DESIGN AND CONSTRUCTION OF GENERAL PURPOSE COMPUTING RESOURCES FOR LINUX BASED COMPUTER SCIENCE EDUCATION*

Richard Sharp
Department of Mathematics, Computer Science, and Statistics
St. Lawrence University
23 Romoda Dr.
Canton, NY 13617
315 229-5345
rsharp@stlawu.edu

Ed Harcourt
Department of Mathematics, Computer Science, and Statistics
St. Lawrence University
23 Romoda Dr.
Canton, NY 13617
315 229-5444
ehar@stlawu.edu

ABSTRACT

For six years our computer science program had no dedicated computing laboratories and limited influence on the software that could be installed on university wide workstations. In the fall of 2009 we remedied this situation by teaching a course where computer science students built their own labs. We describe the design process including physical plant, hardware and software configurations of workstations and servers, and the supporting course which students took to assemble workstations and learn concurrent programming. Our result is a medium scale computing resource (58 CPUs / GPUs; 232 general purpose computing cores and 12,528 GPU cores) ideal for classroom instruction, student projects, and single CPU or grid computing.

1 INTRODUCTION AND MOTIVATION

As an undergraduate liberal arts institution, we teach an introductory level CS service course and a major that follows the traditional curriculum for a liberal arts degree in CS [1]. The CS department teaches 15 courses, 5-10 senior research projects, and 1-3 summer student fellowships per year.

Our computational needs for teaching a CS education have been refined by 14 years of combined experience with a Microsoft based computing resources in education and industry as well as 25 combined years of teaching experience. As CS educators we feel that it is best to expose as much functionality of the operating system to students as possible. Linux exposes almost every aspect of the computing process for inspection and configuration by the end user. We feel this philosophy is a superior one to teach CS because it embodies a major portion of what computer science is about: the practical implementation of the theoretical foundations of computation.

Computational Needs

Our day to day computational needs not only include a programming environment and a central file system but also the individual research requirements of the CS faculty and those of our colleagues. Such computational needs are difficult for a centralized university IT department to meet as they have campus wide goals of maintainability, reliability, and consistency in end user experiences. An advantage of using Linux for our computational needs is that we, the primary users, can configure and maintain the overall environment to suit the needs of CS education and research as opposed to meeting the needs of a general university population.

Administrative Needs

Part of the motivation for this project stemmed from six years of frustrating teaching experiences at our university involving Microsoft Windows based workstations that were centrally maintained by a university-wide Information Technology (IT) department. A CS teaching environment needs flexibility in the configuration of computing resources. We, and perhaps many readers, have experienced frustration involving the need to install software or configure OS settings for classes but being hampered by bureaucratic processes in a rigid IT service model. Additionally, some software configurations such as open source software are technically challenging to install and configure which can be difficult for an IT department to handle.

2 RELATED WORK

We are unaware of other institutions who developed and constructed general purpose computing resources for an undergraduate CS program that was built by faculty and students. However, the are many sources related to Linux supporting CS education that we discuss below.

Eastman makes a case for teaching Linux as part of the CS curriculum due to its prevalence in industry which has been turning to Linux for years [2]. Gaspar and Godwin teach OS kernel programming by "hacking" Linux through student written loadable kernel modules [3]. This approach to teaching fundamental OS concepts would be difficult using Windows. Talton and Fitzpatrick describe their experiences in teaching computer graphics using the OpenGL shading language (GLSL) in [5]. We note this work as we encountered this problem in teaching our own OpenGL course, however, we

found in a Linux based environment we were able to install OpenGL extensions, like GLSL, with the package manager.

As Yue and Ding note [6], one of the primary foundations of web development is the Linux/PHP/Apache / MySQL (LAMP) stack. Although these tools can be installed in a Windows based environment it is often non-trivial to do so, and may be difficult for a university wide IT department to maintain. On the other hand a LAMP environment is easy to install in a Linux environment as many Linux distributions come with the option to install LAMP as a default.

While there are many other examples of Linux being used in CS education, we are unaware of any work describing the construction of a general purpose Linux based computing lab built and maintained for and by faculty and students.

3 CONSTRUCATION OF COMPUTING LABORATORIES

Students and faculty built a total of 58 workstations and four severs to fill three classroom spaces. Workstation design and construction was only a part of the effort required to build our entire laboratory space. Since we were taking over abandoned biology and chemistry labs, we also accounted for physical plant issues such as demolition of lab benches and sink work, asbestos abatement, power and data network installation, furniture and layout, projector and screen installation, keyless entry systems, and others.

Workstation Configuration and Construction

Our goals for selecting hardware for workstations were: (1) Workstations should be built by students from commodity components. (2) The ability to absorb a hardware failure rate of 10% across components in 5 years. (3) Once purchased and assembled, workstations should be maintained solely by CS faculty.

Custom built desktop computers are relatively commonplace and many online dealers offer components directly to the consumer. Although there are non-trivial dependencies among the components, they are often well documented and the selection process is relatively straightforward. To handle inevitable hardware failures we purchased spare components which account for 10% of all our components purchased. As components fail, we expect to minimize the impact by swapping out components.

Our goals when considering software capabilities were: (1) Faculty should be able to install software on all workstations at any time, even during class sessions. (2) Software configurations should be maintained solely by CS faculty. (3) Students must be able to login using their existing university accounts. (4) Students must have access to their university wide student network drives. (5) Students must have access to a local CS network drive space. (6) Students must have access to shared resources such as class materials and software. (7) Workstation software configurations must be identical.

The first two goals are one of our stronger reasons for using Linux as our OS. Unfortunately this choice complicated the third goal as our university's IT infrastructure is Microsoft Windows based. Fortunately, we were able to craft a solution using Samba (a collection of programs that implement Microsoft's SMB protocol for Unix systems)

to join workstations to the university's Windows domain, winbind (which provides services to negotiate user and group information from a Windows server once joined to that domain), and configuring PAM (pluggable authentication modules for Linux which handle the authentication tasks of various Linux services.), to use winbind for user authentication.

Student access to IT served network drives were realized with Samba's client software. Local CS network drive space was implemented using a central file server we built that serves the home directories to the workstations through NFS. Shared resources are also distributed through the file server as a globally readable mount shared through NFS.

Finally, we maintain identical workstation configurations by administering all stations remotely and simultaneously using cluster SSH which allows for control of multiple remote shell windows connected to different machines through a single main console.

Our backup scheme consists of two identically configured file servers one primary and one mirror with each server also containing its own backup partition. Each server contains four 1TB hard drives configured in two partitions. Each partition consists of two drives configured as a RAID1 disk array. The first partition is the primary storage with the second partition being the backup. We use rsnapshot (www.rsnapshot.org) and rsync to maintain hourly, daily, weekly, and monthly backups.

4 PARALLEL COMPUTING COURSE

A key aspect of the implementation of our laboratories was that not only that students built the machines but also learned the foundations of parallel computing, including software that: use multiple cores on one machine through software threads and processes, manually distribute calculations to multiple machines using TCP/IP sockets, automatically distribute calculations using MPI, and execute in parallel the GPU using NVidia's CUDA interface. Applications developed in class included classic concurrent programs such as producer/consumer and readers/writers but also a chat program, and parallel integer factorization. We also used this as an opportunity to teach scripting and using ssh to distribute programs across machines.

Students were required to keep a laboratory notebook, assemble three workstations, and complete a concurrent programming project that used all local computing resources. Student projects included parallel password cracking, a parallel n-body simulation, and a parallel DNA string matching algorithm.

5 STUDENT PERCEPTIONS

During the course of the semester we had students take a survey to determine their perceptions of the importance of the OS in their productivity. Students took the survey twice, at the first and last week of the semester. Although our sample size (n=12) was too small for any statistical significance, we did note a trend toward the preference of Linux over Microsoft Windows. We describe our results below.

OS Preference Survey

Here we discuss each question on our survey and the aggregate results for the semester in question. The aggregate results of our survey are listed in Figure 1. First we present the questions, then discuss the results as aggregated by students' reported preference of OS.

Please rate your relative level of experience with the Linux OS. (1) No experience. (2) Have used a Linux live-CD. (3) I installed Linux at least once. (4) I use Linux as my primary OS. (5) I have written a large program using only Linux. *Students scored an average of 2.5 on week 1 and 4.6 on week 15.*

How do you perceive the choice of OS on your productivity when programming? (1) Does not affect my productivity. (2) I prefer a particular OS because I am not familiar enough with another to be productive with it. (3) I prefer one OS over another, but for reasons unrelated to productivity. (4) The choice of OS is important to my level of productivity. (5) The choice of OS is critical to my level of productivity. *Students scored an average of 3.2 on week 1 and 3.8 on week 15.*

Consider two job offers, one which paid $100k/yr but required you to use an OS you didn't like and another which paid ($100k-x)/yr but you were allowed to choose your own OS. What is the largest non-negative value of x such that you would take the second job?

In a future job, what OS would you prefer to work with?

Figure 1 reflects how students' perceptions and preferences of OSs changed throughout the semester. To summarize, five students preferred Linux as their primary OS at the beginning of the semester while seven did at the end. Students who preferred Linux also thought their choice of OS was more important to their productivity than those who preferred Windows. Additionally, those who preferred Linux were willing to reduce $8k more off a base salary of $100k in order to work in it than those who preferred Windows. Although our sample size is too small to determine any statistical significance, anecdotally there was a shift toward preferring Linux as a primary work OS.

Preference	***N***	**reported**	**Experience**		**Productivity**		**Monetary**	
	1	15	1	15	1	15	1	15
none	3	3	2.0	4.3	3.0	2.7	$37k	$27k
Linux	5	7	3.2	4.7	3.6	43	$10k	$18k
Windows	4	2	2.0	4.5	2.75	4.0	$20k	$10k

Figure 1 Aggregate results of survey data described in Section 5.1. The 1 and 15 under the second row indicate results taken from that particular week of the course (there are 15 weeks in total).

6 DISCUSSION

With a faculty driven effort we built, from scratch, computing laboratories with significant help from students. We feel the novelty of our project comes not from teaching students how to configure Linux or a network for a specialized course, but the

fact that students and faculty constructed, configured, and now maintain the complete computing infrastructure for all CS courses and CS faculty research. Below we briefly discuss our personal expectations, experiences, and reflections that may be useful to others who wish to implement a similar environment for a similar institution.

Damage During Construction We were concerned that student involvement in the construction of workstations could result in significant accidental damage of components. Surprisingly, we experienced only one incident of processor socket damage on a motherboard which was replaced by the manufacturer.

Conversion to Linux Based Environment Anecdotally, the conversion to an open source Linux based computing environment has made a dramatic and consequential impact to our ability to teach CS. In Linux programming languages feel "close" as though at your fingertips on the command line (Python, C,Bash, Unix pipes, etc.). In terms of student response, our shift to Linux was surprisingly smooth. The students in upper level courses took to programming at the command line quickly. In the introductory course we are able to abstract much of the command line experience away from students.

Laboratory Administration We were initially concerned that laboratory administration could turn into an overwhelming task. In practice administration has been no more than 30 minutes a week and usually involves installing software, something easily done with cluster ssh. In short, after the initial software configurations, administration has been nearly trivial involving only checking the Munin monitor on a day to day basis.

Future Work We hope to offer a workshop for faculty at similar institutions which will involve each participant building a workstation from components, and learning the basics of Linux and network administration for a medium scale teaching lab.

BIBLIOGRAPHY

[1] L. A. C. S. Consortium, A 2007 model curriculum for a liberal arts degree in computer science, *Journal on Educational Resources in Computing*, 7 (2), 2, 2007.

[2] Eastman, E. G., Exploring Linux as an operating system in the CS curriculum, *J. Comput. Small Coll.*, 21 (4), 83-89, 2006.

[3] Gaspar, A., Godwin, C., Root-kits & loadable kernel modules: exploiting the Linux kernel for fun and (educational) profit, *J. Comput. Small Coll.*, 22 (2), 244-250, 2006.

[4] Norris, J. S., Kamp, P.-H., Mission-critical development with open source software: Lessons learned, *IEEE Software*, 21 (1), 42-49, 2004.

[5] Talton, J. O., Fitzpatrick, D., Teaching graphics with the opengl shading language, *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer Science education*, 259-263, 2007.

[6] Yue, K.-B., Ding, W., Design and evolution of an undergraduate course on web application development, *ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, 22-26, 2004.