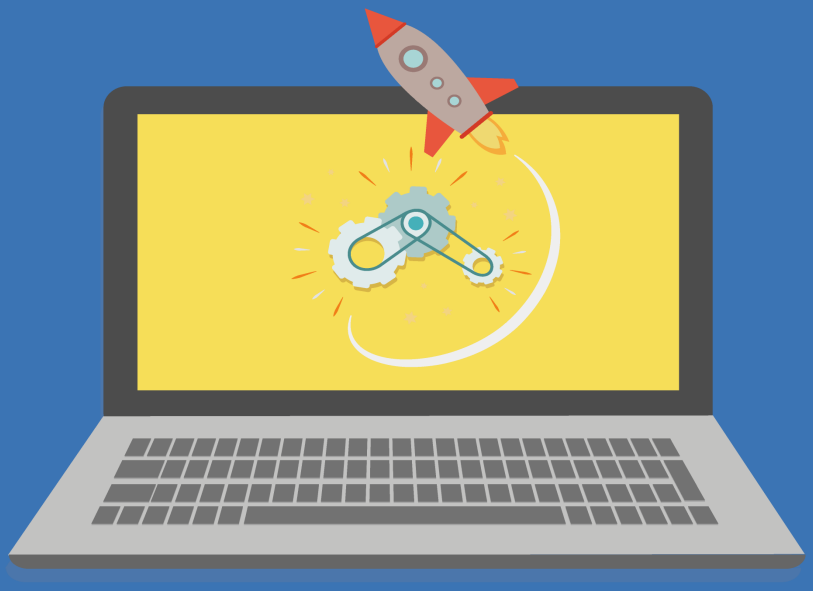
Scientific Computing

LEARN HIGH PERFORMANCE COMPUTING

Supercomputing

Job Scheduling | Parallel Programming | GPUs



Scientific Programmer

Supercomputing

Learn to Use HPC Systems

Scientific Programmer

Contents

Supercomputing

Supercomputers play an important role in today's research world. They aid us to solve compute-intensive problems such as physical simulation, climate research, molecular modeling and so on. Before we get into how to operate on a supercomputer, let's revisit its history a bit.

History of Supercomputing

The history of supercomputing started in 1960s! with the `Atlas`at the **University of Manchester** and a series of computers at **Control Data Corporation (CDC)**, originally designed by *Seymour Cray*. The `Atlas` was a joint venture between **Ferranti** (a UK electrical engineering and equipment firm) and the **Manchester University** and was designed to operate at processing speeds approaching one microsecond per instruction, about one million instructions per second. The first Atlas was officially commissioned 1962 as one of the world's first supercomputers – considered to be the most powerful computer in the world at that time by a considerable margin. However, Cray left CDC in 1972 to form his own company, **Cray Research**. Four years after leaving CDC, Cray delivered the 80 MHz `Cray 1` in 1976, and it became one of the most successful supercomputers in history.

The `Cray-2` released in 1985 was an 8 processor liquid cooled computer and Fluorinert was pumped through it as it operated. It performed at `1.9 gigaflops` and was the world's second fastest after `M-13` supercomputer in Moscow.

While the supercomputers of the 1980s used only a few processors, in the 1990s, machines with thousands of processors began to appear in **Japan and the United States**, setting new computational performance records. For example, the Intel's `Paragon` had 1,000-4,000 Intel i860 processors in various configurations, and was ranked the **fastest in the world** in 1993. The Paragon was a MIMD machine which connected processors via a high speed two dimensional mesh, allowing processes to execute on separate nodes, communicating via the **Message Passing Interface** or `MPI` (details to be discussed later). If you want to know more about the supercomputing and its glorious past feel free to visit this website[1] in relation to history of computers.

HPC cluster computers

The term **cluster computing** is used to denote nothing but two or more computers that are networked together to provide solutions as required. However, this idea should **not be confused** with a more general **client-server** model of computing as the idea behind clusters is quite unique.

A cluster of computers **joins** computational powers of the **compute nodes** to provide a more **combined computational power**. Therefore, as in the client-server model, rather than a simple

[1] http://history-computer.com/ModernComputer/Electronic/Cray.html

client making requests of one or more servers, cluster computing utilize multiple machines to provide a more powerful computing environment perhaps through a **single operating system**.

In its simplest structure, HPC clusters are intended to utilize **parallel computing** to apply more processor force for the arrangement (solution) of a problem. HPC clusters typically have a large number of computers (often called '**nodes**') and, in general, most of these nodes would be configured **identically**. Though from the out side the cluster may **look like a single system**, the internal workings to make this happen can be quite complex.

Computer clusters emerged as a result of convergence of a number of computing trends including the availability of low-cost microprocessors, high speed networks, and software for **high-performance distributed computing**. They have a wide range of applicability and deployment, ranging from small business clusters with a handful of nodes to some of the fastest supercomputers in the world!

Supercomputers vs. HPC clusters

According to Jan Christian Meyer[2]'s Quora response: "**Supercomputer** isn't a name for one particular type of computer, it's a term that refers to computers used to solve problems which require processing capabilities nearly as big as anyone can build at a given time. The types of systems that people call supercomputers change over time, supercomputers of yesteryear aren't that super any more.

Cluster computers are loosely coupled parallel computers where the computing nodes have individual memory and instances of the operating system, but typically share a file system, and use an explicitly programmed high-speed network for communication. The term loosely refers to the technicalities of how such machines are constructed."

> "clusters" and "high-performance computing" are synonymous.

All the ideas that went into the design of supercomputers are now part of your everyday personal computer. So there is no real distinction anymore between personal computers and super computers: a supercomputer is just a cluster with a large number of ordinary processors. So the super prefix comes from the days when high performance computing was only done on very special machines called supercomputers. Those days are largely gone. Only NEC still makes processors that are descendants of the old supers, but even they put them in a cluster. Now-a-days, every "super" computer these days is a cluster.

Benefits of using cluster computing

HPC systems derive their computational power by exploiting parallelism. Programs for HPC systems must be **splitted up** into many smaller **sub-programs** which can be executed in parallel on different processors. HPC systems can offer parallelism at a much larger scale, with 100's or 1000's, or even

[2] https://www.quora.com/What-is-the-difference-between-a-supercomputer-and-a-computer-cluster

millions of tasks running concurrently.However, writing parallel software can be challenging, and many existing software packages do not already support parallelism & may require development.

Therefore, HPCs are userful when you have:

- A program that can be recompiled or reconfigured to use optimized **numerical libraries that are available on HPC systems** but not on your own system;
- You have a **parallel** problem, e.g. you have a single application that needs to be rerun many times with different parameters;
- You have an **application that has already been designed with parallelism**;
- To make use of the **large memory** available;
- Your problem solutions require **backups** for future use. HPC facilities are reliable and regularly backed up.

When not to use HPC systems?

- You have a **single threaded job** which will only run one job at a time (typical of MatLab users);
- You rely on **DBMS/ databases**;
- You have a lot of **data to transfer** between your local machine and the HPC on a continuous basis (e.g. per job);
- You need to have a **GUI** to interact with your program.

Components of a HPC cluster

A cluster is two or more (often many more) computers working as a single logical system to provide services. Though from the outside the cluster may look like a single system, the internal workings to make this happen can be quite complex.

The figure above presents the logical functions that a physical node in a cluster can provide. Remember, these are logical functions; in some cases, multiple logical functions may reside on the same physical node, and in other cases, a logical function may be spread across multiple physical nodes.

Login node(s)

Individual nodes of a cluster are often on a private network that cannot be accessed directly from the outside or corporate network. Even if they are accessible, most cluster nodes would not necessarily be configured to provide an optimal user interface. The Login node is the one type of node that is configured to provide that interface for users (possibly on outside networks) who may gain access to the cluster to request that a job be run, or to access the results of a previously run job.

Master node(s)

Clusters are complex environments, and the management of the individual components is very important. The management node (generally named as the: `master` or `head` node) provides many important capabilities, including:

- **Monitoring** the status of individual nodes (typically compute nodes).
- Issuing management commands to individual nodes to correct problems or to provide commands to perform management functions, such as **power on/off**.
- In most clusters, the compute nodes (and other nodes) may need to be reconfigured and/or reinstalled (**provisioned**) with a new OS image relatively often. The master node provides the images and the mechanism for easily and quickly installing or reinstalling software on the cluster nodes.
- The master node can also provide **networking services** that help the other nodes in the cluster work together to obtain the desired result. Control nodes can provide two sets of functions: Dynamic Host Configuration Protocol (`DHCP`), Domain Name System (`DNS`), and other similar functions for the cluster. These functions enable the nodes to easily be added to the cluster and to ensure they can communicate with the other nodes.
- **Scheduling tasks** (to be discussed in detail soon!) can be another important role for a master node. For instance, if a compute node finishes one task and is available to do additional work, the control node may assign that node the next task requiring work. However, sysadmins may decide to allocate a separate node (Scheduler node) for this purpse to reduce loads on the master node.

To create high availability (HA) , the master node can have a backup master node. Also, in some systems, master node's tasks may be distributed over various nodes such as Job-scheduler node, log-server nodes, monitoring node, etc.

A free cluster management software stack like **OSCAR**[3] can fulfill a lot of the above needs, but there are other commerical software tools, such as **Bright Cluster Manager**[4] from Bright Computing or **HPC Cluster Manager**[5] from Microsoft that can help sysadmins to deploy and mange a HPC system. Further details on the HPC management is out-of-the-scope for this particular couse.

Compute node(s)

A compute node is where the real computing is performed. The majority of the nodes in a cluster are typically compute nodes. In order to provide an overall solution, a compute node can execute one or more tasks, based on the scheduling system.

[3] http://svn.oscar.openclustergroup.org/trac/oscar
[4] http://www.brightcomputing.com/product-offerings/bright-cluster-manager-for-hpc
[5] https://technet.microsoft.com/library/jj899572.aspx

Storage node(s)

Compute nodes must have fast, reliable, and simultaneous access to the storage system. This can be accomplished in a variety of ways depending on the specific requirements of the application. Storage devices may be directly attached to the nodes or connected only to a centralized node (storage node) that is responsible for hosting the storage requests through **Networked file system** (`NFS`) mounts.

The use of a clustered file system is essential in modern computer clusters.Examples include the IBM's **General Parallel File System** (`GPFS`), Microsoft's **Cluster Shared Volumes** or the Oracle **Cluster File System**. The storage node in turn can be connnected to tape libararies for further backups.

Other components

Aside from the cluster nodes (management node, compute nodes, and storage nodes) that make up a cluster, there are several other key components that must also be considered. The following sub-sections discuss some of these components.

- Ethernet switches Ethernet switches are included to provide the necessary node-to-node (1/10 GB) communication.
- Infiniband switch: For faster networks (56/100 GB), mainly used by MPI enabled software.

Keyboard, video, mouse, etc. connected to a terminal server or master node. Mainly for systems maintenance purposes.

Access to HPC

To access the HPC systems at your institution, you generally need an account which your HPC authority would create for you.

Once you have obtained an account, the only method available to connect to the HPC is to use a secure shell (`ssh`) client, no matter what operating system you are using. However, note that HPC facilities around the work is generally Linux-based because that makes the building cost cheaper!

Accessing HPC systems on campus

Linux: To connect to the HPC from a Linux based computer, open a terminal window and simply type `ssh hpc-login-node-name` and then press enter.

```
1 ssh username@login-hostname
```

Windows: To connect to the HPC from a computer using Microsoft Windows requires the use of a `SSH` client, e.g. **PuTTY** or **MobaXterm**.

PuTTY is a free implementation of Telnet and SSH for Windows platforms. It also implements an xterm terminal emulator. However, while using PuTTY, if you need to forward the GUI interfaces from the HPC through the `X11` forwarding, you will also need a software tool called **XMing**. Xming is an X Window Server for Microsoft Windows. XMing is built from the `X.Org` canonical X-server software with modifications so that it can run under Windows. XMing can be used to remotely run Linux programs and display their X-Window output under Windows.

I personally prefer the MobaXterm, as it can provide enhanced terminal for Windows with `X11` server, tabbed `SSH` client, network tools and much more!

Passwordless HPC Login

Password-less login: SSH can provide authentication and secure communications over insecure channels. To enable passwordless logins, you have to generate a pair of keys, one public and the other private. The public key authentication practiced on the HPC platforms assumes that the public key is known by the system in order to operate an authentiation based on a challenge/response protocol instead of the classical password-based protocol.

To generate an SSH keys, just use the `ssh-keygen` command, typically as follows:

```
ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
de:e2:21:df:38:49:3a:99:d7:85:4e:c3:85:c8:24:5b
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|      . E        |
|       * . .     |
|      . o . .    |
|        S. o     |
|       .. = .    |
|       =.= o     |
|      * ==o      |
|       B=.o      |
+-----------------+
```

After the execution of ssh-keygen command, the keys are generated and stored in the following files:

- `SSH RSA` Private key: `~/.ssh/id_rsa`. Do not transmit this file
- `SSH RSA` Public key: `~/.ssh/id_rsa.pub`. This file is the file safe to distribute.

Next you should have sent through the account request form by mail and the public key (i.e. `id_rsa.pub`) to your HPC authority, enabling them to configure the `~/.ssh/authorized_keys` file of your account.

Finally, to be able to use your SSH key in a public-key authentication scheme, it must be loaded by an SSH agent. Just execute the following:

```
ssh-add ~/.ssh/id_rsa
```

Acessing HPC systems off your institution (campus)

Access to HPC systems at your institution from off-campus should be available for staffs and research students via some kinds of VPN services. If you do not know what is this - VPN stands for virtual private network and is a type of network connection that allows users to access computer networks from anywhere in the world. It is also called IP tunneling, is a secure method of accessing your institution's computing resources such as HPC systems.

There are various VPN clients that can you use to tunnel through a VPN. For example, the **Cisco AnyConnect Secure Mobility** client is a web-based VPN client that does not require user configuration. You generally download the client by pointing your browser to vpn.yourinstitution.edu.

Alternatively, in a Linux environment, you can use free VPN clients like **OpenVPN**. On the `Ubuntu` or `Debian` systems, you can install it by typing:

```
sudo apt-get update
sudo apt-get install openvpn
```

Finally on the Mac OSX systems you can use tools like **Tunnelblick** to create the IP tunnels.

HPC Data Transfer

On HPC systems, we can interact with the command line over SSH. We can also create and move file and directories. We can run a program and get some output. But the question is how can we

- Upload files, say our datasets, to the HPC machine?
- Download files, say the results of our analysis, from the HPC machine for further analysis locally?

There are two popular methods to do these:

- FTP = File Transfer Protocol; and
- SCP = Secure Copy

FTP file transfer

You need an FTP client, such as FileZilla. You tell the client which server to connect to e.g,: hpc.yourinstitution.edu and port: 22. You then authenticate with your username and password. Note that All FTP communication to the HPC machine will be 100% secure.

Secure Copy (`scp` command)

The secure Copy (`scp`) command is another way to transfer files to and from the HPC machine. Like FTP, it is secure, but you will need to invoke it from the command line. Therefore SCP is non-interactive, but the good news is- it can easily be scripted.

```
scp <file_name.ext> <user_name>@hpc.yourinstitution.edu:<dest_dir>
```

Get files from the Internet

To do this, we need to be at the command line (i.e. logged into the HPC machine by `SSH`) and type: `wget <URL_to_file>`. This will download the file (e.g., a dataset) to your current working directory.

Dealing with archived files

In order to transfer data and save space, you will often find .zip files every where. This signifies that the file actually contains one or more files bundled for convenience and then compressed. It's called an archive. Common archive formats have extensions `tar`, `tgz`, `gz`, `bz2`. You can easily unzip these files as follows:

```
unzip <file_name>.zip
tar xvf <file_name>.tar
tar xzvf <file_name>.tgz
gunzip <file_name>.gz
bunzip2 <file_name>.bz2
```

Notice that the file ends with an extension in the left, you can use the corresponding command (`unzip` for `zip`, etc.) in the right!

HPC software list

Your institute may have a different list of software, but following are commonly available software that should be readily available on any HPC systems and may loaded through software `modules` (see the next section). The commercial software are noted with a ©:

Programming Language Compilers

- GNU compilers (`gnu-c`, `gnu-cpp` `ForTran`, etc.)
- JDK (`java`)
- Intel compilers (`c++`, `ForTran`, etc.) ©
- SolarisStudio

Scripting

- Guile
- Perl
- Python
- Tcl/Tk
- Bash
- Zsh

File Formats and Data Management

- HDF
- netCDF

Astronomy and Astrophysics

- IDL ©
- Tecplot ©
- DS9
- IRAF
- Figaro
- Rebound

CFD & Engineering and Modelling

- COMSOL ©
- ANSYS Fluent software ©
- APSIM
- Cantera
- Converge CFG
- Eilmer

Climate Modelling

- GMT
- Opengrads

Mathematics and Statistics

- Matlab ©
- R and RStudio
- Scilab
- Numpy
- Scipy

Graphics

- Ferret
- Gnuplot
- Paraview
- Atlas
- NCL
- Wine
- NetworkX
- Gephi
- yED

Editors

- Vim
- EMACS
- Atom

Parallel Programming Libraries/Tools

- Intel MPI ©
- Open MPI
- MPICH
- CUDA Toolkit

Schedulers (any one)

- PBS
- PBS Pro
- Slurm

Utils

- SFTP
- SSH
- sZip

Note that a commercially purchased software may be controlled by their license keys that can either limit the number of concurrent users or the toolboxes/modules available for use (e.g., `Matlab toolbox`). Please contact your institutions HPC authority for further information regarding the availability of a specific software of your need.

HPC software modules

On a HPC system, the user `environment` is setup using environment `modules`. By default, a number of modules are automatically loaded to configure the environment to allow running of applications and the submission of jobs to the cluster.

What is an environment `Module`?

On a HPC system, it is necessary to make available a wide choice of software packages in multiple versions, it can be quite difficult to set up the user environment so as to always find the required executables and libraries.

This is particularly true where different implementations or versions use the same names for files. Environment modules provide a way to selectively activate and deactivate modifications to the user environment which allow particular packages and versions to be found.

The basic command to use is module:

```
module
   (no arguments)            // print usage instructions
   avail or av               // list available software modules
   whatis                    // as above with brief descriptions
   load <modulename>         // add a module to your environment
   unload <modulename>       // remove a module
   purge                     // remove all modules
```

Modules work by setting environment variables such as `PATH` and `LD_LIBRARY_PATH`. Therefore if you need to modify variables directly it is essential to retain the original values to avoid breaking loaded modules (and potentially rendering essential software “not found”) - e.g. do the following:

```
export PATH=$PATH:/home/abc123/custom_bin_directory
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/abc123/custom_lib_directory
```

But do not do:

```
export PATH=/home/abc123/custom_bin_directory // overwrite PATH
export LD_LIBRARY_PATH=/home/abc123/custom_lib_directory // overwrite LD_LIBRARY_PATH
```

Some modules refer to **administrative software** and are not of interest to users and also some modules load other modules. It is possible to make use of various versions of **Intel compiler** and parallel libraries by explicitly loading some of the above modules.

By default the login environment loads several modules required for the normal operation of the account: e.g. the default versions of the **Intel compilers** and **Intel Math Kernel Library**, batch scheduling system and the recommended `MPI` for the particular flavour of compute node hardware.

It is possible to change the environment which is loaded when logging in, by editing the shell initialisation file `~/.bashrc` (or `~/.cshrc`, `~/.tcsh` if using csh/tcsh as your shell). If your shell initialisation file is modified it will only effect all future login sessions, and all batch jobs not yet started. Since some modules are required for proper operation of the account, caution is needed before removing any autoloaded modules.

One can list the modules actually loaded by issuing the command `module list`. Which produce the following example output:

```
module list
Currently Loaded Modulefiles:
    1) kvm/3.4.5+6            4) switcher/1.0.13
    2) default-manpath/1.0.1  5) oscar-modules/1.0.5
    3) torque-pbs/2.3.7
```

Using modules in batch files

To use software package xxx in a batch job script it is a simple matter of specifying `module load xxx` See the appropriate packages entry in the software page (and the output of the `module list`) for details of the expected package name for this flag.

Creating your own modules

Apart from the available system environment modules, you can define your own modules and load `module load use.own`. This will set up the `$HOME/privatemodules` directory with an initial module file called null. It will also change your `MODULEPATH` environment variable to ensure that the module command looks for the modules in your home directory. See `man modulefile` for further information on writing your own modules.

HPC Job Schedulers

What is a HPC Job?

In the arena of HPC, we talk a lot about **jobs**, these are simply commands we wish to run and requests for resources (e.g. `compute time`, `disk space`, `memory amount`, `software environments` etc.). HPC jobs are generally time consuming and resource intensive run **non-interactively**. However, they can be run **interactively**, but mainly for testing purposes.You need add your jobs to a queue and when machines have free resources jobs run. Once jobs are complete, you can inspect their output.

The **batch system** is a program (typically resides on the head node) that lets you add and remove jobs from the **queue** and monitor the queue. It's a script/command line driven program.

The **Job schedulers** manages job queues. Typically, the scheduler will schedule jobs from the queue as sufficient resources (more precisely - cluster compute `nodes`) become idle. You do not need to interact with schedulers directly.

Some widely used cluster batch systems are:

- **Portable Batch System (PBS)**
- **Simple Linux Utility for Resource Management (SLURM)**
- Moab
- Univa Grid Engine
- LoadLeveler, Condor
- OpenLava
- IBM's Platform LSF

The first two in the above list are popular. Therefore, in this course we will restrict our discussion among these two.