Introduction to Unity

Arts and Sciences Technology Services

Version 1.0

Overview

The Unity cluster is a high-performance computing (HPC) environment maintained by Arts and Sciences Technology Services (ASCTech). Unity mirrors the environment at the Ohio Supercomputer Center (OSC), and provides researchers with convenient computational resources while relieving them of the burdens of administering servers and storage, maintaining an appropriate physical environment, and complying with many security requirements.

The cluster consists of many individual computers called nodes that handle specific tasks. All of the nodes run RedHat Enterprise Linux 7. Research groups can buy a compute node and have exclusive use of it or allow shared use by all Unity users (or alternate those models—exclusive use when needed by the group, shared when not in use by the group). ASCTech provides several compute nodes for shared use. See the Unity online documentation for a list of nodes and specifications.

ASCTech maintains a software stack of popular research applications; you can request that we install additional software. See the Module environment section below for how to list and use these applications. In addition, you can install your own software in your home directory.

You get a 100-GB home directory that is available on all the nodes in the cluster. Additional project storage is available on request.

Getting Started

You log in to Unity with your ASC account using ssh (we need to enable your account first—just send a request to asctech@osu.edu).

```
ssh name.#@unity.asc.ohio-state.edu
```

This will log you in to the head (or login) node. You may be able to tell that from the operating system prompt that you see—by default, your prompt is your name.# followed

by the name of the node that you're on. So on the head node, your prompt will be like `name.#@unity-1`, where `unity-1` is the name of the head node.

If you are not on an ASC network (for example, from OSU Wireless or from home), you'll need to connect to the ASC VPN first. If you're using a computer on which you cannot install the VPN client, such as from an off-site lab, or if you're concerned about moving large data sets through the VPN, you can use ASCTech's jump host.

Basic Use

Module environment

In order to facilitate a variety of users who may want different versions of applications, Unity (like OSC) uses the Lmod Environment Modules package. This provides a common stack of requested software, but it requires that you explicitly load many applications before you can use them.

To see the modules that are available on Unity, at a command prompt type

```
module avail
```

This reports the applications (and their versions) that are installed in the module system. For example, `module avail` reports two versions of python, `python/2.7` and `python/3.5`, and `python/3.5` is followed by `(D)`. The `(D)` indicates that 3.5 is the default version of python; you can load python 3.5 for use by typing

```
module load python
```

If you really want python 2.7, enter

```
module load python/2.7
```

You can use `module load` either at the command line (on the head node or in interactive mode—see below) or in a batch script.

Head node

As noted above, when you initially log in, you're on the head node. You'll be able to interact with Unity through the Linux command line.

The head node is a reasonable place to copy data to or from the cluster using `scp` or `sftp`

or to download data from public repositories. You can also edit text files and compile code on the head node.

However, the head node is not where you want to run computations. The head node is limited in its capabilities; any computation run on this node could impact other customers. Instead, you'll want to use either interactive mode or submit a job using the scheduler.

Interactive mode

You can get an interactive session on a compute node by typing (on the head node)

```
qsub -I
```

The default prompt will change to your name.# at the name of the compute node (see the Unity online documentation for a list of nodes and specifications).

By default in interactive mode you get one hour with one core on one node with 3 GB of memory. You can request more cores and more RAM by including limits in the command (note the placement of the colon and the comma):

```
qsub -I -l nodes=1:ppn=4,mem=16GB
```

Type `exit` to leave the interactive session.

By default, interactive sessions are limited to 60 minutes, but you can ask for a longer interactive job by defining additional resources.

Batch mode

In typical use, you submit jobs to Unity using the `qsub` command. While you can do everything as options to `qsub`, it's much more convenient to write scripts using a text editor and submit them with `qsub`. You then submit the script by typing (from the command prompt on the head node) `qsub myscript.pbs` where `myscript.pbs` is the name of your script.

A simple script file might look like this:

```
#!/usr/bin/bash
#
# PBS file to convert SAM file to BAM file.
# Invoke like
#   qsub -v DATAFILE=datafile_value samtools-sam-to-bam.pbs
#
# I expect datafile_value to be the full path of a SAM file
# excluding the .sam extension

#PBS -l walltime=1:00:00
#PBS -l nodes=1:ppn=4,mem=16GB
#PBS -m abe
#PBS -M shew.1@osu.edu

module load samtools/1.5
samtools view -b -S -@ 3 -o ${DATAFILE}.bam ${DATAFILE}.sam
```

The lines that begin with `#PBS` are directives that set options in the scheduling system. For example, the first line that begins `#PBS -l` sets a walltime limit of one hour. If your job is not complete in one hour it will abort, so it's important to make a generous estimate how long you expect your job to take, but just guessing the maximum wall time (which is 336 hours, or 14 days) plays havoc with the scheduler. The second line that begins `#PBS -l` sets limits of 1 node, 4 cores on that node, and 16 GB of RAM. The line `#PBS -m abe` tells the scheduler to send email when the job aborts (`a`), begins (`b`) or ends (`e`), and the line that begins `#PBS -M` tells the schedule where to send those emails (put your own email address in your script). Other lines that begin with `#` are comments.

Next comes the executable part of the script. Here we load version 1.5 of the `samtools` module and run a command, just as we would from the command line.

This script shows the capability of specifying variables on the command line that are passed to the script. This is useful for running the same operations on different files, for example, which can itself be scripted by writing a Python or other script to loop over files and repeatedly call `qsub`.

The job enters Unity's queue and runs when resources become available. You'll get an email when the job begins and an email when it either aborts with an error or completes successfully.

You can get more immediate information about the progress of your job by using the `showq` or `qstat` command. OSC has a page describing tools for monitoring your job.

More Information

Because Unity is modeled after OSC, much of OSC's excellent documentation is applicable to Unity.

We have limited documentation at go.osu.edu/unityhelp. We're concentrating on documenting differences with OSC, but if there's something that you think should be documented, please let us know.

OSC has a PBS directives summary. Adaptive Computing, the distributor of the Torque and Moab resource management packages used to schedule jobs, has an overview of commands.

Adam Lauretig, a Ph.D. student in Political Science, has published an introduction to R on Unity on github.

Victor Eijkhout at the Texas Advanced Computing Center has written a Creative Commons-licensed (CC BY) introduction to HPC.

You can see long-term usage statistics for Unity.

To request access to Unity or for help with a specific issue, submit a ticket or send email to asctech@osu.edu and include the word "Unity" in the subject.