI queue contains only standard memory machines. The high memory machines are a different architecture with different performance characteristics and are contained in the UI-HM queue. Investor queues can have a mix of these nodes depending on what was purchased.

The all.q queue contains all three memory classes of machines. Any one class can be selected with the appropriate option to '-l'. They are all Boolean resources (True/False) so an exclusion can also be specified. For instance, if submitting a job to the all.q queue, one could request standard memory and mid memory machines, excluding high memory machines from the pool of available machines for the job. This may be desirable to keep those machines available as they are a limited resource. That is probably most useful in investor queues that have a mix of node types.

```
qsub -l high_mem=false -q all.q myscript.job
```

The same applies for investor queues that have a mix of node types. Use the '-l' flag to control the machines that you want to be considered for any particular job. You can put any of the resource requests in your job script as well.

```
#$ -l mid_mem
```

Memory Limits

See the table above for the limit values. There are times when jobs use more memory than what is expected. This can lead to a situation where the node(s) the job is running on swap memory to disk. This causes a huge performance hit on the job at best. In the worst case scenario the node becomes unresponsive and both the node and job fail. In the latter case there is no report of a problem and as far as SGE is concerned the job is running. Unless you are checking the output from the job there is not an indication that there is a problem until we notice that the node is non-responsive. The rationale for instituting memory limits is that the job will fail before performance deteriorates and provide a notification to the user.

> To receive notification you must have the appropriate mail options set up in your job script.

```
#$ -M email-name@uiowa.edu
#$ -m a
```

Depending on how many jobs you are submitting it may be useful to also get mail notifications at job launch and end.

```
#$ -m bea
```

One consequence of having memory limits would be cases where a job requests more memory than it actually uses. The memory request would trigger the limit even though the job does use an amount of memory below the limit. These cases would have to be evaluated and adjustments to code and parameters may be needed. They may also require a larger resource request from SGE to insure that they fit within the limits at all times.

Checking Job Status

You can check the **status of submitted jobs** by running the command:

```
qstat
```

This will show a list of all running and pending jobs, displaying the job ID (a number on the far left, next to each job), priority, name, user id, and current state.

Some of the most common states are:

-**qw**, which indicates that there are not currently enough resources to run your job.

-**Eqw**, which indicates that the job failed to start, and is now in an error state.

-**r**, which indicates that the job is currently running.

To get information on just **your own jobs**, run:

```
qstat -u $USER
```

where $USER is your user account name. The first column of the output will have the JobID.

To get more information on **a specific job**, including why it is still in qw, or why it is in an error state run the command:

```
qstat -j JobID
```

If you need to **delete** a job for some reason, you can use the qdel command with the JobID that corresponds to the job to be deleted.

```
qdel JobID
```

If a job has entered the error state due to bad permissions or a similar error, you can clear the error and the same job will try to start again.

It will be restarted with the same script it used when it was first submitted, so if you have modified you're script to fix the error, it will need to be resubmitted instead.

To **clear the error**, run the command

```
qmod -cj JobID
```

For options to display more advanced information, consult the manual by using the command "man qstat"

Completed Jobs

To get information on **completed jobs**, including runtime, memory usage, start and stop times, run the command:

```
qacct -o $USER -j
```

This will provide detail for each of your jobs.To get information on a **specific completed job**, run the command:

```
qacct -j Jobid
```

To list your **overall usage**, run the command:

```
qacct -o $USER
```

Software Installations and Preinstalled Software

A number of software packages have already been installed on the HPC cluster systems and have been made available using environment modules. The software programs are generally installed in the /opt directory and all available modules can be seen using:

```
module avail
```

Modules must be loaded before being used:

```
module load modulename
```

If software packages that you need are not available from the list of environmental modules we encourage you to install the software in your home account directory. If the software installation requires root access or you need assistance with the installation please send a request to hpc-sysadmins@iowa.uiowa.edu. All software packages installed on an HPC system must be compliant with the University of Iowa policy on software licensing. For more details on software installation please see this page.

Data Storage and Backups

Home accounts on each cluster system have automatic snapshots taken periodically that save the state of your home account at that point in time. However, these snapshots expire and are not sent to a second location and are therefore **not** backups. More details on home account

storage are available here.

All users also have scratch space available in /nfsscratch/Users/HawkID and local scratch. A cleanup script is run that deletes all files in scratch that were created more than 60 days prior to the script run.

Although home accounts and scratch **are not backed up**, additional HPC-connected storage with backups can be purchased. More information can be found here.