NYU HPC, within IT, operates and supports high performance computing resources and assists the NYU research community in their use. **HPC resources are open to NYU faculty and staff as well as faculty-sponsored students, and may be used for class instruction.** IT is also available to partner with faculty as they seek funding for research with substantial technology components—see **HPC Stakeholders** and also **ITS Research Computing**. We can also assist in access to and collaboration with a number of **national and state HPC facilities**.

### Upcoming Events

- **Looking for one-on-one HPC help? Email us to book an HPC Consultation**

### New Pages

- **Data Transfer Nodes**
- **Acknowledge Statement**
- **Slurm Tutorial**
- **Programming for Biologists**
- **Licensed Software Available on the HPC Cluster**
- **Scratch Area Cleanup**

### cluster maintenance

**The Prince cluster maintenance on Monday 1/13/2020 from 9 AM to 5 PM**

The HPC Prince cluster will undergo maintenance and will not be available on Monday Jan. 13th, 2020 from 9am to 5pm. During the maintenance period, users will not be able to login or submit jobs. All running jobs will be terminated. JupyterHub servers with URL https://*.jupyter.hpc.nyu.edu will not be available either.

The following HPC-provided services will not be impacted by the maintenance of the HPC cluster:

- Hadoop cluster (Dumbo)
- OpenStack cluster (Brooklyn Research Cluster)

### public dataset

- ImageNet dataset is available on prince cluster for public use: `/scratch/work/public/imagenet/train`
File purging for /beegfs will start from Thursday 6/20/2019

Files in /beegfs that have not been accessed for more than 60 days will be removed at any time from Thursday 6/20. Please always back up important data to /archive.

Service Announcement

Dedicated HPC Data Transfer Nodes are online

Cluster status

- Prince cluster is operating normally today
- Dumbo cluster is operating normally today

Newly available service

Overleaf: NYU’s New Online, Collaborative LaTeX Writing and Publishing Service

Overleaf is an online LaTeX and Rich Text collaborative writing and publishing service that makes the whole process of creating and sharing scientific documents much quicker and easier. NYU now provides the Professional version of the software to students, faculty, researchers, and staff. This version of Overleaf offers the following features to help you and your collaborators create beautiful projects, some of which are only available to Overleaf Professional users:

- Real-time collaboration in your web browser
- Effortless and flexible sharing options
- Real-time preview
- Rich Text mode
- Fast error finding
- Great for papers, theses and presentations
- Publish to many academic journals, repositories and authoring services
- Advanced access control for protected projects
- Full project and document history
- Quick save to Dropbox option
- Priority support directly from Overleaf

For information about creating a new account or converting an existing one to the NYU license, visit www.nyu.edu/it/software/overleaf.
Prince How-to Articles
Logging in
  Windows
  Mac / Linux
Clusters and Storage
  Prince (HPC)
  Dumbo (Hadoop)
  Brooklyn (OpenStack)
  Dalma (NYU Abu Dhabi)
Transferring data to/from the clusters
Transferring data to/from Prince cluster using Globus
Submitting jobs with sbatch
Available software
Licensed Software
Available on the HPC Cluster
Building Software
Slurm Tutorial
Tutorials
FAQs
Scratch Area Cleanup
Programming for Biologists
Acknowledge Statement
Getting and Renewing Access

For how to log in, see Logging in to the NYU HPC Clusters

Bulk HPC Accounts for Courses

HPC bulk accounts request is disabled for HPC sponsors. Please fill out this Request Form for the course, we'll create HPC accounts for the class per request

Click here to request HPC accounts

Who is eligible for an HPC account?

NYU HPC resources are available at no charge to full-time NYU faculty and to all other NYU staff and students with full-time NYU faculty sponsorship (more...)

Getting an account on the NYU HPC clusters

First you need a valid NYU NetID. Your HPC sponsor can request one for you here. You also need a valid NYU Google account to receive emails, as does your HPC sponsor - contact us if you need assistance with this.

Next you need a faculty sponsor.

Finally, log into the NYU Identity Management service and follow the link to "Request HPC account". We have a walkthrough of the process here.

Renewing your HPC account

Each year, non-faculty users must renew their HPC account by filling in the account renewal form from the NYU Identity Management service. See Renewing your HPC account with IIQ for a walk-through of the process.

Information for faculty who sponsor HPC users

You can request a NetID for your student or collaborator here. The request form has additional information about affiliates.

Each year, your sponsored users must renew their account. You will need to approve the renewal by logging into the NYU Identity Management service. We have a walkthrough of the process, with screenshots, here.
Pre-approving a list of netids for class HPC accounts (see notice above)

Faculty (who can sponsor HPC accounts) can pre-approve requests in bulk - this is intended to streamline the process of registering a class to use the HPC facilities. Faculty can set this up via the NYU Identity Management service. We also have a walkthrough of the process here.

Getting an account with one of NYU partners

NYU partners with many state and national facilities with a variety of HPC systems and expertise. Contact us for assistance setting up a collaboration with any these.

- The Open Science Data Cloud
  Provides 1TB free storage for science data. We encourage researchers to publish datasets associated with published research as "Public Data" on OSDC

- The NY State High Performance Computing Consortium (hpc^2)
  Provides high performance computing resources for New York State industry and academic institutions:
  - Rensselaer Polytechnic Institute
  - Stony Brook University - Dave Ecker
  - University at Buffalo
  - Brookhaven National Lab
  - NYSERNet

- The Extreme Science and Engineering Discovery Environment (XSEDE)
  The most advanced, powerful, and robust collection of integrated advanced digital resources and services in the world; a single virtual system that scientists can use to interactively share computing resources, data, and expertise.

- Open Science Grid
  A national, distributed computing grid for data-intensive research.

- The Common Solutions Group
  for cooperative exploration of common solutions to IT challenges in higher education

- The Open Science Project
  is dedicated to writing and releasing free and Open Source scientific software.

- NYSERNet
  is a private not-for-profit corporation created to foster science and education in New York State

- The National Science Foundation
  An independent federal agency created by Congress in 1950 "to promote the progress of science; to advance the national health, prosperity, and welfare; to secure the national defense."

- Oak Ridge National Laboratory
  The Department of Energy's largest science and energy laboratory.

- Argonne National Laboratory
  One of the U.S. Department of Energy's largest research centers. It is also the nation's first national laboratory, chartered in 1946.

- TOP500 Supercomputer Sites
  A project started in 1993 to provide a reliable basis for tracking and detecting trends in high-performance computing.

Contact us
Request Form

HPC Stakeholders

Introduction

NYU IT Research Technology Services (RTS) supports and encourages a model of hosting and managing clusters for research groups or departments in return for making their unused cluster cycles available to other NYU HPC users. These research groups and departments are our HPC Stakeholders, for whom NYU HPC manages the computing resources and provides priority access. If you are interested in becoming a stakeholder, please contact us at hpc@nyu.edu for details. We can discuss your research computing needs, develop a Service Agreement, and work with you in the planning and purchase of servers. Stakeholders are part of the NYU HPC governance process and they meet once per semester with the HPC team to discuss and approve proposed changes and adjustments to HPC policies (upgrades to the clusters, security, job scheduling, downtime, etc.).
## Current HPC Stakeholders

<table>
<thead>
<tr>
<th>Research Group/Lab</th>
<th>Contact Person(s)</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center for Neural Science (CNS - XJ Wang Lab)</td>
<td>Xiao-Jing Wang, Vishwa Goudar</td>
<td>12 Prince Compute Nodes with 256GB of RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 Prince Compute Nodes each equipped with 4x P40 NVIDIA GPU cards</td>
</tr>
<tr>
<td>Tandon CSE/ViDA Project</td>
<td>Claudio Silva, Yitzchak Lockerman</td>
<td>OpenStack - Brooklyn Research Cluster - <a href="#">hardware</a></td>
</tr>
<tr>
<td>Center for Genomics and Systems Biology (CGSB)</td>
<td>David Gresham, Mohammed Khalfan</td>
<td>8 Prince compute Nodes with 128GB RAM</td>
</tr>
<tr>
<td></td>
<td>John Bako</td>
<td>4 Prince Compute Nodes with 256GB RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50 TB of backed up data storage</td>
</tr>
<tr>
<td>Stern School of Business - Research Computing</td>
<td>Norm White, Robin Wurl</td>
<td>2 Prince Compute Nodes with Xeon PHI KNL</td>
</tr>
<tr>
<td>Center for Data Science (CDS)</td>
<td>Sam Bowman, Kyunghyun Cho, Tom Kirchner</td>
<td>4 Prince Compute Nodes each equipped with 4x 1080 NVIDIA GPU cards</td>
</tr>
<tr>
<td>Center for Urban Science and Progress (CUSP)</td>
<td></td>
<td>NYULMC</td>
</tr>
<tr>
<td>Social Media and Political Participation (SMaPP)</td>
<td>Leon Yin</td>
<td>3 nodes on the MongoDB cluster (Hades)</td>
</tr>
<tr>
<td>Department of Biology</td>
<td>Edo Kussell</td>
<td>4 Prince Compute Nodes (Dell PowerEdge 6140 in a C6400 enclosure)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(December 2018)</td>
</tr>
</tbody>
</table>

(more...)

### Compute and Storage Facilities

[Prince] [Dumbo] [Brooklyn]

The NYU HPC team currently maintains three clusters: The HPC cluster **Prince**, the Hadoop cluster **Dumbo** and OpenStack cluster **Brooklyn**

### HPC user accounts

An HPC User account provides access to all NYU HPC and Big Data clusters. If you don't have a user account, you may [apply for an HPC user account](#).
Old HPC clusters

NYU HPC team has retired its older clusters (Union Square, Cardiac, Bowery, Mercer). The current production HPC cluster is Prince.

Prince

Prince is the main NYU HPC cluster.

- For a description of the HPC Prince cluster, see Clusters - Prince.
- For information on how to access and use the HPC Prince cluster, see Getting started on Prince.

Dumbo

Dumbo is a 44 data node Hadoop cluster running Cloudera Distribution of Hadoop (CDH).

- For a detailed description of dumbo and how to access it, please see the dumbo wiki pages.

Brooklyn

- Brooklyn is an OpenStack cluster consisting of 25 compute nodes, each equipped with 4 GPUs.
- For a detailed description of the Brooklyn Research cluster, please see Clusters - Brooklyn Research Cluster
- Presentation Slides

Logging in to the NYU HPC Clusters

The HPC clusters (Prince and Dumbo) are not directly visible to the internet (outside the NYU Network). If you are outside NYU's Network (off-campus) you must first login to a bastion host named gw.hpc.nyu.edu

The diagram below illustrates the login path.
**NOTE:** The clusters can still access the internet *directly*. This may be useful when copying data from servers outside the NYU Network - see: How to copy files to and from the HPC clusters.

**NOTE:** Alternatively, instead of login to the *bastion host*, you can use *VPN* to get inside NYU's network and access the HPC clusters directly. Instructions on how to install and use the VPN client are available here.

**NOTE:** You can't do anything on the *bastion host*, except *ssh* to the HPC clusters.

In a nutshell
• From **within the NYU network** that is, from an on-campus location, or after you **VPN** inside NYU’s network, you can login to the HPC clusters **directly**.

To login to the HPC cluster **Prince**, simply use **replace NYUNetID with your NetId**:::

```bash
ssh NYUNetID@prince.hpc.nyu.edu
```

To login to the Hadoop cluster (**Dumbo**):

```bash
ssh NYUNetID@dumbo.hpc.nyu.edu
```

• From an **off-campus location (outside NYU-NET)**, logging in to the HPC clusters is a **two-step process**:

  a. First login to the bastion host, **gw.hpc.nyu.edu**. From a Mac or Linux workstation, this is a simple terminal command (**replace NYUNetID with your NetId**). Your password is the same password you use for NYU Home:

```bash
ssh NYUNetID@gw.hpc.nyu.edu
```

  ! You can’t do anything on the bastion host, except `ssh` to the cluster

  b. Next login to the cluster. For Prince, this is done with:

```bash
ssh prince.hpc.nyu.edu
```

For Dumbo, this is done with:

```bash
ssh dumbo.hpc.nyu.edu
```

The full story

You need to ensure your workstation has the necessary software and settings to connect to the clusters and to use graphical interfaces. Here are instructions for preparing your workstation and logging in from a **Windows / Linux / Mac**.

**SSH tunneling for easier login and data transfer**

The two-stage access can be inconvenient, especially when transferring files to and from the clusters. Secure direct access and file transfer is possible by setting up **SSH tunneling** from your workstation to the HPC clusters. We have instructions on setting this up for **Windows / Linux / Mac** workstations.

**What can I do on the login node?**

The **login nodes** (**prince** and **dumbo**) are for preparing, submitting and monitoring scripts, analyzing results, moving data around and code development and simple compilation. **Login nodes are Not suitable for running computational workloads!** - for Prince use this batch system.

Compiling a large source codebase, especially with heavy use of optimization or -ipo (interprocedural optimization), can use much memory and CPU time. In such circumstances it is best to use the batch system for compilation too, perhaps via an interactive batch job. Click here for more info about **interactive batch jobs**.

**Finding and Using Software**

A variety of commercial and open-source software is available on the NYU HPC clusters, and can be **accessed via Environment Modules**.
NYU HPC hosts licenses for a number of commercial software packages which are suitable for workstation as well as HPC use, such as Matlab, COMSOL and Mathematica. Contact us about accessing these packages.

If you need a free or open source software package which is not currently available on the HPC clusters, contact us. Usually we can install it for you, or suggest an alternative which is already available.

Our ability to buy and install commercial software depends on the cost and on how widely it will be used. We may also be able to host licenses or share costs with you in return for making the software available also to the NYU research community, so if you need a specific commercial package contact us to discuss it.

Intel and GNU compilers are available on the clusters. For most code, we recommend the Intel compilers.

For debugging we have the GNU debugger gdb, the Intel debugger idb and Totalview by Roguewave. Debugging is best performed with an interactive batch session.

There is more about compiling and debugging on the old wiki pages.

Managing data: Storage, collaboration and moving data around

Filesystems, their optimal usage and your space allocation are described under Storage July 2017.

Quotas

On Mercer, enter 'myquota' at the prompt to see how much space you have used and available on each filesystem.

Security and collaboration: file permissions and ACL on NYU HPC clusters

By default, only you can edit, or even see, your files. You can grant permission for your colleagues to see or edit files with `setfacl`, and you can check the permissions on a file or directory with `getfacl`.

An access control list (or ACL) gives per-file, per-directory and per-user control over who can read, write and execute files. You can see the ACL for a file or directory with the `getfacl` command:

```
$ getfacl myfile.txt
```

To modify permissions for files or directories, use `setfacl`. For a detailed description, see 'man setfacl'. In the example below, I give read permission on `dummy.txt` to user `bob123`:

```
$ setfacl -m u:bob123:r dummy.txt
```

For setting execute permission on files - useful for scripts, and for allowing directories to be entered - `chmod` is still used.

(more...)

Transferring files to and from the HPC clusters

To copy data between your workstation and the NYU HPC clusters, you must set up and start an SSH tunnel on the workstation. We have instructions for this for Windows, Mac and Linux workstations.

Once you have an SSH tunnel, you can transfer files to and from the HPC clusters - including BuTinah at NYUAD.

Submitting jobs with sbatch: How to use the batch system
Quick Links

HPC Home
Getting an account
Gentle Introduction to using HPC
Getting started on Prince
Prince How-to Articles
Logging in
  Windows
  Mac / Linux
Clusters and Storage
  Prince (HPC)
  Dumbo (Hadoop)
  Brooklyn (OpenStack)
  Dalma (NYU Abu Dhabi)
Transferring data to/from the clusters
Transferring data to/from Prince cluster using Globus
Submitting jobs with sbatch
<table>
<thead>
<tr>
<th>Running jobs on the Prince Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accessing the Prince Cluster</strong></td>
</tr>
<tr>
<td>From Windows workstation</td>
</tr>
<tr>
<td>From Mac workstation</td>
</tr>
<tr>
<td><strong>Software and Environment Module</strong></td>
</tr>
<tr>
<td><strong>Job script and resource request</strong></td>
</tr>
<tr>
<td>Introduction to job scheduling</td>
</tr>
<tr>
<td>Submitting jobs with sbatch</td>
</tr>
<tr>
<td>Requesting resources</td>
</tr>
<tr>
<td>Using computing nodes interactively</td>
</tr>
<tr>
<td><strong>Monitoring batch jobs</strong></td>
</tr>
<tr>
<td>Monitoring batch jobs - squeue</td>
</tr>
<tr>
<td>What is running and where? slurmtop</td>
</tr>
<tr>
<td><strong>Canceling your jobs</strong></td>
</tr>
<tr>
<td><strong>Compiling your own software</strong></td>
</tr>
<tr>
<td><strong>Putting all pieces together</strong></td>
</tr>
<tr>
<td>An Amber example</td>
</tr>
<tr>
<td>An R example</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
</tr>
</tbody>
</table>
**Batch vs interactive**

The working pattern we are all familiar with is *interactive* - I type (or click) something, and the computer performs the associated action. Then I type (or click) the next thing.

You may recall this from the first tutorial.

**The trouble with interactive environments**

There is another reason why GUIs are less common in HPC environments: **point-and-click is necessarily interactive**. In HPC environments (as we'll see in session 3) work is scheduled in order to allow exclusive use of the shared resources. On a busy system there may be several hours wait between when you submit a job and when the resources become available, so a reliance on user interaction is not viable. In Unix, commands need not be run interactively at the prompt, you can write a sequence of commands into a file to be run as a script, either manually (for sequences you find yourself repeating frequently) or by another program such as the batch system.

The job might not start immediately, and might take hours or days, so we prefer a *batch* approach:

- plan the sequence of commands which will perform the actions we need
- write them into a script

I can now run the script interactively, which is a great way to save effort if I frequently use the same workflow, or ...

- submit the script to a batch system, to run on dedicated resources when they become available

**Where does the output go?**

- The batch system writes stdout and stderr from a job to a file named "slurm-12345.out"
  - Which you can change, using sbatch options
- While a job is running, it caches the stdout and stderr in the job working directory
- You can use redirection (See Tutorial 1) to send output of a specific command into a file

**Writing and Submitting a job**

There are two aspects to a batch job script:

- A set of SBATCH directives describing the resources required and other information about the job
- The script itself, comprised of commands to setup and perform the computations without additional user interaction

**A simple example**

A typical batch script on an NYU Prince cluster looks something like these:

<table>
<thead>
<tr>
<th>Using precompiled third-party software</th>
<th>Using self-developed or built software</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We’ll work through them more closely in a moment.

You submit the job with sbatch:

```bash
$ sbatch myscript.s
```

And monitor its progress (as is discussed further in [here](#)) with:

```bash
$ squeue -u $USER
```

**What just happened?** Here’s an annotated version of the first script:
#!/bin/bash

# This line tells the shell how to execute this script, and is unrelated to SLURM.

# at the beginning of the script, lines beginning with "#!/SBATCH" are read by
# SLURM and used to set queueing options. You can comment out a SBATCH
# directive with a second leading #, eg:
##SBATCH --nodes=1

# we need 1 node, will launch a maximum of one task and use one cpu for the task:
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1

# we expect the job to finish within 5 hours. If it takes longer than 5
# hours, SLURM can kill it:
#SBATCH --time=5:00:00

# we expect the job to use no more than 2GB of memory:
#SBATCH --mem=2GB

# we want the job to be named "myTest" rather than something generated
# from the script name. This will affect the name of the job as reported
# by squeue:
#SBATCH --job-name=myTest

# when the job ends, send me an email at this email address.
#SBATCH --mail-type=END
#SBATCH --mail-user=bob.smith@nyu.edu

# both standard output and standard error are directed to the same file.
# It will be placed in the directory I submitted the job from and will have a name like slurm_12345.out
#SBATCH --output=slurm_%j.out

# once the first non-comment, non-SBATCH-directive line is encountered, SLURM stops looking for SBATCH directives.
# The remainder of the script is executed as a normal Unix shell script

# first we ensure a clean running environment:
module purge

# and load the module for the software we are using:
module load stata/14.2

# next we create a unique directory to run this job in. We will record its name in the shell variable "RUNDIR", for better readability.
# SLURM sets SLURM_JOB_ID to the job id, ${SLURM_JOB_ID/.*} expands to the job id up to the first '.' We make the run directory in our area under $SCRATCH, because at NYU HPC
# $SCRATCH is configured for the disk space and speed required by HPC jobs.
RUNDIR=${SCRATCH}/my_project/run-$(${SLURM_JOB_ID/.*})
mkdir $RUNDIR

# we will be reading data in from somewhere, so define that too:
DATADIR=${SCRATCH}/my_project/data

# the script will have started running in $HOME, so we need to move into the
# unique directory we just created
cd $RUNDIR

# now start the Stata job:
stata -b do $DATADIR/data_0706.do
Submitting a job

Jobs are submitted with the sbatch command:

```
$ sbatch options job-script
```

The options tell SLURM information about the job, such as what resources will be needed. **These can be specified in the job-script as SBATCH directives, or on the command line as options, or both** (in which case the command line options take precedence should the two contradict each other). For each option there is a corresponding SBATCH directive with the syntax:

```
#SBATCH option
```

For example, you can specify that a job needs 2 nodes and 4 cores on each node (by default one CPU core per task) on each node by adding to the script the directive:

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
```

or as a command-line option to sbatch when you submit the job:

```
$ sbatch --nodes=2 --ntasks-per-node=4 my_script.s
```

Options to manage job output:

- `-J` `jobname`
  
  Give the job a name. The default is the filename of the job script. Within the job, `SSBATCH_JOB_NAME` expands to the job name
- `-o` `path/for/stdout`
  
  Send stdout to `path/for/stdout`. The default filename is `slurm-$(SLURM_JOB_ID).out`, e.g. `slurm-12345.out`, in the directory from which the job was submitted
- `-e` `path/for/stdioerr`
  
  Send stderr to `path/for/stdioerr`.
Options to set the job environment:

- --export=VAR1,VAR2="some value",VAR3
  Pass variables to the job, either with a specific value (the VAR= form) or from the submitting environment (without "=")
- --get-user-env=[=timeout][mode]
  Run something like su -c /usr/bin/env and parse the output. Default timeout is 8 seconds.

The mode value can be "S", or "L" in which case "su" is executed with "-" option

Options to request compute resources:

- -t, --time=time
  Set a limit on the total run time. Acceptable formats include "minutes", "minutes:seconds", "hours:minutes:seconds", "days-hours", "days-hours:minutes" and "days-hours:minutes:seconds"
- --mem=MB
  Maximum memory per node the job will need in MegaBytes
- --mem-per-cpu=MB
  Memory required per allocated CPU in MegaBytes
- -N, --nodes=num
  Number of nodes are required. Default is 1 node
- -n, --ntasks=num
  Maximum number tasks will be launched. Default is one task per node
- --ntasks-per-node=ntasks
  Request that ntasks be invoked on each node
- --cpus-per-task=ncpus
  Require ncpus number of CPU cores per task. Without this option, allocate one core per task

Requesting the resources you need, as accurately as possible, allows your job to be started at the earliest opportunity as well as helping the system to schedule work efficiently to everyone's benefit.

Options for running interactively on the compute nodes with srun:

- -n num
  Specify the number of tasks to run, e.g. -n4. Default is one CPU core per task. Don’t just submit the job, but also wait for it to start and connect stdout, stderr and stdin to the current terminal
- -t time
  Request job running duration, e.g. -t1:30:00
- --mem=MB
  Specify the real memory required per node in MegaBytes, e.g. --mem=4000
- --pty
  Execute the first task in pseudo terminal mode, e.g. --pty /bin/bash, to start a bash command shell
- --x11
  Enable X forwarding, so programs using a GUI can be used during the session (provided you have X forwarding to your workstation set up)

To leave an interactive batch session, type exit at the command prompt.

Options for delaying starting a job:

- -d, --dependency=dependency_list
  For example, --dependency=afterok:12345, to delay starting this job until the job 12345 has completed successfully.
- --begin=time
  Delay starting this job until after the specified date and time, e.g. --begin=9:42:00, to start the job at 9:42 am.

Options for running many similar jobs:

- -a, --array=indexes
  Submit an array of jobs with array ids as specified. Array ids can be specified as a numerical range, a comma-separated list of numbers, or as some combination of the two. Each job instance will have an environment variable SLURM_ARRAY_JOB_ID and SLURM_ARRAY_TASK_ID. For example:
  --array=1-11, to start an array job with index from 1 to 11
  --array=1-7:2, to submit an array job with index step size 2
  --array=1-9:4, to submit an array job with simultaneously running job elements set to 4

The srun command is similar to pbsdsh. It launches tasks on allocated resources.

---

---mail-user=\my_email_address@nyu.edu

Send email to \my_email_address@nyu.edu when certain events occur.

---mail-type=type

Valid type values are NONE, BEGIN, END, FAIL, REQUEUE, ALL...

Options to set the job environment:

- --export=VAR1,VAR2="some value",VAR3
  Pass variables to the job, either with a specific value (the VAR= form) or from the submitting environment (without "=")
- --get-user-env=[=timeout][mode]
  Run something like su -c /usr/bin/env and parse the output. Default timeout is 8 seconds.

The mode value can be "S", or "L" in which case "su" is executed with "-" option

Options to request compute resources:

- -t, --time=time
  Set a limit on the total run time. Acceptable formats include "minutes", "minutes:seconds", "hours:minutes:seconds", "days-hours", "days-hours:minutes" and "days-hours:minutes:seconds"
- --mem=MB
  Maximum memory per node the job will need in MegaBytes
- --mem-per-cpu=MB
  Memory required per allocated CPU in MegaBytes
- -N, --nodes=num
  Number of nodes are required. Default is 1 node
- -n, --ntasks=num
  Maximum number tasks will be launched. Default is one task per node
- --ntasks-per-node=ntasks
  Request that ntasks be invoked on each node
- --cpus-per-task=ncpus
  Require ncpus number of CPU cores per task. Without this option, allocate one core per task

Requesting the resources you need, as accurately as possible, allows your job to be started at the earliest opportunity as well as helping the system to schedule work efficiently to everyone's benefit.

Options for running interactively on the compute nodes with srun:

- -n num
  Specify the number of tasks to run, e.g. -n4. Default is one CPU core per task. Don’t just submit the job, but also wait for it to start and connect stdout, stderr and stdin to the current terminal
- -t time
  Request job running duration, e.g. -t1:30:00
- --mem=MB
  Specify the real memory required per node in MegaBytes, e.g. --mem=4000
- --pty
  Execute the first task in pseudo terminal mode, e.g. --pty /bin/bash, to start a bash command shell
- --x11
  Enable X forwarding, so programs using a GUI can be used during the session (provided you have X forwarding to your workstation set up)

To leave an interactive batch session, type exit at the command prompt.

Options for delaying starting a job:

- -d, --dependency=dependency_list
  For example, --dependency=afterok:12345, to delay starting this job until the job 12345 has completed successfully.
- --begin=time
  Delay starting this job until after the specified date and time, e.g. --begin=9:42:00, to start the job at 9:42 am.

Options for running many similar jobs:

- -a, --array=indexes
  Submit an array of jobs with array ids as specified. Array ids can be specified as a numerical range, a comma-separated list of numbers, or as some combination of the two. Each job instance will have an environment variable SLURM_ARRAY_JOB_ID and SLURM_ARRAY_TASK_ID. For example:
  --array=1-11, to start an array job with index from 1 to 11
  --array=1-7:2, to submit an array job with index step size 2
  --array=1-9:4, to submit an array job with simultaneously running job elements set to 4

The srun command is similar to pbsdsh. It launches tasks on allocated resources.
We are developing a set of tutorials to help NYU HPC users make the most of the facilities. Tutorials are suitable for self-directed learning and are also periodically run as classes in the library. NYU Data Services also provides tutorials for a range of scientific software - for dates and times of upcoming HPC classes check our calendar, or see NYU Data Services for a wider schedule of classes.

If you want to schedule an Information session apart from the regular HPC training offerings, please fill out the form

Currently available HPC tutorials are:

**Tutorial 0: Introduction to Unix/Linux**

**Tutorial 1: A Hands-On introduction to Unix/Linux**

**Tutorial 2: Getting Started in the NYU HPC environment**

**The NYU HPC sbatch tutorial is also available, covering:**

- Declare the date/time a job becomes eligible for execution
- Defining the working directory path to be used for the job
- Manipulate the output files
- Mail job status at the start and end of a job
- Submit a job to a specific queue
- Submitting a job that is dependent on the output of another
- Submitting multiple jobs in a loop that depend on output of another job
- Opening an interactive shell to the compute node
- Passing an environment variable to your job
- Passing your environment to your job
- Submitting an array job: Managing groups of jobs

**Getting Started on Dumbo: How to login**

**Tutorial 1: MapReduce**

**Tutorial 2: Hive**

**Tutorial 3: Spark**

FAQ
Something went wrong!

Why does running "ls" on /scratch take so long?

I can't login

When trying to login, I get warnings about "HOST IDENTIFICATION HAS CHANGED"

What happened to my data on /scratch?

In the library, my wireless connection keeps dropping out. How can I fix it?

I'm getting a "module: command not found" error

Warning: no access to tty (Bad file descriptor), Thus no job control in this shell

I get an error "Warning: no display specified." when I use -X flag with ssh

Who killed my job, and why?

I got an email "Please do not run jobs on login nodes"

Running jobs

What resources can and should I request?

Can I make sure a job gets executed only after another one completes?

How do I log in to a specific node?

How can I make sure my job is running smoothly?

My job will take longer than 48 hours, what should I do?

My job needs (MySQL, some other service) to be running
I want to run a job at 9am every day

Using software

How do I run ... (esp, needs a license)

a STATA job?

a Gaussian job?

a Matlab job?

a parallel, non MPI job (eg Julia)?

I can't find (some software package)

Can you install (some software package)?

How can I view a PDF file on Prince?

Managing data

How much of my file/space quota have I used?

How do I give my colleague access to my files?

How do I get the best transfer speed to or from BuTinah?

I have a huge amount of data that I want to compress for storage or transfer

Monthly Maintenance Window

To provide the best possible service, ITS must regularly update and perform routine maintenance on its systems and networks. Some of these activities require that the affected systems and networks be shut down. While this work is essential, we also recognize that it presents an inconvenience. To enable those who use these systems to better plan for maintenance, we have guidelines for scheduling routine maintenance and upgrades to the HPC clusters as described below.

A MONTHLY SCHEDULED MAINTENANCE OF UP TO 12 HOURS WILL BE TAKEN, IF NEEDED, BEGINNING AT 8AM ON THE FIRST MONDAY OF EACH MONTH

Major scheduled maintenance and upgrade activities will take place, if needed, once per month. These will be scheduled for the first Monday of each month at 8am to noon to start these scheduled maintenance and upgrade activities. The maintenance period may often be brief or not used at all, but can last up to 12 hours if this amount of time is needed to complete the work.

We have chosen early morning on the first Monday of each month for our maintenance work as it has been the time period during the week which has low usage on our clusters.
A notification will be sent to all HPC account holders announcing any scheduled maintenance work in advance.

**A WEEKLY SCHEDULED MAINTENANCE OF UP TO FOUR HOURS (MONDAY 8 AM to NOON) MAY BE USED TO ADDRESS SMALLER MAINTENANCE AND UPGRADE NEEDS.**

This time will not be used if not needed.

**Featured Research**

- The *Brooklyn* Research Cluster provides flexible computing, including for NYU Tandon’s Visualization, Imaging, and Data Analytics research center
- An app that translates sign language into spoken English using Brooklyn Research Cluster Platform
- An Event-Driven Model for Estimation of Phase-Amplitude Coupling at Time Scales of Cognitive Phenomena
- Drug design for treatment of heart attack and stroke - How the hSCAN-1 enzyme is activated
- How carcinogenic chemicals slip past DNA repair mechanisms
- Seeing below the resolution of MRI
- The link between Atlantic Ocean warming and Antarctic climate change