International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

A Simplified Introduction to the Architecture of High Performance Computing

Utkarsh Rastogi

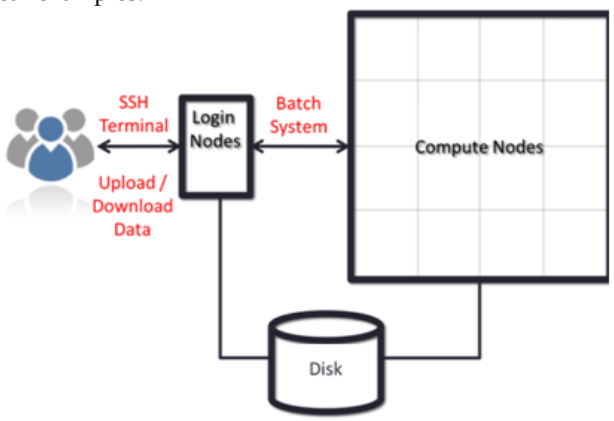
Department of Computer Engineering and Application, GLA University, India

Abstract: ***High-performance computing (HPC) is the ability to process data and perform complex calculations at high speeds. To put it into perspective, a laptop or desktop with a 3 GHz processor can perform around 3 billion calculations per second. While that is much faster than any human can achieve, it pales in comparison to HPC solutions that can perform quadrillions of calculations per second. One of the best-known types of HPC solutions is the supercomputer. A supercomputer contains thousands of compute nodes that work together to complete one or more tasks. This is called parallel processing. It's similar to having thousands of PCs networked together, combining compute power to complete tasks faster.***

1. Introduction

The paper is focused on IST Gravity Group Computer Cluster, Baltasar Sete S´ois and study the impact of a decoupled high performance computing system architecture for data-intensive sciences. Baltasar is part of the "DyBHo" ERC Starting Grant awarded to Prof. Vitor Cardoso, to study and understand dynamical black holes. Baltasar was designed to help solve a specific problem with a Cactus-based implementation. The goal was to have 4GB per processor, a minimum of 200 processors and at least 500MB/s of network storage capability. Even though it was designed to address a specific application, , it exceeded –by far– the group's expectations to a point that it can be considered an all-around cluster.

This paper starts by going through the construction and fundamental computer notions one needs in order to fully understand the parallel computation paradigms. It also cover some of the essential tools necessary for an efficient workflow in the HPC realm. Finally, it tries to demystify the "beast" of shared computer cluster's usage with simple and clear examples.



A simplified diagram of high performance computing

1) Construction

To build a high-performance computing architecture, compute servers are networked together into a cluster. Software programs and algorithms are run simultaneously on the servers in the cluster. The cluster is networked to the data storage to capture the output. Together, these components operate seamlessly to complete a diverse set of tasks. HPC systems often include several different types of nodes, which are specialised for different purposes. Head (or front-end or login) nodes are where you login to interact with the HPC system. Compute nodes are where the real computing is done. You generally do not have access to the compute nodes directly - access to these resources is controlled by a scheduler or batch system (more on this later!). Depending on the HPC system, the compute nodes, even individually, might be much more powerful than a typical personal computer. They often have multiple processors (each with many cores), and may have accelerators (such as Graphics Processing Units (GPUs)) and other capabilities less common on personal computers.

To operate at maximum performance, each component must keep pace with the others. For example, the storage component must be able to feed and ingest data to and from the compute servers as quickly as it is processed. Likewise, the networking components must be able to support the high-speed transportation of data between compute servers and the data storage. If one component cannot keep up with the rest, the performance of the entire HPC infrastructure suffers.

2) Cluster

An HPC cluster consists of hundreds or thousands of compute servers that are networked together. Each server is called a node. The nodes in each cluster work in parallel with each other, boosting processing speed to deliver high-performance computing. Deployed on premises, at the edge, or in the cloud. A typical cluster architecture has 3 node types. The entry nodes, the storage nodes and the worker nodes. The entry nodes are the nodes we connect to in order to use the computer cluster. The storage nodes are the nodes that permanently store data. The worker nodes are the nodes that run programs in the cluster environment and have limited to no local storage space. Usually, small clusters have only one entry node while larger clusters can have more than one for availability purposes. On small clusters like Baltasar, the entry node is also one of the storage nodes. In this typical architecture scenario, storage nodes' access is shared across all worker nodes so that programs can read and write data independently of the worker node they are

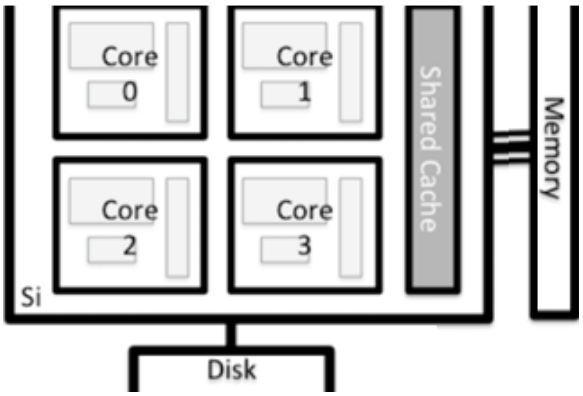
Licensed Under Creative Commons Attribution CC BY

running in. Although typical, we may find different architectures in different clusters.

3) Nodes

Each individual "computer" component of an HPC system is known as a node. Different types of node exist for different tasks. The nodes are connected together by a network usually known as interconnect. Each node on an HPC system is essentially an individual computer. The processor contains multiple compute cores (usually shortened to core); 4 in the diagram below. Each core contains a floating point unit (FPU) which is responsible for actually performing the computations on the data and various fast memory caches which are responsible for holding data that is currently being worked on. The compute power of a processor generally depends on three things:

The speed of the processor (2-3 GHz are common speeds on modern processors), The power of the floating point unit (generally the more modern the processor, the more powerful the FPU is), The number of cores available (12-16 cores are typical on modern processors). Often, HPC nodes have multiple processors (usually 2 processors per node) so the number of cores available on a node is doubled (i.e. 24-26 cores per node, rather than 12-16 cores per node). This configuration can have implications for performance. Each node also has a certain amount of memory available (also referred to as RAM or DRAM) in addition to the processor memory caches. Modern compute nodes typically have in the range 64-256 GB of memory per node. Finally, each node also has access to storage (also called disk or file system) for persistent storage of data. As we shall see later, this storage is often shared across all nodes and there are often multiple different types of storage connected to a node.



A simplified diagram of a node

4) Scheduling

In order to share these large systems among many users, it is common to allocate subsets of the compute nodes to tasks (or jobs), based on requests from users. These jobs may take a long time to complete, so they come and go in time. To manage the sharing of the compute nodes among all of the jobs, HPC systems use a batch system or scheduler. The batch system usually has commands for submitting jobs, inquiring about their status, and modifying them. The HPC center defines the priorities of different jobs for execution on the compute nodes, while ensuring that the compute nodes are not overloaded. Scheduling is an essential component in computer clusters in order to make full use of the available resources while reducing resource concurrency to an optimal level. As an example, if no scheduling is present on a computer cluster, there would occasionally be so many simultaneously running programs that none would have acceptable performance, as well as occasions where the opposite is true: large amounts of free resources and no programs taking advantage of them. For example, a typical HPC workflow could look something like this:

a) Transfer input datasets to the HPC system (via the login nodes)
b) Create a job submission script to perform your computation (on the login nodes)
c) Submit your job submission script to the scheduler (on the login nodes)
d) Scheduler runs your computation (on the compute nodes)
e) Analyse results from your computation (on the login or compute nodes, or transfer data for analysis elsewhere)

Computer cluster scheduling can be done in various ways. Most clusters use some Portable Batch System (PBS) based queue and scheduler4 but there are alternatives and different approach solutions like HTCondor. In Baltasar, we use PBS Torque, with the Maui Scheduler.

5) Storage and File System

The kind of computing that people do on HPC systems often involves very large files, and/or many of them. Further, the files have to be accessible from all of the front-end and compute nodes on the system. So most HPC systems have specialized file systems that are designed to meet these needs. Frequently, these specialized file systems are intended to be used only for short- or medium-term storage, not permanent storage. As a consequence of this, most HPC systems often have several different file systems available – for example home, and scratch file systems. It can be very important to select the right file system to get the results you want (performance or permanence are the typical trade-offs).

6) Parallel Computing

By definition, parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones. As simple as it may seem, splitting efficiently a large problem into smaller ones so that they can be truly solved parallelly, is almost an art. While "parallelly" may seem equivalent to "concurrently", both are quite different things. Solving a problem parallelly implies splitting the problem into smaller, completely independent tasks. If, for some reason, one task depends on some other task we can consider both as concurrent tasks. Since there are few problems that can be completely split into loosely-coupled tasks, the ideal scenario to bear in mind would be "use pure parallelism if you can, concurrency otherwise".

7) Accessing Software

Because HPC systems serve many users with different software needs, HPC systems often have multiple versions of commonly used software packages installed. Since you cannot easily install and use different versions of a package at the same time without causing potential issues, HPC systems often use environment modules (often shortened to modules) that allow you to configure your software

Licensed Under Creative Commons Attribution CC BY

environment with the particular versions of software that you need. We will learn more about modules and how they work later in this lesson. Many HPC systems also have a custom environment that means that binary software packages (for example, those you may download from websites) will not simply work "out-of-the-box". They may need different options or settings in your job script to make them work or, at worst, may need to be recompiled from source code to work on the HPC system.

8) Performance

The "P" in HPC does stand for performance and the makeup of an HPC system is designed to allow researchers to access higher performance (or capabilities) than they could on their local systems. The way in which an HPC system is put together does have an impact on performance and workflows are often categorised according to which part of the HPC system constrains their performance. You may see the following terms used to describe performance on an HPC system:

- Compute bound. The performance of the workflow is limited by the floating point (FPU) performance of the nodes. This is typically seen when performing math-heavy operations such as diagonalising matrices in computational chemistry software or computing pairwise force interactions in biomolecular simulations.
- Memory bound. The performance of the workflow is limited by access to memory (usually in terms of bandwidth: how much data can you access at one time). Almost all current HPC applications have a part of their workflow that is memory bound. A typical example of a memory bound application would be something like computational fluid dynamics.
- I/O bound. The performance of the workflow is limited by access to storage (usually in terms of bandwidth but sometimes in terms of numbers of files being accessed simultaneously). This is often seen in climate modelling and bioinformatics workflows.
- Communication bound. The performance of the workflow is limited by how quickly the parallel tasks can exchange information across the interconnect. This is often seen when single applications scale out to very large numbers of nodes and the amount of traffic flowing across the interconnect becomes high. Particular common algorithms, such as multidimensional Fourier transformations, can also exhibit this behaviour.

9) Baltasar sete sóis Specs and Uses

It consists of total of 272 CPUs. 2,6 TB RAM and 89 TB of Distributed Filesystem Storage. It has 3 Entry Nodes (Baltasar-1) consist of 24 GB Ram, storage of 23 TB and Intel(R) Xeon(R) CPU W3565 @3.20GHz (raid 6 redundant). It also has 4 High-Ram Computational Nodes(nodes 1,2,3,4) of specs 4 AMD Opteron(tm) processor 6180 SE @ 2.5 Ghz (12 cores each, 48 total) 256 GB of Ram and 14 GB SWAP. Node 5 is a Medium RAM computational node with specs as node 1, 2, 3 and 4 but with 128 GB RAM. Node 6-10 are 5 Medium- RAM Computational Nodes. They consist of 4 AMD Opteron (tm) Processor 6344 @ 3.2 Ghz (max clock) - 12 cores each (48 total). 128 GB RAM and 14 GB SWAP. Node 11 and 12 are 2 High-RAM Computational Nodes with specs as 2 AMD EPYC 7410 24-Core Processor @ 3.0 GHz(max clock) - 48 cores each (96 total) with 256 GB RAM and 14 GB SWAP.

Baltasar sete sóis is used by Centra which is a research unit of Instituto Superior Técnico (IST), with a branch at Faculdade de Ciências (FCUL), and a leading center for astrophysics and gravitation. The main topics of research are black hole physics, gravitational waves, big bang and the inflationary universe, supernovae, stellar physics, galaxies, and dark matter.

10) Performance Complexity Problem

There is also more performance complexity. Consider a simple real-world problem like putting 1,000 index cards into alphabetical order (imagine a bunch of vocabulary words for a foreign language class, for example). This problem would take a long time for one person to do, but it's not at all hard to figure out how to do it. One person simply sits down and alphabetises the cards using knowledge she already has in her head (the order of the alphabet) and without the need to talk to or coordinate with anyone else. If we wanted to get it done faster, we could recruit more people to help with the problem. But even adding a single new person dramatically adds to the complexity. The two people now must work together in some way: even if they each start by sorting their stack individually without speaking to one another, they're still going to have to cooperate to get their two stacks sorted into one. And there are a bunch of ways that even that seemingly easy thing can be done.

The exact same problems exist in cluster computing. The problem of getting the processors to work together on a single problem is largely a software problem, but the software needs hardware in order to work: two processors cannot speak to one another unless they are both connected to the same network. You'll need to make sure you have the right processors for the compute task you are running, the right amount of memory and disk, the right cluster interconnect, and so on.

2. Conclusion

In this paper, I present my research study trying to answer the HPC system architecture. I studied a decoupled HPC system architecture for scientific applications. How a decoupled architecture builds separate data processing nodes and compute nodes, with computation-intensive and data-intensive operations mapped to compute nodes and data processing nodes respectively. The data processing nodes and compute nodes collectively provide a balanced system design for data-intensive applications. I have presented modelling to study the potential. The result has shown a promising potential of such a HPC system architecture. I was able to draw important conclusions for HPC system design and development, and these conclusions can guide the configuration and deployment of future HPC systems for solving data intensive scientific problems.

Licensed Under Creative Commons Attribution CC BY

3. Acknowledgement

I would like to say Thanks to Centra, Universidade Téchnia de Lisboa, Sergio Almeida, Lisbon Gravity Group, Professor Vitor Cardoso for letting me access the relevant data and information for my research. I further want to thank my mentor Dr. Manoj Choubey who had introduced me to High Performance computing and also helped me on parallel/concurrent computing that really helped to clarify the subject. The cluster "Baltasar Sete-S´ois" is supported by the DyBHo256667 ERC Starting Grant

References

[1] An Introduction to High Performance Computing- Sergio Almeida
[2] Lisbon Gravity Group, http://blackholes.ist.utl.pt/ .
[3] Baltasar Sete-Sóis Cluster, http://blackholes.ist.utl.pt/baltasar/ .
[4] Cactus Webpage http://cactuscode.org/ .
[5] Portable Bactch System http://en.wikipedia.org/wiki/Portable_Batch_System .
[6] HTCondor http://research.cs.wisc.edu/htcondor/
[7] Torque Resource Manager, http://www.adaptivecomputing.com/products/opensource/torque/ .
[8] Maui Cluster Scheduler, http://www.adaptivecomputing.com/products/opensource/maui/ .
[9] Parallelism and Concurrency, http://www.yosefk.com/blog/parallelismandconcurrencyneeddifferenttools.html .
[10] Baltasar Sete S´ois Cluster Wiki, http://blackholes.ist.utl.pt/wiki/index.php/Baltasar_Sete_Sois .
[11] Centra, https:centra.tecnio.ulisboa.pt/ .

Licensed Under Creative Commons Attribution CC BY