

# Tuning Networked Unix Systems For Modern Applications

Tony Dale

March 24, 1993

# Tuning networked Unix systems for modern applications

Tony Dale,  
Computer Science Dept,  
University of Canterbury

March 24, 1993

## Introduction

The complexity of a network of workstations and modern networked applications can make performance tuning difficult. A telling comment from one networker was “You know you’re on a network when some system you’ve never heard of comes up on your console and hangs your machine!”. This paper aims to provide techniques for troubleshooting and tuning your network, and is organised into the following sections:

### 1. Background

- Characteristics of old style applications, how modern applications are different to these and what the next few years are likely to bring.
- Greed of modern applications versus speed of modern hardware. What technology is keeping up with demands, and what is dropping behind.
- An example of a real network (from the Computer Science Dept at Canterbury, referred to here as “Cosc”)

### 2. Configuring a networked system

- Centralized or distributed services. Configuring servers of several kinds, for file serving, CPU serving and network gateway services).
- Siting discs, memory and CPU: who should get the resources?
- Configuring discs: mixing “hot” and “cool” partitions and using I/O bandwidth effectively.
- Planning and expanding your network topology, using ethernet or FDDI.

### 3. Tuning a networked system

- Monitoring and controlling your network (why is it going slow?).
- Using vmstat, iostat, netstat and sar. How clients slow down servers.
- Tuning for various types of applications such as client-server or X windows.
- Determining what resources to add, and where to add them.

### 4. Summary

# 1 Background

## Typical Characteristics of old-style computer usage

Each user had one character-mode terminal talking over a serial line to a central, stand-alone computer. You would login on the terminal and run one application at a time. Possibly the terminal would have graphics capability, and you could receive graphs or line drawings in a vector-graphics format.

## Characteristics of modern applications

Things have become more complicated in recent times, see figure 1. A revolutionary change for computing has been the advent of the Local Area Network, and especially ethernet, with all its associated sharing of files, CPUs and I/O devices. Modern "terminals" might be Xterminals, or discless Unix (or Apple Macintosh or MS-DOS) workstations that run some applications locally and others remotely, on networked compute servers that mount filesystems from several other machines. Some of the applications (eg databases) could be client-server, using one or more networked servers talking to a local client program. Each user can run many applications simultaneously.

## Coming soon (or here already)

Here are a number of prime candidates to overuse CPU, RAM and disc space: Realtime video and CD-quality sound are being added to applications such as E-mail and databases. Three dimensional GUIs are being developed, and PEX, the 3-D extension to Xwindows, is available on some high-end Xterminals. Already ethernet is too slow in some applications (eg, for the above three), and 100 Mbits/sec networks are becoming available.

## Effects of system slow-down

Character oriented applications were arguably less sensitive to slow system response: you could usually type ahead when the system was slow, and even correct mistakes, because the terminal driver would echo characters. If nothing was echoed to your typing, you knew that the system had crashed...

Modern GUI applications are extremely sensitive to slow system response: You can't "mouse-ahead" when your mouse won't move! Slow and erratic GUI response is extremely irritating to users. Even when operating from an Xterminal with its local server, a window that won't respond to mouse clicks or typing is pretty distressing.

## What is a "typical" GUI user?

The *modus operandi* of the users at Cosc that makes them "typical" is that they are usually running an interactive text-based program; either an editor or mail reader or something similar. These programs don't actually use much in the way of CPU or RAM (eg, the editing session on this file is using about 360 kbytes). What accounts for the bulk of the RAM and CPU in our user sessions is the Xwindows (or OpenWindows) interface. An xterm and window manager on their own account for 300-500 kbytes of memory each, plus the CPU and network load as they control the complicated X GUI. An Openwindows session can easily use 16 Mbytes of RAM and all the CPU on a Sun ELC.

Many users can be characterized this way at present. The aspects of a GUI that will make it truly different from an ascii terminal — Real time video, CD-quality sound and 3-D displays — are just starting to make an impact on networked systems. The very high network and CPU performance required for these types of applications has only recently started to be available, so most networked systems limit users to GUIs that display text and static two dimensional pictures.

A desktop workstation runs multiuser Unix, and for that reason the GUI response can be intermittent if the CPU has to run many processes, or wait for remote paging. This has a drastic effect on GUI performance: the workstation may "freeze" for seconds at a time. An Xterminal is single-user CPU, and always provides good response because the local CPU has nothing else to do.

## Greed ...

The one word that might describe the trend of all computer applications is GREED. Memory requirements in the tens of megabytes are common these days, and Sparc CPUs of less than 20 MIPS are rather sluggish running Sun's latest release of Openwindows. Disc space usage is on the increase, although not to the same extent as CPU and memory requirements.

## ... v's speed and space

Fortunately CPU power is much cheaper these days: ten years ago very few users would have had a graphical simulation of an analogue clock on their screen. In 1993 individual CPUs are running in the 100 MIPS range, and multiprocessing can put hundreds of MIPS on a user's

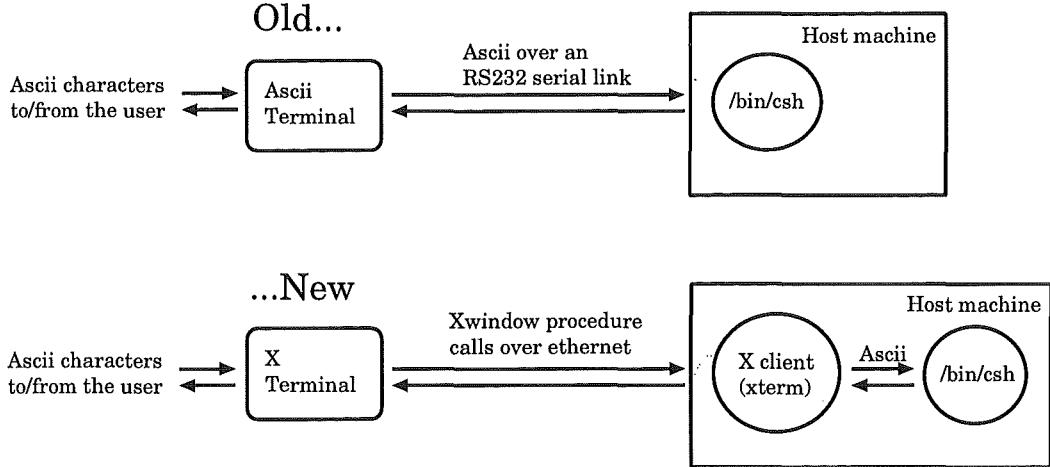


Figure 1: A login session at Cosc, then and now.

desktop. All that CPU power can be utilized by one user running a complicated GUI, though.

Likewise RAM is becoming cheaper in leaps and bounds, currently costing about \$NZ100 per megabyte. An individual user workstation is underconfigured with less than 16 Mbytes of RAM, and maximum capacities are going up into the gigabyte range.

Disc space is becoming cheaper but not faster. Nowadays discs of 2 Gbytes and up are readily available, but disc access times of less than 10 mS are still hard to come by.

## The Computer Science Network

The local network at Cosc is illustrated in figure 2. The physical layout of the network is quite simple: two logical ethernets local to Cosc, and an external ethernet backbone to all extra-department machines, including the internet. These three ethernets are gatewayed<sup>1</sup> together by one machine.

Why is our ethernet split up the way it is? There are very good reasons to gateway ourselves off the backbone:

1. Performance. The backbone ethernet regularly sees an average load of 20% or more, which we don't want to see on our ethernet.
2. Reliability. Terrible things can happen on an ethernet shared by many machines and protocols, as our backbone is. Quite apart from the occasional ARP storm<sup>2</sup>, it only takes one broken ethernet interface on a

PC to kill every machine on the same ethernet segment. It's hard enough to track down ethernet problems in our own department, let alone in another building...

Our Intra-departmental ethernet is split into two segments to spread the ethernet load. As you can see, most of the Xterminals are on one segment and most of the discless clients are on the other. There are a number of reasons for doing things this way:

1. There are a lot of Xterminals, but the ethernet traffic they generate is quite low, around 6400 bits/sec each [2].
2. The discless clients are on the other network, for quite the opposite reason: there aren't many of them, but they can generate a *lot* of traffic, up to two Mbits/sec each.
3. The Xterminals are physically grouped together (in two labs), as are the discless clients (on one floor), so it was easy to run an ethernet that grouped them logically together.

This layout keeps the load down to about 5–10% on each ethernet segment. There are also several “dataless” clients, that have the root and swap partitions on a local disc, and in some cases a local /usr. Having the root and swap partitions local cuts down the NFS traffic generated by a client hugely, because of the large amount of I/O traffic to these partitions. root and swap partitions are referred to as “hot” [5], whereas partitions with executables and user data on them are referred to as “cool”, because of the much lower I/O traffic to them (usually – there are special cases like database partitions). Because of these characteristics the dataless clients are not much of a load on the ethernet at all.

<sup>1</sup>Nomenclature for this term varies: other people might call our gateway a “router”

<sup>2</sup>An ethernet “storm” refers to a huge ethernet load, caused in this case by two different ethernet addresses being supplied in response to the same Address Resolution Protocol request.

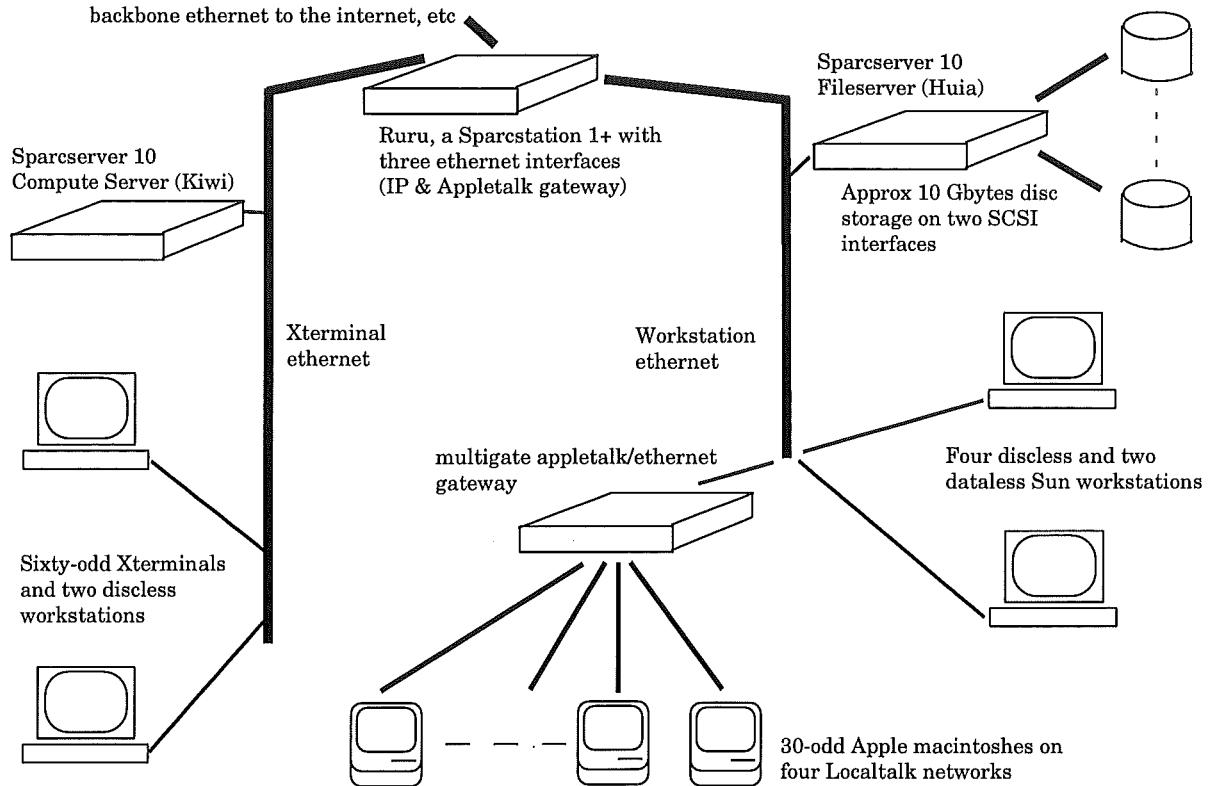


Figure 2: The Computer Science Dept network

### The Users

Our staff and students run a variety of applications interactively from an Xterminal or workstation. Their usage patterns are typical of the classical “edit, compile, run, edit” user, and during a busy day, such as just before an assignment is due, there might be 60 simultaneous users of this type. Most of these sessions are in the middleweight league – about 1-2 Mbytes resident text/data and maybe needing a MIP or two to run OK.

Another class of users run large applications at night, in batch mode. Examples of this usage are large simulations and tests of compression algorithms. The heaviest jobs of this type might need up to 100 Mbytes of swap space and all of the CPU. All of this class of jobs are CPU-bound, provided they can get enough RAM.

The most resource-hungry “user” on our system is RTI ingres, the database system. Ingres has a disc partition to itself for database files, which is utilized 100% (mostly seeking) during database lookups, etc. The transaction logfile is a raw partition on another disc (ingres does the housekeeping for this heavily used file).

A significant background job is the system disc backups, which are run twice a week. The disc being backed up gets 100% usage, and the CPU associated with that disc runs a compress program as well. The effect is something like a

malignant demon that slows down response for classes of users associated with a particular disc (staff, undergrads, postgrads...), one after the other, all through the night.

### The Machines

We have one fileserver, Huia, which is available also as a general-purpose machine. We have three compute servers: Kiwi and Mohua, which are Sparc 10/30s, and Kahu, a Sparcstation 2. We have one router, Ruru, performing gateway functions for TCP-IP and appletalk, and it is an NIS slave server. It doesn’t do much else.

### The Network as it was

The local network at Cosc *circa* 1989 is illustrated in figure 3. While it looks not too different to what we have today there are some major differences.

### The users

Interestingly enough the applications used on these machines were almost the same as used today (editors, compilers, mail readers, etc). The big change from 1989 to 1993 is in the user interface: in those days login sessions were al-

<sup>3</sup> Chooser presents a menu of machines, from which a user selects one on which to start their X session.

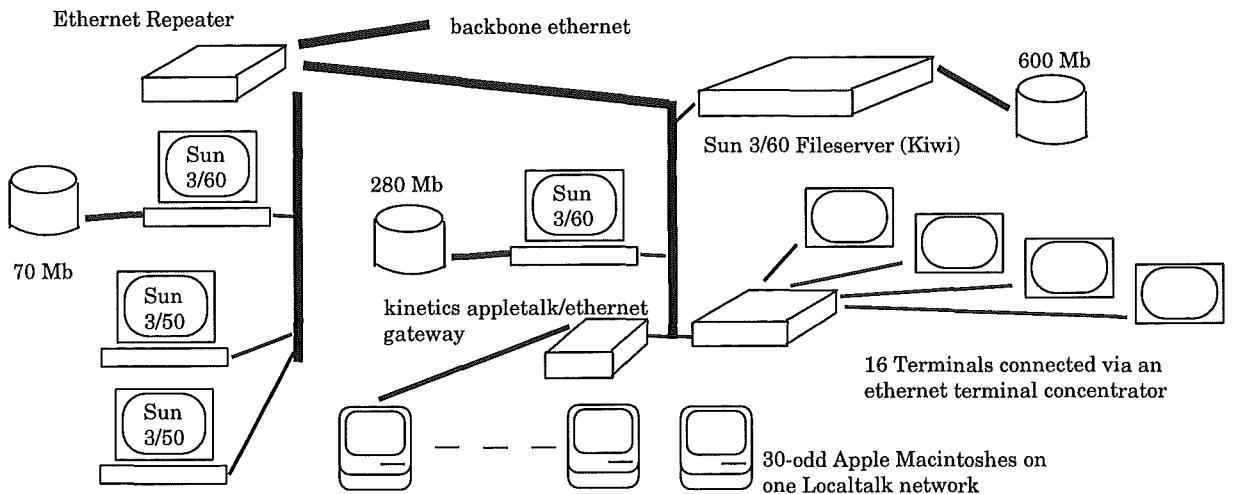


Figure 3: The Computer Science Dept network, 1989

### Load balancing

As soon as we had more than one machine on our network we became concerned about load balancing. The method of balancing loads among our machines has always been by distributing login sessions, and so we implemented a "loads" command and login ID, which would display a list of machines to log in to, sorted by load. The command was just a script which used the "ruptime -rl" command. Users (students) were instructed to always use the loads login before starting a session. This works reasonably well except when a large number of users log in at once, and of course they all select the same machines. Fortunately the only common occurrence of this is during student lab sessions, and in this case the tutor in charge might want to override the choice suggested by the load balancing system, and tell the students to log in to a specific machine.

The loads command was reasonably convenient for logins from ascii terminals, when logging in was quite a quick process, but wasn't much use for Xterminals because establishing an X session takes a minute or so on our network. In mid 1992 one of our honours students modified the Xterminal chooser<sup>3</sup> program to present a "login best" button to start an X session on the least loaded machine [1]. This modified chooser works very well and is available via anonymous ftp from <cantva.csc.canterbury.ac.nz> in the COMPETITION directory as CLB1.TAR.Z;1 (yes, it's a VMS machine).

most always via ascii terminals, which resulted in far lower resource usage: perhaps 1/2 Mbyte of memory and 1/10 MIPS per user.

### The machines

The main file/compute servers were Sun 3/60s, with 8-12 Mbytes of memory, along with two discless Sun 3/50s. Almost all of these machines were accessed via ascii terminals or Apple Macs running telnet. This configuration gave good response time for up to 40 users because of the character mode user interface. The few machines with consoles could support the Suntools GUI, but this produced a heavy load on the host machine, so good response time was only available if there weren't many remote login sessions. Our ethernet segments were joined by a repeater, and so there was no isolation either from the ethernet backbone or between machines in the department, with all the attendant problems (described earlier).

### The problems

The Suns (especially the 3/50s) were lacking memory and very vulnerable to excessive paging. As well, the fastest of them (the 3/60) ran at 3 MIPS. The result was that the bulk of users were limited to 1970's-style computing, apart from the privileged few who had access to a Sun graphics console.

When we began using RTI ingres the memory shortage on the 3/60 servers became very obvious. RTI Ingres is a client-server program, with the back end running the database engine and the front end assembling queries and interacting with the user. We found that more than one or two users would completely bog down the ingres server with paging, as the back end on

Kiwi expanded beyond the limits of RAM. This crippled response for the entire networked system because Kiwi was our main fileserver and we could not dedicate another machine to running ingres.

## 2 Configuring a networked system

### Centralized v's distributed services

The main thrust of networking and putting high performance workstations on users' desktops is towards providing distributed services. This approach is in fact very wasteful of computing resources, especially CPU cycles, but fortunately CPU cycles are very cheap, and getting cheaper. The gain provided by distributed workstations is in power and flexibility of the user interface.

The other extreme is to centralize, with all the CPU, RAM and discs on one machine. Users would use ascii or X terminals to log in to the central server. This approach is more economical than the first, but the central server has to do all the work and it may not be possible to configure a fast enough system for a large user community.

We have found that the current generation of Xterminals provide a good compromise between cost and providing a powerful and flexible GUI. Our two student laboratories have 24 Xterminals each and there are Xterminals and workstations on the desks of postgraduate students and staff. These terminals are served by one file and three compute servers, as in figure 2.

### Some real server configurations

Referring to figure 2,

- Our fileservers Huia is a Sparc 10/30 used as a general purpose machine (compute and file serving) and operates reasonably happily in this capacity. Huia has 64 Mbytes of memory which ought to be increased to 128 or 256 Mbytes in order to eliminate paging — this would also provide lots of memory for file caching. There are two SCSI interfaces on Huia although it really needs four (for 11 discs) to cut down the SCSI load.
- Our Sparc 10/30 compute servers, Kiwi and Mohua<sup>4</sup>, should be configured with at least 128 Mbytes of memory to support many X users or large programs. This

### Server reliability

For 1993 we have implemented a strategy for making our essential servers (and thereby all our networked machines) more reliable. Our essential servers provide network gateway, NFS file-sharing and NIS services, and if they become unavailable the machine requesting them "hangs" in most cases.

The strategy requires, logically enough, that no server providing one or more of the above services is dependent on another machine for any of them. The outcome of this requirement is:

- The servers must not mount any remote NFS filesystems.
- Each server must be a NIS server, either master or slave.
- Our network gateway machine, as well as fulfilling the above conditions, is not available for general use.

would provide 2–3 Mbytes of memory and 1–2 MIPS for up to 60 Xwindows users per server. Kahu, our Sparcstation 2, has 64 Mbytes of memory, providing the MIPS and RAM for up to 25 Xwindows users.

- Our gateway, Ruru, has 24 Mbytes of memory and is dedicated to providing gateway services for three ethernets, which can produce a very heavy CPU load as the kernel forwards ethernet packets.

### Who gets the RAM?

When configuring a Unix box it's important not to skimp on the RAM. These days RAM chips or SIMMs, especially third-party ones, are quite cheap, and cost is becoming less of a reason to under-configure the RAM in a Unix box.

What you really need is to know how much RAM all your applications are going to need, then add a megabyte or two for the operating system. [5] and [4] give extensive information and techniques for this. Special applications are often easier to configure for in this way because you have a good idea of what the machine will be doing. The hardest machine to configure is a general-purpose one, running applications, doing fileserving and possibly running a GUI, and often all that can be done is to start with 32 Mbytes and monitor the paging.

Many modern GUI applications are mem-

<sup>4</sup>Mohua can be set aside to run large simulation programs that would load down the CPU unacceptably for interactive users.

ory bound<sup>5</sup>, or CPU bound if they have enough memory, so it's important to have enough RAM to keep paging levels down, likewise, too little RAM in discless workstations can cause heavy performance penalties on the server and network, due to paging traffic.

## Who gets the CPU?

The CPU should go where the programs are. If your network has lots of Xterminals running from a central machine then that machine had better have a fast CPU, and multiple CPUs would be a good avenue to investigate. If you have lots of workstations running programs locally on a user's desktop then they need reasonably fast CPUs (and lots of RAM). The latter approach is more expensive than using Xterminals and a central server, though.

## Configuring disc drives

### Hot and cool disc partitions

A "hot" partition is one that has a steady, moderate to high level of utilization. The standard examples of these are root and swap partitions, but a DBMS partition would be another example of this. A "cool" partition has a low average utilization, and although on occasion the disc utilization might be high, accesses tend to be very intermittent in nature. A cool partition might be a users home directory, or a partition of application binaries.

How to mix hot and cool partitions? We take the approach of trying to configure no more than two<sup>6</sup> hot partitions on a disc, and of having them physically contiguous. Usually we use the rest of the disc as a cool partition for data or binaries. This approach seems to work well, for the following reasons:

- A typical (hot) root partition for a discless Sun client is 15-30 Mbytes in size, and the associated swap partition is 30-100 Mbytes in size
- (Cool) user data partitions and partitions for program binaries tend to be large.
- Disc space gets cheaper the larger the disc is, and so most of our discs are at least one Gbyte in size.
- Having two hot partitions contiguous cuts down the number of long seeks as the disc heads move from one partition to the other.

<sup>5</sup>meaning that memory is the resource that determines their performance

<sup>6</sup>ideally no more than one, but in the past we haven't had the discs available for this.

Here is a partition map from a 1.4 Gbyte disc I configured recently:

```
1890 cylinders 15 heads 95 sectors/track
a: 28500 sectors (20 cyls)
    starting cylinder 0
b: 99750 sectors (70 cyls)
    starting cylinder 20
g: 2565000 sectors (1800 cyls)
    starting cylinder 90
```

Partition *a* is 14 Mbytes, and is intended for a discless client root partition. Partition *b* (48 Mbytes) could be used for a client swap space, but in this case is used as a transaction file for RTI Ingres. The transaction file is heavily utilized, and in fact for efficiency Ingres uses the raw partition for the file; there is no unix filesystem there. Partition *g* (1.2 Gbytes) is used for the home directories of the Cosc staff. So here we have our standard mixture of partitions: two small, hot partitions at the start of the disc and one large, cool partition taking up the rest of the space.

### I/O bandwidth and SCSI

We use SCSI discs exclusively for our machines. SCSI discs are fast approaching the performance of IPI discs and, more importantly, they are cheap.

Although the SCSI standard allows up to 7 units on one interface, there are lower limits beyond which it is not sensible to go. Sun recommends [5] no more than two random-access<sup>7</sup> or four sequential-access (typical of a "datafull" workstation, where user data is being written to disc) disc drives per SCSI bus<sup>8</sup>.

Surprisingly, slow transfer rate devices such as tape drives should be kept off a SCSI bus you want high performance from. An exabyte 8200 tape drive can use 18% of SCSI bandwidth (220 Kbytes/sec at 1.25 Mbytes/sec) on its own, which could seriously increase the latency of any discs sharing that bus.

## Planning your LAN topology

### Ethernet

With careful planning ethernets can become very large and still provide good service. Some US institutions run thousands of workstations off one carefully subnetted ethernet. Ethernet in

<sup>7</sup>Most NFS partitions are random access. To provide good response time for random-access applications the SCSI bus bandwidth load should be kept less than 20%.

<sup>8</sup>Cosc are breaking all the rules here: one of our file-server SCSI buses has seven discs and the other has four! This heavy load on the SCSI bus increases the time for a disc request to be serviced.

its thinnet or twisted pair form is also a very inexpensive network to install.

It is difficult to decide how to partition a new ethernet because it is difficult to estimate the traffic it will have to carry. Things are much easier if you have an existing system to expand or if you can configure a trail system, because then you have some network traffic figures to extrapolate. An ethernet with an average utilization of 10% is moderately loaded, and this is a good maximum per-segment load to aim for when partitioning an ethernet.

If you are installing a large number of discless clients then it is probably not wise to go above six per ethernet segment (our present network has four 16 Mbyte discless Sun SLCs and ELCs on one ethernet segment, and this generates about a 5% ethernet load for "typical" usage). Six discless clients would also be a heavy load for one server, so the ethernet could be partitioned along the lines of one segment/six clients/one server.

Janson and Loyzaga [2] state that an active Xterminal has an "average" bit rate of 6400 bits/sec, rising to 38,400 bits/sec during window creation. An upper limit of 182.4 kbytes was found by running a particular graphics demonstration program. To use the word "average" for X traffic is very misleading though: the traffic between an X terminal and its host is generally very "bursty" in nature, which is ideally suited to ethernet), and at present we have 60 Xterminals and two discless clients working off one ethernet with a moderate (less than 5%) average load.

## FDDI

At present the main characteristic of FDDI optical fiber seems not to be blindingly fast response time, but rather a very consistent response under a heavy load. This is because many workstations (e.g. Suns) have a bottleneck in the FDDI interface - the network can deliver packets faster than the interface can process them. FDDI is in use on our campus as a network backbone and this is an ideal application for it, as the aggregate traffic from the several ethernets it services would hopelessly overload an ethernet backbone.

FDDI and other 100 Mbits/sec networks will undoubtedly be used more in the future, as workstations speed up and as the GUI they run provides more facilities. Future generations of the X protocol will provide audio and real time video, for example, which will substantially increase network load.

## 3 Tuning a networked system

### System monitoring

BSD Unix introduced a number of very useful utilities for reporting system activity: vmstat(8), iostat(8), and netstat(8). System III and V Unix has sar(1) (System Activity Reporter), which can provide most of the information that the BSD utilities can. Both BSD and system V have the "ps" command. You should make yourself *very* familiar with the manual pages for these utilities. An excellent book about the above utilities, and about Unix tuning in general, is [4].

Another utility we use a lot is top, a public domain system monitor. It gives a very user-friendly display of the activity of the most greedy programs on your system, and is available from many anonymous FTP sites [3]. All of the information that top displays is available from the aforementioned utilities, however.

Some things to bear in mind when interpreting the output of vmstat or top:

- The pageout rate should be low, preferably zero. If more than a few tens of Kbytes of pageout traffic is occurring you are close to losing a lot of performance because there is not enough RAM to go round, which results in runnable programs being paged out of memory. Incidentally, there should be no process *swapping* (as opposed to paging) on your system. If processes are being swapped (use vmstat -S or sar -w) you are very short of RAM indeed.
- The information about the percentage of CPU in system, user and idle states is sometimes useful. If the CPU is idle a lot of the time it could mean one or more of a number of things:
  - No one much is using that machine (unlikely...)
  - The machine is desperately short of RAM, and is paging heavily. In this case the CPU spends a lot of time idle, the rest of the time in system state and almost no time running user programs. The system load will likely be fairly moderate, but response time will be *bad*.
  - If more than a few percent of CPU time is spent in user state, then things are probably OK. Check the load factor on that machine though: if it is high then some other resource is short.

- Keep an eye on the information about interrupts, system calls and context switches. Sooner or later one of these parameters will reach its limit, and you will encounter a dramatic CPU slowdown<sup>9</sup>.

## Client-Server Applications

Client-Server applications can go slow because either the client, the server or the communications link goes slow. Usually the server goes slow first because of excessive greed on the part of the client, which asks (and in some cases keeps on asking) for far more resources than the server can provide. There seems to be no easy way of controlling this, although the nature of the problem provides the method for solving it:

- starting at the server, look for the overused resource and find out what remote clients are associated with it.
- Look at network traffic to identify which client(s) are making the most use of the resource.
- Try and find a way to reduce the demands of the client or increase the capabilities of the server.

### An example of client-server slowdown

Recently one of our lecturers started to write a learning environment for our stage one students, not wishing to fully expose them to the X windows system at this tender stage. He decided to use a development environment called "suit" which creates an X application according to your specification, and wrote a simple GUI that edited, compiled and ran programs, plus a few other things.

Things worked fine while he did the development on his Sparc ELC, which of course ran both an X server (openwindows) and the client program (suit) on the same machine. Communication was via a TCP/IP socket on the ELC. When he ran the client program for a demonstration, and attempted to use a remote X terminal for the server, the GUI was *very* slow to respond to user input, even though the client program was hardly using any resources on its host, and the X server was working fine. The problem was that suit used the X server (badly) in a way that required very timely communication between the client and server. This was

---

<sup>9</sup>Example: A Sparcstation 1 can't context switch much faster than 1000 times/sec. This was the limiting factor, we found, for the number of processes we could run, given enough RAM.

## Dealing with NFS overloads

We have found it very difficult to track down NFS overloads. My method for doing so is:

1. Use "etherfind -r |& more" to find the source and destination of the bulk of the NFS traffic. Often "iostat -D 1" can reveal which disc, and associated partitions, are being overloaded.
2. Log in to the offending client machine and look for likely NFS hogs. A classic one is the find(1) command.
3. Take appropriate action...

fast enough when both were on the same machine, but when a communications link (ethernet) was used, X server output lagged way behind user input. Eventually the learning environment was re-written using a more efficient development system that made efficient use of the X protocol.

## Effects of network filesharing

NFS is a client-server application for files. Frequently it happens that clients demand far more from the server, in the way of file reads and writes, than the server can provide. The result is that the clients slow down, as they wait for file access. The server may also be slow in providing access to disc I/O, as it tries to satisfy disc requests. One disc may slow down, or all the discs may slow down if the I/O channel they are on is being used excessively by NFS. Moral: client greed slows down networks and servers (and thereby other clients as well), from a distance.

## Tuning and making changes

Sometimes the only changes you can make when your system is performing badly are to change the habits of your users. The next best thing to do is to add RAM, eg to heavily paging discless clients, and finally adding CPU power may be necessary.

## Tuning kernels

With most versions of Unix it soon becomes necessary to reconfigure your kernel. For instance, Sun's GENERIC kernel has its MAXUSERS variable set to eight, and with this you will soon find yourself running out of process table slots.

## OS upgrades

We have upgraded the operating system on our network (SunOS) six times, with varying degrees of success. One memorable upgrade attempt kept our network down for a week, and was ended only by a hurried restore of the previous version. These days OS upgrades go somewhat more smoothly, even though there are more machines to upgrade, by following these guidelines:

1. We tend to be quite conservative about upgrades, something along the lines of: "if it ain't broken, don't fix it".
2. We try to have a standalone machine to install the upgrade on, for trial purposes (this is how we are running Solaris 2.1 at present.)
3. This goes without saying, but I'm going to say it: we back up our discs before an OS upgrade!
4. Starting with file and compute servers, we then upgrade one machine at a time, sometimes over many weeks. Usually the different OS versions work fine together, and upgrading one machine at a time (rather than bringing the whole network down to upgrade everything) seems to cause less total downtime. The only time the whole network is down is when file-servers are being upgraded.

Changing your kernel parameters is easy provided you read the documentation first. Some systems even provide programs to automate the whole procedure as far as possible. Some of the things we do when configuring a new kernel are:

- Increase MAXUSERS. For our machines (SunOS 4.1.3) it is commonly increased to 64 or 96 on servers, and 16 for clients. With the large amount of memory available in modern machines there isn't much reason for keeping MAXUSERS small.
- Increase the number of resources such as pseudo-ttys.
- Remove (comment out) unnecessary device drivers. Obviously a discless client isn't going to need disc or tape drivers. Be careful when removing some of the pseudo-device drivers – it is possible to go too far! You will notice when you have removed too much as your new kernel won't compile.

## Tuning for X

The X windows system is a client-server system of a special kind: the server is a graphics terminal dedicated to supporting an X windows GUI. Client programs ask the server to render graphical output.

### Memory requirements

We have found that memory shortage affects Xterminal response *very* badly. A heavily swapping host machine provides very poor service, because of the very timely response required by GUI interfaces.

Janson and Loyzaga [2] state that around 1.5–2 Mbytes of RAM and 1–3 Mbytes of swap space are used by a "typical" X user, who is running a session of one xdm, four shells, one window manager, one xclock and three xterms. This is a pretty good description of a "typical" student user at Cosc, and the memory requirements are about the same.

### CPU requirements

Most X programs are actually quite light in their CPU and network usage. The CPU load of an X client, in particular, is not much more than an equivalent character-based program. The heaviest common user of CPU we often see are graphical display programs, such as xv, which for a short time can use all the CPU available as they render a graphics image, and as the image is downloaded the ethernet load can be up to 50%. However, in general it is the tasks an X client performs, rather than the nature of the GUI, that slows down the host CPU.

Something that we may do in the future is running some X clients locally, on our Xterminals (which are actually born-again Sun 3/50s). Of course, the prime candidate for this is the window manager. An X window manager is very vulnerable to network or host slowdown, because it has to provide very fast responses to user input (mouse movements and clicks). Running the window manager locally improves the situation greatly, because the Xterminal is effectively a single-user machine. The Apple Macintosh provides good user feedback for the same reason. Some commercial Xterminals have the facility to run a window manager or other X clients locally.

### Adding resources

A properly configured system will only run faster if its users load it less or if you add more resources. Generally the second course of action is easier...

## **Adding RAM**

The main idea of any virtual-memory system is, of course, to keep any accessed program text or data in RAM, and preferably in your CPU caches. Most modern operating systems do a pretty good job of this, and with SunOS and SVR4 there is usually only one reason for excessive (= much more than zero) page out rates: lack of RAM. Reasonably enough, there are only two cures for this: reduce the demands of your users, or buy some more RAM. The first avenue is often quite difficult, although if you can identify one particular user who is using up all the memory something might be done, so the second avenue is usually easier.

The parameter to monitor when deciding how much RAM to add, and where to add it, is almost always the paging rates, especially the pageout rate. If a machine is paging excessively then the users of that machine will complain of poor response time. Look at the machine when performance is at its worst and subtract the sum of the application working sets from the available RAM. There will be a deficit.

## **Adding disc**

There are two reasons to add disc drives to a system: lack of space and lack of performance. Generally the "lack of space" syndrome is an open-and-shut case: you either remove files or buy another disc. If users of an NFS partition complain of slowness then there are a number of possible causes:

- Heavy use of the partition. Check the utilization of the disc partition at the server end.
- Slow general response from an overloaded server. Possibly the CPU, SCSI bus or network interface are too busy doing other things. In this case other services will be slow from this server as well.
- Slow general response from the NFS client machine. As for the server case, other things will be going slow.

A heavily used disc partition could be resited, probably to the client machine (if there is only one). This would be a standard course of action for a heavily used swap partition, for instance. If a number of heavy users want a partition then it could possibly be split between them, spreading the load. Over the years we have moved first our postgrad, then our honours students, to separate discs from undergraduate students.

## **Negotiating with users**

As mentioned earlier, most of our daytime users are quite a light load on the central compute and file servers. They are mostly your archetypal "edit, compile, run, edit" types: staff and students writing programs, reading news, sending mail. By negotiation, our heavy users run only at night, to avoid slowing response time for our interactive daytime users.

When resources are tight a small change to user habits can make a large difference to system response. As an example, in 1992 we had 30 Xterminals running from a few Sparc one and two servers. CPU resources were running low, and so we asked our students to stop running "perfmeter", a performance monitoring application which, ironically, was loading our machines substantially. This produced a significant improvement in system response.

## **Adding CPU**

Generally CPU is the last thing that you will run out of, having added RAM and disc all over the place and carefully configured your network. Eventually some machine will run out of context switches or MIPS or system calls/sec. You will constantly see "0% idle" from top, and the users will be forming a lynch mob.

Time to buy another CPU! Possibly you can trade in your present machine for a faster one, which is a very easy and painless operation: out with the old and in with the new. Often you don't even have to upgrade the operating system. The other way of adding CPU is to buy another machine and partition some of the load from the rest of the network onto it. Some of the ways of doing this might be:

- If you have a central file and compute server then make the new machine the compute server and the old machine the fileservice.
- If there are numerous file and compute servers then make the new machine a compute or fileservice for a particular group. This might go with partitioning the network.

## **Network expansion**

If your ethernet is overloading (more than 2-3% collisions, 20% load) then you need to partition it. Find out who is using the most bandwidth. Can these machines be partitioned into their own network?

### Monitoring your ethernet

Ethernets need to be monitored for integrity and load. Bad connections can cause total failure of an ethernet segment, or worse still degrade performance by corrupting a small percentage of packets. Try to keep within the ethernet specs. It is all too easy to keep lengthening a segment until it is out of spec and mysterious problems happen. Our own record for this is a segment nearly twice the maximum allowable length. Use netstat(8) to monitor loading on an ethernet. An ethernet segment with an average load of 20% of its bandwidth is starting to introduce significant delays. When the average load reaches 50% the segment is unusable. Ethernet collisions are an important statistic as well, with 1-2% being OK and 5% indicating overloading. Ethernet repeaters can be used to expand an ethernet cheaply, but remember to be careful of overloading: would gateways or MAC level ethernet bridges be better, given that they isolate busy segments?

Adding MAC-level (no protocol filtering) ethernet bridges is easy, but buying a second ethernet card for a workstation may be much cheaper. Check that the workstation has the I/O capacity available for a second network interface, though. Configuring an IP gateway and the associated machines requires some care, as well.

## 4 Summary

- Buy more RAM! Modern GUI applications need lots of RAM to work well.
- When looking for resource-hogging clients, start at the server and follow the trail.
- Tuning your kernel parameters is easy, and is essential for most systems.
- If your ethernet is overloading (more than 2-3% collisions, 20% load) then you need to partition it.
- Once your network has enough RAM and network bandwidth, it will need more CPU power.

## Glossary

*Context Switch* When the CPU in a system has to switch to working for a different process.

*CPU* Central Processing Unit, of which multi-processing systems have more than one per machine.

*Disc utilization* The percentage of time that a disc spends seeking and/or transferring data.

*FDDI* Fiber Distributed Data Interface. An industry standard optical fiber communications protocol.

*Gateway* A server that forwards packets between two or more interfaces, thus bridging the associated networks together.

*GUI* Graphical User Interface: What a user works with on their graphical workstation.

*LAN* Local Area Network. A high-speed network (10 Mbits/sec and more) over a small (less than 5 km radius) area, such as ethernet.

*NFS* Sun's Network File System filesharing protocol.

*NIS* Sun's Network Information Service information sharing protocol.

*RAM* Random Access Memory. Often comes in the form of *SIMMs*, Single Inline Memory Modules.

*SCSI* Small Computer Standard Interface, a peripheral bus standard used widely for discs and tapes.

*System load factor* The average number of processes waiting to run: the average length of the CPU run queue.

*SVR4* Unix, System five, release four, (e.g. Sun's Solaris 2).

*TCP-IP* Transmission Control Protocol — Internet Protocol. The standard LAN and WAN communications protocol for Unix.

*WAN* Wide Area Network. A low-speed network (up to 2 Mbits/sec) over a large (more than 5 km radius) area.

## References

- [1] Paul Ashton and Peter Smith. The CLB load balancing system. In *Proc. of UniForum NZ Conference '93*, Masterton, May 1993.
- [2] Bruce Janson and Ray Loyer. X-terminal load analysis. Programmers Notes, Computer Science Dept, University of Sydney, November 1989.
- [3] Willian LeFebvre. Top version 3.0. Anonymous FTP, June 1992.
- [4] Mike Loukides. *System Performance Tuning*. Nutshell Handbooks. O'Reilly and Associates Inc, 1990.
- [5] Brian L Wong, Pat Shuff, and Hal L Stern. *Configuration and Capacity Planning for Sun Servers*. Sun Microsystems, USA, January 1992.