Introduction To HPC For New Users

What Is An HPC Cluster?

At the most basic level, a compute cluster is a group of uniformly configured compute nodes that work in concert, usually over high bandwidth low latency network interconnects, to solve large computational problems which require more memory or more CPU than is available on a typical workstation or single server. When clustered together, these nodes can be thought of as a single system. This allows researchers to solve complex problems in much shorter amounts of time.

Compute clusters can be comprised of anywhere from several to thousands of compute nodes, a few "login" or "staging" nodes, and one or more head nodes, which coordinate the scheduling, distribution and data sharing of jobs.

What Is A Scheduler?

In order to fairly distribute a cluster’s resources amongst a group of users, a software tool known as a scheduler is used. Cluster resources are divided into environments and resources. For example, users may choose between different available parallel environments (known as a PE) or special resources such as a co-processor, GPU, or extra memory. PEs add in extra functionality for parallel processing which manages inter-node process communication. Different types of parallel jobs may require a certain configuration of parallel environment, such as with the use of OpenMPI.

Schedulers generally work with a queueing system (Queues can be organized in various ways, but generally are organized by resource or by user access), which distributes jobs to the cluster as resources become available. Schedulers monitor user job submissions, then check available resources to determine which jobs can be run at a given time.

Other factors are also taken into consideration, depending on how the scheduler may be configured. For example, schedulers can prioritize submissions based on how frequently a particular user may be using the system, and will put a slightly higher priority on jobs from users who do not use the system as frequently. This ensures that the system will not be dominated by only a few very active users.

University of Iowa Cluster Systems

The University of Iowa currently has a shared HPC cluster available for campus researchers to use. The shared system is run primarily by ITS-Research Services. The cluster is capable of running both High Performance jobs and High Throughput jobs. The system is comprised of several hundred compute nodes with several thousands of processor cores.

What are the differences between High Throughput Computing and High Performance Computing?

High Performance computing enables a user to solve a single, large problem by harnessing a large number of processors and memory across multiple compute nodes. These types of problems are typically broken down into pieces and processed in parallel, with different compute nodes working on a different part of the problem. Each node communicates with the other nodes working on the problem via a high-speed interconnect -- in our case, we use OmniPath. Parallel processing typically requires code modification in order to utilize a library such as MPI which in turn facilitates parallel communication between the nodes working to solve the problem. Examples of problems that use High Performance Computing are Computational Fluid Dynamics and Molecular Dynamics. More information on using MPI on our HPC systems is available here: MPI Implementation.

High Throughput computing allows a user to use multiple compute nodes in a coordinated fashion to solve a high number of individual problems. The jobs that make up this sort of computation typically do not communicate with each other. This provides the ability to analyze many data sets simultaneously, and also allows the user to efficiently perform a parameter sweep, which refers to running the same program multiple times, but with varying inputs.

Examples of High Throughput problems are Genomics data processing and Image Analysis.

I just received my HPC account, what are my next steps?

Getting started with Linux
Our HPC cluster runs CentOS Linux, the version being current at the time of system deployment. In order to make use of the cluster, users will need a basic understanding of how to interact with a Linux system at the command line. At a minimum, you will need to know how to move around the system, copy and edit files. There are many resources on the Internet devoted to helping you learn your way around a Linux system. One of the best resources available is a book called *The Linux Command Line*, which is available as a free PDF download [here](the-linux-command-line.pdf). For a quicker overview of basic Linux commands, there is a good Linux Cheat Sheet [here](the-linux-command-line.pdf).

### Mapping your work to one of the clusters

If your compute problem is not tractable on a desktop or lab workstation, uses a large amount of memory, requires a rapid turnaround of results, would benefit from being scheduled, then an HPC cluster may be a good fit for you. The next steps are to determine if your computation runs on Linux, can be run in batch mode (non-interactively) and whether it is a high performance (parallel) or high throughput (serial) job. Determining the answers to these questions will help decide how to go about requesting and utilizing HPC resources. Some additional questions to consider are:

- **Will you need to recompile your code to run on our cluster?**
  If you are bringing code over from another system, you may need to recompile it to work on our systems, especially if you are using MPI (of which we offer a few different varieties). We have some additional notes on compiling here: [Compiling Software](#)

- **What software will your job need, and is it available centrally, or could it be installed in your home directory?**
  Our list of installed software is here: [Software Installations](#). If you don't see a package you need, please [let us know](#), and if it is broadly applicable to a number of users, we may install it centrally, or we will help install it into your home directory.

- **Can you estimate how much memory your job will need?**
  Knowing approximately how many processes you will need or how much memory to request will help ensure you request enough resources to get your job to complete. One way to discover this is to run a small version of the job to see how much memory it uses and then calculate how much it would use if you were to double or triple it in size. We also offer a small development queue on the HPC cluster that you may submit small jobs to see how things go, and then tweak your resource requests accordingly.

### Getting your data into the cluster

If your data is not large, the quickest way to get your data onto one of the clusters is to use scp, rsync or sftp from the command line or via an application such as Fetch (Mac based) or IPSwitch (windows based). If you have larger data sets (larger meaning several Gigabytes or more), then you can utilize our [Globus Online connection](#).

### Storage Options

HPC accounts have a 1TB quota, but there are times when more storage, or a group share might be required for your work. ITS Research Services has made several [options available](#) in an attempt to meet these needs.

### Basic job submission

#### Software

Once your data is uploaded to the cluster, you are ready to work on getting a job submitted to the cluster. If you are going to use one of our centrally installed software packages, you'll need to load the module for it into your environment. More in-depth information on this is on our [Environment Modules](#) page. Basically, you'd use the command

```
module avail
```

To list the modules available to choose from. Then you'd use
module load <module-name>

To load that module into your environment. Note that some modules are not compatible, and will not load together. The environment module system will automatically load the correct dependent modules. You may also use the

module show <module-name>

to see what other modules will be loaded along with it, and also what modifications it will make to your environment.

Sizing Your Job To Fit The Cluster

Another factor to consider is how many resources you need to request from the cluster for your job. In High Performance Computing, resources are parceled out in units called "slots". A slot is a combination of a cpu & memory allocation based on the memory available from the nodes where your job will be running. The cluster has different types of machines inside of it which are defined by the number of cores and the amount of ram that each offers. Slots from each resource will be defined accordingly. For example, for a node with 64G of memory, a slot will be 1CPU & 4G RAM, while for a 256G memory node, a slot would be a proxy for 1CPU & 16G RAM. Once you have an idea of how many processors, and/or how much memory your computation will need, you can use this information to calculate how many slots you will need to request for your job.

For example, if your computational problem is to process data from thousands of large image files, you'd need to first figure out how much memory is required to process one file, and extrapolate accordingly. If processing each image requires 2G RAM, and a node offers 4G per slot, you could request one slot for each image.

You may find that doing small prototyping jobs are necessary in order to come up with an accurate resource request. For this, ITS-RS clusters offer a small development queue where you may run small versions of your jobs. You may also use qlogin to run interactively in order to get an idea of how your job will run on the cluster nodes.

Launching Your Job

Our cluster uses the SGE scheduler to match job submissions with available resources. There is extensive documentation on using SGE and all the options available. We offer pages on Basic Job Submission and Advanced Job Submission for the cluster. Launching jobs is done via qsub with options on the command line or via special commands in the job script which are then passed to the scheduler for controlling your job. A qsub script can be very simple, consisting of a few commands, or very complex, depending on your needs.

Note that if you use qsub options in the job script, then any additional options you pass to qsub on the command line when you launch the script will override those same settings inside the script. For example, if your script specifies the UI queue with #$ -q UI, but you would like to do a submission to the development queue for prototyping, you can override the UI queue on the command line with qsub:

qsub -q UI-DEVELOP <myscript.sh>

Monitoring Your Job

Once you have launched your job using qsub, you will want to be able to monitor its progress on the cluster. There are a couple of simple ways to do this. First, you need to find out what the "jobid" is of your job. A jobid is a unique number assigned by SGE to each job. To get this information, you can use the qstat command like so:

qstat -u <username>

which will produce the following output:

```
 qstat -u aarenas
 job-ID   prior name       user   state       submit/start at     queue         slots ja-task-ID
----------------------------------------------------------------------------------------------------------------
  44348  1.00692 BC80_9     aarenas      r     01/10/2019 11:21:50 IWA@argon-lc-h21-16.hpc           56
```

Note the "job-ID" is in the leftmost column. Other columns of note are the "state" column, which tells you what state your job is in. In this case the job is in "r" or "running" state, which means it has been assigned a node and is running. The "queue" column indicates which queue instance (queue-host) the job is running on. The "slots" column tells how many slots the job is requested. The "prior" field shows the scheduling priority of the job. When a job is in the "running" state, the priority is not really that meaningful, but the value would be more useful for pending jobs. However, syncing the scheduling priority values from the scheduler to the main queue process is a very expensive operation. Due to the size of Argon, and the varied types of jobs that are run on it, the synchronization of the scheduling priorities can cause severe slow downs, and possibly time outs, of SGE commands as SGE spends most of its time updating job priorities for display. Because of this, the synchronization of job priorities from the scheduler to the primary queue process has been turned off. The scheduler still schedules jobs with relative priority as one factor but the values of those priorities are not available to qstat, and display as 0.00000.
Use the `qstat -j <jobid>` command to view additional details about your job (note the below is abbreviated output):

```
$ qstat -j 8853637
==============================================================
job_number: 8853637
exec_file: job_scripts/8853637
submission_time: Tue Jan 6 09:57:10 2015
owner: naomi
uid: 1205679
group: its-rs-neon
gid: 899998927
sge_o_home: /Users/naomi
sge_o_log_name: naomi
sge_o_path: <path information here>
sge_o_shell: /bin/bash
sge_o_workdir: /Users/naomi/jobs/espresso
sge_o_host: neon-login-0-1
account: sge
cwd: /Users/naomi/jobs/espresso
merge: y
mail_options: abes
mail_list: naomi-hospodarsky@uiowa.edu
notify: FALSE
job_name: QE-CO2-Test-time
jobshare: 0
hard_queue_list: sandbox
shell_list: NONE:/bin/bash
env_list: <environment information here>
script_file: espresso-test.sh
parallel_environment: 16cpn range: 16
usage 1: cpu=00:01:57, mem=34.41240 GBs, io=0.18064, vmem=N/A, maxvmem=5.972G
scheduling info: (Collecting of scheduler job information is turned off)
```

The above information gives an overview of how your job looks to the scheduler. You can see job submission & start times, queue requests, slot requests, and the environment loaded at the time of job submission. One of the most useful lines in this output, however, is the "usage" line. This line will show you peak resource usage of your job. Pay special attention to "maxvmem" as this is the peak memory used by your job up to that point; you can use this information to help determine if you have requested enough resources for your job to operate with.

Two additional commands which may be useful are "qdel" for deleting jobs:

```
$ qdel -j <Jobid>  # deletes jobs by jobid
$ qdel -u <username>  # deletes all jobs owned by user
```

and qacct for gathering info about a completed job:

```
$ qacct -o <username> -j -d <days to report>  # shows accounting records for each of the user's job for the last x number of days
$ qacct -j <jobid>  # shows accounting record for a specific job
```

The accounting logs are rolled over and archived to prevent them from getting too large. They can be found in the `/opt/ohpc/admin/sge` directory. They are compressed but can be decompressed on the fly and fed into qacct.

```
zcat /opt/ohpc/admin/sge/accounting-20201201.gz | qacct -f -
```

**Conclusion**

This was a high-level introduction to HPC computing, and there are many topics not covered by this wiki page. Our other wiki pages offer more detail on various aspects of the system and how to use it. We also offer consulting services, so if you have questions about using our resources, HPC in general, or would simply like additional assistance getting started, please do not hesitate to contact our staff: research-computing@uiowa.edu, and one of us will be happy to help you.