

# Network Booting versus hard disks: Costs and Implications

John M. Ostrowick<sup>1</sup>

Mathematical Sciences

University of the Witwatersrand,

*jon@cs.wits.ac.za*

School of Computer Science, Private Bag 3, 2050 WITS

## Abstract

In the African context, it does not necessarily make sense to spend large amounts of money on proprietary computing solutions when low cost solutions exist. This paper explores the experiences we have had in our University implementing both network-booting and hard disk-booting Linux systems. Most Linux systems boot from hard disk. Our system, one of the few in the world, boots mainly from ethernet, completely without hard disks. We are not aware of any such system of this scale; we have approximately 200 computers using this system. We discuss cost implications, maintenance issues, usability issues, and performance issues. We have found that in terms of both hardware and software costs, this is the cheapest possible solution. In terms of maintenance, the initial investment is high, but once the system is established, it is extremely quick to add another client computer. In terms of usability, the variety of Open Source software now available renders the system highly usable for both our specialised needs as well as day-to-day computing. The primary difficulty we experienced was around performance. We found that performance statistics varied greatly and were highly susceptible to the number of users on the system, the number of servers, how the server load was distributed, the software networking configuration, and the protocols used. In particular, we found that dividing the server load up, carefully, had the single biggest impact on the system performance. Another factor which severely impacted the performance was overall network load as determined by the protocols and network software configuration. A well-configured client computer in a lightly-loaded server environment could boot in about 30 seconds. A poorly-configured client in a busy environment could take half an hour. Hard disk boot time performance is typically of the order of slightly under one minute. Thus, in optimal circumstances, network booting is faster than hard disk, whereas in sub-optimal circumstances the performance is worse. We believe that the maintenance overhead and cost savings make the solution viable, but, for smaller sites, we recommend against our solution and advocate a hard disk based solution.

---

<sup>1</sup> The author wishes to acknowledge those persons who contributed to the present article; Conrad Mueller, Brynn Andrew, Scott Hazelhurst, Hilton Currie, and Adi Attar for their academic input, Alexander Holt, Susan McGlashan, Lushanta Moodley and James Ostrowick for technical assistance and the creation of the system under discussion.

## Introduction<sup>2</sup>

For many years, the various departments of Mathematical Sciences at the University of the Witwatersrand in Johannesburg<sup>3</sup> utilised a laboratory system for their undergraduate students which consisted of Microsoft DOS and Windows 3.11 clients that booted over ethernet from a Novell Netware server. What was the then Department of Computer Science, however, utilised primarily free or Open Source solutions (and Silicon Graphics IRIX systems) for their higher computing needs and research facilities. As the various budgetary concerns became more stringent, it became imperative that we find a solution which would reduce our costs. Furthermore, newer PCs had to be purchased as the existing machines were becoming redundant, and with the advent of Windows 95, it became hard to see how the Novell Netware network-booting solution would be sustainable, because of the size of Windows 95. It became clear that we would have to abandon Novell's network-booting solution, and either move to Windows 95 clients with hard disks, or some other solution. The administrator of the time, Susan McGlashan<sup>4</sup>, wished to preserve the network-booting, because, she felt, it reduced the overhead involved in maintaining the computers. This desire to avoid hard disk based solutions<sup>5</sup> and the need for a UNIX-like environment for research purposes led her to propose one of two solutions: Either thin Windows clients using a Windows NT server running the costly Citrix solution, or, some other solution using the Open Source Linux operating system, which, while costing nothing for the software, was more complex to set up. Linux was selected.

McGlashan researched the matter, and discovered that it was possible to boot Linux over a network, however, a specifically-configured bootrom<sup>6</sup> was required. She found the necessary driver software, and purchased the necessary network cards. Thus, while it was tedious having to burn the network booting software onto each bootrom on each network card, the overall benefit was that Mathematical Sciences managed to preserve the network booting environment and avoid software costs. At this time, the author suggested to the management of Mathematical Sciences that the amount of time involved in burning bootroms, sourcing compatible bootroms, physically replacing all the network cards, and so on, was scarcely worth the price difference in terms of person-hour costs. This counter-suggestion was overruled, however, because of the perceived benefit of the network-booting solution. Furthermore, the then Department of Computer Science wished to not utilise proprietary software solutions extensively in its laboratories.

---

<sup>2</sup> The following introduction is somewhat detailed so that the reader may come to understand the context of our environment, and the particular needs behind our various decisions, which determined our choice of solution.

<sup>3</sup> Mathematics, Computational and Applied Mathematics, Statistics, and Computer Science.

<sup>4</sup> Presently at the University of Toronto, Canada. Email [smcglash@utm.utoronto.ca](mailto:smcglash@utm.utoronto.ca)

<sup>5</sup> We do not presently discuss whether hard-disk-less solutions are reasonable. That discussion appears later.

<sup>6</sup> A chip on the network card which enables network booting.

After a number of years of running the ingenious solution created by McGlashan, it became apparent that the software in use was again becoming redundant. Furthermore, certain old Windows/DOS packages were still in use, and there was still one laboratory of machines booting from the original Novell server. While in the employ of the University, McGlashan had extensively investigated Windows emulator or compatibility environments, including *Wine* and *VMware*<sup>7</sup>, but no usable system could be found in which to run the old but nonetheless popular Windows/DOS packages which were still required. The subsequent administrator who succeeded McGlashan also failed to find a replacement, and it looked as if we were doomed to use the original system for the rest of time; until a disaster struck and the Novell server was destroyed by a hardware failure. We were unable to resurrect or recreate it, and we were thus forced to find a new solution. We had since been advised by Alexander Holt<sup>8</sup> of the University of Edinburgh that Intel had released a chipset for network booting, called PXE - *preboot execution environment*. Ethernet cards equipped with PXE could, when powered-up, send a DHCP request, and on receipt of the answer, use the TFTP protocol to download some code, including a Linux kernel. A solution using a customised, stripped-down RedHat Linux 8 with NFS-root<sup>9</sup> abilities, was built by James Ostrowick. The packages which required Windows were upgraded and replaced. Windows machines were installed with hard disks, but they were installed with Windows 2000 so as to minimise the ability of the non-administrative users to customise the hard disk contents. The central Linux server had the *Samba* Windows filesharing package installed, and would authenticate the Windows computers, thereby preventing casual users from using the laboratories. We thereby managed to retain the usage of the specialised Windows-based packages, but also managed to avoid a Windows server solution, and the customisation problems typical of hard disk environments.<sup>10</sup>

At this time, the *Ltsp* (*Linux terminal server project*) solution was available. We decided against it because of the scale of our operation. We were skeptical that any single server, within the price range we could afford, would be able to cope with the RAM requirements of running approximately two hundred PCs. We worked with the assumption (which we later tested), that most of the larger user-level packages such as the KDE or Gnome graphical environments do

---

<sup>7</sup> There is another package called *Bochs* which also purports to offer a compatibility environment.

<sup>8</sup> Email: [lex@cs.wits.ac.za](mailto:lex@cs.wits.ac.za)

<sup>9</sup> *NFS-root* is where the Linux/UNIX machine has its *root* (top-most) file system on the network - from an NFS server - a network filesystem server. Having an NFS-root means that your entire operating system, and everything on or below the top of the filesystem that is part of the same filesystem, comes from an NFS server. NFS is a network protocol which gives machines non-authenticated filesystem access. It is similar to Windows SMB, Netware or AppleShare. The primary difference, from an end-user perspective, is that it does not authenticate the user. For more information on NFS, see Currie, (attached), p12.

<sup>10</sup> Specific details and instructions on how to create such an environment are attached as an appendix, authored by James Ostrowick. Email: [james@guests.cs.wits.ac.za](mailto:james@guests.cs.wits.ac.za). See also <http://www.etherboot.org>

not have much re-entrant<sup>11</sup> code. Taking the assumption, therefore, that a minimal graphical environment with a few user-level programs ranges between 30 and 100 MB of non-re-entrant code, we'd need a server with between six and twenty gigabytes of RAM. It was our understanding that Linux at the time could only address up to four gigabytes. Thus we rejected *ltsp*.<sup>12</sup> Note that in *ltsp*, the server itself runs the processes belonging to each user. The client computers are true thin clients, merely acting as terminals to display what is being executed in the server's address space (RAM). All the data processing and storage is handled by the server.

“LTSP is an add-on package for Linux that allows you to connect lots of low-powered thin client terminals to a Linux server. Applications typically run on the server, and accept input and display their output on the thin client display.” - <http://www.ltsp.org><sup>13</sup>

On our solution, however, the client PC itself runs the user's processes in its own address space, even though it reads and writes data exclusively to a network drive. The server, on our solution, is primarily for data storage, not data processing. This does not mean, however, that anyone considering a network-booting solution should not consider *ltsp*, or that we ourselves rule it out. We excluded it, initially, on the *prima facie* assumption that it would not cope at a large scale. We have not experimentally proven this.

Any person or institution wishing to embark on the installation of a substantial computing environment needs to pay heed to a variety of considerations, not the least of which are *costing*, *sustainability*, *maintainability*, and *scalability*. We will now compare a hard disk solution to our existing network-booting solution, and discuss the details of the problems we encountered. We have not experimented with *ltsp*, thus we will omit further discussion of it.

## Costing and Setup Issues

We have already touched on some of the issues affecting costing. There are three areas where costing is relevant: person-hours, hardware, and software.

<sup>11</sup> We have tested this. KDE seems to not be re-entrant, and nor does OpenOffice/StarOffice, whereas, for example, another popular user program, Mozilla/Netscape, does seem to be re-entrant.

“**Re-entrant** - Used to describe code which can have multiple simultaneous, interleaved, or nested invocations which will not interfere with each other. This is important for parallel processing, recursive functions or subroutines, and interrupt handling.... It is usually easy to arrange for multiple invocations (e.g. calls to a subroutine) to share one copy of the code and any read-only data but, for the code to be re-entrant, each invocation must use its own copy of any modifiable data (or synchronised access to shared data). ...”.

<http://computing-dictionary.thefreedictionary.com/re-entrant>

<sup>12</sup> We have not since seriously re-considered it, however, we shall soon have to.

<sup>13</sup> Apparently Novell plans to use *ltsp* extensively. [http://www.ltsp.org/reaction\\_to\\_novell.html](http://www.ltsp.org/reaction_to_novell.html)

As we have mentioned in the introduction, it was in part the attraction of the low software cost of Linux that drew us to it as our solution of choice. While we have, of course, had to purchase *some* expensive commercial packages needed in our research areas, we have managed to save an estimated cost of R 1000 - R 2000 per-seat, by avoiding the option of proprietary client software. Were we to run proprietary software, we would have to not only pay for the server operating system, client operating system, and the Office suite, but we would also have to pay license fees for the Windows Server Client or Novell Netware Client packages as well. Given that we have approximately 200 PCs, we estimate that we have saved between R 200 000 and R 400 000 on software costs.

Then, considering the hardware, we obviated the costs involved in purchasing hard disks for each PC, a saving of between R 500 and R 1000 per seat at the current pricing. This in turn has saved us a further R 100 000 to R 200 000.<sup>14</sup>

The question that suggests itself at this point, however, is whether this software and hardware cost saving, which are both one-off savings, are sufficient to warrant the person-hours cost involved in setting up such an environment. It is not immediately obvious that this is the case. Firstly, there is the setup or installation cost. Does it make sense, we need ask, to save R 200 000 - R 400 000 if it takes an equivalent amount of money in person-hours to set the software up? We found that the person-hours expended, however, were *not* comparable to this figure; we conservatively estimate that the cost was around R 10 000 to R 20 000 - ie., one or two month's salary for one person. But there is a further question. Does this type of software *prevent* or *cause* future problems which cost person-hours to the company or institution? Another way to put the question is: *What is the total cost of ownership of the system (TCO)?* We believe our software choice has prevented future costs for a few years; we have avoided problems associated with proprietary systems, such as user tampering and viruses. Assuming, in addition, that the software cost savings reflected here represent the equivalent of one or two persons' annual salaries, we have effectively earned enough to employ another technician, even though we do not need one. However, were we to use a proprietary solution, it is possible that *not only* would we have to pay for the software, but we would also probably need an additional technician because of the person-hours cost involved in maintaining such a solution.

There is, however, a substantial problem of *obscurity*. Linux, and network-booting Linux in particular, is very complex and difficult to set up. It requires a technician with a high level of expertise, patience, and a large amount of time to dedicate to the task. Because of the complexity of the Linux network booting architecture, it is a very risky business to make changes to the system, or to add or remove features or packages. One has to proceed with great caution, and in the event of human (administrator) error, it takes a lot of time to set the packages up again and ensure they are functioning correctly. This is a setup issue that anyone interested in this

---

<sup>14</sup> Actually, the hard disks *were* purchased, but were deployed elsewhere or exchanged.

solution, needs to pay heed to. But one is not compelled, strictly, to utilise Linux:

“Etherboot is usually used to load Linux, FreeBSD or DOS. However the protocol and boot file formats are general, so there is no reason why it could not be used to load arbitrary images to a PC, including other OSes.” - <http://www.etherboot.org>

On the matter of the hardware expenses, there are more grounds for skepticism, however, as to whether our approach has saved money. Consider that we are saving a cost of approximately R 1-200 000 on hardware. This is counter-balanced, however, by the costs involved in our case in having to set up the specialised hardware. Furthermore, the PXE network cards cost approximately the same as a hard disk. So in losing the hard disk, and saving money, we have spent approximately the same amount of money on upgrading to, and installing, the PXE network cards, which would not have been necessary were we booting off hard disk. So on the hardware issue, it is much less obvious that we save money.

The saving, therefore, in implementing a network-booting Linux and PXE solution, comes in through the specific choice of the Open Source software, and that it boots over a network. Because we do not have to pay for the software, we save money. And because it boots over a network, the laboratory users are not able to tamper with the installed operating system; thus, we save person-hours in not having to regularly reinstall or mirror the hard disk contents from a server. But it is worth noting that this does not entirely preclude the use of disks.

“Etherboot can work with RAM disks, NFS filesystems, or even local disks, if desired. It's a component technology and can be combined with other technologies to do things the way you want.” - <http://www.etherboot.org>

## **Sustainability**

The question of sustainability, in this circumstance, is one around how easy it is to perpetuate such a system into the future. This is a question we have not answered to our own satisfaction. Part of the difficulty in answering it revolves around the continued support for the technologies in use. For example, we were not able to continue to use the original network-booting ethernet cards set up by McGlashan; we experienced problems with the newer version of Linux. Furthermore, as network speeds had increased, and the demand for bandwidth increased, we had found it necessary to replace the network cards used in the PCs *anyway* to cope with the increasing demand. This too, entailed the purchase of newer network cards, which, in turn, entailed the redundancy of the existing operating system installations. In the Mathematical Sciences at the University of the Witwatersrand, we have had three laboratory system installations<sup>15</sup> in the past six years, or an average of a new one every two years. Each time, this

<sup>15</sup> Novell and DOS (1), RedHat Linux 6 and customised bootroms (2), RedHat Linux 8 and PXE (3), and we are shortly going to be implementing a new solution.

has been because the hardware we had did not meet the specification of the software we wished to run. This is further exacerbated by the complexity of the network booting software installations. Of our laboratories in the same part of the building, only the laboratory which has hard disks dates back to the time when McGlashan began her experimental work, and is still in operation. Since the beginning of her work, all the network-booting laboratories have each been replaced or upgraded once or twice. This implies that hard disk based solutions have a longer life expectancy, and that that network-booting solutions have a more limited useful lifespan, because of software changes and incompatibilities.

## **Maintainability**

Over and above issues of compatibility, and long-term sustainability, a network-booting system has one distinct advantage over a hard disk based solution: Day-to-day maintenance. At one stage, Mathematical Sciences had a laboratory of Apple Macintosh computers. At the time, the Macintosh Operating System did not have true local access privileges or a true login and authentication system. As a result, the machines were typically littered with peoples' private documents, web downloads, games, IRC tools, streaming audio tools, and so on. The author was forced, because of this, to create login and authentication systems, but even that didn't completely solve the problem. It was still difficult to keep the machines tidy and in a working order. We have had similar experiences, even now, with our Windows 2000 laboratory. Despite the presence of a true authentication and access privilege system in Windows 2000, we have had maintainability problems such as these. Users are also able to infect the computers with viruses. And the systems nonetheless still do spontaneously seem to fail or acquire errors, and have to be reinstalled. Although we have an elegant solution to this (the machines can be induced to mirror their drive from the server, thus overwriting and cleaning it) - the computers still have to be manually reconfigured after mirroring, to have the correct IP address, administrator password, domain server, and so on. It typically takes half an hour to an hour to restore just one PC with a hard disk. We also sometimes encounter similar problems with our Linux systems which have hard disks.

By comparison, with our network-booting Linux clients, it takes approximately one minute or less to recreate a machine which has become corrupt. Because the data representing the operating system is stored on the server, a user of a client machine is unable to modify the client machine's contents without gaining administrative permissions on the server first.<sup>16</sup> Even if, on the rare occasion where some damage has occurred, it is simply a matter of running a script (with some parameters) - to restore the particular machine to full normal functionality. Furthermore, the amount of data actually required to boot Linux is quite small - of the order of 50 MB per PC. For 200 PCs this entails that a total of 10 GB of storage is used - as compared to the standard RedHat 8 installation on a hard disk which requires 2 GB or so *per* PC. Part of

---

<sup>16</sup> This is not strictly true. In running Linux, `/var` and `/tmp` do always get modified, even in our design.

the reason that the data uses up as much as 50 MB per PC is that the top-level libraries cannot be used simultaneously by more than one PC. Peculiar behaviour resulted when we attempted this. But if it were possible, we'd need even less than 50 MB per PC. But as far as we can tell, this represents the approximate minimum per-client data size. The `/usr` partition, however, is shared directly from the server - meaning that the bulk of the OS and software is shared over all machines, and to install a package requires only that we install it once - on the server itself - since on Linux, newly-installed packages typically are installed into `/usr`. On a hard disk based solution, we'd have to use `rdist` or `scp -r` to transfer new software to all machines. Or even worse; we'd have to go to each machine, one at a time, and do the installation manually on each machine. This is, in our case, a 200-fold time-cost saving. So whereas the server is hard to maintain (as we mentioned earlier), the saving on maintaining the clients more than compensates for that.

There is a further advantage to using network-booting clients. Since the operating system for each machine is almost identical (barring certain hardware differences which need to be accounted for), it is extremely fast to add a new machine to the pool. This brings us to the question of scalability.<sup>17</sup>

## Scalability

One of the important things in considering a large site installation of PCs is how scalable your server/client architecture is. There are two issues here, over and above maintaining or sustaining the pool of machines: Firstly, how easy or time-consuming it is to add another machine to the pool. Secondly, there is a question of what impact adding a client has, to the pool. A poor solution is one in which adding a client to the pool of clients, will slow the system down, and/or be hard *to* add to the pool in the first place. In this section, we consider the scalability of our own network-booting solution as compared against hard disk based solutions.

### *Ease of addition*

In the case of a solution involving hard disks, it is quite tedious, generally, to add another

<sup>17</sup> The University of Edinburgh runs a package of their own authoring called *lcfg* - Local Configuration System. This package works similarly to the model we use. When a computer boots up, it uses PXE to acquire its operating system, with the correct unique hardware configurations specified in XML format on the server. Each particular machine is preconfigured and preinstalled by the server. When the client machine boots for the first time, it contacts the server, acquires its operating system, and installs onto its hard disk, obviating the tedious, multi-CD Linux installation process.

“New machines can be installed automatically by creating the appropriate source files and booting from a network filesystem using a boot floppy, or PXE. The required package set is installed according to the profile, and the components configure the system in exactly the same way as a fully-installed machine.” - <http://www.lcfg.org/faq>

machine to the pool of clients. The machine has to be set up - which generally involves having to answer dozens of questions and give the machine many CDs. We experience precisely this tedium when we have to deal with the PCs in our Windows 2000 lab. If one of these particular machines becomes corrupted, or we want to add a new machine, we have to either install a CD-ROM drive or persuade the machine to download its OS from the server. Either way, it takes a good hour or more of time and effort before the machine is usable. By comparison, in our Linux network-booting lab, it is very easy to add a new machine or correct it. We need ensure only that we use the same standard cards (network and graphics) - and then we run a script which creates the necessary OS entry, DHCP entry, etc., for the machine. Between placing the hardware-complete PC on the desk, and being able to login, is a matter of about two minutes. Our solution therefore scales very well for large organisations where ease of addition is a prerequisite. Even the *lcfg* solution is slower than this. That solution, however, does not experience some of the other scalability problems we experience.

“Compared to booting from solid-state devices, e.g. Flash disks, Etherboot has the advantage of centralising software administration, the tradeoff being the dependence on a server. This can be partly alleviated by providing redundant servers.” - <http://www.etherboot.org>

As we can see, while ethernet bootup does save on some administrative overheads, there are some server-related problems which impact performance and scalability.

### *Performance impact*

A poorly scalable solution, as we have said, is one which takes a performance hit when a new client is added to the pool. In our case, unfortunately, our solution does not scale well in this regard. Whereas a client PC using a hard disk will generally perform quite consistently, our client machines' performance depends very heavily on a number of factors. The poor performance scalability of our solution seems to be due to a wide variety of complex factors, namely:

- i. The OS used
- ii. The number of client machines
- iii. The network protocols in use, packet sizes, and MTU
- iv. The network topology
- v. Thrashing on the server
- vi. The number of servers
- vii. The division of labour between the servers

This list is not exhaustive. It represents a list of our prime suspects as it presently stands. The reader should be perfectly clear on this matter. Our solution works well generally, and is generally easy to maintain, however, it has noticeable performance peculiarities, which detract

from its overall desirability. The remainder of this paper discusses these performance-detracting factors and our attempts to eliminate them.

*i. The OS used*

We have noticed that the performance under RedHat Linux version 6 was better than under RedHat 8. This is despite having upgraded all the PCs - in terms of their RAM, processor speed, and ethernet card speed. We currently have the server on a gigabit line, and the clients on 100 Mbps lines. Previously, the server was on 100 Mbps, and the clients on 10 Mbps. Yet the performance is worse. Since the hardware is better, and the performance worse, we assume that it may have something to do with the newer OS. We have not attempted to replace the server or client OS to ascertain whether in fact it is a problem *with* RedHat 8, because we cannot afford the downtime. Part of the performance issue, however, may relate to what is called in UNIX/Linux, the “run level”. Under RedHat 6, set up by McGlashan, we used “run level 3”. This is where the client machine boots to a commandline, a text interface. We observed, however, that most users, on logging in, would immediately run the graphical interface. We felt that since the users would inevitably do this (for web browsing and graphical wordprocessing), that we may as well set the PCs up to do this by default. So under RedHat 8, we set the machines to “run level 5”, which switches to the graphical interface after booting the basic core operating system. The problem this caused, of course, is that the user cannot use the machine until the graphical interface appears, and, they also assume that they *have to* use the graphical interface. This means, on a laboratory setup of 200 PCs, that when the system is in full use, approximately 200 times the copy of the whole operating system, including the full logged-in graphical interface, has to be transferred over the network from the server. This, unsurprisingly, places a heavy load on the system, to the detriment of performance.

Assuming the basic commandline OS is 30 MB, and the GUI, logged in, is a further 70, that means that in a typical laboratory boot up and login, 20 GB is transferred from the server. Given that it is on a gigabit ethernet link, which can transfer *in theory* over 100 MB/sec, the whole laboratory system should be logged in, in a best-case scenario, within 200 seconds (3m 20s). Yet in a hard disk based solution, the best-case scenario would be about half that. So in theory, a full network boot at full speed would still be slower than hard disk, in our circumstances, because of the number of client PCs involved. In practice, however, network booting large numbers of machines is far worse than this. Firstly, the server does not deliver anywhere near its full performance on the ethernet card. We believe the OS does not properly support it. Changing the driver software does not seem to help. In practice, the server’s ethernet card seems to perform only in the region of a 100 Mbps card - about 12 MB/sec. Thus we see, in actuality, that it takes over half an hour for the whole laboratory system to boot and log in. For this reason, we generally do not allow users to reboot their computers. In practice, it is not unusable, because the users do not login simultaneously, so each client generally only has to share the server response with a few other machines. But the problem is real, and it is particularly

noticeable in the case of a power failure: When all the machines come back online simultaneously, the whole system does become unusable for almost an hour.

## *ii. The number of client machines*

We have noticed that the number of client machines attempting to access the server is one primary cause of the slowdown. If one client machine is added or booted, it takes about 30 seconds to reach the login prompt. This is superior performance to hard disk; the same OS installation on hard disk always takes about a minute.<sup>18</sup>

“Etherboot can boot computers faster than from a disk because there are no delays in spinning up disks, etc. A moment’s calculation will show that even with a 10Mbit Ethernet, sending a 500kB kernel will take only a couple of seconds typically. With 100Mbit Ethernet it gets even better.” - <http://www.etherboot.org>

However, when there are more than two client machines, it takes slightly more than a minute.

The point to realise is that although ethernet can be faster than hard disk, in practice, *all the client machines are contending for data from a single ethernet card and a single hard disk* - that of the fileserver.

So effectively, when ten machines boot simultaneously, the fileserver’s ethernet card, even though it is a gigabit card, will respond to each, in the best case, as if it were merely a 100 Mbit card. Whilst this would still give good performance at a delivery rate of about 12 MB/sec to one PC, it is clear that additional machines booting will dramatically reduce the throughput.

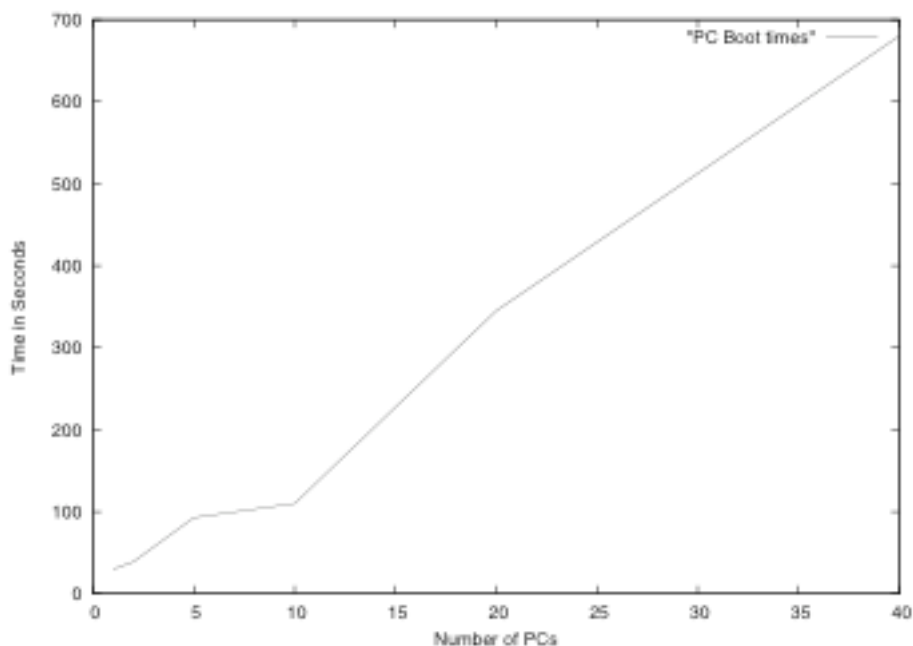
In reality, however, the card only seems to perform at 100 Mbps. We have tested this and found that the card does not perform at its advertised speed. We believe this is a software issue. Thus, since the card is only performing in the 100 Mbps region, when there are 100 machines, the server at best only gives one percent of its performance to each client machine. And when all 200 PCs boot, each can at best only have the benefit of 0.5% of the server’s network data delivery capacity. Even if the server were running at full gigabit speed, this would entail a best-case of a per-client throughput of 5 Mbps - which is substantially slower than hard disk.

The following table and graph represent some empirical measurements we took of the change in boot times against the number of machines.

---

<sup>18</sup> To be fair, our network booting PCs have had their `/etc/rc.d` scripts drastically reduced to speed up the boot time even more.

Number of PCs	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average	
1	35	32	27	27	30	30	
2	35	35	45	45	35	39	
5	110	90	95	70	100	93	
10	125	90	105	100	130	110	
<b>Trial with 20 PCs</b>							
Time:	1:20	2:00	3:30	4:00	4:30	5:00	5:45
Avg No. Booted	2	3.5	6.5	10	14	16.5	20
<i>Took 345 sec to boot all 20 PCs</i>							
<b>Trial with 40 PCs</b>							
Time:	1:20	10:30	11:20				
Avg No. Booted	1	38	40				
<i>Took 680 sec to boot all 40 PCs</i>							



The graph is approximately linear - ie., the number of machines is directly proportional to the time taken. We can therefore predict that two laboratories like this one would take 20 minutes, and four laboratories would take approximately 40 minutes. The low time taken for ten machines to boot simultaneously may be an outlier; a side-effect of the method used to boot those ten; they were not all booted really simultaneously, but rather in series, so as a result, some

finished booting sooner than others, relieving the network congestion and allowing a faster boot time for the remaining machines, than if they had actually been booted simultaneously. However, in the case of 20 and 40 PCs, even though they were booted in series, there were so many booting simultaneously that the network was thoroughly congested during the boot up of the majority of machines. In the case of the lower numbers of PCs (ie., when we booted 1, 2 or 5) - they were booted almost exactly simultaneously, so their figures give an accurate indication of the impact their network activities had on each other.

### *iii. The network protocols in use, packet sizes, and MTU sizes*

We discovered that a factor which substantially impacts the performance of the laboratory system is the kind of networking being done, viz., what protocol, and certain details about the packet sizes. The first consideration is whether to use NFS/UDP or NFS/TCP. The former has the advantage in that it has little processing overhead, and thus experiences lower latency<sup>19</sup>. The disadvantage, however, is that since no error-correction is undertaken, packet retransmissions may occur, which may increase the unnecessary amount of data on the network. NFS/TCP, while providing a measure of error correction, has the opposite problem of processing overheads involved in dealing with the TCP headers. If large portions of data are being transmitted, clearly, TCP is more reliable, and the sacrifice involved in the header processing is minimal. However, if many small portions of data are being sent (as is often the case with small user files), the overhead involved in the TCP header processing may render the advantage of the reliability, somewhat irrelevant in the face of the larger problem. (Currie, p12 et seq.).

Hilton Currie<sup>20</sup> experimented with using NFS over TCP/IP.<sup>21</sup> The server and client machines had to have specially-enabled TCP/IP NFS kernels compiled for them. Currie's findings were that NFS over IP offered no particular advantage (p15). He found that the 8k block size, using the UDP protocol, yielded the optimal performance (p23). Because of the computational overheads involved in TCP/IP, it was found that NFS over IP exacerbated the problem we were experiencing. Currie suggests however that the TCP implementation of NFS may be preferable

<sup>19</sup> “**latency - 1.** The time it takes for a packet to cross a network connection, from sender to receiver.

**2.** The period of time that a frame is held by a network device before it is forwarded.

Two of the most important parameters of a communications channel are its latency, which should be low, and its bandwidth, which should be high. Latency is particularly important for a synchronous protocol where each packet must be acknowledged before the next can be transmitted.”

<http://computing-dictionary.thefreedictionary.com/latency>

<sup>20</sup> Unpublished research paper, 2003, attached as an appendix.

<sup>21</sup> At no stage does NFS request a password. NFS by default makes use of the UDP protocol, which does not have any provisions for data reliability. TCP/IP, however, does. TCP/IP is more reliable over long distances and poor connections, or under high network congestion. See Currie (attached), p11.

on networks with higher packet losses, eg., the Internet itself (p25). While Currie's results are correct, one may dispute his conclusion (p26) that NFS is not responsible for the slow performance of the network. The TFTP phase of the boot sequence, using TCP/IP, is reasonably fast, no matter what the server load. So the conclusion that the slowness has *something* to do with NFS is hard to avoid. Currie also suggests that client caching may improve the performance. This however, is already implemented to a certain extent, using the NFS update time value. Setting this value causes the client to only send updates at the specified interval. And since data on the client, is not sent constantly, it must be the case that it is cached by the client until it the update time is reached. Similarly, Linux does, by its nature, cache the data that was recently in RAM. Lastly, Currie gestures towards increasing the number of threads available on the server to handle NFS requests. This was also taken into consideration by the author; hence the generous number of instances of nfs server processes which run in the server address space.<sup>22</sup> Yet despite taking these measures, the problem persists.

Three factors which this author experimented with, however, *did* seem to have some positive effects, further improving on the UDP/NFS performance:

- a) the ethernet MTU (maximum transmission unit) size
- b) the NFS packet size<sup>23</sup>
- c) the NFS update time value (`actimeo`)

In the case of (a), we found that by default it is set to 1500 bytes. If we substantially increased this, eg., to 8k, it drastically improved the performance of a single client machine. Eg., in one test, a machine booted in approximately ten seconds (as opposed to 30 or so). However, when the MTU reached about 16k, it seemed to no longer make a difference. We also noticed that this setting had a dramatic effect on other machines: they would all suffer slowdowns to pay for the speed increase of the altered machine. The effect was that, with a group of such machines, if they were booted more-or-less simultaneously, they would *queue*. One machine would boot, then the next, then the next. We felt that this behaviour was undesirable.

Regarding (b), the NFS packet size, this can evidently only be specified in the client's `/etc/fstab` file entry for the NFS mount in question. All the documentation on the Internet suggested the optimal size was 8k. There was a noticeable performance improvement on following that advice, as thoroughly demonstrated by Currie.

In the case of (c), the NFS update time value (which appears as `actimeo` in `/etc/fstab`); this, apparently, is significant but we did not find a substantial performance difference by modifying it. We set it to zero seconds for the shared mail spool directory (so that the user would experience no delays in being notified of the arrival of new mail), but on the user home

---

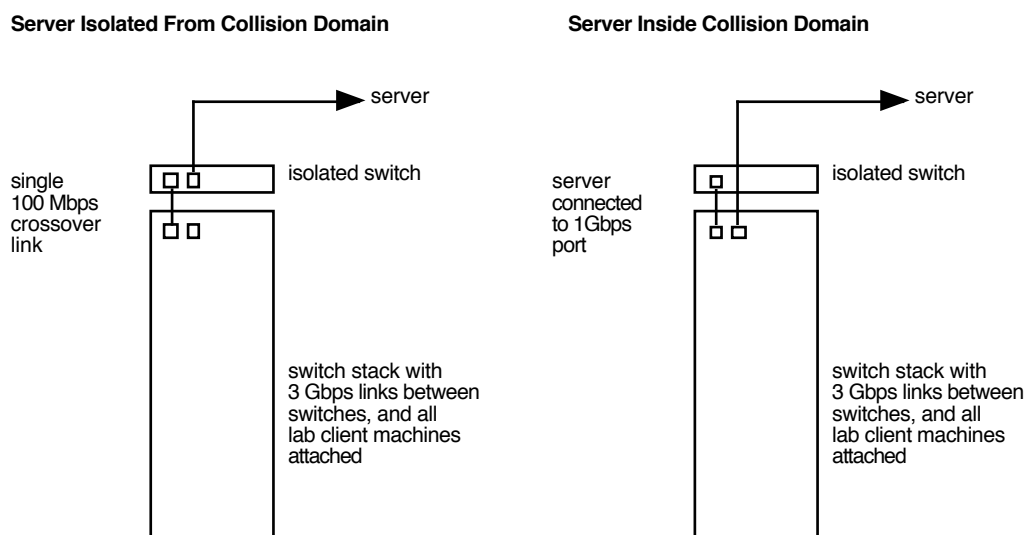
<sup>22</sup> `actimeo` and the number of nfs server processes is discussed shortly.

<sup>23</sup> Currie experimented with this and found, as mentioned above, that 8k is the optimal size.

directories, where the users store their documents and data, we left the default. On parts of the filesystem which seldom are modified, however, we set the update delay to several minutes. Our reasoning was that if we know a certain part of the filesystem would not experience changes often, then it is pointless for the client machines to poll the server often for queries as to whether there are changes to any particular filesystem. We felt that even if it was a theoretical improvement, it was at least the right thing to do, to try reduce the amount of NFS traffic on the LAN. A scan which we performed using *ethereal* (amongst other tools) showed the traffic to consist of 95% NFS/UDP traffic, with extraordinarily small packet sizes in the 150 byte size region. We assume that these are NFS update request packets. We therefore attempted to reduce excessive polling from the client machines by manipulating the `actimeo` entry.<sup>24</sup> We did not experiment further to see if this reduced the number of small-sized UDP traffic packets.

#### iv. The network topology

Whereas in theory this should make a difference to the performance of a network, we did not, in practice, find it to be so. We structured the cabling and switches so that the server would be on a gigabit port on one of the switches, linked via cascade cable (3 Gbps) to the other switches. We attempted to create a spider-diagram topology or radial topology; ie., the client machines all have, as far as possible, a direct line to the server. We avoided placing the server on any switch which was isolated from the laboratory's stack with a lower-bandwidth line. But either way, moving the server to or from such isolation did not seem to make any difference. Whether the server was so isolated or not, the contention for its network card did not seem related to the network topology. To diagramatise it, we attempted the following two arrangements:



<sup>24</sup> The following entry is an example of the syntax used in the `/etc/fstab` file:

```
# server:      mount      target      protocol    options
boot:         /home      /home      nfs         intr,actimeo=3
```

The purpose of this experiment was to ascertain whether being inside the collision domain of the laboratory stack was negatively impacting the server's performance. We found that isolating it over a single 100 Mbps line made no substantial difference. This could be because of a combination of a variety of possible reasons:

- a) Despite the bottleneck imposed on it by the 100 Mbps line, the server performed better because it was no longer inside the collision domain
- b) The server in the collision domain should perform better in theory since it was on a 1 Gbps link, but because of the large numbers of ethernet traffic collisions in the stack it was not practically, *actually* able to perform at its theoretical maximum speed
- c) The single gigabit port module that the server was plugged into, acted like an isolating switch or line, thus enforcing queueing behaviour anyway, just like plugging it into the 100 Mbps line did

However, the switches we have in service at the moment are very new and should not allow large amounts of traffic collisions. They are intelligent devices, not mere hubs, so there should be no reason for the server to underperform inside the collision domain. We could not escape the conclusion that the server was simply not delivering gigabit performance for some software or disk-related reason, perhaps:

- d) The server ethernet card *driver software* can only really cope with 100 Mbps
- e) The server hard disk can only deliver at 100 Mbps physically; contention for files on the server hard disk which is causing the server's ethernet card to effectively only deliver at best what the hard disk is delivering
- f) The kernel can not cope with the number of queries or kernel tasks.

#### v. Thrashing on the server<sup>25</sup>

---

<sup>25</sup> **“thrash** - To move wildly or violently, without accomplishing anything useful. Paging or swapping systems that are overloaded waste most of their time moving data into and out of core (rather than performing useful computation) and are therefore said to thrash. Thrashing can also occur in a cache due to cache conflict or in a multiprocessor (see ping-pong).

Someone who keeps changing his mind (especially about what to work on next) is said to be thrashing. A person frantically trying to execute too many tasks at once (and not spending enough time on any single task) may also be described as thrashing.

Compare multitask.” - <http://computing-dictionary.thefreedictionary.com/thrash>

It is apparent that our boot server is thrashing; ie., switching between tasks but not actually handling them. This author holds that this may be the root of the problem, and short of reprogramming the Linux kernel, there is no simple software solution. The reason for suggesting that this may be the problem, is as follows. On UNIX/Linux, when `nfsd` is started up, one specifies to the OS how many `nfsd` server processes are required. The default that most administrators use, is four. It was apparent very soon into the creation of our system, however, that four `nfsd` processes were not enough to cope with the NFS demands of 200 PCs. We therefore increased the number to fifty, approximately the number of machines per laboratory room.<sup>26</sup> We assumed that each of the fifty `nfsd` processes would then service the queries of one machine, and that there would therefore be no machine in the laboratory which would have to wait for its service. However, we found that doing this did not make much difference to the performance of the system. Instead, the server process load went up to approximately 50 (as ascertained by the program `top`). This meant, in effect, that the server was not actually handling the processing requests of the individual NFS daemons, but to a certain extent seemed processing them serially rather than in parallel. While most UNIX/Linux system administrators would be reaching for the power button on their servers at a load of 50 - the server, we found, was idling. It showed up at about 95% CPU idle with no obviously excessive disk I/O. In other words, it was queueing up the NFS jobs to a load of fifty still to do, but not actually doing them. We could not think of a way to solve this problem, and reducing the number of NFS server processes did not seem to help, either. Furthermore, the problem was not related to TFTP boot up. The TFTP phase of the delivery of the OS kernel always proceeds promptly and efficiently. It is after the kernel is loaded, when the client machine reaches the "Welcome to Red Hat Linux" point, that the client slows down drastically - ie., once it reaches the point of NFS-root, and loading data over NFS.<sup>27</sup>

#### *vi. The number of servers*

This was one of the most significant factors in improving the performance and scalability of our system. We found that there was a limit on how *slowly* a single room of fifty PCs would boot. We therefore reasoned that we could reduce the "worst-case scenario" by having more than one boot server. In the worst case, we had four rooms of fifty PCs booting from one server. This would take about half an hour to an hour. We have almost halved the time to boot the whole system of all the client machines by adding another boot server.

#### *vii. The division of labour between the servers*

---

<sup>26</sup> The server has 1 GB of RAM so it easily manages this number of processes.

<sup>27</sup> We recognise that alternatives to NFS, such as Coda, exist. However, for the purpose of the creation of a networked root filesystem, the only supported filesystem is NFS.

The single biggest performance improvement, we found, was to pay close attention to which drives on the server, the clients were contending for. There are, in our system design, two performance problems. The most conspicuous one is in the boot sequence. If the laboratories are busy, the bootup is slow. But there is a further performance issue which is much easier to understand, namely, the performance lag experienced after bootup. Obviously, if the LAN itself is flooded with traffic in a busy laboratory circumstance, the performance will be poor - not only in bootup but during general usage; because, recall, on our system being described here, all files reside on the server and are delivered via NFS. We found that separating the bootup function from the user data repository function, drastically improved performance all round. The reason is obvious: those users who were busy booting up would not have to compete with those users who were already logged in, for access to the server's drives. Bootup would not have to compete with application launching, file saving, etc. Similarly, we found that dividing up the functions of the drives inside the servers made a substantial difference as well. The reason for that is also obvious. If user data is stored on one particular drive, and application software is stored on another, then there is no contention for one particular drive when loading an application; since no user data is flooding in and out of the drive at the same time. We found that partitioning a single disk helped slightly, but not enough. It is far better to separate the drives, or even servers, into functions, namely, one for bootup (at least), one for application software, and one for user data. No single change to our architecture or design of our laboratory system made as substantial an improvement as this.

## **Conclusion**

In optimal circumstances, network booting is faster than hard disk, whereas in sub-optimal circumstances the performance is worse. The questions one need ask are:

- a) How important is performance?
- b) How important is scalability?
- c) How important is ease of day-to-day maintenance?
- d) How important is cost?
- e) How important is ease of installation or setup?

On the matter of performance, our system generally performs slower than hard disk systems, because of its size. And thus it is precisely large organisations such as ours, which would want the ease of maintenance offered by network booting, who would suffer poor performance. So the issue amounts to the trade-off between ease of maintenance and speed.

The solution used by Mathematical Sciences at the University of the Witwatersrand, has some advantages and disadvantages, as we have seen. It performs well, sometimes, but performs poorly when busy. It is very scalable, but scaling up usually involves buying more servers. We believe that the maintenance overhead and cost savings make the solution viable. The saving on

not having to use proprietary software is a real saving, and an important saving in Africa, where we can scarcely afford to give away money to multinational corporations, and waste time on viruses. But this is counter-balanced by the need for highly-skilled and competent technicians, as Linux is a complex operating system to administer, and this solution we use is very complex to set up. Furthermore, anyone wishing to install such a system need also pay heed to the various ways mentioned above to optimise the performance of NFS, since by default, NFS is relatively slow, but, since Linux requires NFS-root to boot up, NFS is nonetheless required at least for the boot sequence phase.

For smaller sites, therefore, we recommend against our solution and advocate a hard disk based solution (or *ltsp*), as the ease of scaling up is less important to a smaller organisation, than ease of daily maintenance. Similarly, in environments where performance is an issue, the question becomes one of ease of maintenance and whether it is worth sacrificing speed for that sake. For larger sites, the issue is a matter of how many person-hours are expended maintaining a conventional system.

## **Future Research**

We believe the primary cause of the generally slow performance is the amount of data being contended for by the large number of clients, coming from one boot server. We could investigate whether the `hdparm` tool might have a positive impact on those servers which utilise IDE/ATAPI drives to see if this will improve the server performance. Superficial preliminary testing indicates that the server hard disk is substantially under-performing - below 100 Mbit speeds - compared to the server used previously, which had delivered appropriate performance. Also, further testing indicated that the gigabit card was indeed performing below 100 Mbit speed as well - at best, in the region of 80 Mbit. Thus we may need to try a different card. We could also try different ethernet card driver software and test the server gigabit card more thoroughly to ascertain why it is underperforming. We also plan to investigate a variety of other options, including *ltsp*, newer versions of Linux, adding further servers, and testing to see whether the `actimeo` entry makes a difference to the amount of UDP traffic on the LAN. Finally, we could investigate whether using alternative filesystems after bootup make a difference.<sup>28</sup>

---

<sup>28</sup> Amdahl's law may be useful in calculating the performance enhancement achieved through a feature modification. It is calculated by dividing the execution time of some task which is influenced by the feature modification, where the modification is not present, by the time taken for the task to be performed when the feature modification *is* present. (Hennessy, J. L., Patterson, D. A. p40).

## References

<http://www.etherboot.org>

<http://www.ltsp.org>

<http://www.lcfg.org>

<http://computing-dictionary.thefreedictionary.com>

**Currie, H.** (2003). *Performance evaluation of NFS/TCP versus NFS/UDP on the Mathematical Sciences Network at the University of the Witwatersrand.*

<http://www.cs.wits.ac.za/~jon/help/unix/other/pxe-tftp-dhcp/>

**Hennessy, J. L., Patterson, D. A.** (2003). *Computer Architecture - A Quantitative Approach.* Morgan Kaufmann.

**Ostrowick, J. L.** (2002). *How to get PXE network cards to boot under etherboot.*

<http://www.cs.wits.ac.za/~jon/help/unix/other/pxe-tftp-dhcp/>