Standardized Construction of HPC Clusters for Academic Usage

Bradford Bazemore
bradford_w_bazemore@georgiasouthern.edu

Standardized Construction of HPC Clusters for Academic Usage

ABSTRACT

A model for the standardization of design and implementation of HPC clusters to be used in universities is presented. Standardization is achieved by using an open-source operating system, network infrastructure, and software packages. The cluster is configured for universities intending to implement an HPC cluster for research or teaching use. No prior understanding of clusters is assumed but a basic understanding of programming, networking and computers in general is required.

INTRODUCTION

In all universities across the country, computing centers that once were the heart of university research have been replaced with server rooms and data centers. For years these same computing centers housed mainframes and supercomputers that supported faculty and student research programs. The jobs computed everything from astrophysics simulations to economic predictions, and were the driving force of science. As mainframes were replaced by a campus-wide network of PCs, there came a rise of specialty supercomputers, but costing millions of dollars and being limited to scientific calculations.

Universities could not afford to accommodate such machines due to their cost. This lack of High Performance Computing (HPC) power left a gap at most universities, a cheaper alternative was needed to bridge this gap. That alternative came in the form of

computing clusters. These are smaller, cheaper computers linked together by a conventional network, all sharing the workload of a particular program. Clusters have many benefits over traditional supercomputers. They are cheaper to own, and can be expanded to nearly unlimited size. For many universities clusters are the best solution to give their faculty and students the HPC power they need.

In years past, creating clusters was something that the faint of heart would not attempt. The setup and configuration would be left to well-trained computing professionals. This was due to the fact that clusters are complicated interconnected machines, needing sophisticated software for the most basic uses. Over the years this task has been simplified by a preconfigured "cluster on CD" [1], the ROCKS system is an example of one. In addition to the simplified software, hardware for clusters has greatly reduced in cost and setup time. With this greater availability of different hardware and software options a new problem arises, a lack of consistency between clusters.

Figure 1. Rocks Logo ®

This paper describes the development of a standardized cluster design and implementation. The design evolved from observations made while designing and deploying an HPC cluster for Georgia Southern University.

Benefits of Clusters

Research is something that is expected to be done at any university. If you are in any science let it be chemistry, physics, engineering, or even economics, there are large amounts of data that must be analyzed. Another large-scale task involves creating models and simulating them on a scale necessary to give meaningful results. These tasks, with sufficient size, can become daunting and even impossible without HPC computation.

Another computational challenge arises when the research requires experimentation. Anyone that is in science knows that experimentation is expensive and very time consuming. This is due to the equipment needs and the time it takes for experiments to complete. These problems can be greatly reduced if not overcome by computational models. When you create models, you can try thousands if not millions of parameters in the simulation before finding feasible solutions. There are cases, such as those in physics, where you could never conduct a physical experiment, such as galaxy collision, or subatomic particle interactions. These are problems that clusters are designed to solve.

Right now the trend at most universities is to pay for remote access to supercomputers such as a CRAY to do their research. Over time this is costly and very limiting to the faculty. The first and most problematic limitation is delay on the job queue. Most of the large supercomputing facilities have thousands of job requests a week. This creates a large time delay on receiving data back from simulations. A computing cluster local to the researcher's university can greatly alleviate this issue.

In the world of science and engineering, supercomputers and clusters are being used with a higher prevalence then in years past. Students in these fields will need an understanding of the programming challenges that one faces when working with these types of machines. This will require real-world exercises to be done in the classroom. Having students submit their work to an off-site system where there is no guarantee of a speedy return is impractical at best. The way for students to achieve the skills needed is for them to use a local system that can allow for swift return of their work. For any student in science or engineering, experience writing programs for clusters is an indispensable ability.

Overview of Hardware and Software

In today's world of endless choices of hardware, let it be servers, workstations or just a simple desktop, having the hardware to build a cluster is rarely the problem for most universities. The problem comes from designing and implementing a system of any scale. Most clusters are used by people that lack advance knowledge in the fields of programming or networking needed to efficiently implement such machines. This can make the task of building a cluster very daunting for most.

Operating System

As in most research-oriented computers, the operating system of choice is Linux. This has many advantages over the other options. Linux is free to the public, saving on the expense of licenses. It is also open source software. This is a critical requirement for clusters and research in general.

Master Node

In all cluster designs there is a computer that manages the system and distributes jobs, we call this the master node. The first and foremost job of this node is to manage and distribute jobs to the compute nodes. In this design, the master node takes in jobs from the users in the form of a program and a request. This request is added to the queue and run when the job scheduler allows. The node then has to gather the resulting data and return it to the user.

One of the other tasks is web front-end, this is an alternative to a simple ssh terminal and it has several benefits. When the master node has a web-based front-end, this creates transparency of the cluster as show in Figure 2. By not having the users directly log into the machine but use a web submission, you take away access to parts of the system that could be manipulated or misused. For the user, this greatly simplifies the task of submitting a job and receiving the data. In order for there to be a web-based system, there will have to be a web server running, which will put a load on the master node, yet the security benefits clearly outweighs the minimal overhead that a simple web server would have.

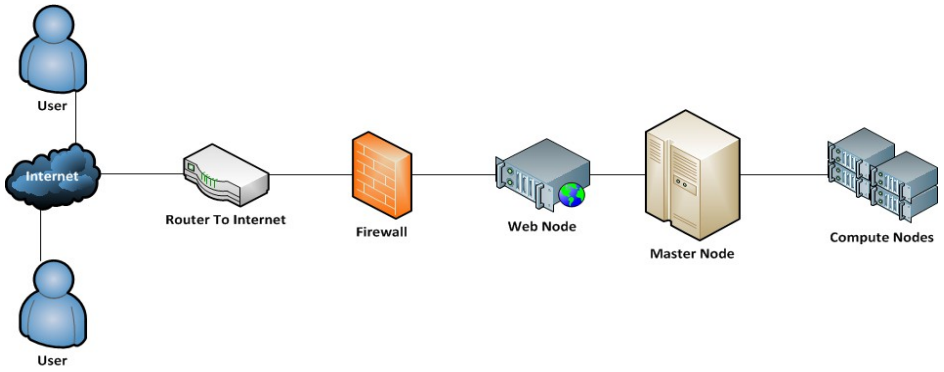

Figure 2. Design of Web Node Network

Compute Nodes

With all clusters there are compute nodes [6]. These are what make up most of a cluster and do most of the work. In this design, the compute node has one main focus, to do computations as fast as possible.

All compute nodes take commands from the master node, this could be a program, or a command to load/retrieve data from a local disk. After executing the command, all processes will return a call to the master node with a signal that it is ready for more work [6]. Due to this simple function, the hardware of the compute node will focus on the processor, a large RAM size, and a simple SSD for storage of the minimal OS and utilities.

In most cluster designs, a standard Linux operating system is installed with the graphical user interface removed as well as most of the utilities of a normal Linux machine. This is the right idea, removing the unneeded utilities, yet this may not be enough. If the compute nodes are to utilize the system with the greatest efficiency, changes to the operating system kernel may be needed.

Networking

Due to how clusters function, the network is the largest bottleneck of any system and has to be designed and configured properly or the entire system will be hindered. The proposed standard model takes this fact into consideration and has addressed it with a dual-network design. The network is set up into two sectors, the communication and maintenance sectors, these sectors are physically separated.

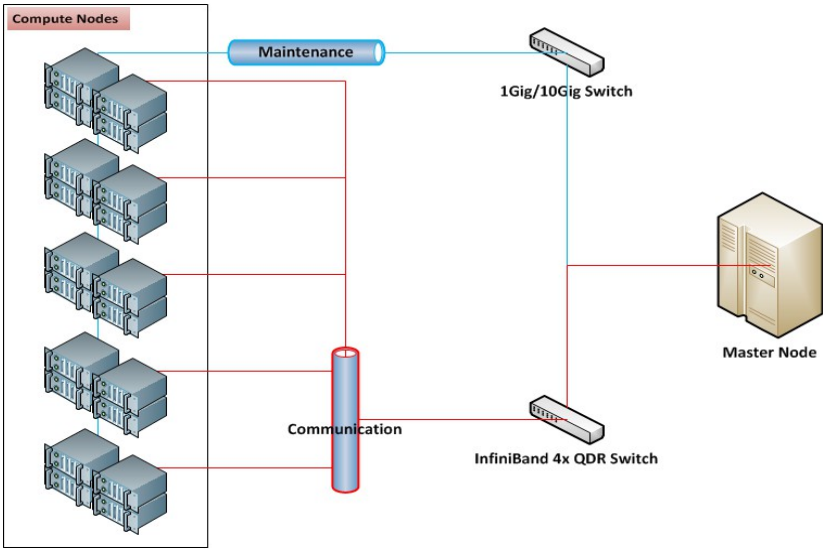

Figure 3. Diagram for Cluster Network

The communication sector is an Infiniband 4x QDR connection, achieving theoretical speeds of 32 Gbit/s [2]. Compared to 10Gig Ethernet, the latency is reduced by 5 to 6 times [3]. This connection is where all distributed jobs will communicate. It will transmit the instructions for the compute nodes and the data needed to carry out the tasks. With all compute nodes connected via Infiniband to a main switch, where the master node also is connected, the system is able to eliminate a significant bottleneck normally found when using 1Gig/10Gig Ethernet.

The second connection, called maintenance, is a 1Gig/10Gig Ethernet connection, spanning the entire cluster. The maintenance network is used for all operations needing to be done on the system without interrupting jobs currently in execution. Examples of this would be system updates, monitoring the progress of jobs, and hardware status checks.

Data Node

Research simulations and calculations require and produce large amounts of data, sometimes in the range of terabytes. This data needs to be stored while a program is waiting to be executed, stored after execution, and remain until the user can retrieve the results. Data of this size must be stored in a location other than the master node, otherwise it will use too much time being processed, moved, and stored.

The data node is tasked with managing all program data, input and output. The other goal is to allow the cluster's storage size to increase as needed without halting operations. If the master node is holding all the data and the system admin sees a need to expand, the entire system will have to be stopped in order to accommodate the addition. With the data being held on a collection of servers a new server can be added and the program redirected to it during execution. The data node is not a required piece for all clusters, only those that see significant usage with many users.

Parallel Tool

Clusters are unique machines, and so are the methods of writing programs to run on them. There are several tools that are used in this cluster design, MPI, OpenMP, Python, and MatLab. These are not the only parallel programming environments that can be used on a cluster, these four are what appear to be the most widely used. These tools are also used on most supercomputers, allowing already created code to work on this cluster design. In this paper we will only discuss the most popular of the four, that is, MPI, and OpenMP.

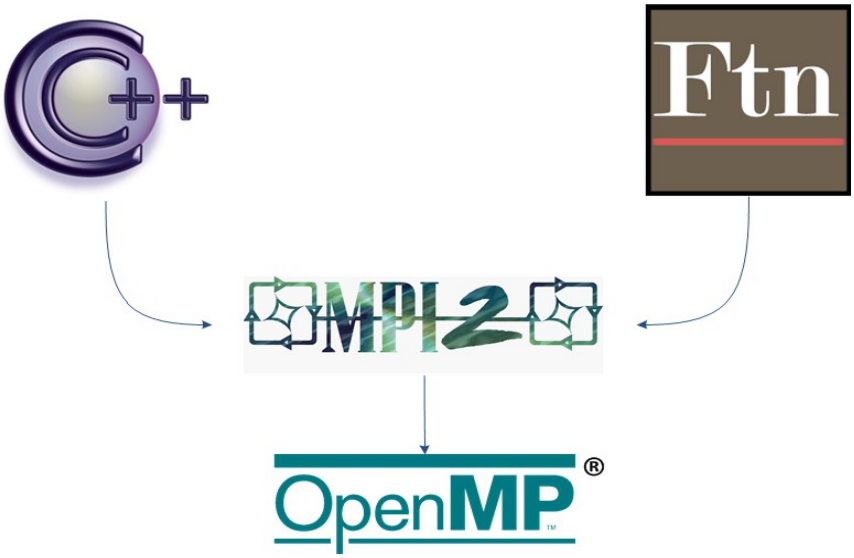

Figure 4. Interaction of distributed programming APIs, (OpenMP®)

MPI or Message Passing Interface, is the main tool used by most programmers, it is also the oldest of the group. MPI was created with the idea of a portable distributed programming environment. It can be used with C/C++ and FORTRAN. The API for MPI is almost the same for all the languages, this is due to the fact that MPI is language independent [5]. Working with FORTRAN and C can be a simpler and easier process because of this.

MPI works by sending messages from process to process on different machines. When you run a program the has been written with MPI, the creator of the processes (Master Node) sends the program to all the Compute Nodes. When they get the program they are given a rank. This rank is how a programmer controls the flow of the program, and all of this exists in the MPI communication world. You then simply state that if the machine has some rank or some range of ranks, then it should do some task. MPI contains over 125 different methods, although only 6 are needed to complete most tasks

[5].

MPI is very useful to transfer data from machine to machine, but it lacks a simple way of creating threads. Threads are very important to clusters now. This is because almost all processors are made with more than one core. This means that if a cluster consisted of N nodes and each node had just 2 cores, then this simple fact doubles the power of the cluster. To get access to these extra cores, a separate interface will be needed. This is where OpenMP comes into the picture.

OpenMP or Open Multiprocessing, is an API allowing the programmer to use the multicore capabilities of the processor. Unlike MPI, OpenMP is much simpler to use, most of the modification to the code needed for multithreading is done automatically. The user simply marks the segment of code that they want multithreaded with a preprocessor directive. When the code is run, OpenMP sets up the threads using the runtime environment, prior to the specified section of code executing [7]. By having the complicated parts of thread creation abstracted by OpenMP, the programmer can focus on the primary purpose of the program.

All of these tools are integrated into the cluster. When the user compiles their C/C++ or FORTRAN program, MPI and OpenMP are all built into the compilation process, greatly reducing the complexity of building the program from source.

Job Scheduling

The last part of any cluster, yet one of the most important, is the job scheduling program. Its job is to do several things. First, it has to create a list of all programs that

need to run and what they will need to run. Second, it has to determine the resources that are available to the cluster at any given moment. Thirdly, it needs to determine the order of user program execution based on parameters given by the system admin. Fourth, it will then queue jobs giving them the resources needed and setting up all storage systems as needed.

With the job scheduler being tasked to do so much, it is difficult to create a truly efficient design. Several companies offer pre-created job schedulers. This can be an option for some, but creating your own with your specific needs in mind will greatly out-perform most commercial software.

Preliminary Findings

The test environment that has been used to confirm the hypotheses stated above was created with limited funding, therefore not all components of the design were implemented.

The EHPRC (Eagle High Performance Research Cluster) Figure 5, was the testing environment created to analyze a few of the hypotheses presented in this paper. The EHPRC consisted of twenty compute node servers. The servers were dual core single processor machines.

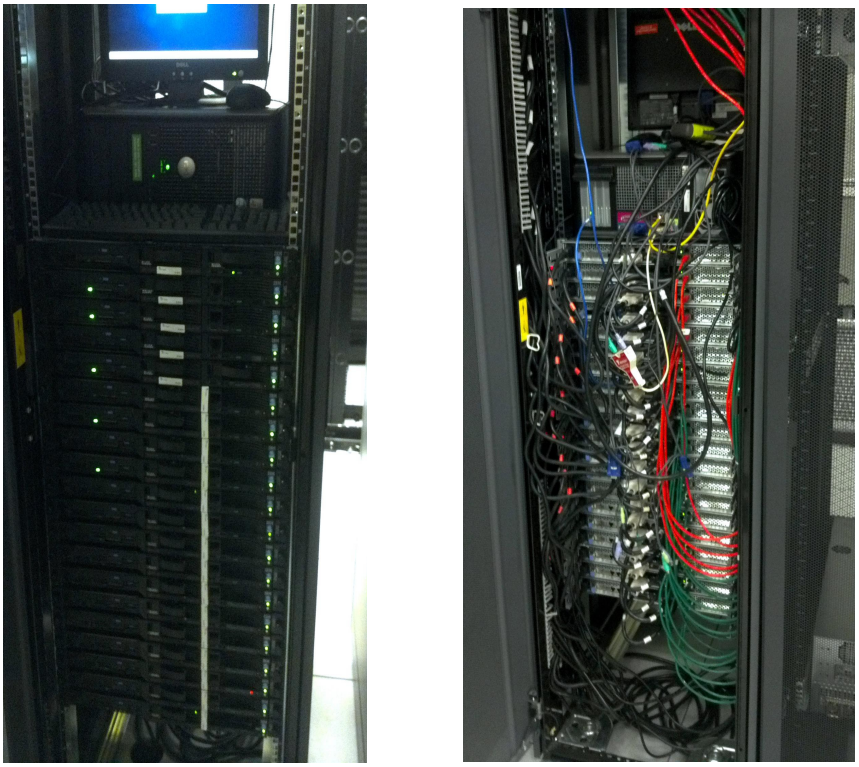

Figure 5. EHPRC (Eagle High Performance Research Cluster) is the system created for Georgia Southern University and was the system used to create all data that appears in the paper.

The master node was also a single processor, dual core machine. The networking was 10/100Mbs Ethernet, using a slower bit rate allowed for better prediction of the effects of the network. If a faster speed, such as 1Gbs, was used and a discrepancy was found in the data, pinpointing the problem would be an issue. If the bottleneck is forced on the network than that can be added into performance calculations, creating a greater accuracy.

To test the current design and implementation of the cluster, a prime number

search program was used. The program was written in C, and MPI was used to distribute the program for the cluster. It was run on a single machine, and also distributed on this cluster. There was no job scheduling program used, this allowed for greater control of the test program execution. Do to the fact that this testing environment is not ideal, the predictions were made using Amdahl's Law [3]. This allows for the current implementation to make possible prediction of its speed up and efficiency.

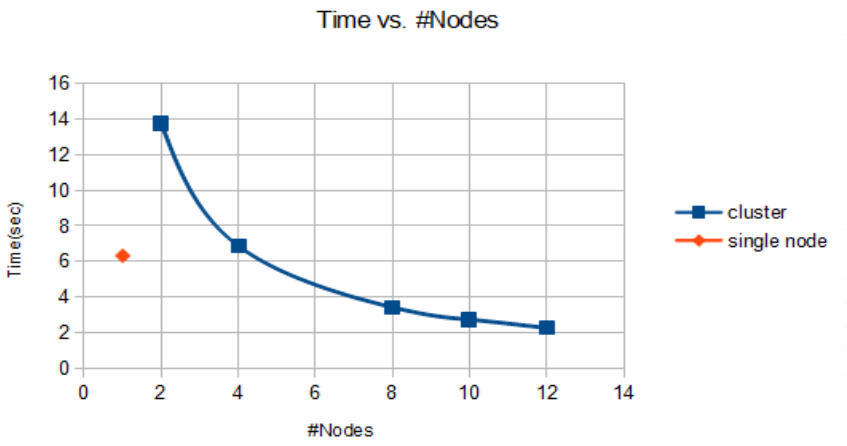

Figure 6. Time vs #Nodes, found using the current implementation

The graph shows the rate of change of the program. As the number of nodes increases, the time of completion drops, yet as Amdahl's Law shows, the curve is asymptotic in nature due to the point of maximum speed up. The curve also follows the predictive calculations, made by using Amdahl's Law, in figure 7. This is based on the current percent of parallelism in the program. The system is running at the predicted speed, with the assumption of a networking bottleneck.

This data was calculated by first determining the speedup using Equation 1. With

that actual speedup found, you can now use Equation 2 to find the estimated proportion of the program that is parallelized. Now using Equation 3, the proportion is used to predict the speedup with N number of nodes.

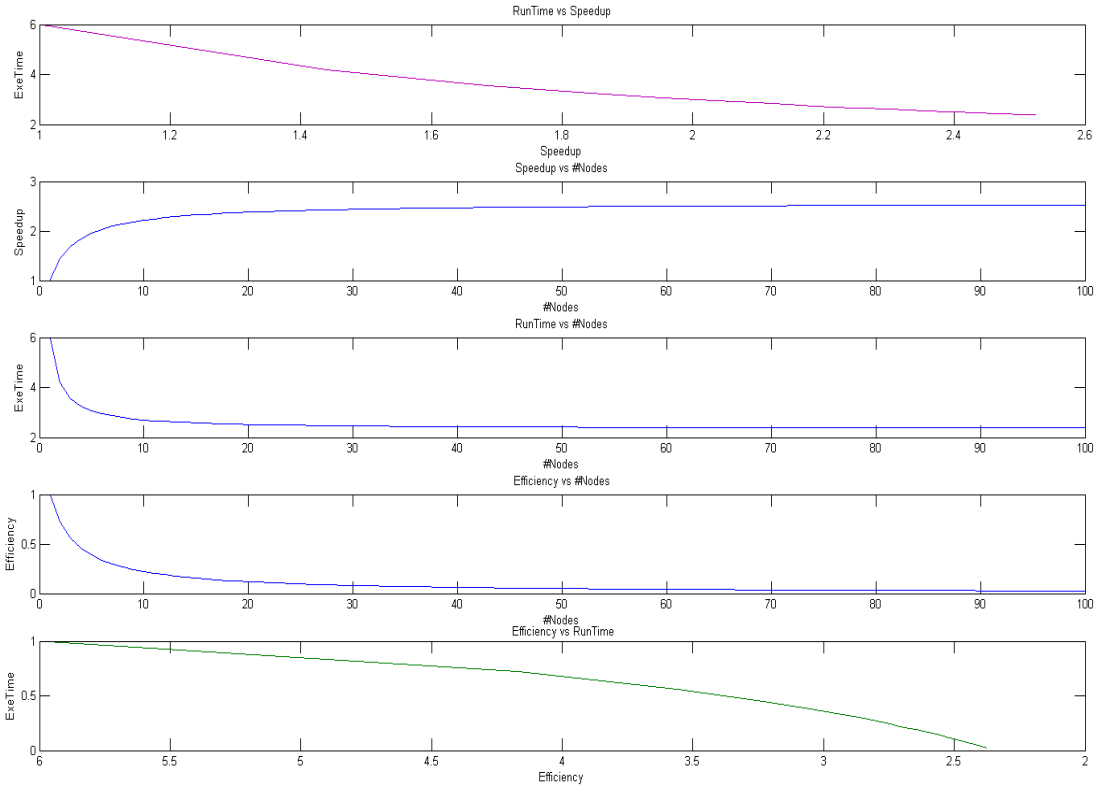

Figure 7. The collection of predictions made using Amdahl's Law and current data.

$$S_p = \frac{T_1}{T_p} \qquad P_{\text{estimated}} = \frac{\frac{1}{SU} - 1}{\frac{1}{NP} - 1}. \qquad S(N) = \frac{1}{(1-P) + \frac{P}{N}}.$$

Equation 1 [4]. **Equation 2 [4].** **Equation 3 [4].**

This data shows that with the current implementation, the system has maximized its processing power, and reached the theoretical limit of speed up.

Conclusion

In the past, a large powerful central computer was expected at all universities.

That trend stopped with the advent of the personal computer, yet the need for these large scale computers did not diminish. An alternative was created to fill this need, the HPC cluster. The use of clusters in the scientific community for simulations and experiments is a vital part of research. In some cases, it is the only way for a researcher to conduct experiments.

The need for clusters requires a simple means of constructing them. This has been done with systems such as Rocks, and Pelican, requiring minimal knowledge of computers or networks. This simplicity has given way to a lack of consistency between machines with each configuration running different utilities, requiring users to learn many different setups and programming environments. A standard model of cluster design is needed to bridge these gaps of design.

In the model proposed all systems will from the users perspective, run the same. A program will execute the same on any system, with out configuration changes. In every system a standard set of utilities are provided that cover the needs of many programs. A maximization of the hardware is achieved by modifying the OS and careful detail made to the network configuration. These modifications are universal and apply to all hardware designs. Preliminary tests show this design to be efficient and with limited bottlenecks. Predictions from the data gathered show that as the system grows, it follows Amdahal's Laws, meaning that the system currently in operation is predictable and running with little wasted time. For science to progress, standardization of these machines is needed. Scientists need to be left to focus on their work with the support of an efficient and consistent infrastructure.

References

1. Sacerdoti, F., Chandra, S., & Bhatia, K. (2004). Grid systems deployment & management using rocks.*IEEE International Conference on Cluster Computing*, Retrieved from
2. Tantisiriroj, W., & Gibson, G. (2008). *Network file system (nfs) in high performance networks*. Informally published manuscript, Computer Science, Carnegie Mellon, Retrieved from institute.lanl.gov/isti/irhpit/projects/nfshpn.pdf
3. Interconnect analysis: 10gige and infiniband in high performance computing. In (2009). Sunnyvale: HPC Advisory Council. Retrieved from http://www.hpcadvisorycouncil.com/pdf/IB_and_10GigE_in_HPC.pdf
4. Amdahl, G. (1967). *Validity of the single processor approach to achieving large scale computing capabilities*. AFIPS spring joint computer conference, Sunnyvale, California. Retrieved from http://www-inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf
5. Barney, B. (2010, July 10). *Message passing interface (mpi)*. Retrieved from https://computing.llnl.gov/tutorials/mpi/
6. Barney, B. (2012, July 03). *Linux clusters overview*. Retrieved from https://computing.llnl.gov/tutorials/linux_clusters/

7. Barney, B. (2012, July 12). *Openmp*. Retrieved from https://computing.llnl.gov/tutorials/openMP/