Getting Started Guide

September 2019

Northeastern University
Research Computing

Contents

Introduction

This guide will help get you started with using the Discovery cluster system. It contains information and instructions on getting an account, connecting to the system, loading modules, running jobs, and storing data. If you are not familiar with high performance computing, you should take our training courses before working on the system. You can find information about the training and services that the Research Computing team provides at our website: http://www.rc.northeastern.edu.

We want your feedback! If you have any comments or suggestions, email us at rchelp@northeastern.edu and specify Documentation Request in the subject line.

What is Discovery

Discovery is a high performance computing (HPC) resource for the Northeastern University research community. The Discovery cluster is located in the Massachusetts Green High Performance Computing Center (MGHPCC) in Holyoke, MA. MGHPCC is a 90,000 square foot, 15 megawatt research computing and data center facility that houses computing resources for five institutions: Northeastern, BU, Harvard, MIT, and UMass.

See Hardware overview for more information on Discovery's hardware configuration.

See Software overview for more information on Discovery's available software.

Linux

Discovery is a Linux-based cluster. You should have some knowledge of Linux before attempting to use Discovery. The Research Computing group offers training sessions on Linux, as well as training on using Discovery and other topics related to the high performance computing environment. See the Research Computing website for more information on our training and services. You can also find information on how to use Linux on sites such as Code Academy.

Hardware overview

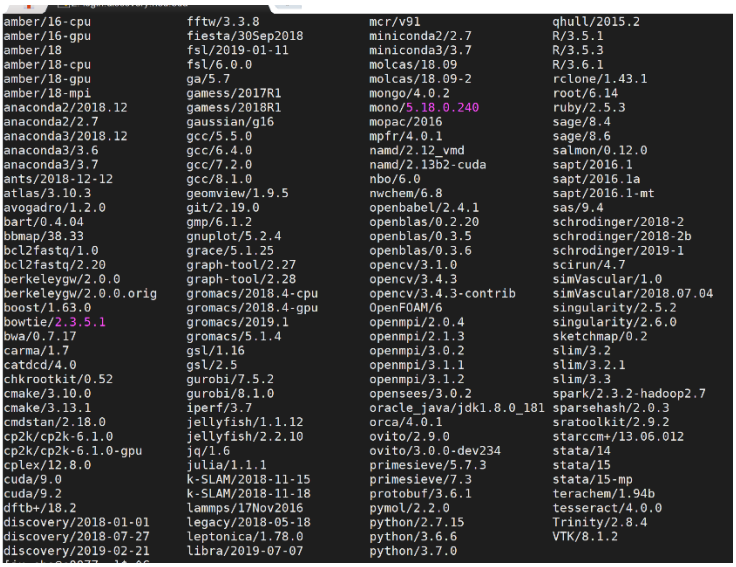
The Discovery cluster consists of a combination of the following CPUs and GPUs:

2.4 GHz Intel E5-2680 v4 CPUs
2.1 GHz Intel Xeon Platinum 8176 CPUs
a selection of NVIDIA K80, P100, V100, and T4 GPUs

This system provides you with access to over 20,000 CPU cores and over 200 GPUs. Discovery is connected to the university network over 10 Gbps Ethernet (GbE) for high-speed data transfer, and Discovery provides 3 PB of available storage on a high-performance GPFS parallel filesystem. Compute nodes are connected with either 10 GbE or a high-performance HDR100 InfiniBand (IB) interconnect running at 100 Gbps, supporting all types and scales of computational workloads. Full HDR IB connections (200 Gbps) are also available, if needed.

Software overview

Discovery has a number of software applications that are free for you to use. You can find and load the currently available software on Discovery using the `module avail` command (see Figure 1). See Using Module for more information.



```
amber/16-cpu              fftw/3.3.8              mcr/v91                 qhull/2015.2
amber/16-gpu              fiesta/30Sep2018        miniconda2/2.7          R/3.5.1
amber/18                  fsl/2019-01-11          miniconda3/3.7          R/3.5.3
amber/18-cpu              fsl/6.0.0               molcas/18.09            R/3.6.1
amber/18-gpu              ga/5.7                  molcas/18.09-2          rclone/1.43.1
amber/18-mpi              gamess/2017R1           mongo/4.0.2             root/6.14
anaconda2/2018.12         gamess/2018R1           mono/5.18.0.240         ruby/2.5.3
anaconda2/2.7             gaussian/g16            mopac/2016              sage/8.4
anaconda3/2018.12         gcc/5.5.0               mpfr/4.0.1              sage/8.6
anaconda3/3.6             gcc/6.4.0               namd/2.12_vmd           salmon/0.12.0
anaconda3/3.7             gcc/7.2.0               namd/2.13b2-cuda        sapt/2016.1
ants/2018-12-12           gcc/8.1.0               nbo/6.0                 sapt/2016.1a
atlas/3.10.3              geomview/1.9.5          nwchem/6.8              sapt/2016.1-mt
avogadro/1.2.0            git/2.19.0              openbabel/2.4.1         sas/9.4
bart/0.4.04               gmp/6.1.2               openblas/0.2.20         schrodinger/2018-2
bbmap/38.33               gnuplot/5.2.4           openblas/0.3.5          schrodinger/2018-2b
bcl2fastq/1.0             grace/5.1.25            openblas/0.3.6          schrodinger/2019-1
bcl2fastq/2.20            graph-tool/2.27         opencv/3.1.0            scirun/4.7
berkeleygw/2.0.0          graph-tool/2.28         opencv/3.4.3            simVascular/1.0
berkeleygw/2.0.0.orig     gromacs/2018.4-cpu      opencv/3.4.3-contrib    simVascular/2018.07.04
boost/1.63.0              gromacs/2018.4-gpu      OpenFOAM/6              singularity/2.5.2
bowtie/2.3.5.1            gromacs/2019.1          openmpi/2.0.4           singularity/2.6.0
bwa/0.7.17                gromacs/5.1.4           openmpi/2.1.3           sketchmap/0.2
carma/1.7                 gsl/1.16                openmpi/3.0.2           slim/3.2
catdcd/4.0                gsl/2.5                 openmpi/3.1.1           slim/3.2.1
chkrootkit/0.52           gurobi/7.5.2            openmpi/3.1.2           slim/3.3
cmake/3.10.0              gurobi/8.1.0            opensees/3.0.2          spark/2.3.2-hadoop2.7
cmake/3.13.1              iperf/3.7               oracle_java/jdk1.8.0_181 sparsehash/2.0.3
cmdstan/2.18.0            jellyfish/1.1.12        orca/4.0.1              sratoolkit/2.9.2
cp2k/cp2k-6.1.0           jellyfish/2.2.10        ovito/2.9.0             starccm+/13.06.012
cp2k/cp2k-6.1.0-gpu       jq/1.6                  ovito/3.0.0-dev234      stata/14
cplex/12.8.0              julia/1.1.1             primesieve/5.7.3        stata/15
cuda/9.0                  k-SLAM/2018-11-15       primesieve/7.3          stata/15-mp
cuda/9.2                  k-SLAM/2018-11-18       protobuf/3.6.1          terachem/1.94b
dftb+/18.2                lammps/17Nov2016        pymol/2.2.0             tesseract/4.0.0
discovery/2018-01-01      legacy/2018-05-18       python/2.7.15           Trinity/2.8.4
discovery/2018-07-27      leptonica/1.78.0        python/3.6.6            VTK/8.1.2
discovery/2019-02-21      libra/2019-07-07        python/3.7.0
```

Figure 1: Using module avail to list the software available on Discovery

Discovery partitions

The compute nodes on Discovery are organized into partitions. Different partitions have different combinations of hardware, such as having only CPUs or GPUs or a combination of both. The following table shows the current partition configuration on Discovery.

Name	Nodes	CPUs	GPUs
general	316	6828	0
gpu	48	896	48
test	2	40	0
interactive	2	40	0
infiniband	64	1024	0
multigpu	24	688	120
fullnode	408	11424	0

FIRST TASKS

Getting Access

You must first have an account before you can access Discovery. You request an account through ServiceNow. You will need a Northeastern username and password to access ServiceNow. If you are new to the university or a visiting researcher, work with your sponsor to get an Northeastern username and password.

To request an account, follow these steps:

1. Go to ServiceNow.
2. Fill out the form, check the acknowledgement box, and click Submit.

Your request can take up to 24 hours to process. You will receive a confirmation email when your access has been granted.

After you have access, if you are not familiar with using Discovery, high performance computing, or Linux, you should take one of our training courses. See the Research Computing website for more information on our training and services.

FullNode Access

If you require access to the fullnode partition, you will need to download and fill out the fullnode partition application. You can find the application in the Policies section on the Research Computing website.Then, email the application to p.whitford@northeastern.edu. This application should only be filled out after you have general access to Discovery. See Getting Access for instructions on getting an account on Discovery.

MultiGPU Access

If you require access to the multigpu partition, you will need to download and fill out the multigpu partition application. You can find the application in the Policies section on the Research Computing website. Then, email the application to d.kaeli@northeastern.edu and ioannidis@ece.neu.edu. This application should only be filled out after you have general access to Discovery. See Getting Access for instructions on getting an account on Discovery.

Connecting to Discovery

You connect to Discovery using a secure shell program and initiate an SSH session to log into Discovery.

Mac

Mac computers come with a Secure Shell program called Terminal that you use to connect to Discovery using SSH.

To connect to Discovery on a Mac:

1. Open Terminal.
2. Type `ssh <username>@login.discovery.neu.edu` , where `<username>` is your Northeastern username.

3. Type your Northeastern password and press Enter.

You are now connected to Discovery at a login node.

✕ **Caution:** Never launch a job from the login node. Jobs that are launched from the login node will be terminated. Move to a compute node before running any jobs.

Windows

Before you can connect to Discovery on a Windows computer, you'll need to download a terminal program, such as MobaXterm or PuTTY. We recommend MobaXterm, as you can also use it for file transfer, whereas with other SSH programs, you would need a separate file transfer program.

To connect to Discovery with MobaXterm:

1. Open MobaXterm.
2. Click **Session**, then click **SSH** as the connection type.
3. In **Remote Host**, type `login.discovery.neu.edu`, make sure **Port** is set to 22, and click **OK**. You do not need to enter a username.
4. At the prompt, type your Northeastern username and press Enter.
5. Type your Northeastern password and press Enter. Note that the cursor does not move as you type your password. This is expected behavior.

You are now connected to Discovery at a login node.

✕ **Caution**: Never launch a job from the login node. Jobs that are launched from the login node will be terminated. Move to a compute node before running any jobs.

Move to a compute node

You should never launch any jobs from the login node `[username@login-00~]`. Any job launched from the login node will be terminated. You can move to a compute node using the `srun` command or use the `sbatch` command to run a job on an interactive compute node.

- To move to a compute node, at the command prompt type `srun --pty /bin/bash`
- See Submitting Jobs for information about using `sbatch`.

Using Discovery

Transferring Data

Discovery has a dedicated transfer node that you must use to transfer data to and from the cluster. You are not allowed to transfer data from any other node. The node name is `<username>@xfer.discovery.neu.edu`: where `<username>` is your Northeastern username.

✕ **CAUTION**: The /scratch directory is for temporary file storage only. It is not backed up. You should immediately transfer any data off /scratch to another location.

Data transfer node using Mac

You can use Terminal to transfer data to and from Discovery.

Use this command to transfer a file to your /scratch space:

```
scp <filename> <yourusername>@xfer.discovery.neu.edu:/scratch/<yourusername>
```

Where `<filename>` is the name of the file you want to transfer and `<yourusername>` is your Northeastern username.

Data transfer node using Windows

You can use MobaXterm to transfer data to and from Discovery. You can also use a file transfer program, such as Filezilla.

To transfer data using MobaXterm:

1. Open MobaXterm.
2. In the **Host** field, type sftp://xfer.discovery.neu.edu
3. In the **Username** filed, type your Northeastern username.
4. In the **Port** field, type 22.
5. In the **Password** field, type your Northeastern password.

You will now be connected to the transfer node and can transfer files.

Using Module

The `module` command allows you to easily install software packages in your path on Discovery. Discovery has several software packages available for you to use. See Software for a list of software or use the `module avail` command to list the software on Discovery.

Module Command	What it does
`module avail`	Shows a list of available modules
`module whatis <modulename>`	Provides information about the module that you specify
`module load/add`	Loads the module in your path
`module list`	Lists the modules that you have loaded
`module delete`	Deletes a module

Using Slurm

Slurm (Simple Linux Utility Resource Management) is the scheduling software that you use to schedule your jobs on Discovery. This section provides you with a few simple examples to help get you familiar with Slurm and be able to submit and monitor basic jobs on Discovery. You should refer to Slurm's documentation for detailed information on the commands and parameters available for more complex scheduling jobs.

Commonly used commands include the following:

Command	What it does
`sbatch <filename>`	Sends the job to the scheduler
`srun`	Launches an interactive session
`squeue`	Allows you to see what jobs are currently waiting and running
`scancel <job id>`	Removes a running or pending job from the queue
`scontrol <flags>`	Displays more information about the machine configuration and job settings
`seff <job id>`	Reports the computational efficiency of your calculations

Submitting Jobs

You use Slurm to submit jobs to Discovery. Slurm commands allow you to specify the resources you need to run your jobs, such as which nodes you want to run you jobs on and how much memory you'll need. Slurm then schedules your job based on the availability of the resources you've specified.

The general format for submitting a job to the scheduler is as follows:

```
sbatch example.script
```

Where `example.script` is a script detailing the parameters of the job you want to run. See Example Job ScriptsExample Job Scripts for more information on writing scheduler scripts.

ⓘ **NOTE**: The default time limit for submitted jobs is 24 hours. If your job does not complete within the requested time limit, Slurm will automatically terminate the job. You also cannot request more than 24 hours, as your job will not launch.

TIP: Slurm looks at the time you've requested when scheduling jobs. If you request less time, Slurm may be able to schedule your job sooner. For example, if there is a high priority job that requires a specific number of nodes, and Slurm anticipates that those nodes will be available in 6 hours, Slurm will let lower priority jobs run if they

will complete within the 6 hours the higher priority job has to wait for the specific nodes to be available.

Example Job Scripts

The following are example job scripts for submitting to Slurm using the `sbatch` command. You should review these to have a general understanding of the types of jobs you can submit to the scheduler. See Using Slurm Using Slurmand Submitting Jobs for more information on using the scheduler on Discovery.

1-core job

Run a 1-core job for 4 hours on the general partition:

```
-----------------------------
#!/bin/bash
#SBATCH -nodes=1
#SBATCH -time=4:00:00
#SBATCH -job-name=MyJobName
#SBATCH -partition=general
<commands to execute>
-----------------------------
```

1-core job and additional memory

The default memory per allocated core is 1GB. If your calculations try to use more memory than what is allocated, Slurm automatically kills your job. You should request a specific amount of memory in your job script if your calculations need more memory than the default. The example script below is requesting 100GB of memory (mem=100Gb).

```
-----------------------------
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --mem=100Gb
#SBATCH --partition=general
<commands to execute>
-----------------------------
```

1-core job with exclusive use of a node

If you need exclusive use of a node, such as when you have a job that has high I/O requirements, you can use the exclusive flag. The example script below specifies exclusive use of 1 node in the general partition for four hours.

```
-----------------------------
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --exclusive
#SBATCH --partition=general
```

```
<commands to execute>
-----------------------------
```

Example Parallel Job Scripts

Parallel jobs should be used with code that is configured to use the reserved resources. If your code is not optimized for running in parallel, your job could fail.

The following script examples all allocate additional memory. The default memory per allocated core is 1GB. If your calculations try to use more memory than what is allocated, Slurm automatically kills your job. You should request a specific amount of memory in your job script if your calculations need more memory than the default.

8-task job, one node and additional memory

```
-----------------------------
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --tasks-per-node=8
#SBATCH --cpus-per-task=1
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --mem=100Gb
#SBATCH --partition=general
<commands to execute>
-----------------------------
```

8-task job, multiple nodes and additional memory

```
-----------------------------
#!/bin/bash
#SBATCH --nodes=4
#SBATCH --tasks-per-node=2
#SBATCH --cpus-per-task=1
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --mem=100Gb
#SBATCH --partition=general
<commands to execute>
-----------------------------
```

8-task job, multiple nodes, additional memory, and exclusive

If you need exclusive use of a node, such as when you have a job that has high I/O requirements, you can use the exclusive flag.

```
-----------------------------
#!/bin/bash
#SBATCH --nodes=4
#SBATCH --tasks-per-node=2
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --mem=100Gb
#SBATCH --exclusive
#SBATCH --partition=general
<commands to execute>
```

```
------------------------------
```

Submitting GPU Jobs

If you need to submit your job to a GPU node, you need to specify this in your script. Refer to the following script examples for how to specify a GPU node.

1 Node, 1 Core, and 1 GPU

```
------------------------------
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --partition=gpu
#SBATCH --gres=gpu:1
<commands to execute>
------------------------------
```

1 Node, All Compute Cores, and 1 GPU

```
------------------------------
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --partition=gpu
#SBATCH --gres=gpu:1
#SBATCH -exclusive
<commands to execute>
------------------------------
```

4 Nodes, All Compute Cores, and 1 GPU Per Node

```
------------------------------
#!/bin/bash
#SBATCH --nodes=4
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --partition=gpu
#SBATCH --gres=gpu:1
#SBATCH -exclusive
<commands to execute>
------------------------------
```

4 Nodes, All Compute Cores, and 1 GPU Per Node

```
------------------------------
#!/bin/bash
#SBATCH --nodes=4
#SBATCH --tasks-per-node=2
#SBATCH --cpus-per-task=6
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --partition gpu
#SBATCH --gres=gpu:1
<commands to execute>
```

```
------------------------------
```

Specifying Required GPU Types

You can specify that a specific GPU type be used in your job. The following GPU designations are available:

- gpu partition: K20, k40m
- multigpu partition: k80, p100 (4 per node)

```
------------------------------
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --tasks-per-node=8
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --partition=gpu
#SBATCH --gres=gpu:k20:1
<commands to execute>
------------------------------
```

Redirecting stdout/stderr

By default, Slurm writes all `stdout` and `stderr` to a single file called `slurm-<jobid>.out`, where `<jobid>` is assigned at the time of submission. If you want to write `stderr` and `stdout` to specific files, use the flags as shown in the example:

```
------------------------------
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --partition=general
#SBATCH --output=myjob.%j.out
#SBATCH --error=myjob.%j.err
<commands to execute>
------------------------------
```

Specifying Required Node Features

You can use the `sinfo` command with the `--format="%b"` flag to display the features of the node.

Using Globus

As a member of the Northeastern Research community, you have access to Globus. If you would like to use this resource, email rchelp@northeastern.edu with the details of this request. A ServiceNow ticket will be generated for you, and a member of the RC team will help get you connected to Globus.

Getting Help

You can get help from the Research Computing team by sending an email, submitting a ticket on ServiceNow, or making an appointment to meet with one of our consultants through our Bookings page.

ⓘ **NOTE**: To request an account, you must complete a ServiceNow ticket. See Requesting an Account for more information.

Email

You can email the Research Computing team at rchelp@northeastern.edu. This will automatically generate a ticket for you in ServiceNow. Make sure to include details about your question or issue, including any commands or scripts that you are trying to use, so that we can best connect you with the right person to help you.

ServiceNow

You can submit a ticket on ServiceNow by selecting from the Research Computing ServiceNow catalog.

Walk in (Bookings)

We offer scheduled, walk-in consultation hours at the Snell Library most weekdays during normal business hours (9AM to 5PM). Note that we follow the Northeastern University holiday schedule, so there are no walk-in hours on holidays or during breaks. Use our Bookings page to see our availability and to schedule an appointment.