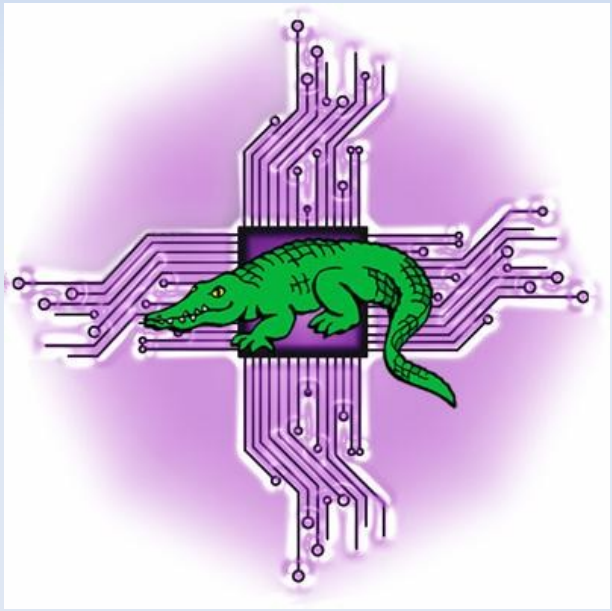
An Introduction to Supercomputers

Presentation to the Linux Users of Victoria Beginners Workshop



May 17, 2013

http://levlafayette.com

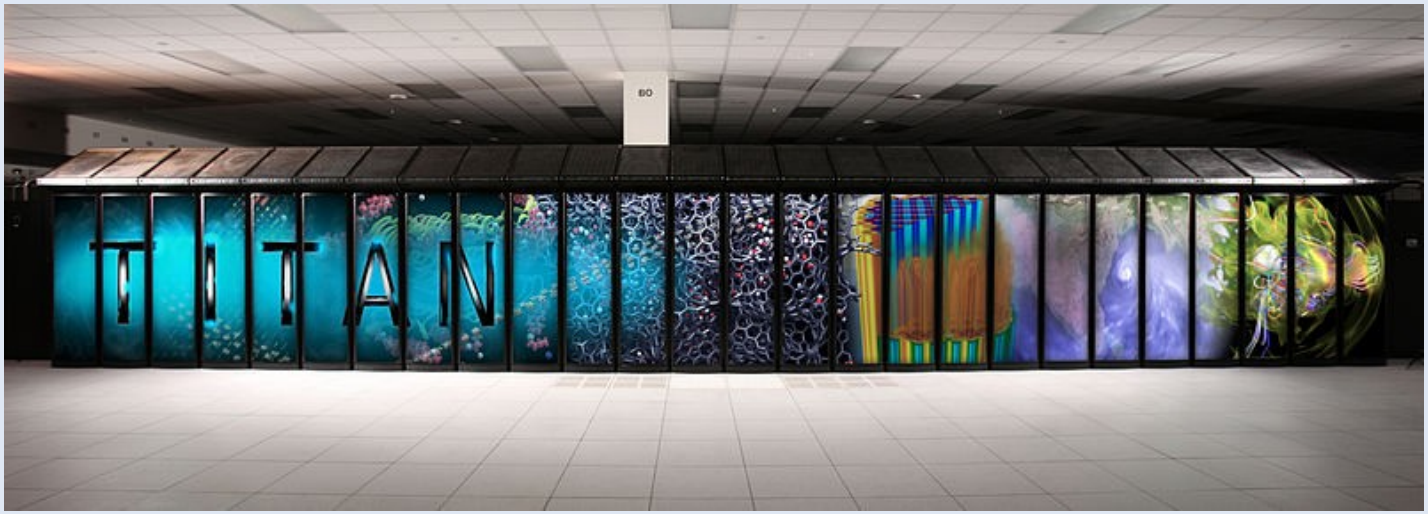
1. A Definition

1.1 What Is A Supercomputer Anyway?

An arbitrary term with no specific definition. In general use it means any single computer system (itself a contested term) that has exceptional processing power for its time. A well-adopted metric is the number of floating-point operations per second (FLOPS) such a system can carry out.

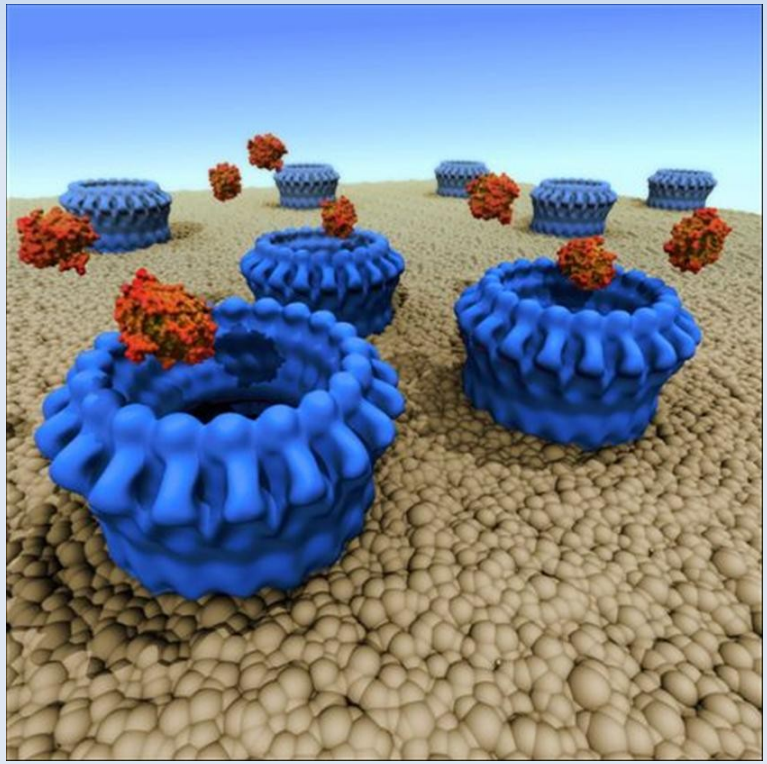
Supercomputers, like any other computing system, have improved significantly over time. The Top500 list has seen as doubling of FLOPS roughly every 14 months. The theortical maxima for a system is cores * clock speed * FLOPS/cycle. The following is some metrics for illustrated purposes from the Top500.

1993: 124.50 GFLOPS
1994: 170.40 GFLOPS
1996: 368.20 GFLOPS
1997: 1.338 TFLOPS
1999: 2.3796 TFLOPS
2000: 7.226 TFLOPS
2004; 70.72 TFLOPS
2005: 280.6 TFLOPS
2007: 478.2 TFLOPS
2008: 1.105 PFLOPS
2009: 1.759 PFLOPS
2010: 2.566 PFLOPS
2011: 10.51 PFLOPS
2012: 17.59 PFLOPS



1. A Definition

The current number #1 system (pictured above) is "Titan", built by Cray at Oak Ridge Nation Laboratory, a hybrid system using graphics processing units (GPUs) and well as CPUs. An upgraded version of a previous number one holder, Jaguar (from Nov 2009 and Jun 2010), the upgrade cost was $60 million USD, primarily from the US Department of Energy.



1.2 Why Supercomputers Are Important

The importance of supercomputing cannot be measured simply by their processing capacity, but rather what is done with that capacity. Essentially however, any hard number-crunching or processor intensive visualisation requires a supercomputer for both speed and effectiveness. Supercomputers are typically (but not exclusively) used for scientific computing. Some applications have included weather forecasting, aerodynamic design, fluid mechanics, radiation modelling, molecullar dynamics, CGI rendering for popular movies.

A local example, researchers from Monash University, the Peter MacCallum Cancer Institute in Melbourne, the Birkbeck College in London, and VPAC in 2010 unravelled the structure the protein perforin to determine how pathogenic cells (c.f., http://www.nature.com/nature/journal/v468/n7322/full/nature09518.html).

2. The Contemporary Supercomputer

2.1 The Teamster Analogy

The contemporary supercomputer is a high performance cluster with a tightly-coupled high-speed interconnect that uses parallel applications. This can be explained as a teamster analogy; if a compute task is a considered to be the equivalent of moving goods from 'a' to 'b' using a horse and cart you can either (a) get a really big horse and cart (a bigger and more powerful single system)., (b) perfect the distribution of the load on the horse and cart (code optimisation)., (c) use a large number of horses and carts and distribute the task between them with a teamster used to manage the convoy.

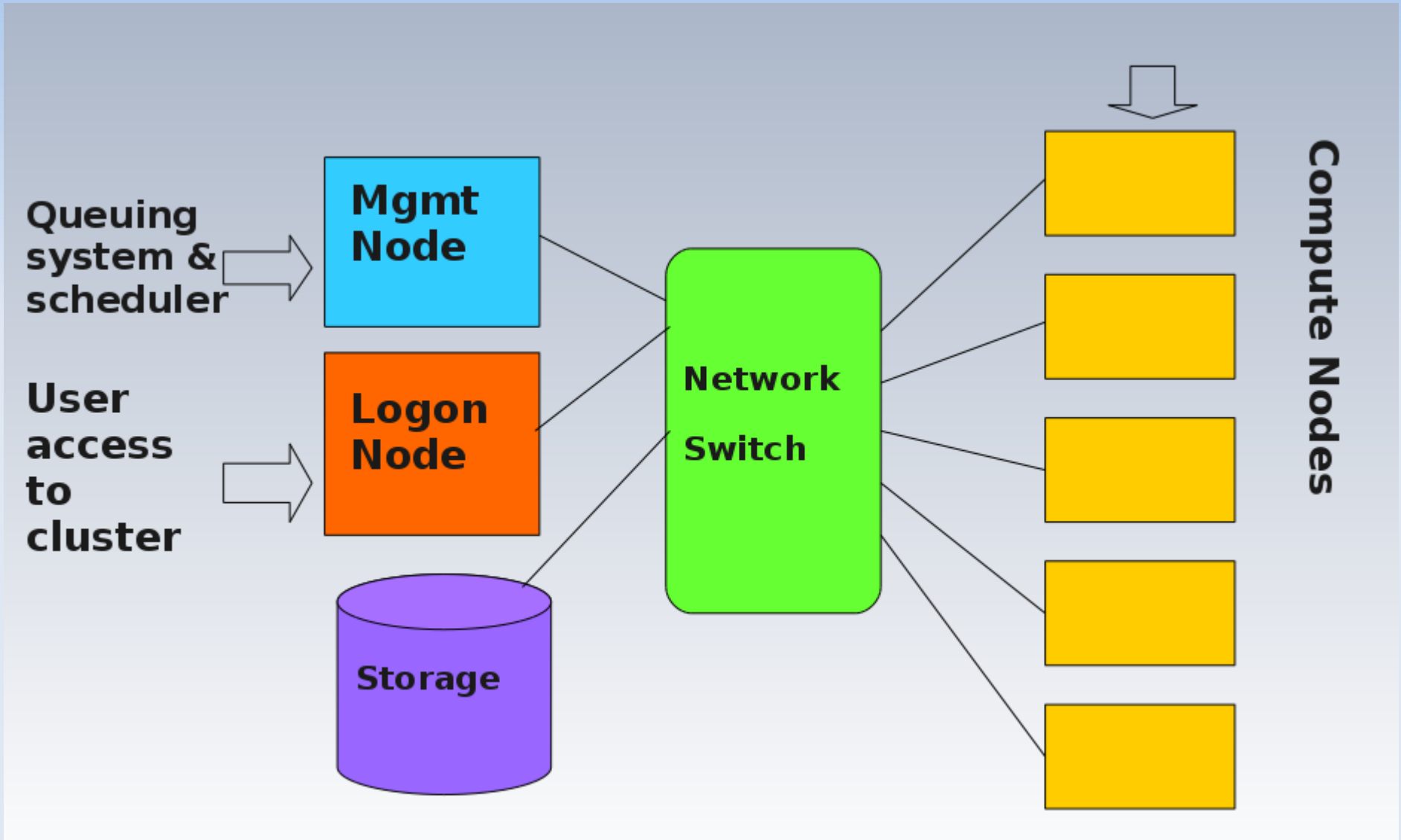
The best supercomputer uses all three. But with (c) it doesn't matter if one set breaks down, as there are others to take their place (high performance); normal horses and carts are readily and cheaply available (low cost).

2.2 Clusters (and their friends)

In the 1960s the big innovation was pipelining, putting data processing in a series, where the output of one is the input of the next. In the 1970s vector processors were common - using a instruction set on a one-dimensional arrays of data (vectors) - single instruction, multiple data. In the 1980s multicore systems with distributed memory and file systems became more common. As multicore systems became increasingly common so did distributed shared memory (physically separate memories can be addressed as one logically shared address space).

A cluster is defined of individual computer nodes designed to operate a single system, whether tightly connected or loosley. Despite being potentially loosley connected it is usually considered different to peer-to-peer or grid computing, which are typically heterogeneous, and geographically dispersed.

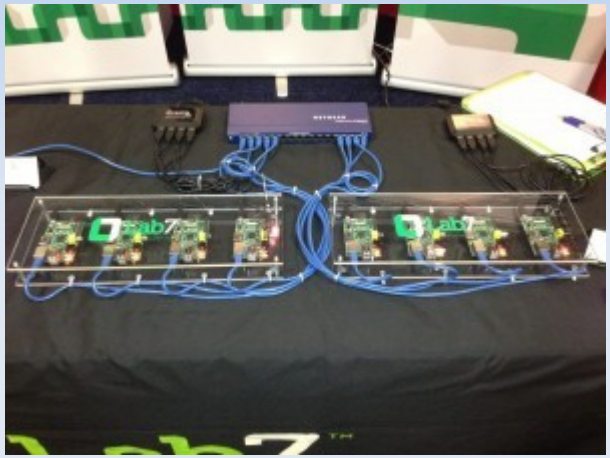
2. The Contemporary Supercomputer



2. The Contemporary Supercomputer

2.2 The Interconnect

The interconnect is the communication chanel between the nodes of the cluster. A cluster can be made from something a simple as Cat-5e Ethernet, but it won't be very fast! The use of switched fabric (rather than Ethernet's hierarchical architecture) is common.



A common form of interconnect is Infiniband (manufactured by Mellanox and Intel), with a theoretical peak of 300 Gbit/s, and a minimum of latency of of less than 1 microsecond for RDMA (remote direct memory access) operations. Others use a torus interconnnect topology. The Tianhe-I used a proprietary interconnect called Arch that runs at 160 Gbit/s, twice the bandwidth of Infiniband (at the time)

2.3 Parallel Applications

Parallel applications break down large problems can often be divided into smaller ones, which are then solved concurrently. A simple example would be to generated a pool of 1000 random die rolls. These could be solved serially, rolling one dice, recording the result, then rolling the next one. Or it could be solved by rolling 1000 dice simultaenously.

Parallel programming is not easy! A parallel programmer may have to worry about the order that results come back, the propsect of bottlenecks and deadlocks in the system, and conflicts in memory.

3. The Contribution of GNU/Linux

3.1 GNU/Linux: The Operating System of Choice

In November 2012 of the Top 500 Supercomputers worldwide, only about 0.6% did not use a "UNIX-like" operating system (a decline from 1.4% on November 2007 and 1.0% in November 2012) and nearly all use an operating system that is entirely "free and open source" (a small percentage use a combination of free and proprietary systems).

3.2 Efficiency of the Command Line

For most users a Graphic User Interface (GUI) is how they interact with a computer system, and there are some advantages with this, not the least being a usually intuitive visual representation for actions. However this takes up significant computer resources. In contrast a command-line interface provides a great deal more power and is very resource efficient. Running supercomputers with a GUI is not a sound policy.

3.3 GNU/Linux Scales, It's Stable and It's Open

GNU/Linux scales and does so with stability and efficiency. Secondly, critical software such as the Message Parsing Interface (MPI) and nearly all scientific programs are designed to work with GNU/Linux. Thirdly, the operating system and many applications are provided as "free and open source", which means that not only are there are some financial savings, were also much better placed to improve, optimize and maintain specific programs.

3.4 Compilation and Optimisation

Free or open source software (not always the same thing) can be can be compiled from source for the specific hardware and operating system configuration, and can be optimised according to compiler flags. There is necessary where every clock cycle is important.

4.0 A Hands-On Experience

4.1 Logging On

Open up two instances of a terminal. One will be a logon to the local machine, the other will be a logon to the cluster.

On one of the terminal windows run:

```
ssh trainXX@trifid.in.vpac.org
```

4.2 Environment Modules

Environment modules provide for the dynamic modification of the user's environment via module files. Each module contains the necessary configuration information for the user's session to operate according according to the modules loaded, such as the location of the application, its manual path, LD_LIBRARY_PATH and so forth.

Run the following commands.

```
module avail
module load vpac
module list
module display gcc/4.7.2
```

4.0 A Hands-On Experience

4.3 PBS Scripts

The Portable Batch System (PBS) is the name of a utility software that performs job scheduling among the available resources. The schedular provides for paramterisation of computer resources, an automatic submission of execution tasks, and a notification system for incidents. There is a variety of PBS applications available (TORQUE, Slurm, etc). We'll use TORQUE today (supported and maintained by Cluster Resources Inc).

A sample PBS script; review its content.

```
cp /common/examples/NAMD_training_example/pbs_example_script .
less pbs_example_script
```

Note the loading of the module, the processors it expects, the walltime (the job ends if this is exceeded), and the mpiexec running namd with a configuration file.

Who is running jobs at the moment?

```
showq | less
```

Other command commands include qsub (submits a job) and qdel (deletes a job).

5.0 Two Example Applications

5.1 Submitting and Reviewing a Job (R)

Go to the home directory on the cluster and copy the R directory to the home directory

```
cd ~
cp -r /common/examples/R/ .
```

Look at the file pbs-script to see what it is doing. The comments should be self-explanatory;

```
less pbs-script
```

Look at the tutorial.R script (less tutorial.R). Firstly, it imports the w1.dat and trees91.csv files into appropriate variables. Then it plots a histogram, breaks, a box plot, normal quantiles, a scatter plot relationship. The output file will also record the correlation of the scatter diagram.

Submit the job

```
qsub pbs-script
```

5.2 Collecting and Viewing Results (Evince)

When it is complete (check with showq -u [username] run and ls on the directory. Apart from the error/output files, there will also be an a PDF file. Copy this to the local system and view it with evince.

```
scp trainXX@trifid.in.vpac.org:R/Rplots.pdf .
evince Rplots.pdf
```

5.0 Two Example Applications

5.3 Submitting and Reviewing a Job (NAMD)

Copy the full NAMD training example to the home directory.

```
cp -r /common/examples/NAMD_training_example  .
```

View the configuration file

```
cd NAMD_training_example
less Ubiquitin_example.conf
```

Review the PBS script

```
less pbs_example_script
```

Submit the job

```
qsub pbs_example_script
```

5.4 Collecting and Viewing Results (VMD)

When it is complete (check with showq -u [username] run and ls on the directory. There will be an error and output file and a lot of other files (structure, .

Copy these to the local machine

```
scp  -r trainXX@trifid.vpac.org:NAMD_training_example .
```

Start up vmd, load the protein structure file, (1ubq_example.psf), then the protein starting position (1ubq_example.pdb), then the trajectory data (1ubq_example_output_01.dcd).

Thank You VPAC!

VPAC
INFINITE INNOVATION