UCRL-TR-155566

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

High Performance Diskless Linux Workstation in AX-Division

E. Councell, L. Busby

September 30, 2003

UCRL-TR-155566

High Performance Diskless Linux Workstations in AX Division

E. L. Councell
L. E. Busby

Lawrence Livermore National Laboratory

ABSTRACT

AX Division has recently installed a number of diskless Linux workstations to meet the needs of its scientific staff for classified processing. Results so far are quite positive, although problems do remain. Some unusual requirements were met using a novel, but simple, design: Each diskless client has a dedicated partition on a server disk that contains a *complete* Linux distribution.

October 10, 2003

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Lawrence Livermore National Laboratory

subject: **High Performance Diskless Linux Workstations in AX Division**

date: **October 10, 2003**

from: **E. L. Councell**

L. E. Busby

ABSTRACT

AX Division has recently installed a number of diskless Linux workstations to meet the needs of its scientific staff for classified processing. Results so far are quite positive, although problems do remain. Some unusual requirements were met using a novel, but simple, design: Each diskless client has a dedicated partition on a server disk that contains a *complete* Linux distribution.

1. Introduction

AX Division is part of the Defense and Nuclear Technologies directorate at Lawrence Livermore National Laboratory. We have about 200 technical staff members, a mix of physicists, engineers, computer scientists and system support personnel, and other scientific specialities. Our broad mission of nuclear stockpile stewardship brings a strong requirement for numerical simulation, scientific visualization, and many related needs in code development.

Much of the work in AX Division is classified. Access to classified computing resources has traditionally been carried out using either an X-terminal or a desktop PC with removable storage media. Each of those approaches has significant disadvantages: X-terminals have low market appeal and are only available in a limited number of configurations. They are expensive and perform poorly compared to standard PC hardware. Placing classified information on portable computer media triggers an elaborate formal system of handling protocols, with checks, counterchecks, audits, and other safeguards. In spite of the system, some risk remains. It was our determination that a policy of following best available practices required that we attempt to *eliminate* that risk by removing classified computer media from user offices.

The introduction of diskless Linux workstations to meet our classified computing needs combines most of the strong points of the two preceding desktop devices, and avoids their most significant weaknesses.

2. Project history

The general goals of this project were laid out in an informal proposal in June, 1998. It was obvious that we could combine off-the-shelf PC hardware with accelerated 3D graphics and 100Mbit (or better) switched networks to the desktop. Linux was a low-cost operating system and development environment, otherwise compatible with our major in-house software applications. This idea took hold as the ‘‘Next generation designer workstation’’ project, and several proof-of-principle machines, software systems, and small standalone networks were constructed over the next three years. The project accelerated in the spring of 2001, when it became clear that a network merger would force the replacement or reconfiguration of several dozen desktop machines by the summer of 2002. Initial hardware procurement began in summer, 2001. The software configuration was designed and assembled during that spring and summer, and our initial group of 8 machines was fielded in the fall of 2001.

We are currently running about 75 clients booting from 7 servers, with plans to expand this to about 100 clients and 10 servers over the next 12 months.

3. Related work

Diskless workstations date back at least 15−20 years, to very shortly after the time that NFS (Network File System) was introduced by Sun Microsystems. There are many preceding experiments and implementations specifically developed for Linux.

The Linux Terminal Server Project (*ltsp.org*) is the most active and comprehensive diskless Linux project of which we are currently aware. The LTSP system runs on a variety of Linux distributions in several configurations and combinations. It might be more accurate to use the term *thin client* for the primary configuration supported by LTSP. In this form, the client runs a kernel with a small root filesystem in ram disk. The only significant application running directly on the client is the X server. All other applications run on the LTSP boot server, or on other servers according to the local network needs and resources. LTSP also can be configured to support additional applications running on the local diskless client.

There are many sites (dozens or hundreds) running LTSP in some form; it is hard to generalize about their characteristics and requirements. LTSP definitely is an excellent choice for users who have older, slower, or smaller hardware. It centralizes administration on the boot server and achieves a high level of sharing for disk resources among the clients. Using LTSP, a less powerful client desktop can continue to provide speedy and reliable access to computing resources across the network. This is a good model for many organizations, even if their desktop machines are not older or slower.

There are several Linux HOWTO's and mini-HOWTO's available to help you roll your own diskless configuration. See the *Resources* section of this document for some references. These documents may help a motivated practitioner construct a one-off solution for a particular need, but generally fall short of the community of resources provided by LTSP. In each of them, the diskless clients share one or more filesystems with each other or with the boot server itself over the network.

There are other ways to get a running Linux system in a machine that has no hard drive, such as booting from a CD-ROM distribution. One example is the Knoppix Linux distribution (see *Resources*). Such a system could boot from CD-ROM on an otherwise diskless workstation, and mount user data using NFS from other servers. This approach does away with centralized boot servers, at some cost in performance and effort to produce bootable CD-ROM distributions. Such CD-ROM's can vary in complexity from a simple network boot loader (which still requires a boot server for other resources) up to a full Linux distribution customized for a given user on a particular machine.

Another well documented diskless Linux system was constructed by Jamie Zawinski. This was a set of ‘‘kiosk’’ systems that provide internet access to club customers. It is distinct from LTSP in that most application software does run on the local host. Like LTSP, however, it is built around a customized filesystem, many parts of which are shared among all clients. This application has interesting security and system recovery requirements because of its operating environment in a nightclub, with anonymous users.

4. Goals and constraints

We had a somewhat different set of goals and constraints than most or all the work previously cited. High performance, simplicity in construction, and ease of reconstruction are very important to our organization. We were willing to invest in hardware to help meet those goals, (although our hardware costs have not been excessive.) The following list of requirements came clearly into focus about half-way through the design process:

1. The system should be capable of running most software on the local client, including significant numerical simulations and graphic post-processing;
2. It should be possible to upgrade an entire software distribution on a per-client basis;
3. It should be possible to upgrade just the system kernel, or any other set of software packages, per-client;
4. The system has to run on a variety of hardware. Although we generally control our purchases, upgrade cycles and individual needs cause quite a bit of variation, especially as time goes by. For example, we expect to upgrade video adaptors with greater frequency than the rest of the chassis, to meet new demands and capabilities in this important area.

5. To support various video adaptors, we therefore require the ability to modify or update the X server, or graphics drivers, per-client;

6. The system should allow the management of any per-client software customizations using the standard (RPM-based, in our case) tools of the system administration team.

All the above requirements are fairly easy to meet using standard PC hardware running Linux, *if* you have a local disk. They are much harder to meet if you are constrained to diskless operation, using any of the architectures discussed earlier.

As noted, previous diskless systems are organized around the idea of sharing one (disk) copy of some data among as many clients as feasible. This is reasonable, if you are limited by disk space or by the ability to manage the data on the disk. We built our system around two contrary observations:

1. Disk space is *not* presently a constraint, for many practical purposes;

2. Software management tools − *rpm* and its associated database − make it reasonable to create, maintain, and control multiple copies of a Linux distribution that vary in an arbitrary (typically small) number of files.

One other pre-existing feature of our network environment increases the simplicity and robustness of the diskless architecture: All user data is made available from NFS file servers separate from our Linux boot machines. Home directories, */usr/local/*, other filesystems dedicated to application development, and a Citrix™ (Windows) server are all accessed from separate hosts, independent of our Linux diskless system.

5. *System description*

5.1 *Hardware*

For the first 6 months, we booted clients from Dell model 1550 servers. Those have been replaced with model 1650 machines that cost about $5000 each, configured as follows:

Dual Pentium III 1.2GHz processors
2 Gigabytes of RAM
Three 36 or 73GB SCSI disk drives
Intel Pro 1000 network adaptor
Redundant AC power supplies
PERC3-D1 raid controller installed, but unused so far.

Our diskless clients are a mix of Dell Optiplex and Precision model machines. They currently cost about $2500 apiece configured roughly as follows:

3.06GHz Pentium 4 processor
1 Gigabyte non-ECC Rambus memory
nVidia Quadro4 900XGL 128MB video adaptor
Intel Pro 1000 integrated network adaptor
20 inch color flat panel monitor.

In addition, we support a few high-end customers with dual-processor SMP diskless workstations and 2GB memory.

5.2 *Software*

We run Redhat Linux on both clients and servers, and are in the process of upgrading from version 7 to 9. The only non-standard software on boot servers is the DHCP server, for which we substituted the reference implementation available from the Internet Software Consortium (see *Resources*). Due to past problems with the stock *tftpd* server, we have continued to use the version from Redhat 6.2. We configure our own kernels and construct network boot ROM drivers using the tools available from the Etherboot project.

5.3 *Operation*

On the boot server, the DHCP configuration file contains essentially all the data needed to define the clients supported by that machine. DHCP assigns static IP addresses per machine and specifies the location of a ROM driver, kernel, and root filesystem for each client. The number of NFS daemons is 32 instead of the default 8.

There are 3 disks in each server, one dedicated to the server itself. The other two are divided up to serve multiple clients: We started out cautiously assigning 4 clients per server disk, then went to 8 as the load still seemed low, and now are benchmarking 10 clients per 73GB SCSI disk in the latest incarnation.

Client hardware is generally received ready-to-go from Dell, save for resetting the BIOS boot sequence to enable network booting. Machines that are slated to dual boot (a handful) are fitted with a removable sled drive enclosure for the additional (non-Linux) operating system. For client software, each disk image is cloned from that for a similar machine, with a handful of customizations required to identify the new host.

If a significantly different client disk image is required, say to support new hardware, it is easiest to build the distribution on a machine with a local disk, then copy the entire disk to an appropriate client slot on a server.

It is important to note (we were surprised) that our client disk images on the boot server are almost completely stock Redhat distributions, in every detail of the file system organization and installed software packages. The *only* difference between our client disk images and a stock distribution is the customized kernel, to support network booting. Those kernels also boot just fine from a local hard disk, so we can as a matter of course build workstations that run identically whether booted from local disk, or from a network server containing a bit-for-bit copy of that disk. The relative lack of customization in our system significantly reduces administrative effort.

Installation or updating of software to a client disk image can be done either from the server or from the client, using *rpm*. Working from the server, we use the *--root* option to relocate the package (and its database changes) to a given client image. It is easy to script this operation to update multiple clients if necessary. Rpm updates can also be performed while logged in on the diskless client, if that is convenient. A simple system looks for and installs rpm package updates automatically each time a client boots. It is quite possible to build and install a new kernel on a running diskless client, (although the consequences of a mistake may require a trip to another machine to undo, of course.)

Booting a diskless client is a two-stage process for our system. The network interface card (NIC) shipped from Dell contains netboot code conforming to the Intel *Pre-Boot Execution Environment*[1] (PXE). After BIOS initialization, this code sends a network request to the DHCP server. Our DHCP server returns an Etherboot net loader, which is loaded into memory, and immediately repeats the process of sending a DHCP request. At the second request, our DHCP server recognizes Etherboot, and returns a net-bootable kernel image. The kernel boots, mounts an NFS root file system, mounts user data from additional servers on the network, and proceeds through the usual *init* sequence to a login screen.

We store almost no user data on the disks devoted to booting and running Linux clients. User mail is routed to a spool file in each HOME directory, and system log files are sent to a central log server. We have a pre-existing configuration database where we note any specific modifications made to a client image. This information proves invaluable when a non-standard client moves between servers.

6. Experiences

We did not have serious doubts about whether our approach to diskless clients would *work*. However, we have been surprised and pleased, on the whole, by their performance, ease of construction and administration.

Construction of a new server is standard except for cloning client filesystems. This should be done using a client filesystem from a client that is currently powered off. Otherwise, lock files, */proc* entries and other artifacts of a live system will leave many problems.

Replacing or upgrading clients has been easier than expected. Replacing similar hardware takes only a change to the ethernet address in the server DHCP configuration file. Upgrading to newer hardware often just works: *Kudzu* has been quite reliable at detecting new devices and making the appropriate changes. We have successfully used the same image on Dell Optiplex GX260, GX300, GX400, Precision 340 and 350 hardware. Patches can be applied to clients at any time, even if the client machine itself is powered off in a locked office.

It is easy to experiment with a new operating system for a customer: Just point their client at a new image on the server. If they don't like it, reverting is just as simple. Since our boot servers are *dataless*, from the user's point of view, this really is a seamless operation.

The lack of (NFS) swap on the client is occasionally a problem. This is vexing because the user process cannot detect running out of memory − *malloc(3)* never fails on RedHat 9 (not, at least, from lack of apparent memory.) The typical failure mode we observe is a single user process dominating memory. When the operating system runs out of memory, it generally selects the 2nd largest process for destruction. This is usually the window manager or the X server itself. If swap were available, the dominant user process would swap out, slow down, and the user could notice and take steps to fix the problem. (We are aware that a kernel patch for NFS swap is available, and plan to evaluate it in the future.

1. See *http://www.intel.com/labs/manage/wfm/index.htm*.

Adding more client RAM can also be a practical solution.)

Performance overall is excellent. We are mostly interested in large data sets that reside on remote servers and networks, so bandwidth and latency are usually the bottlenecks. For the most part, our clients perform just as if they had a local disk.

Although our architecture supports per-client customization of video or network interface drivers, we have appreciated the unified driver software for our Nvidia graphics cards. It has made it easier to mix multiple cards across the set of client workstations.

The most irritating problems so far are unrelated to diskless operation: Providing a good default user environment across an organization that is customized to institutional requirements and the high expectations of our technical customers is quite hard.

There are still a few things we want, but don't have: ‘‘productivity’’ applications such as Adobe Photoshop™ and Illustrator™, and an office suite compatible with the Microsoft offerings. Our documents often include equations, figures, and diagrams. Although we might like to standardize on OpenOffice, the complexity of our documents has made it difficult to share them with other workers still using Microsoft products.

We continue to investigate the possibility of running Apple OS X in diskless fashion. Many of our staff have used Macintosh hardware for years, and would prefer that option if it were available.

7. *Resources*

1. http://www.ltsp.org/: Home site for the ‘‘Linux Terminal Server Project’’. Documentation for installing LTSP is available in several formats and languages. Chapter 6 (*Troubleshooting*), and Chapter 7 (*Kernels*) of the documentation are useful for any diskless Linux project. The troubleshooting chapter in particular is excellent, and saved us many hours of work.

   See also *pxe.howto.html*, available in the documentation section. It contains instructions for setting up a DHCP server and configuration file to support the two-stage boot process used in our system.

2. http://www.rom-o-matic.net/: A clever use of web technology to provide etherboot binary ROM loaders for almost any network interface adaptor, on demand.

3. http://etherboot.sourceforge.net/: Home for the *Etherboot* project.

4. http://www.isc.org/products/DHCP/dhcp-v3.html: The Internet software consortium version of DHCP. We require a DHCP server that supports the *vendor-class-identifier* option, and this is one that does.

5. www.tldp.org/HOWTO/Diskless-HOWTO.html: A general discussion and useful background regarding many aspects of Linux diskless systems.

6. http://www.dnalounge.com/backstage/-src/kiosk/: Discussion, instructions, and source code for the diskless system built by J. Zawinski.

7. http://www.knoppix.org/: A Linux distribution that runs entirely from CD-ROM.

8. *Author information*

Eric Councell is a system administrator and Lee Busby is a computer scientist at Lawrence Livermore National Laboratory, both matrixed to AX Division. Email them at *councell1@llnl.gov*, or *busby1@llnl.gov*, respectively.

9. *Acknowledgements*

Steven H. Langer of AX Division provided programmatic support for the project from its earliest days. His persistent focus on practical outcomes was fundamental to its success. We are grateful to the past and present managers of AX Division for taking the risk of funding this work. Langer, along with David H. Munro, also of AX Division, contributed important components to the user environment.

James McQuillan, of the LTSP project, has led a renaissance in diskless workstations and thin clients. We are grateful to him and to the many other workers who created LTSP, Etherboot, and the various pieces of GNU/Linux, who have made it such a robust and versatile system.