Linux and Open Source in the Academic Enterprise

Mike Davis
Virginia Commonwealth University
P.O. Box 980016
Richmond, VA 23298
(804) 828-9843
jmdavis@hsc.vcu.edu

Will O'Donovan
Virginia Commonwealth University
P.O. Box 980016
Richmond, VA 23298
(804) 828-9843
wjo@hsc.vcu.edu

John Fritz
Virginia Commonwealth University
P.O. Box 980016
Richmond, VA 23298
(804) 828-9843
fritz@hsc.vcu.edu

Carlisle Childress
Virginia Commonwealth University
P.O. Box 980016
Richmond, VA 23298
(804) 828-9843
cgchildr@hsc.vcu.edu

ABSTRACT

Open Source Software (OSS) has made great strides toward mainstream acceptance over the past two years. However, many IT managers, both in business and academia, are still cautious about OSS. Is it reliable? Is there support? Will it last? Linux has further complicated the issue not only because its operating system is OSS, but because it runs on inexpensive commodity hardware. Often IT managers are hesitant to move from long trusted proprietary hardware and software and trust major projects to OSS and commodity hardware.

Past SIGUCCS presentations by Virginia Commonwealth University have detailed our use of standards based email and directory services to replace legacy systems. That email migration put us on a path toward the implementation of a variety of Open Source Software and commodity hardware solutions.

In 1997, the primary Open Source Software in use on our campus was Perl. In the past three years, we have implemented OSS solutions for email, webserving, webmail, software development, directory services, and database development. While implementing OSS we have also begun to implement commodity hardware solutions running the Linux Operating System in those areas where it provides benefits. While Linux webservers have become the norm, we have brought other Linux based machines online for directory services, webmail, and research. Recently, we investigated, benchmarked and purchased a Beowulf Linux cluster to significantly expand our ability to provide resources for our computationally intensive research.

Open Source Software and the Linux operating system provide two very important tools to allow universities to leverage skilled and trained staffs to meet user needs and expectations in a highly cost effective manner without sacrificing quality of service. This paper will examine VCU's transition from proprietary hardware and software solutions to OSS and commodity hardware. It will focus on selection criteria, testing methods, implementation, the evaluation process and the "selling" of OSS and commodity hardware to IT managers.

Keywords

Linux, Open Source, Beowulf, commodity hardware..

Introduction

VCU has traditionally met its users computing needs with a variety of proprietary hardware (Pyramid, OS 390, VAX/VMS, IBM SP, SGI) and a combination of proprietary and custom software. Over the past 15 years the VCU Academic Technology unit has fought to keep pace with the ever-expanding general and research needs of the University community. Through the use of case studies, this paper will show the effectiveness of Open Source Software, commodity hardware and the Linux operating system to solve real world problems of varying natures.

Hardware:

As desktop computers became commonplace (and the need for centralized office applications diminished), the primary uses for the central computing resources became email, database storage and analysis, statistical analysis, and computationally intensive research such as drug design, chemical and physical simulations, mechanical simulations, bioinformatics, and Monte Carlo type applications for a variety of disciplines.

In 1998 email needs for faculty and students were met by four machines and two distinct protocols. On the Medical College of Virginia Campus of VCU, centralized email service was provided using the IMAP standard on a dedicated SGI Origin 200 (more

information about the selection and implementation of IMAP for the campus may be found in the SIGUCCS proceedings for 1997 and 1998). On the Academic Campus of VCU, email service was provided by three multi-function IBM SP2 nodes using sendmail, POP, and later IMAP. These three machines also provided a wide variety of general and research services.

The university offered three machines to meet the needs of its computationally intensive researchers. Two eight processor SGI Origin 2000's (one on each campus) and a four processor SGI Power Challenge L were available to researchers on the two campuses.

For statistical and database research, two one processor IBM SP wide nodes were available (one of these IBM SP's also provided email and web service for faculty and staff on the Academic campus). In addition, statistical software was available on one eight processor Origin and the four processor Power Challenge L.

A small number of workstations were available on the Medical College of Virginia campus for limited visualization and general usage. In addition an IBM SP node was setup to provide login services for users needing dial-in access to email.

By late 1998, all of this hardware was moving toward 100% utilization. The four IBM SP nodes were often taxed by 300% to 400% usage (ie. running three to four times the optimum number of jobs) , the SGI research machines were closer to 100% usage, and the SGI Origin 200 IMAP commercial mailserver software was beginning to cause load problems. To meet the needs of users, an additional four processors were added to one of the Origin 2000's and planning began to replace the mailserver software on the Origin 200 as well as to secure replacements for the aging IBM SP nodes.

Until 1998 the use of commodity hardware was limited to the PC and Novel LAN groups at VCU. One of the authors, Mike Davis, used Intel hardware and Linux as his primary desktop machine and two low-cost Cobalt RAQ machines were purchased to provide support for University FTP and a Community Web Service Project. But, most users and computing staff had little interest in commodity hardware or Open Source operating systems.

Software:

Ironically, early 1998 was the point at which a number of Linux advocates made the decision to supplement the traditionally bottom-up strategy of unix implementation and attempt to directly interest media in Linux to increase the operating systems publicity (Eric Raymond's book *The Cathedral and the Bazaar* details this decision). These advocates hoped that increased publicity would interest Chief Information Officers and higher level managers in Linux while engineers, developers and administrators continued their grass-roots advocacy and implementation. As press reports about Linux increased, its acceptance at VCU increased as well.

In addition the increased publicity of Linux helped propel the concept of Open Source Software from the domain of software developers and administrators to the attention of both the public and information technology managers. Open Source Software has a variety of definitions. In general, Open Source is "free" software which allows: free distribution, access to source code, the right to modify the source code, no discrimination against persons or groups, and no restrictions on fields of endeavor.

Open Source Software goes beyond a simple definition. Key to the Open Source movement are several concepts which interested Academic Technology. Stability, portability, support, and access to source code are the reasons that we find open source software appealing.

Stability:

Most open source projects rate stability at least as highly as performance. With the price of hardware decreasing rapidly, and staff salaries increasing, we'll gladly put up with a 30% performance hit if it will eliminate even one show-stopper bug. Even with the stress on stability, we find that we often get much better performance from Open Source Software.

Portability:

Open Source Software is portable to many platforms, including commodity hardware. This portability allows Academic Technology to leverage the skills of its staff.

Support:

We've found that when we can't solve a problem with commercial software, the problem often also tends to be beyond the capacity of the commercial vendor's support staff. Universities can seldom pay the large fees required for direct access to developers. In one case, we later learned that the support engineer that we consulted met once a week with a manager who met once a week with developers. Ironically that problem was finally resolved in the same manner as most Open Source issues, an engineer at the vendor read our Usenet posting and responded with the appropriate information. In general Open Source authors are more accessible than developers for commercial vendors.

Source Code:

While source code may not be the “Holy Grail” as some Open Source advocated describe, it is important. We use the source code, sometimes to fix problems, and sometimes to diagnose them

Case Studies:

Linux, and Apache to the rescue:

In the summer of 1998, the Academic Technology unit on the Medical College of Virginia Campus at Virginia Commonwealth University made the decision to purchase a dedicated Linux server. The machine was a dual Pentium II- 400 scheduled to be a test unit for the replacement of an SGI Indigo2 webserver that was overloaded by a combination of general web pages and the distance education software "Web Course in a Box." This new server would run the Redhat 5.2 version of Linux and use the Open Source Apache webserver to provide central support for departmental, staff, and student web pages on the campus. The machine would represent the first centralized use of Intel commodity hardware and totally Open Source software at VCU. As is often the case, these plans needed to be changed due to changing situations.

By late fall of 1998 the SGI webserver was at a critical point

primarily due to a dramatic increase in. "Web Course in a Box" (WCB) usage and the resultant poor performance of WCB. The time required to seamlessly move both services from the machine was too long to make a transition that was transparent to users, at least several days would be required. In addition, we worried about some of the older cgi scripts using early versions of Perl on the SGI. We decided that moving everything to the new machine was not a viable option.

The Instructional Development Center (IDC) at VCU was created to explore new instructional technology, and to initiate and demonstrate the effective use of innovative educational methods for the delivery of instruction. Communications with the IDC (co-developers with Madduck Technologies of "Web Course in a Box") revealed that a test server was running using the Apache webserver and the BSD operating system at a test site. Once we new that "Web Course in a Box" would function reliably with Apache, we were able to plan for the transition. In coordination with IDC, we set out to determine possible problems with the move.

Database issues were the first difficulty that we encountered. The SGI machine used a different version of unix databases than RedHat Linux. We would need to create Perl scripts to convert the databases to text format and then convert them to the new database format on the Linux box. The IDC produced these scripts, then we tested them, worked together to make corrections and re-tested. When we had these tools in place, the rest of the move was simple though it required much attention to detail. During this time, we practiced the move and configuration repeatedly to try to make it as quick as possible. In all, we would need to move almost a gigabyte of data and convert hundreds of small databases. Each move was timed and the data spot tested to insure integrity. The process would be to convert and move WCB, configure Apache and then set a redirect on the commercial webserver on the SGI. At this time, we would also begin the use of a new, dedicated url for "Web Course in a Box." The dedicated url would make any future moves of the server much easier.

When the time came for the actual move, down-time was limited to three hours and the process proceeded flawlessly. By noon on the day of the move, the new server was handling its duties extremely efficiently. The one issue that we experienced was that the new server was too fast to track the 5000 line cgi processes when they ran. While the SGI and commercial webserver had required 15 to 30 seconds to run the cgi, the Linux box and Apache required less than two seconds. It tool several frantic and comedic minutes to realize that all was fine and that the new machine was just too fast to watch the processes run using unix process monitoring tools.

The success of the "Webcourse in a Box" move was noticed and commended by both IDC and the Office of Information Technology senior management. The level of respect for commodity hardware, the Linux operating system, and Open Source software increased dramatically within OIT. Two more dedicated Linux servers were ordered, a rack-mount dual Pentium II-450 machine to finally replace the general webserver still running on the SGI and a rack-mount single processor Pentium II-450 test machine.

Improving IMAP mailserver performance:

As previously mentioned, the commercial email server software at Academic Technology on the Medical College of Virginia campus was having problems supporting the load of approximately 7000 users. The major issue was that every access of a folder required a number of reads to add new messages. Every time a user opened a his inbox, the server went through the process of performing a STAT (examining the file's system information) of each message followed by a READ of the headers of each message. When users had large numbers of messages in a folder, this overhead proved significant.

We determined that when messages in a folder numbered less than 200, performance was satisfactory. If more than 200 messages were present, the wait became a problem. In the some cases, opening a folder with several thousand messages could require five minutes or more. In addition, during times of heavy use, the cache saved by the server could be overwritten causing all of the messages to need to be reread. Worse than this was that users deleting messages one at a time were forced to wait as the system updated itself.

Originally, we tried to improve performance by tuning the system parameters to better handle these mailserver issues. When that proved less than satisfactory, we began to break the mail directories up over a number of separate disk partitions. These changes were nothing more than "first aid". The problems still existed, but we attempted to spread users across partitions in such a way that we could minimize the wait experienced as a whole. We knew that we needed to solve this problem that was endemic to the commercial mail server.

In December of 1998, our Unix team began to investigate various mailserver's IMAP implementations. We looked at PMDF, Simeon (ExecMail), Cyrus, and finally the updated version of the Netscape Mailserver 4.0. Our investigations led us to reduce the contenders to Simeon, Cyrus, and Netscape Mailserver 4.0. Netscape was a complete rewrite of the program with totally new message handling. Having had problematic experiences with early versions of Netscape's products, Netscape Mailserver 4.0 was soon eliminated. With the choices reduced to two, we obtained copies of both and set about testing and benchmarking them.

Cyrus is an Open Source product originally developed at Carnegie-Mellon. It also served as the basis for the commercial product Simeon. The differences between the two in late 1998 were mostly in the admin interfaces. Simeon had a Graphical User Interface for admin as well as a command line interface. Cyrus on the other hand was managed from a list of simple command line arguments. Simeon did have some minor performance enhancements not available with Cyrus.

Having had problems with our previous email server vendor's response time to bugs and overall general support, we had become somewhat distrustful of commercial support in general. We subscribed to both Simeon and Cyrus Usenet groups and monitored the performance of the company with regard to bug fixes and product updates. In general 30 to 60 days were required for Simeon bug fixes. With the source code, we found that we could fix bugs in Cyrus ourselves or find someone on the Cyrus Usenet group that had already fixed the problem in a much shorter

time.

Having reached a dead end on the issue of a source license for Simeon, we set up a conference call with our salesperson and a Simeon Engineer to determine just what the true differences in the programs were. We determined that there were no substantial differences in the programs. Both programs used the same file layouts. Both used databases to store header information and both were remarkably faster than the commercial product we were using. Simeon offered GUI admin tools, some minor performance enhancements and commercial support but no source code. Cyrus offered no GUI admin tools but had a very active group of developers at Carnegie-Mellon and elsewhere and complete access to all of its source code. With the source code, we felt comfortable that we could work with the Open Source community developing Cyrus to make any fixes necessary.

On April 1, 1999, we made the decision to implement Cyrus. At this point we began two months of stress testing to assure ourselves that Cyrus was a completely stable product. These stress tests included driving the load on our test server to over 200. Even at a system load of 200 (100 times the optimal load for a two processor machine), Cyrus kept functioning reliably.

Satisfied that Cyrus was stable, we could begin the final phases of the email migration. We wrote programs to convert the Netscape folders of our users to the database oriented Cyrus format. The holiday weekend of July 4th, 1999, was set as the migration date and users were informed that email would be out of service for up to 48 hours. A complete backup of all 50 GB of email was made, and the actual migration began. We also used this time as an opportunity to upgrade the email hardware by attaching another set of RAID disks. The process took 41 hours. To users, this migration was transparent. They experienced the migration down time, followed by remarkably better email service with no major changes required on the client side.

After a full year, the performance of Cyrus has exceeded all expectations. Though the use of databases complicates the software, the overall administration is a fraction of that required previously for the "supported" commercial product. Between .25 and .5 of an FTE has been freed by switching to Cyrus. This efficiency has allowed the department to move forward with other longterm goals.

Beowulf cluster for research:

Computationally intensive research has increased dramatically over the past two years. In 1997, the research load on the MCV campus was handled by a single four processor SGI Power Challenge L. In 1998, we added an eight processor Origin 2000. In 1999, the Origin with supplemented by the addition of four processors and of a 180GB Fibre Channel RAID. By the end of 1999, these two machines were running with loads 2 to 4 times optimum usage. Jobs that required 14 to 30 days of time with a full processor were taking two to four times as long. The Origin on the Academic Campus was experiencing similarly high usage.

Plans had been made to combine the research computing resources of the two campuses and to expand this combined resource. Funds were secured to purchase equipment to increase the number of processors in the combined single system image Origin 2000 from 20 to 28. While we believed that this increase should provide adequate resources for current users for a year, we knew that it provided no resources for new research and did nothing to alleviate chronic resource shortfalls on the four IBM SP2 nodes in research use by the two campuses. To expand our services, we would need to find another solution to these performance problems.

Commodity hardware, Open Source software and research by NASA and a variety of other institutions seemed to provide an appropriate solution. We could use a Beowulf cluster to meet our increasing needs. A Beowulf is defined in *How to Build a Beowulf* by Sterling et al, as "a collection of personal computers (PCs) interconnected by widely available networking technology running any one of several open-source Unix-like operating systems."

Instead of merely adding eight processors to the Origin 2000's, the same funds could be used to secure a 32 processor Beowulf cluster (for computational research), a four processor Sun Enterprise server (to replace the obsolescent SP nodes) , and 324 GB of disks. With these purchases, we would be able to expand our services.

We began to research and document the performance of Beowulf clusters in a wide variety of research including chemical, physical, mechanical, biological, and geological. Research from around the world was used to provide a firm foundation on the real-world capabilities of Beowulf clusters.

Finally, we needed to benchmark applications in actual use by VCU researchers. Our benchmarking efforts received invaluable assistance from a VCU researcher in the School of Engineering who had purchased a Beowulf for his personal research. We provided the researcher with assistance in setting up, managing, and maintaining this six processor cluster, as well as porting software to it which provided us both experience and increased comfort level as we moved ahead. In addition, we were able to run benchmarks of engineering and computational chemistry codes on this small cluster.

The initial benchmarks for clusters centered around substantiating the belief that commodity Intel hardware and Linux had the capabilities required for computational research. These benchmarks consisted of the test programs of the GAMESS US software. GAMESS is one of the standard tools used in computational chemistry and has been developed by the Ames lab at the University of Iowa. This software runs on a wide variety of platforms including the SGI Origin, and Linux clusters.

The newest version of GAMESS was obtained, compiled, installed, and optimized for shared memory message passing on the 12 processor Origin. In addition the standard version was obtained, compiled and installed on a node of the six processor Beowulf in the School of Engineering.

The Beowulf cluster performance exceeded that of the Origin 2000 on all of the short tests while performing at greater than 66% of the larger machine on the longer tests. This provided important evidence that a two processor $4000 node of a Beowulf had the capacity to perform well when compared with a $29000 node on

the SGI Origin 2000. Some representative samples are below.

Table 1. Initial results:

Test	Origin Time	PIII-700 Time
E01	0.9 seconds	0.4 seconds
E04	0.3 seconds	0.1 seconds
E07	1.4 seconds	1.2 seconds
E10	0.5 seconds	0.2 seconds
E13	1.0 seconds	0.7 seconds
E16	0.3 seconds	0.1 seconds
E19	0.9 seconds	0.5 seconds
E22	2.5 seconds	2.1 seconds
E25	4.0 seconds	1.5 seconds
E28	1.4 seconds	1.2 seconds
E31	17 seconds	25.4 seconds

After apparent success with the sample programs further computational chemistry benchmarks were run using GAMESS on real world molecules. These benchmarks compute the direct rhf energy of a crown ether molecule. The results of these benchmarks showed that the Beowulf was significantly faster than the Origin in both single and dual processor tests.

Table 2. Single processor results:

Machine	Cpu Time	Wall Time	CPU use
Origin 2000	4493.9 sec	4531.0sec	99.18%
PIII-700	2908.5 sec	2908.5 sec	99.35%

Table 3. Dual processor results:

Machine	Cpu Time	Wall Time	CPU use
Origin 2000	2342.7 sec	2355.0 sec	99.48%
PIII-700	1581.8 sec	1601.8 sec	98.76%

Within the PIII-700 node, a speedup of 183% was experienced when testing two processors versus one. The SGI Origin scaled slightly better at 192% percent, but was still significantly slower in dual processor tests than the PIII-700 dual processor machine. These results mirror similar results from a University of Adelaide sponsored paper "Commodity Cluster Computing for Computational Chemistry." With these results and similar results for mechanical engineering applications and Monte Carlo applications, we moved ahead with our purchase of a 32 processor Beowulf cluster.

Our beowulf, hydra.vcu.edu, consists of 16 dual processor 600mhz nodes (32 processors total). It uses two 100Mbit fast Ethernet switches, two KVM (keyboard, video, mouse) controllers to allow all nodes to use one monitor, keyboard and mouse, and two remote power controllers to provide a remote startup, shutdown, and restart capability.

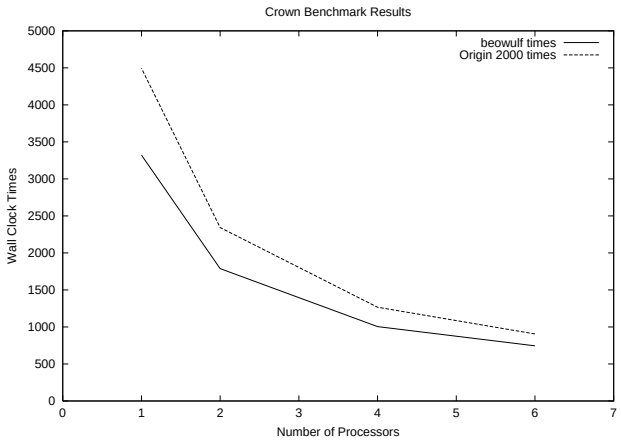
The cluster is networked by bonding two 100Mbit Ethernet cards in each node to create a single channel. Originally designed by NASA researcher Donald Becker for 10Mbit Ethernet, this network topology still continues to prove its efficiency. Tests on our Beowulf show that we can obtain 184.3Mbits/ second of TCP throughput for a cost of $248 per node. Due to bus and timing issues, current Gigabit Ethernet will support 240Mbit/second of TCP throughput at a cost of $1650 per node. We are quite pleased with the price/performance of channel bonding in our cluster.

One issue that may cause some resistance to the implementation of a beowulf solution is a belief that beowulf clusters are less scaleable than various traditional vendors hardware. Clearly, these arguments must be addressed with real world data and should be examined from a price/performance point of view.

Since our purchase and installation of a Beowulf cluster, we have spent time determining the performance of that cluster. We have examined how fast the network is, how the processors perform, how the network mounted files systems perform, and how the system performs overall. In a number of benchmarks, the Beowulf performance is superior to machines ranging from a Cray Research T3E, to an IBM SP cluster, and to an SGI Origin 2000.

In multi-processor testing of Gamess-US software using the previously referenced crown molecule, the cluster scales well and with an extremely attractive price/performance.

Figure 1. Beowulf and Origin Scaling



In these tests, the Beowulf performed approximately 25% faster in jobs run on one to six processors. The paper "Cluster Computing for Computational Chemistry," showed good scalability for individual GAMESS-US jobs of up to 20 processors on Beowulf clusters. When this data is put into the perspective of cost per processor, one realizes that the cluster can both scale well and perform well when compared to machines costing up to an order of magnitude more.

Conclusion:

These case studies show that the use of Linux, Open Source Software (OSS) and commodity hardware have allowed VCU to improve both its efficiency and capabilities. OSS programs such as the Apache Web Server and Cyrus have allowed VCU to provide web and email services more efficiently and with a lower cost. Open Source software provides stability, portability, support and access to source code. The performance of the Beowulf cluster has provided us with the ability to offer more computing resources for our researchers. These features allow VCU to leverage its staff knowledge and hardware dollars to better meet the needs of its users.

References:

[1] Raymond, E., *The Cathedral and the Bazaar*, 1st ed., O'Reilly and Associated, Sebastopol, CA (1999)

[2] Sterling, T., et al., "How to Build a Beowulf," Cluster Computing Conference- 1997, Emory University, Atlanta, GA (1997)

[3] Hawick, K., D. Grove, P. Coddington, and M. Buntine, "Cluster Computing for Computational Chemistry," DHPC Technical Report DHPC-073, University of Adelaide, Adelaide, South Australia (2000)