Procedia Computer Science

Volume 51, 2015, Pages 1907–1916

ICCS 2015 International Conference On Computational Science

Developing a Hands-On Course Around Building and Testing High Performance Computing Clusters

Karl Frinkle, Mike Morris

Southeastern Oklahoma State University, Durant, Oklahoma, USA
kfrinkle@se.edu, mmorris@se.edu

Abstract

We describe a successful approach to designing and implementing a High Performance Computing (HPC) class focused on creating competency in building, configuring, programming, troubleshooting, and benchmarking HPC clusters. By coordinating with campus services, we were able to avoid any additional costs to the students or the university. Students built three twelve-unit independently-operating clusters. Working groups were formed for each cluster and they installed the operating system, created users, connected to the campus network and wrote a variety of scripts and parallel programs while documenting the process. We describe how we solved unexpected problems encountered along the way. We illustrate through pre- and post-course surveys that students gained substantial knowledge in fundamental aspects of HPC through the hands-on approach of creating their own clusters.

Keywords: High Performance Computing, Supercomputing, Beowulf Cluster, HPC, HPC Education, Hardware Management, Parallel Programming, Cluster Operating Systems, Curriculum Development, BCCD, Course Assessment

1 Introduction

Southeastern Oklahoma State University (Southeastern) is a regional liberal arts university. Teaching HPC at such an institution presents challenges for both students and instructors. Two major challenges for this experiment were funding and course structure. After four years of teaching HPC topics with positive results[8], the authors realized a need for students to have a more hands-on experience in the HPC environment, one in which students do not exclusively write programs, but one in which students understand the basic components of creating and utilizing an HPC system.

In many HPC courses, students interact with a reliably stable cluster that has been tested and benchmarked. At Southeastern, our HPC machinery was limited to a single LittleFE [7] unit, which has proven to be an excellent teaching tool, but was our only teaching tool. Our goals for this course were to have students build clusters from basic computer components, benchmark performance, and be involved in the rudimentary aspects of establishing a HPC

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2015

doi:10.1016/j.procs.2015.05.455

cluster. A subsequent benefit to this approach was an additional set of HPC clusters to utilize as teaching tools in future classes. We were able to accomplish this with no funding by harvesting hardware from machines designated for recycling, and working closely with the Southeastern IT department for incidental items such as cables and switches.

2 The Hardware

Southeastern deploys several hundred computers that are periodically retired after two or three years. When these machines are replaced, they usually end up in a storage area until they are picked up for recycling. By coordinating with IT, we were able to retrieve approximately forty computers, extra memory from older non-working computers, network switches, cables, monitors, keyboards, and mice.

Space is at a premium and our experiment was not a proven asset to the students or the university. We were only offered an unused chemistry lab containing four long benches, ample power plugs, and access to the network. Students found it amusing that there was an abundance of sinks and gas nozzles, and although not ideal, we found the room workable and eagerly accepted it.

The first task our students undertook was to verify which of the computers were fully operational. For the non-functional machines, students determined causes of problems. This was a useful exercise because the problems were plentiful and varied, including non-functioning power supplies, damaged RAM and motherboards, and inoperable hard drives. Parts were harvested from the non-functioning machines to construct the largest collection of working machines.

Once a computer was classified as working, the students who certified it as such were required to put their names on the outside of the machine and record specifications to include the amount of RAM, CPU type, speed, and model. This process took more time than we originally planned, but in three weeks, the task was complete. For many students, this was their first exposure to working with the internal components of computers and the students learned basic approaches to analyzing hardware problems and repairing them.

After all of the machines had been cataloged, the students decided the best way to group the computers to create the clusters, primarily considering the variations in the computer specifications. One student shared his observation that since some of the computers were almost identical, we might consider building a Beowulf cluster[11]. Eventually it was decided that the machines would be best broken up into three groups, thus creating the need to build three clusters. At this point, the students formed three mutable teams and proceeded to take their designated machines to their work benches. Once all machines were lined up on each bench, each machine was connected to a local switch. There were a few other minor missing hardware items, but students remedied the situation by contacting the Southeastern “Help Desk”, which was in itself a good communication exercise. Table 1 lists the hardware specifications for each of the three clusters.

3 The Software

After spending a few years working on a LittleFE unit which utilizes the Bootable Cluster CD (BCCD)[3] operating system, we decided our familiarity with BCCD would provide a better chance for success if we used it on our three new clusters. The BCCD operating system is typically used in the setting where all the machines load the operating system into memory

Model	CPU(Intel)	Cores	Clock Speed	RAM	HD Size	Node
Cluster A						
Optiplex755	Core2 Duo	2	2.7 GHz	8 GB	320 GB	master
GX620	P4	1	3.2 GHz	4 GB	80 GB	client(×2)
GX620	P4	2	3.2 GHz	4 GB	160 GB	client(×2)
Optiplex755	Core2 Duo	2	3.4 GHz	4 GB	160 GB	client(×2)
GX620	P4	1	2.8 GHz	4 GB	320 GB	client
GX620	P4	1	3.4 GHz	4 GB	40 GB	client
GX620	P4	1	2.7 GHz	4 GB	160 GB	client
Optiplex755	Core2 Duo	2	2.7 GHz	4 GB	320 GB	client
Optiplex755	Core2 Duo	2	2.7 GHz	4 GB	160 GB	client
Cluster B						
GX280	Pentium	1	2.8 GHz	1 GB	40 GB	master
GX280	Pentium	1	2.8 GHz	1 GB	40 GB	client(×7)
GX280	Pentium	1	2.8 GHz	512 MB	40 GB	client(×4)
Cluster C						
Optiplex7010	I5	4	3.4 GHz	8 GB	80 GB	master
Optiplex755	Core2 Duo	2	2.8 GHz	2 GB	500 GB	client
Optiplex760	Core2 Duo	2	2.8 GHz	4 GB	320 GB	client
Optiplex760	I2	2	2.3 GHz	2 GB	320 GB	client
Optiplex760	I2	2	2.3 GHz	4 GB	500 GB	client
Optiplex760	I2	2	2.3 GHz	8 GB	320 GB	client
Optiplex760	I2	2	2.3 GHz	8 GB	40 GB	client
Optiplex755	Core2 Duo	2	2.7 GHz	1 GB	160 GB	client
Optiplex755	Core2 Duo	2	2.7 GHz	4 GB	80 GB	client
Optiplex755	Core2 Duo	2	2.7 GHz	2 GB	500 GB	client
Optiplex760	Core2 Duo	2	2.8 GHz	4 GB	250 GB	client

Table 1: Table of Hardware Specifications

from a flash drive or from a CD. At most one computer will be "liberated", which means that the operating system is installed onto the hard drive. In this scenario, the other machines can be booted in the Preboot eXecution Environment (PXE)[6] from the liberated machine. This is casually referred to as "pixie booting" and is how the LitteFE unit is configured.

Since each of our machines had a hard drive and we planned on working with relatively large data sets, we decided to liberate each node on the clusters. When we began the liberation process, we used BCCD version 3.3.1.4582. Students installed the operating system on the first computer, which was designated as the master node. Upon installing BCCD on the first client node, problems were encountered. The two machines had communication issues and parallel programs could not be executed across this small 2-node network.

After a week of troubleshooting, students formulated questions and presented them to the BCCD developers forum for assistance. From these detailed questions, the developers determined that problems existed with the operating system being installed across multiple liberated nodes. Over the next few weeks, multiple revisions of BCCD were compiled by the developers and were tested by our students. Results from the updates were reported to the developers, and as a group, the students troubleshot a plethora of issues. Finally, by BCCD revision 3.3.2.4920, the students successfully installed BCCD on all machines. Again, this was an excellent real-

world technical communication exercise.

An interesting anecdote is the fact that much of the BCCD developers' work is done on virtual machines[10]. Problems arose on the physical machines which could not be replicated on the developers' virtual machines, illustrating an important lesson – just because something works on a virtual machine does not mean it is going to work on a physical machine. As an example, one cluster had to be setup by a different process than the others. For a still unknown reason, client nodes on this cluster had to have their IP addresses manually assigned instead of having them assigned by the master node through its DHCP server. Neither the students, the instructors, nor the BCCD developers have discovered why this situation exists.

A proper installation was determined to be one where all the nodes were liberated with one node designated as the master node and the remainder as client nodes. All nodes were connected through a local network switch and could communicate with each other. Finally, parallel programs could be compiled and executed across all the nodes with verification of program execution across all nodes using the standard BCCD user created at installation.

4 Network Connectivity

After reaching the milestone where all three clusters were proven to function, the next step was to add a second network interface card (NIC) to the master node of each cluster. This would allow the systems to be connected to the University's network, and with the correct configuration, be accessible from off-campus as well. When we originally set up our LittleFE unit on the campus network, the procedure was documented and the IT department was made aware of the special settings required for the process.

Next, the students requested permission from the IT department to connect the clusters to the network. The two ports in our lab were activated, along with a port in an adjoining room. Students cut, capped, and tested their own network cables, and then ran the cables through the ceiling to the designated ports.

The IT department gave each cluster a fixed IP on the same domain. Students were allowed to choose a name for their cluster and the names were added to the DNS server; although a modicum of caution was necessary as student-inspired names tend to be a bit risqué at times. After the secondary NIC was set up correctly, each group was required to SSH into their cluster from their laptops while in the lab and connected to the campus network. Afterwards, students verified off-campus access from their smartphones.

It was at this point that another interesting issue was encountered. Since all of the clusters were on the same domain, we had three master nodes trying to assign IP addresses to all of the client nodes on the three systems. After some searching through the BCCD operating files, the groups were able to locate the DHCP file and determined that the settings for the DHCP servers were for both the local cluster network and the domain group. The students disabled the domain DHCP server by commenting out two lines in the BCCD conf.net file located in the /etc/dhcp folder. At this point, all three clusters were completely rebooted and were verified to be running correctly. By utilizing communication, ingenuity, and basic troubleshooting skills, after nine weeks of class, the students had built three functional connected clusters from hardware scavenged from materials destined to be recycled.

5 Cluster Customization

The BCCD operating system had not been used in an all-liberated-node configuration since a much earlier version, and many of the recent revisions to the operating system did not take this configuration into account. As a result, important commands and operations, such as creating user accounts and locating all of the machines on the cluster, did not function correctly.

During the construction phase of the course, no user accounts had been created due to the likelihood of machines being reset. The BCCD Wiki[4] has a detailed set of instructions for creating users on a liberated cluster. Unfortunately, this set of instructions refers to one machine being liberated and the rest PXE booting from the one liberated machine – the standard configuration for a liberated BCCD cluster. After following the instructions on the BCCD Wiki, students attempted to compile and run code across all nodes on their cluster and were unable to do so.

For the standard BCCD cluster, users are created on the master node and cloned to the temporary directories on the master node for each client node that is successfully PXE booted. The BCCD developers originally thought this process should work in the fully liberated mode as well, but it turned out not to be the case. In a first attempt at solving this problem, the groups attempted to SSH into several of the client nodes and manually create the same test user account as was created on the master node. This approach met with limited success. Synching directories across all nodes before executing a parallel program required a password for each node before copying to that node would commence. This would prove problematic if a program was executed across many nodes. Furthermore, at execution time, passwords were once again required for each client node involved in the process. With twelve client nodes this quickly became cumbersome.

The class decided to work on a scripting solution to the user accounts problem instead of trying a new revision of the BCCD operating system since the clusters were functioning with the current version. The idea of writing a script was new to most of the students, but as a group, they learned the syntax required to code exactly what they needed. During the development of the "create user" script, approximately 100 test users were created which did not satisfy the criteria needed to be considered a viable user account. Finally, after much trial and error, the script was completed. A complete "create user" script created a user account across all nodes and generated and copied SSH keys to all nodes, making it possible to both synch directories and execute programs in parallel without requiring passwords.

In the process of writing the "create user" script, another script was written to locate all active nodes on the cluster. The standard BCCD command which accomplishes this, "bccd-snarfhosts", did not work with the configuration of our clusters. Finally, since there were so many test users created during the "create user" script process, another script was written to delete a user completely from each node. By the end of this stage, students had successfully wrote scripts specific to the particular system and particular need.

6 Benchmarking

Encountering problems with the operating system and the user accounts ended up taking four weeks. With just over two weeks left in the semester, there were two tasks left: documenting the process of setting up the clusters, and benchmarking them. The BCCD operating system comes with many programs ready to compile. One of these programs, GalaxSee[5], emulates the n-body problem in parallel. Using the time option when executing the GalaxSee program in parallel, students were able to compile run time data based on the number of particles in the

simulation and the number of processes. These data sets were then plotted in *Mathematica* for analysis by the individual groups and they were asked to analyze the results in the context of their cluster. Finally, results were compared across all the clusters.

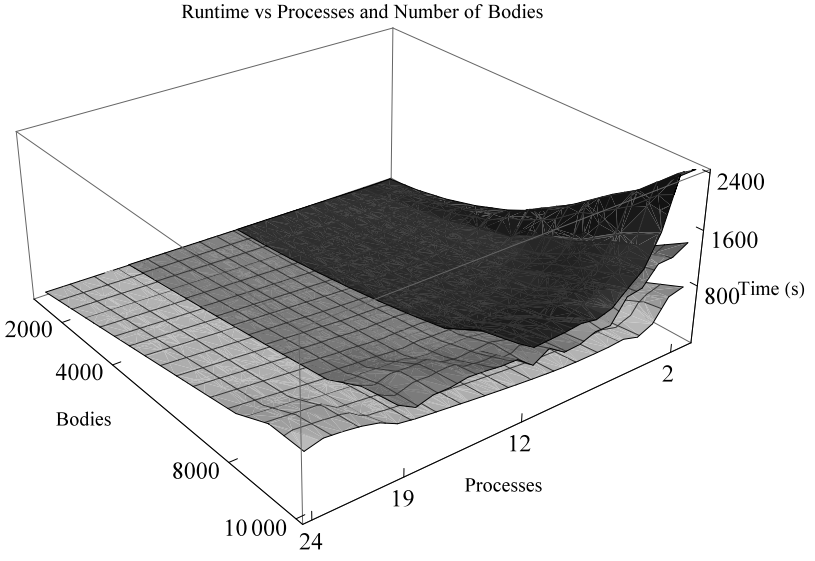

Figure 1: Comparison of real runtimes of the GalaxSee simulation versus the number of bodies and number of processes for all three clusters. The top surface corresponds to cluster B, the middle surface is cluster A, and bottom surface cluster C.

In Figure 1, the three clusters real runtimes were compared based on number or processes and number of bodies in the simulation. Cluster B had a maximum of 12 processes and is the top most surface. Cluster A had a maximum of 19 processes and is the middle surface. Cluster C had 24 processes available and is the bottom surface. Overall, cluster C had the fastest set of CPUs and, as a consequence, had the shortest runtimes. Cluster B had the slowest machines, all single core, and as a result their runtimes were the longest. Figure 1 was shown to the students and they were asked to analyze it, and arrived at the conclusions already given.

Three-dimensional graphs can be rather complicated to analyze, so specific cuts of Figure 1 were made by holding the number of processes fixed. These two-dimensional plots can be seen in Figure 2. Of particular interest is the case of $n = 2$ processes, corresponding to the top row of graphs in Figure 2. Students were asked to interpret why the system time graph (middle) appeared to have only one curve with non-negligible values and to determine which cluster it corresponded to. For cluster B, each machine has only a single-core processor which means that when trying to execute the GalaxSee program across two processes, cluster B must go through the local switch and initiate the program on a client machine. The other two clusters have master nodes with at least two processes, and could execute the program exclusively on the master node.

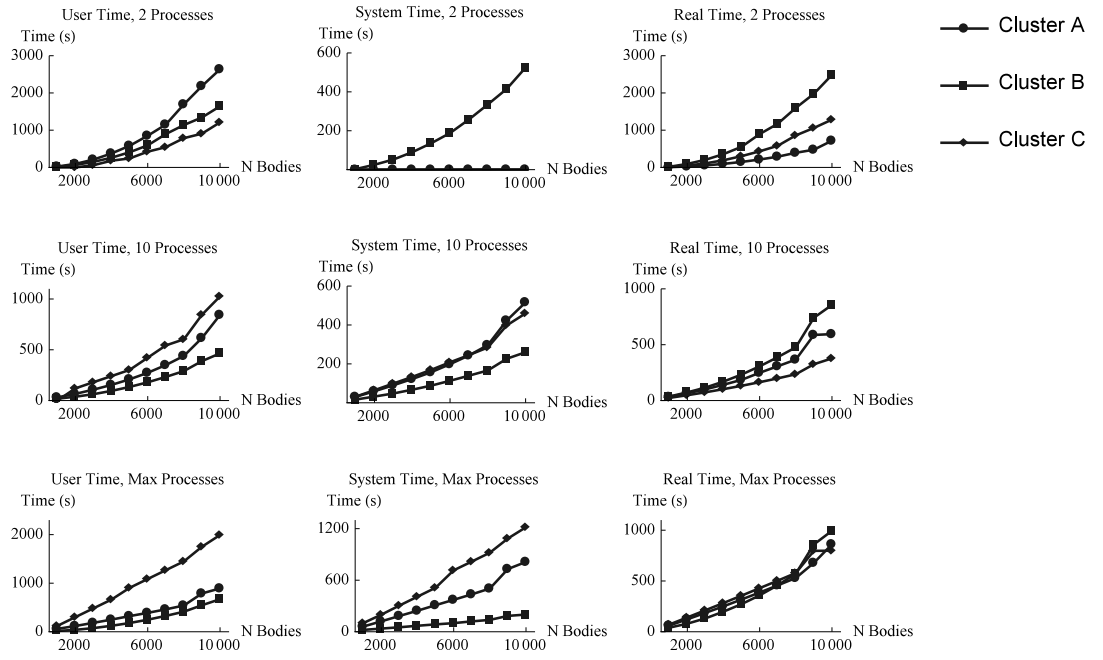

Figure 2: Plots of user, system, and real runtimes versus the number of bodies in the GalaxSee simulation for select numbers of processes. First row is with two processes, second row with 10 processes, and third row with the maximum number of processes. The maximum number of processes was $n = 19$ for cluster A, $n = 12$ for cluster B, and $n = 24$ for cluster C.

7 Process Documentation

Our campus requires specific networking settings to operate in its particular environment. Taking these facts into account, we wanted to create documentation that would ensure that the process of setting up a cluster will be easy to reproduce in the future or in the event that a cluster must be rebuilt during a semester.

Students were required to document each and every step in the cluster build. During the installation process of the BCCD operating system, there are several stages at which user interaction is required, mainly in the networking settings. Documentation for liberating a single node can be found at the BCCD Wiki[4]. However, there is no public documentation for setting up an entirely liberated cluster. Each group submitted their documentation to the instructors and were subsequently required to add information to their write-ups until deemed complete by the instructors. Once complete, the documentation was sent to the BCCD developers to help the developers build a complete set of instructions for setting up a fully liberated BCCD cluster. As a consequence, our students were able to contribute directly to the BCCD project.

8 Learning Outcomes

At the beginning of the semester, students were given a survey to assess their knowledge of all of the aspects of what they would experience throughout the semester. Questions ranged over

topics including hardware, software, networking, benchmarking and documenting procedures.

Students were to assign values to each question. A '0' was a "strongly disagree", or "very little experience", a '5' was "average" while a '10' was a "strongly agree", or "very good at".

1. I have built a computer from a pile of components.
2. I have set up a network on a single computer.
3. I have set up a network on several computers.
4. I have installed a Linux OS distribution on one or more computers.
5. I have worked with campus IT people and basically understand how our network infrastructure is designed.
6. I have "benchmarked" computers.
7. I have written C++ programs that work.
8. I have written C programs that work.
9. I have written documentation for computer projects.
10. I know how basic HPC architecture works.
11. I have written my own HPC code.
12. I have compiled and run my own HPC code.
13. I can understand HPC code (even if I am not that good at writing it).
14. I can execute and understand basic command-line Linux commands.
15. I can handle file management in a Linux environment.

The survey was given once again at the end of the semester. There were 16 students who took the survey at the beginning of the semester, and 16 who took it at the end of the semester. Mean values for each question were calculated.

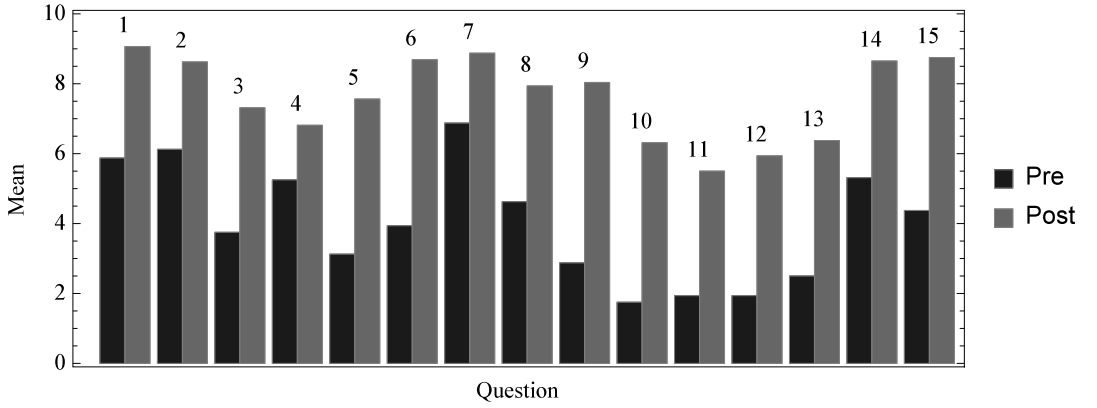

Figure 3: Comparisons of Means for Survey Outcomes

As is illustrated in Figure 3, mean scores for each category increased, with the largest increases occurring for the HPC specific questions and those related to the Linux based operating system environment. Recent ACM guidelines[1, 2] for program accreditation now include HPC. We tailored this course to comply with the most recent guidelines. In this course, students were exposed to more HPC content and as a result, feel more confident with the HPC environment, as is shown by the pre- and post-survey data for questions 10 through 15.

9 Conclusions

By the end of the semester, the students had a variety of learning experiences. Students had learned to set up the hardware required to run a cluster. Many hardware problems were encountered, assessed and rectified. Installing a specialized operating system required communication with the developers of the operating system. Setting up a cluster network by connecting to a local switch, and then creating connectivity to the outside world necessitated interacting with the IT department. Customizing the clusters after they became operational required the students to learn about scripts, and also taught the students how to work within the confines of the given operating system.

The authors plan to continue this project and offer a follow-up course where the students will write parallel programs on the machines they or their peers have built. These clusters are not extremely large or powerful, but they demonstrate the concepts of HPC. As in past endeavors, we can use these clusters to debug parallel programs and after thoroughly debugging them, execute them on Boomer, which is the supercomputer managed by the Oklahoma Supercomputing Center for Education and Research[9] at Oklahoma University in Norman.

As a final remark, we dispel the widely held myth that utilization of HPC resources requires a substantial expenditure of money and resources. In this experiment, we were able to construct three dedicated HPC clusters by working closely with our University and taking advantage of the HPC resources available in the public domain. All necessary hardware was harvested from our University's obsolescence program, a dedicated room was procured for our modest requirements of a HPC lab, and the software was obtained at no cost. All of the problems we encountered while building the clusters were turned into opportunities for the students to learn more about HPC and give them insight into what they may encounter in industry. The end result was a net expenditure of zero dollars to accomplish the successes described in this paper.

10 Acknowledgments

The production and presentation of this paper at the ICCS 2015 event would not have been possible without a generous award of a Southeastern Oklahoma State University Faculty Research Grant for each of the authors of this paper.

This experiment is an ongoing project that will develop different versions of what we perceive to be effective teaching techniques for high performance computing classes. Curriculum development is in its infancy in the HPC field and we plan to contribute to the industry by preparing students to work in the industry. None of this would be possible without the valuable help we have received from Henry Neeman, director of OSCER and all the priceless time we have been given on OU's supercomputer, Boomer.

We also appreciate Charlie Peck and all the individuals involved with the LittleFE group for legitimizing our early efforts by awarding us a LittleFE supercomputer. The BCCD development team worked closely with us on fine tuning the operating system so that it would meet our needs.

On the local level, the network operations staff at Southeastern allowed us to connect with the outside world and gave us permissions and ability to access any resources we needed to carry out our projects. The CS 4973 – Supercomputing Architecture and Development class is also to be thanked for enrolling in this course, and successfully creating three working clusters. One student in particular, Keith Pearce, went above and beyond what was expected of a student in this course and really helped get these three clusters online. Finally, we thank Bobbi Page for her insightful comments and suggestions regarding the contents and structure of this article.

References

[1] ACM. Computer Science Curricula 2008. [online], 2008. http://www.acm.org/education/curricula/ComputerScience2008.pdf, last viewed December 2014.

[2] ACM. Computer Science Curricula 2013, Final Report 0.9. [online], 2013. http://www.acm.org/education/CS2013-final-report.pdf, last viewed December 2014.

[3] BCCD. Bootable Cluster CD. [online], 2014. http://bccd.net, last viewed December 2014.

[4] BCCD. Bootable Cluster CD Wiki. [online], 2014. http://bccd.net/wiki/, last viewed December 2014.

[5] BCCD. GalaxSee. [online], 2014. http://bccd.net/wiki/index.php/GalaxSee, last viewed December 2014.

[6] Intel Corporation. Preboot Execution Environment (PXE) Specification, Version 2.1. [online], 1999. http://download.intel.com/design/archives/wfm/downloads/pxespec.pdf, last viewed December 2014.

[7] LittleFE. Parallel and Cluster Computing Education on the Move. [online], 2014. http://littlefe.net, last viewed December 2014.

[8] Mike Morris and Karl Frinkle. A Three-Semester, Interdisciplinary Approach to Parallel Programming in a Liberal Arts University Setting. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, page 66. ACM, 2014.

[9] The University of Oklahoma. OSCER – OU Supercomputing Center for Education Research. [online], 2014. http://oscer.ou.edu/, last viewed December 2014.

[10] James E Smith and Ravi Nair. The Architecture of Virtual Machines. *Computer*, 38(5):32–38, 2005.

[11] Wikibooks. Building a Beowulf Cluster — Wikibooks, The Free Textbook Project. [online], 2011. http://en.wikibooks.org/w/index.php?title=Building_a_Beowulf_Cluster&oldid=2210594, last viewed December 2014.