

**Performance Tuning
with
LINUX inside**

Copyright

© 2001 Adam Tauno Williams (awilliam@whitemice.org)

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. You may obtain a copy of the GNU Free Documentation License from the Free Software Foundation by visiting their Web site or by writing to: Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Key



2.2

This indicates that the parameter(s) discussed on the slide is appropriate only to the 2.2.x series of kernel (Redhat Linux 7.0 and prior)



Early
2.4

This indicates that the parameter(s) discussed on the slide is appropriate only to early 2.4.x series kernels (Redhat Linux 7.1 and 7.2). Some 2.4.x kernel parameters changed around the time of the 2.4.9 release.



2.4

This indicates that the parameter(s) discussed on the slide is appropriate only to the 2.4.x series of kernels (Redhat Linux 7.1 and 7.2).

Basic Concepts

Numbers are meaningless

A number standing by itself with no unit (second, mile, packets, light year) or standing without relation to another number has no meaning.

For meaningful performance tuning, you must measure performance in a consistent fashion over a long period of time, both before and after adjustments. These measurements will stand in relation to each other.

OEM benchmarks and performance claims are as meaningless as “14 dentists prefer Crest toothpaste”. A chip vendor's claim that their test indicates their CPU is 13% faster than the competitor's CPU is meaningless unless their instruction sequence is nearly identical to the one generated by **your** applications.

Balance

A single constraint or poorly performing component can throttle the throughput of an entire host, or even network.

Adjusting network, virtual machine, or hardware parameters always needs to be done with consideration for all other subsystems.

An overloaded CPU may take so long to respond to requests that packets are lost, causing network retransmissions, causing more collisions, causing higher latency, and so on, and so on....

A machine heavy into its swap space can leave its blisteringly fast superscalar four processors with nothing to do.

Poorly adjusted network settings can cause a host connected to a high speed WAN to spend more time waiting for packet acknowledgements from remotes than actually processing requests or processing data.

Basic Technical Concepts

System Calls

The UNIX/Linux model (and most other modern operating systems) divides program operations into two basic groups: calls to procedures found in user space (programs, libraries, etc...) and calls to procedures found in kernel space (hardware operations, etc...). The later are referred to as [system calls](#).

An application (particularly an I/O or network intensive one) can often spend more of its time making system calls, then it spends processing data itself.

Making system calls has a penalty. Data must be copied in an out of kernel space. It behoves the developer to move as much data as possible at once.

The [strace](#) command can be used to report the system calls made by a process, as well as, a summary of how many times a particular system call was made, and how much time was spent there.

A strace summary

```
~ $ strace -c nslookup www.kalamazoolinux.org
```

Non-authoritative answer:

Name: www.kalamazoolinux.org

Address: 63.148.122.250

% time	seconds	usecs/call	calls	errors	syscall
-----	-----	-----	-----	-----	-----
37.29	0.010970	522	21		read
14.07	0.004140	159	26	5	open
10.60	0.003118	62	50		old_mmap
5.48	0.001612	230	7	7	rt_sigsuspend
4.09	0.001203	50	24		close
3.50	0.001031	49	21		fstat64
3.48	0.001025	54	19		mprotect

Context Switch

UNIX/Linux is a family of multi-tasking operating systems. The process of suspending one running process and commencing the execution of another process is referred to as a [context switch](#).

Context switches are computationally expensive. The kernel must perform several operations in preparation for running a process other than the one currently executing, including saving and restoring the CPU registers, modifying the address translation table in the MMU, updating the scheduler information, etc...

Simply running “*too many*” processes will harm performance.

On SMP systems, context switches are exceptionally painful if a process is switched from one processor to another, as the advantage of the processors fast cache is negated.

Interrupts

An interrupt is a hardware event (mouse movement) or software event (timer) that forces the system to context switch to a given process or point in the kernel.

When a system is processing a hardware interrupt, it is not doing “*work*”. This, plus the cost of the context switch involved, can impede performance.

The `vmstat in` (interrupt) and `cs` (context switch) columns can be used to monitor how much of this activity is taking place.

~ \$ vmstat 1

procs			memory				swap		io		system			cpu			
r	b	w	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id		
0	1	0	78652	2712	4356	161712			1	1	12	4	206	151	5	3	92
0	1	0	78652	2712	4356	161712			0	0	0	0	125	346	1	1	98
0	1	0	78652	2712	4356	161712			0	0	0	0	138	346	1	0	98
0	1	0	78652	2712	4356	161712			0	0	0	0	130	376	1	2	97
0	1	0	78652	2712	4356	161712			0	0	0	0	132	328	2	1	97

PATH and LD_LIBRARY_PATH

The **PATH** environment variable determines which directories are searched when an executable is requested, and in what order those directories are searched. A lengthy **PATH** will make this search longer, especially on filesystems (such as **ext2**) where searches of large directories (such as **/usr/bin**, **/usr/X11R6/bin**, and possibly **/usr/local/bin**) are inherently slow.

The **LD_LIBRARY_PATH** environment determines which directories are searched when a shared library is requested and in what order before the system library cache is searched. The same performance penalties that apply to **PATH** apply to **LD_LIBRARY_PATH** except that for every executable requested, a dozen or more shared libraries may be loaded.

The **LD_LIBRARY_PATH** can however be used to improve performance if you know before hand where the executables libraries will be found, thus avoiding having to search the system library cache.

'memory squeeze'

A “**memory squeeze**” condition occurs when the kernel (most likely a device driver) is unable to allocate any physical RAM pages. This will very often crash at least the device driver itself, and quite possibly the entire system.

The most common device to generate a “**memory squeeze**” is a high speed network interface card. Particularly when it receives **many** small packets.

Device drivers need to acquire physical RAM pages, not virtual memory. Thus, it is better to swap and avoid consuming all the physical RAM, than to simply avoid swapping.

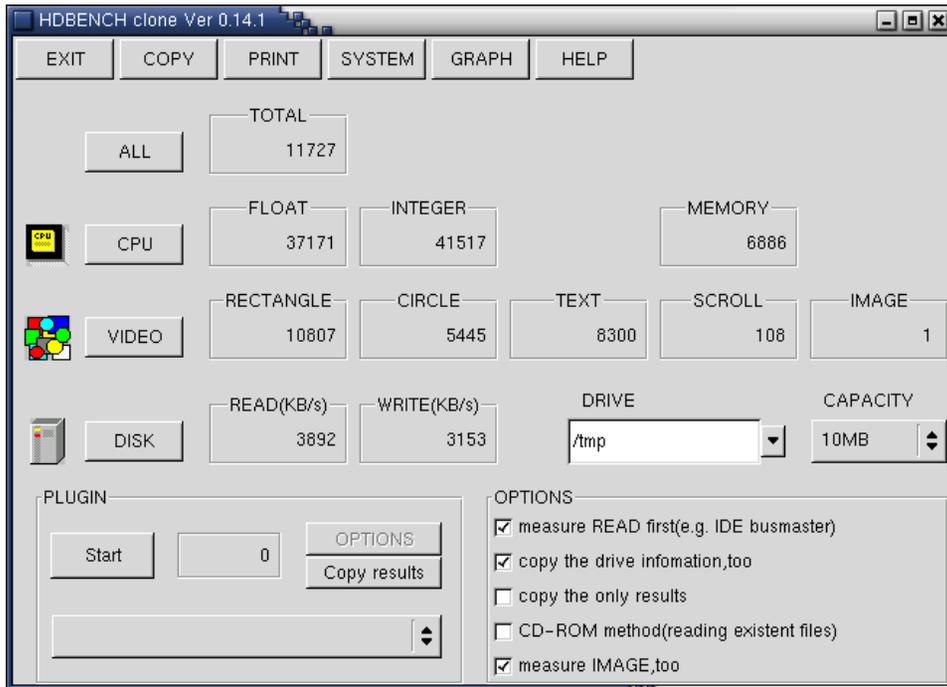
There are several parameters mentioned in this presentation in regards to avoiding “**memory squeeze**” conditions.

Benchmark Tools

hdbench

<http://freshmeat.net/projects/hdbenchclone/>

hdbench is a GTK/UN*X port of the once popular winbench tool for providing a very basic measurement of a desktop machine's performance.

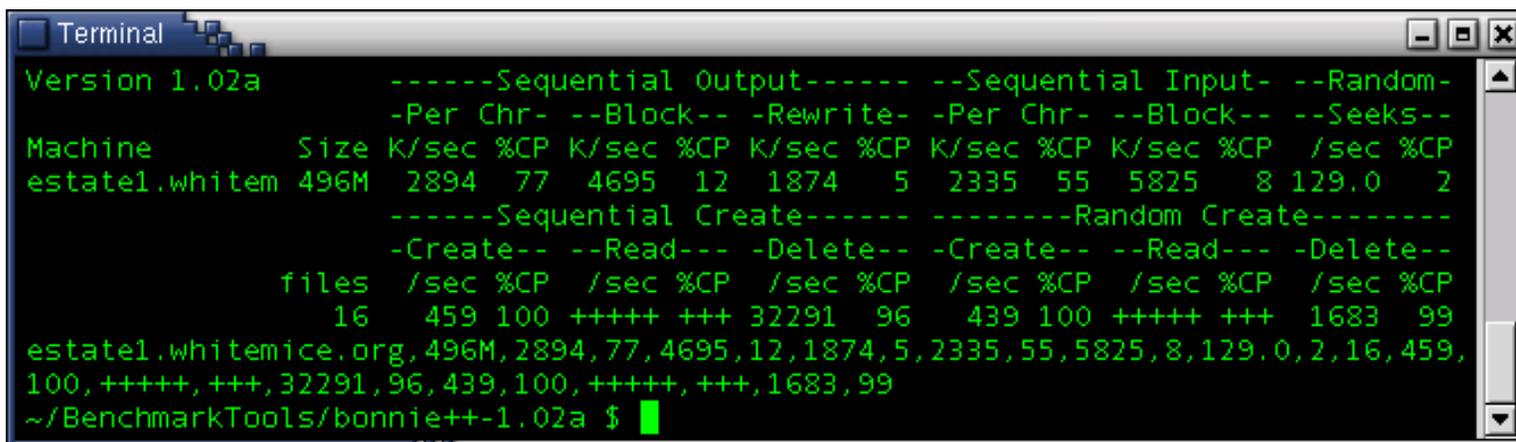


Parts of hdbench are written in assembly language to avoid compiler ideosyncracies. For this reason, it will only work on Intel based systems.

Disk I/O and graphical benchmarks are rather primitive, but better than nothing.

bonnie++

<http://www.coker.com.au/bonnie++/>



```
Terminal
Version 1.02a
-----Sequential Output----- --Sequential Input- --Random-
-Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine      Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP /sec %CP
estate1.whitem 496M 2894 77 4695 12 1874 5 2335 55 5825 8 129.0 2
-----Sequential Create----- -----Random Create-----
-Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
files /sec %CP /sec %CP /sec %CP /sec %CP /sec %CP /sec %CP
16 459 100 +++++ +++ 32291 96 439 100 +++++ +++ 1683 99
estate1.whitemice.org, 496M, 2894, 77, 4695, 12, 1874, 5, 2335, 55, 5825, 8, 129.0, 2, 16, 459,
100, +++++, +++, 32291, 96, 439, 100, +++++, +++, 1683, 99
~/BenchmarkTools/bonnie++-1.02a $
```

bonnie++ is the defacto standard for measuring I/O throughput. As you can see from the above example, it gives a rather detailed synopsis of performance.

Use the “-b” option when testing performance on database and mail servers. This causes a `fsync()` after each I/O operation which is typical for such services. Otherwise, write-back caching is permitted.

NetBench

NetBench is the commercial benchmark used to measure the performance of a fileserver for Microsoft Windows' clients.

Running NetBench requires a very expensive lab configuration and a great deal of Windows specific knowledge.

The Samba team has developed a set of tools for measuring the performance of a Samba filesystem that emulates the NetBench tests/.

dbench & tbench

<ftp://ftp.samba.org/pub/tridge/dbench/>

dbench performs all the disk I/O operations that a Samba server would perform during a NetBench run.

tbench performs all the TCP network calls that a client would make to a Samba server during a NetBench run.

Both **dbench** and **tbench** can simulate the load of a given number of clients. (Assuming that you're running the test from a workstation that is powerful enough and has a fast enough network connection.)

```
~/dbench $ ./dbench -c client.txt 4
```

```
4 clients started
```

```
Throughput 17.2396 MB/sec (NB=21.5495 MB/sec 172.396 MBit/sec) 4 procs
```

```
~/dbench $ ./tbench_srv &
```

```
~/dbench $ ./tbench 4 localhost
```

```
^.^.^.^4 clients started
```

```
Throughput 24.9053 MB/sec (NB=31.1317 MB/sec 249.053 MBit/sec) 4 procs
```

smbtorture

<ftp://ftp.samba.org/pub/tridge/dbench/>

`smbtorture` is a Samba server benchmarking tool and stress tester. It requires the `client.txt` file from the `dbench` and `tbench` packages.

The administrator should create a separate share on the Samba server for the `smbtorture` program to use. It will require roughly 25Mb of disk space per simulated client.

```
~ $ smbtorture //sardine/torture -User%passwd 32 NBW95  
Throughput 2.67016 MB/sec (NB=3.33771 MB/sec 26.7016 MBit/sec)  
NBW95 took 49.4353 secs  
~ $ smbtorture //sardine/torture -User%passwd 32 NBWNT  
Throughput 2.4795 MB/sec (NB=3.09937 MB/sec 24.795 MBit/sec)  
NBWNT took 53.2369 secs
```

pchar

<http://www.employees.org/~bmah/Software/pchar/>

1: 192.168.1.19 (brouter3.morrison.iserv.net)

Partial loss: 0 / 1472 (0%)

Partial char: rtt = 9.666541 ms, (b = 0.038325 ms/B), r2 = 0.999985
stddev rtt = 0.018111, stddev b = 0.000022

Partial queueing: avg = 0.000476 ms (588 bytes)

Hop char: rtt = 8.157720 ms, bw = 210.211991 Kbps

Hop queueing: avg = 0.000321 ms (8 bytes)

2: 192.168.121.2 (wycgate-WAN0.morrison.iserv.net)

Partial loss: 736 / 1472 (50%)

Partial char: rtt = 12.229483 ms, (b = 0.039295 ms/B), r2 = 0.999824
stddev rtt = 0.085249, stddev b = 0.000114

Partial queueing: avg = 0.001103 ms (1234 bytes)

Hop char: rtt = 2.562942 ms, bw = 8243.287278 Kbps

Hop queueing: avg = 0.000627 ms (646 bytes)

3: 192.168.21.1 (wycctysrvr.morrison.iserv.net)

Path length: 3 hops

Path char: rtt = 12.229483 ms r2 = 0.999824

Path bottleneck: 210.211991 Kbps

Path pipe: 321 bytes

Path queueing: average = 0.001103 ms (1234 bytes)

Start time: Wed Jan 30 01:58:06 2002

End time: Wed Jan 30 02:51:33 2002

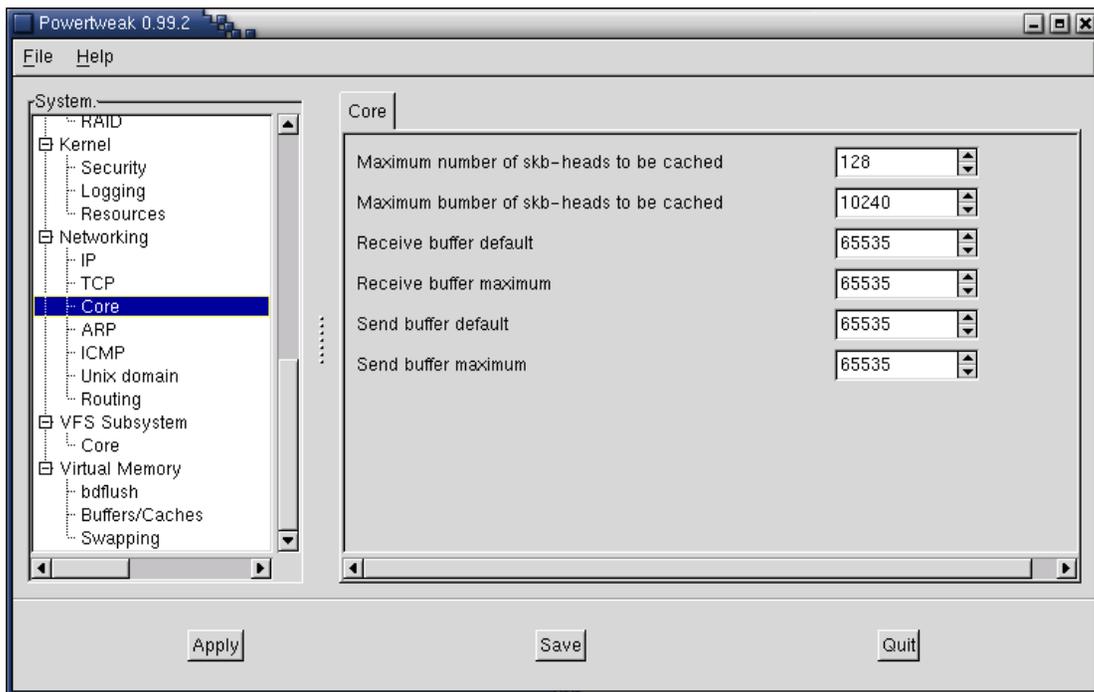
The pchar utility measures the end-to-end characteristics of a network path.

To the left is an example of pchar running against a host on a remote network connected to the host network with a frame relay circuit (256kb port speed, CIR of 128Kb).

powertweak-gtk

<http://powertweak.sourceforge.net/>

powertweak is a daemon and GTK GUI for adjusting system parameters, including the ones discussed in this presentation and more.



powertweak also contains a database of PCI devices and relevant tweaks that can be applied to improve performance.

powertweak is one of the most thorough hardware browsers available.

**I/O, I/O, it's
off to work
we go....**

I/O Tuning

I/O is the ability to move data between a device and system memory. Performance tuning of the I/O subsystems usually focuses on maximizing the throughput of fixed disks, as they account for the preponderance of I/O in a system.

Most computers designed for “personal” use have flakey and poorly designed I/O subsystems. Up until very recently, most desktop operating systems did not support multithreaded I/O and other advanced I/O techniques, which allowed manufactures to get away with simply strapping in an even more over-powered CPU.

If you work with large files, multiple applications, or swap frequently, tuning your I/O subsystems can result in a very significant performance increase.

IDE

IDE consumes a fair amount of the host CPU capacity to perform its tasks. The best course of action for systems dependent on IDE is to minimize disk activity. IDE has poor to no multitasking capability. Avoid attaching more than one device to an IDE bus, as only one drive can be active at a time.

Newer IDE drives and controllers can be significantly faster than older units. Avoid connecting older drives to the same bus as newer drives. The bus will operate at the speed of the slowest device.

Different IDE controller chipsets vary greatly in performance.

Many (most?) IDE drives report success for writes once the data reaches the drive's buffer, but before it is written to disk. This happens even if you tell the drive not to buffer writes. Do **not** store critical information on an IDE drive, or at least, let the system be idle for several seconds after shutdown before powering off.

hdparm

The `hdparm` command is used to display or adjust the features enabled for communication with a fixed disk or fixed disk controller.

`/dev/hda:`

```
multcount   = 16 (on)
I/O support = 1 (32-bit)
unmaskirq   = 0 (off)
using_dma   = 1 (on)
keepsettings = 0 (off)
nowerr      = 0 (off)
readonly    = 0 (off)
readahead   = 8 (on)
geometry    = 524/255/63, sectors = 8421840, start = 0
```

Enabling features not supported by your drive can result in system lockups, poor performance, and/or corrupted filesystems.

Correct adjustment of the feature set can result in a huge increase in system performance and responsiveness.

/etc/sysconfig/harddisks

On a RedHat system the file `/etc/sysconfig/harddisks` is parsed by the startup scripts for the purpose of “permanently” adjusting drive/interface parameters.

`USE_DMA=1` (hdparam -d1)

Enables DMA transfer to/from the controller.

`MULTIPLE_IO=16` (hdparm -m16)

By default, the IDE driver transfers one block of data per interrupt. This enables the driver to transfer 16 (or some other number) blocks per interrupt, which greatly reduces the overhead involved in reading from the disks (10-50%).

`EIDE_32BIT=1` (hdparm -c1)

Enables 32 transfers between main memory and a PCI or VLB IDE interface.

/etc/sysconfig/harddisks

LOOKAHEAD=1 (hdparm -A1)

This enables a drive's read-ahead feature, which causes the drive to read the next 8 sectors after a read request, in anticipation that these blocks will be subsequently requested. The number of blocks read ahead can be adjusted with `hdparm -a#`.

EXTRA_PARAMS=

This option is used to pass additional parameters to `hdparm`.

Other `hdparm` flags of interest:

`-u{0/1}` Allows the system to unmask other interrupts during drive operations. This can result in a significant boost in responsiveness and the performance of other devices, particularly serial port modems. However, some IDE interfaces are not capable of operating unmasked.

`-X34` Enable multi-word DMA mode 2.

`-X66` Enable UltraDMA mode (burst).

SCSI

SCSI is a significantly more “advanced” I/O technology than IDE, and thus is fraught with additional complexities. SCSI does, however, offer superior performance for multitasking environments and heavy I/O loads in addition to greater expandability (more devices per bus).

Various SCSI adapters implement different sets of features and differ greatly in how they perform. Be sure to check if your SCSI card has a corresponding README in `/usr/src/linux/drivers/scsi`.

SCSI: TCQ

Enable “Tagged Command Queueing” if your card supports it. TCQ can increase disk performance under heavy loads by 15-20%.

For the Adaptec 7xxx drivers TCQ needs to be enabled via LILO:

```
append="aic7xxx=tag_info:{{0,0,0,0,0,0}}"
```

The maximum queue size for Adaptec controllers is set at compile time (default 8). The queue size can be increased to 32.

Tape Block Size

Most tape devices under Linux default to a block size of 0, which indicates “variable” block size. Setting your tape devices to a fixed block size will often improve backup and restore performance.

```
/home/awilliam $ mt -f /dev/st0 status
```

```
SCSI 1 tape drive:
```

```
File number=0, block number=0.
```

```
Tape block size 0 bytes. Density code 0x0 (default).
```

```
Soft error count since last status=0
```

```
General status bits on (41010000):
```

```
  BOT ONLINE IM_REP_EN
```

```
/home/awilliam $ mt -f /dev/st0 setblk 4096
```

```
/home/awilliam $ mt -f /dev/st0 status
```

```
SCSI 1 tape drive:
```

```
File number=0, block number=0.
```

```
Tape block size 4096 bytes. Density code 0x0 (default).
```

```
Soft error count since last status=0
```

```
General status bits on (41010000):
```

```
  BOT ONLINE IM_REP_EN
```

Use the `-b` (*Nx512 bytes*) option of the `tar` command to match the block size of your backups to the block size of the tape drive.

```
tar -b8 -cvf /dev/st0 /home
```

The File- system

Swap Striping

The default behavior for the allocation of swap space is that the first activated swap partition is consumed first, then the second, etc...

This places the greatest write load (at least for paging/swapping) upon the drive containing the first swap partition, and subsequent swap partitions may never be used.

Using the `pri={int}` option in `/etc/fstab` with swap partitions you can control how the kernel chooses to allocate swap space. If swap partitions have an equal priority, the kernel will allocate swap from the partitions in a round-robin fashion, balancing out the I/O load.

Higher priority swap partitions are consumed before lower priority swap partitions are used.

Swap priority is a value between 0 and 32,767.

The Elevator

The elevator is the kernel mechanism for scheduling I/O operations out to the hardware. The elevator frequently makes non-optimal choices about the order in which I/O operations are presented to hardware.

For example:

1. A disk has three partitions: blocks 1-1000, 1001-5000, and 5001 – 10000.
2. A process writes to a file found on block 500; another process writes to a file on block 5002; and then the first process writes to its file again, at block 501.
3. These I/O operations may be presented to the drive as writes to blocks 500, 5002, 501. Resulting in excessive head movement, and thus additional latency.

It is best to spread filesystems that receive a significant amount of writes (/home, /var, /tmp, swap) on different drives where possible.

If more than one of these partitions will be on a single drive, attempt to make them adjacent and near the center/start of the drive (which has better seek times).

File Handles

The ability of a process to access a file depends upon its ability to allocate a file handle to represent the file.

```
~ $ sysctl fs.file-nr
```

```
fs.file-nr = 5398 962 16384
```

Value #1 – The number of allocated file handles

Value #2 – The number of used file handles

Value #3 – The maximum number of file handles the kernel will allocate

You can raise the file handle limit with `sysctl -w fs.file-max=32767`

Ideally, the first number should approach the maximum number with the center number far behind. Raise the limit until this occurs.

A system needs to be up for a reasonable period of time before you should measure against these values.

The i-node table

2.2

Files, directories, pipes (including: standard in, standard out, standard error) and sockets are representing in the kernel as [i-nodes](#).

```
~ $ sysctl fs.inode-state
```

```
fs.inode-state = 160946 0 0 0 0 0
```

Value #1 – The number of i-nodes to system has allocated

Value #2 – The number of free i-nodes (?)

Value #3 – Non-zero when the system needs to prune the i-node table

Values #4 through #7 are dummy values

The maximum number of i-nodes that the kernel will allocate into the i-node table is stored in [/proc/sys/fs/inode-max](#), and this limit can be raised by echo-ing a value into this file -

```
~ # echo "32767" > /proc/sys/fs/inode-max
```

The general rule of thumb is that this table should be three times the size of the maximum number of allocated file handles.

Block Size

The block size of a filesystem defines the granularity of disk space allocation and can significantly effect its performance, particularly if it is part of a logical or RAID volume.

Large block sizes use space less efficiently but minimize file fragmentation while smaller block sizes use space more efficiently but increase the probability of filesystem fragmentation.

Filesystems are created with the `mkfs` command whose `-b` option determines the block size. Most filesystems on Linux support 1024, 2048, or 4096 byte blocks.

noatime

Most UNIX filesystems (including ext2) record three time stamps for every file: **creation**, **modification**, and **access**.

The maintenance of the access time stamp requires an I/O operation to be performed to update the files meta-data every time an operation is performed on the file. This can dramatically increase I/O load on a busy filesystem.

The **noatime** mount option disables maintenance of the access time stamp on files in the filesystem.

ext2

Ext2 is (at least until recently) the default and standard Linux filesystem.

Ext2 filesystems begin to slow down after they reach about 1/3 their capacity.

Ext2 filesystems perform poorly when directories contain many (hundreds or thousands) of entries. It is better to have deep directory structures with fewer files in each directory.

Ext2 and RAID

ext2 filesystems on a RAID 4 or 5 volume can be optimized at creation by using the mkfs's “-R stride=###” option. This option allows you to inform mkfs of how many contiguous filesystem blocks will be contained in each volume chunk*.

For example: If your RAID device has a 32k chunk size, and your filesystem has 4k blocks, you would be best to set stride=8. (**mkfs -b 4096 -R stride=8 /dev/md3**)

*A **chunk** is the atomic unit for read/write operations performed on a RAID volume.

For RAID levels other than 4 or 5, the performance effect of the **stride** option is not documented. It is probably a good idea to use it anyway.

XFS

The XFS filesystem is a high-performance journalized filesystem available on some Linux distributions. XFS was donated to the Open Source community by Silicon Graphics, which used it as the default filesystem for their proprietary UNIX version, IRIX.

XFS is generally faster than ext2 for large filesystems in addition to being more robust due to its journaling technology.

XFS on Linux supports only 4096 byte blocks. XFS on IRIX supports blocks as large as 64k.

By default, two log buffers are allocated per mounted XFS filesystem. This can be adjusted to any value between two and eight at mount time via the `logbufs=` mount option. More logbufs may result in increased performance at the expense of RAM consumption

XFS

The default journal for an XFS filesystem is **internal**. That is, it resides on the same partition, or logical volume, as the data. For filesystems with a heavy write load, it is advantageous to move the journal to a partition or logical volume located on a separate (and less utilized) disk or physical volume -

```
mkfs -t xfs -l logdev=/dev/mirror1/journal1,size=50000b /dev/mirror0/home
```

The above would create an XFS filesystem on /dev/mirror0/home, and place the journal on /dev/mirror1/journal1. The journal size is specified to be 50,000 blocks ($4,096 * 50,000 = 204,800,000$ bytes).

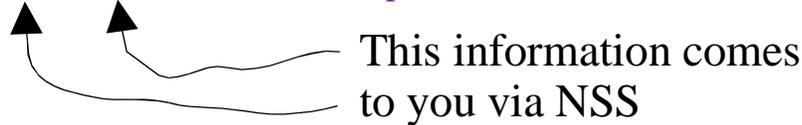
Name
Service
Switch

What is NSS?

NSS (Name Service Switch) provides the system with a common method of acquiring user, group, and host name information.

```
#ls -l
```

```
-rw-r--r--  1  awilliam users  2137 Oct  5 05:35 wds_typer.class  
-rw-r--r--  1  awilliam users  3900 Jul 19 21:48 webclasses.dia~  
drwxr-xr-x 12  awilliam users  4096 Sep 28 06:29 work
```



This information comes to you via NSS

The default NSS module (libnss_files) searches /etc/passwd, /etc/group, and /etc/hosts for the appropriate information.

As your passwd file, or other files grow, the constant need to parse these text files can begin to degrade system performance.

NSS performance effects the performance of the entire system.

nscd

(The Performance Clue Stick)

The nscd cache daemon creates a cache of NSS information. Calls by an application to the NSS API will first check with NSCD to see if it already has the information before querying the NSS modules.

The "`nscd -statistic`" command will display how effectively the NSS cache is performing -

passwd cache:

```
yes  cache is enabled
211  suggested size
600  seconds time to live for positive entries
20   seconds time to live for negative entries
1    cache hits on positive entries
0    cache hits on negative entries
3    cache misses on positive entries
0    cache misses on negative entries
25%  cache hit rate
yes  check /etc/passwd for changes
```

nscd.conf

The nscd daemon's configuration is `/etc/nscd.conf`

```
enable-cache      passwd      yes
positive-time-to-live  passwd      600
negative-time-to-live  passwd      20
suggested-size     passwd      211
check-files        passwd      yes
```

This lets you tune the size of the cache (in case you have a large user database), how long entries live in the cache, etc...

The cache size must be a prime number.

Network Devices

modinfo

Most distributions provide the network interface drivers as modules that are loaded at boot time. Each module has a set of options it supports in order to customize or tune its performance. The options of a module can be interrogated with the `modinfo` command.

```
~ $ /sbin/modinfo -p tulip
```

```
tulip_debug int
```

```
max_interrupt_work int
```

```
rx_copybreak int
```

```
csr0 int
```

```
options int array (min = 1, max = 8)
```

```
full_duplex int array (min = 1, max = 8)
```

If you do not believe your network interface is operating at full duplex (assuming it should), you can *force* full duplex mode with the `full_duplex=1` option when the module is loaded.

rx_copybreak

When an ethernet interface is initialized, the driver typically sets up buffer space for 32 packets. (Each buffer is 1536 bytes.) This *insures* the driver has somewhere to put incoming packets.

In order to conserve this buffer, the interface driver attempts to allocate an additional buffer outside this pool for any packet smaller than the size specified in the [rx_copybreak](#) parameter (default is 200 bytes).

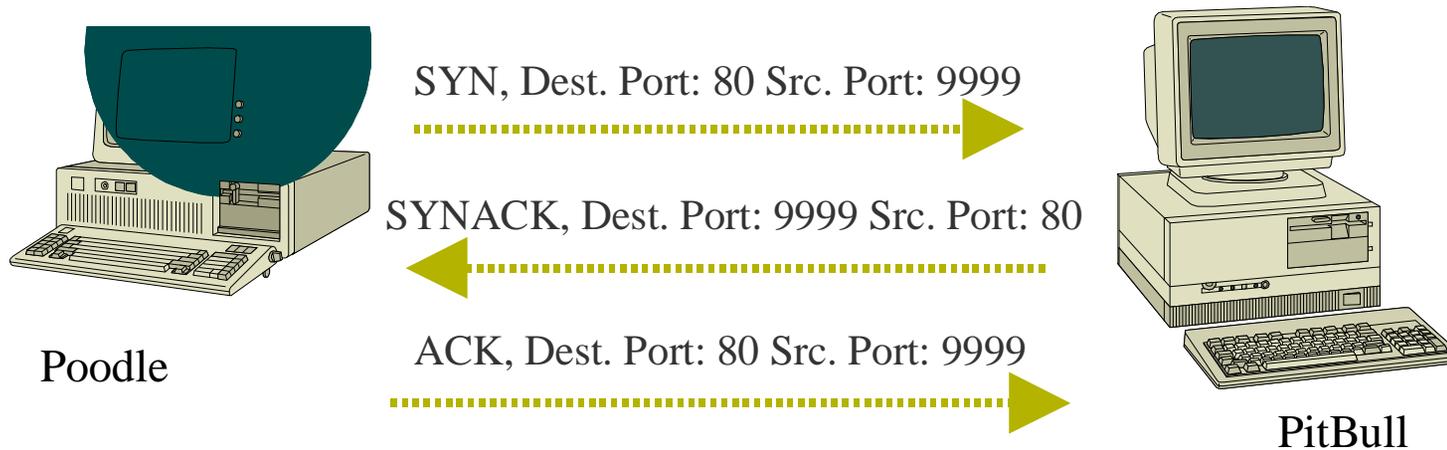
```
~$ modprobe tulip rx_copybreak=400
```

Systems with high memory pressure may be unable to allocate memory for the out-of-buffer buffer if many small packets arrive at the system in short order. This can result in a “[memory squeeze](#)” condition, crashing the module and potentially the entire system. If “[memory squeeze](#)” occurs, it is better to resolve the issue via [vm.freepages](#) (a virtual machine parameter), if possible.

It may be worth while to adjust this parameter for performance reasons on very slow or low memory systems, to avoid performing memory copies.

**The
Network
(TCP, UDP, IP)**

TCP Connection Setup



The three-step connection process must complete before any application data is transferred.

The sequence field is assigned an arbitrary value when the connection is established. The sequence is a 32 bit value.

TCP Retries

The number of attempts to establish a TCP connection is controlled by the [net.ipv4.tcp_syn_retries](#) parameter, which has a default value of 5 (in most cases) and a maximum value of 255. On networks with high rates of packet loss, you may want to increase this number.

[net.ipv4.tcp_max_syn_backlog](#) controls the number of connection attempts that can be queued (received, but not yet acknowledged). If a low-memory server (<128Mb) receives bursts of traffic, increasing this value may assist clients' ability to connect. The maximum value is 1024, which is the default for systems with greater than 128Mb of physical memory.

During a connection, TCP will retransmit unacknowledged packets. The maximum number of retransmissions before the host decides that the connection is dead is controlled via the [net.ipv4.tcp_retries2](#) parameter. Lowering this value allows systems to give up sooner on dead or slow connections/clients.

TCP Keepalive

net.ipv4.tcp_keepalive_time = 7200

How often Linux sends out keepalive probes for idle sockets where keepalive is enabled (7200 seconds / 60 seconds per minute) = 120 minutes

net.ipv4.tcp_keepalive_probes = 9

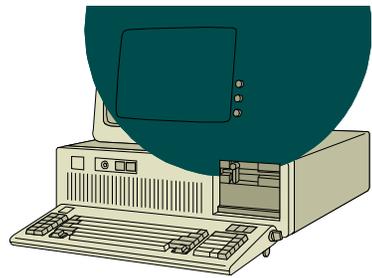
How many keepalive probes to be sent before assuming the connection is dead

net.ipv4.tcp_keepalive_intvl = 75

How long to wait, in seconds, between each of the aforementioned keepalive probes

Allowing idle connections to be truncated sooner, or detecting dead connections faster, can enable a server, which maintains long duration connections (SMB file server), to support more clients.

The Window



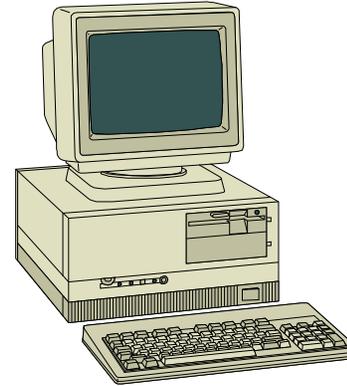
Poodle

Data Packet, Sequence 1

Data Packet, Sequence 2

Data Packet, Sequence 3

ACK, Sequence 4



PitBull

TCP maintains a **Window** which determines how much data can be on the wire before an ACK from the destination is required. A higher window permits faster more efficient data transfer, but makes error recovery more expensive.

The theoretical optimum TCP Window =

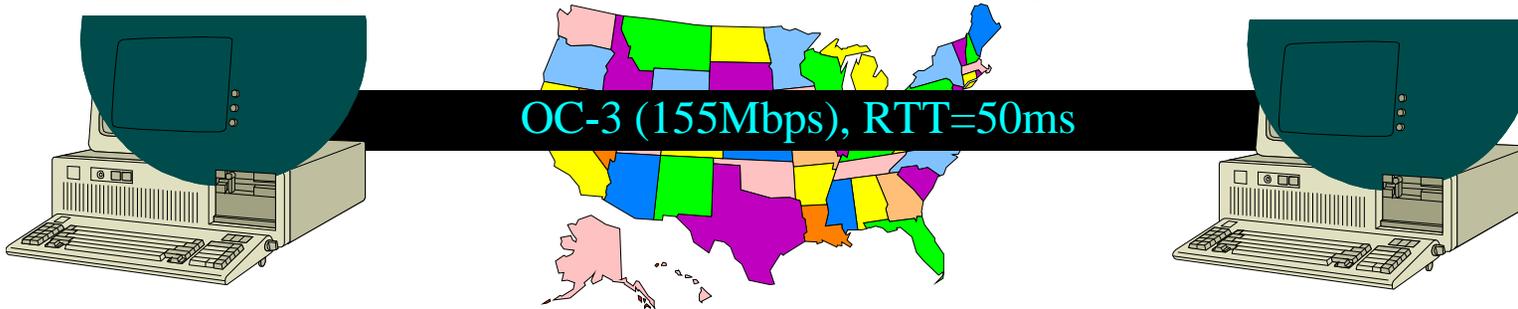
$$\text{Bandwidth (B/sec)} * \text{RTT (seconds)}$$

The maximum window size is determined by the size of the TCP buffers on each host (which should be roughly twice the size of the maximum desired window).

TCP Window Effects

On a low-latency LAN, the TCP window rarely has much of an effect on performance (except introducing lots of unnecessary ACK packets).

Having a restricted TCP Window on a network containing high-speed WAN links will stangle network performance. Here is why....

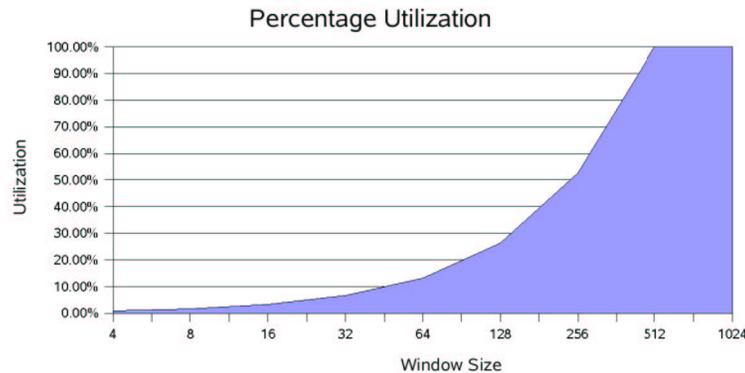


Optimal Window Size

$$(.05s/2) * (155,000,000 / 8) = 484,375$$

Maximum Throughput With
A Given Window

$$\% = \frac{(\text{window in KB})}{(\text{Mb} / 8) * (\text{RTT} / 2)}$$



Input Queue

(Protocols other than TCP)

The default kernel buffer size for incoming network socket operations is 64k. For a busy server, particularly for NFS, this can be quite a constraint.

For example: With a 64k input queue, nfsd's default 8 instances leaves 8k of buffer for each NFS thread.

To set the input queue size to an alternate value (256k/4Mb for example):

```
/sbin/sysctl -w net.core.rmem_default=262144
```

```
/sbin/sysctl -w net.core.rmem_max=4194304
```

All sockets opened after this adjustment will receive (from the kernel) an input queue of the specified size.

You must restart any running services to receive the effect of this change.

Output Queue

The default kernel buffer size for outbound network socket operations is 64k.

The output queue size has the same basic performance impact as the input queue.

To set the output queue size to an alternate value (256k/4Mb for example):

```
/sbin/sysctl -w net.core.wmem_default=262144
```

```
/sbin/sysctl -w net.core.wmem_max=4194304
```

All sockets opened after this adjustment will receive (from the kernel) an output queue of the specified size.

You must restart any running services to receive the effect of this change.

Output Buffer Stuffing

Having large output buffers can increase throughput in the following “common” situation:

1. A client queries a server for a large result set.
2. The client drops off or does not listen for the result.
3. Server attempts to send result to the client, and can only queue X number of bytes of the large result set before waiting for the queue to empty. Because the client is not listening, connection attempts are retried and the queue takes an excessive amount of time to empty. **Other clients' requests and results are tied up while the server thrashes about attempting to service the dead client.**

By making the output buffer large enough to hold any reasonably sized result set, the server can “transmit” the data and return to normal operation, leaving the kernel to deal with most of the difficulty presented by the dead client.

This is particularly useful for services, such as LDAP, that tend to receive numerous queries whose results may vary greatly in size and where responsiveness is imperative.

TCP Input & Output Queues

The size of the TCP input and output queues can be controlled somewhat independently of other protocols via two sysctl values:

```
net.ipv4.tcp_rmem = 4096    87380    174760
net.ipv4.tcp_wmem = 4096    16384    131072
```

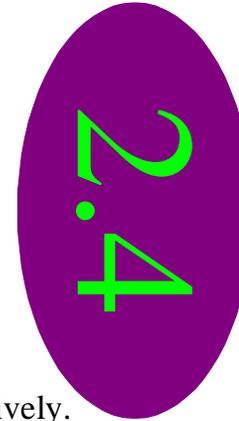
Value #1 – Minimum buffer guaranteed to each TCP socket

Value #2 – The default buffer size for each TCP socket*

Value #3 – The maximum buffer size for each TCP socket**

*This value overrides net.core.rmem_default or net.core.wmem_default respectively.

**This value cannot be greater than net.core.rmem_max or net.core.wmem_max respectively.



The values can be modified with the sysctl command -

```
~ $ /sbin/sysctl -w net.ipv4.tcp_rmem="65535 131072 4194304"
net.ipv4.tcp_rmem = 65535 131072 4194304
~ $ /sbin/sysctl -w net.ipv4.tcp_wmem="65535 131072 194304"
net.ipv4.tcp_wmem = 65535 131072 4194304
```

TCP Overall Memory Consumption

The overall memory consumption of the TCP stack is controlled by the sysctl variable `net.ipv4.tcp_mem`

```
~ $ /sbin/sysctl net.ipv4.tcp_mem  
net.ipv4.tcp_mem = 7168    7680 8192
```

Value #1 – The number of pages guaranteed to the TCP stack

Value #2 – Once the TCP stack has consumed this number of pages, it attempts to moderate its consumption until the number of pages consumed falls below value #1.

Value #3 – The maximum number of pages the TCP is permitted to consume

Pages on the Intel architecture are 4k.



2.4

Values can be adjusted with the sysctl command -

```
~ $ /sbin/sysctl -w net.ipv4.tcp_mem="8192    9216 10150"  
net.ipv4.tcp_mem = 8192    9126 10150
```

TCP Options

Do not disable these features on an Internet server.

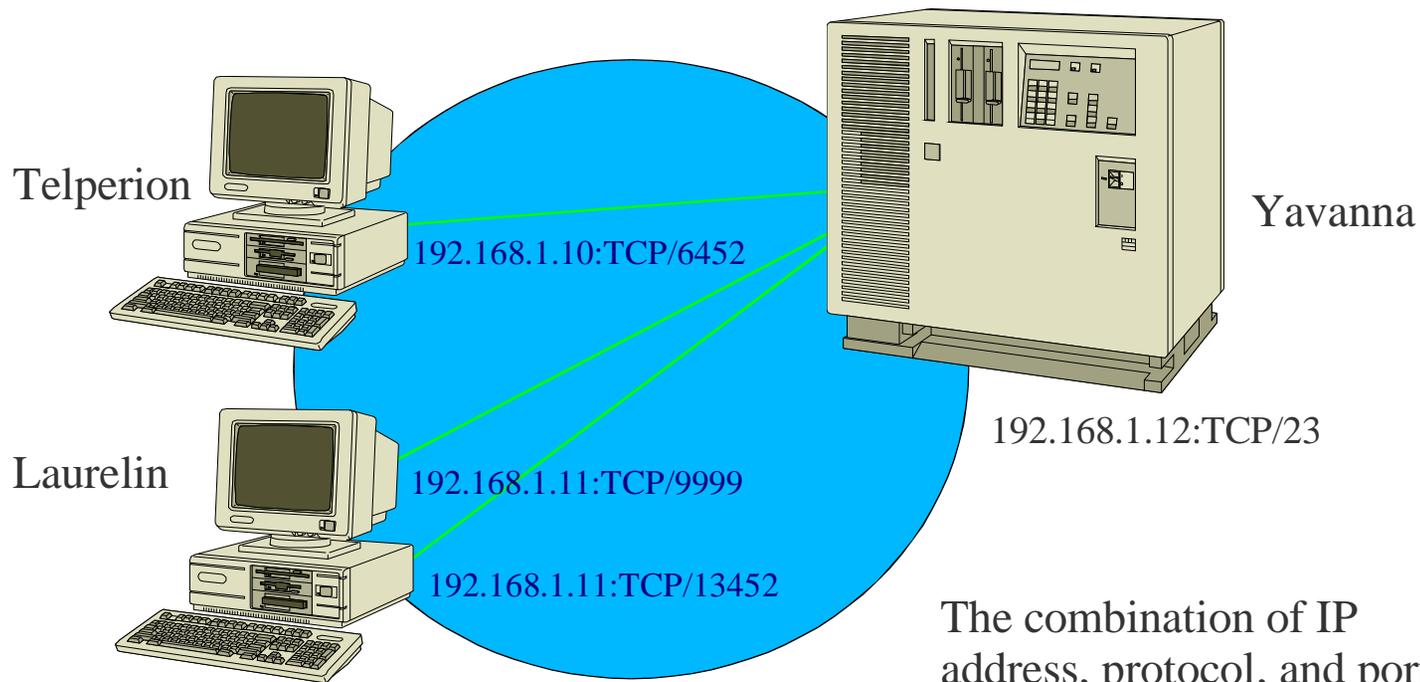
TCP is a complicated protocol with many features. Several of these features were implemented to improve performance on modern high-latency high-throughput WANs. These options increase the CPU resources consumed in processing the TCP stack. If your host communicates only with LAN connected clients, you may be able to gain a small performance increase by disabling these features.

```
~ $ /sbin/sysctl -w net.ipv4.tcp_sack=0  
net.ipv4.tcp_sack = 0  
~ $ /sbin/sysctl -w net.ipv4.tcp_timestamps=0  
net.ipv4.tcp_timestamps = 0
```

Do NOT disable the following if you communicate with any other hosts via a WAN, and be sure to set your buffers to some value between 128kb and 256kb.

```
~ $ /sbin/sysctl -w net.ipv4.tcp_window_scaling=0  
net.ipv4.tcp_window_scaling = 0
```

Port Oriented Communication (TCP & UDP)



A client (the side initiating communication) uses any available port greater than 1024.

The combination of IP address, protocol, and port create a unique identifier for an application.

net.ipv4.ip_local_port_range

To initiate new connections or, for some services, to process additional incoming connections, the IP stack needs to allocate a local port number (either UDP or TCP).

The range from which the kernel allocates these ports is controlled by the `net.ipv4.ip_local_port_range` parameter which contains two values -

Value #1 - The port number at which to begin allocating ports

Value #2 - The upper limit of the local port range

```
# /sbin/sysctl net.ipv4.ip_local_port_range
net.ipv4.ip_local_port_range = 1024 4999
# /sbin/sysctl -w net.ipv4.ip_local_port_range="32767 61000"
net.ipv4.ip_local_port_range = 32767 61000
```

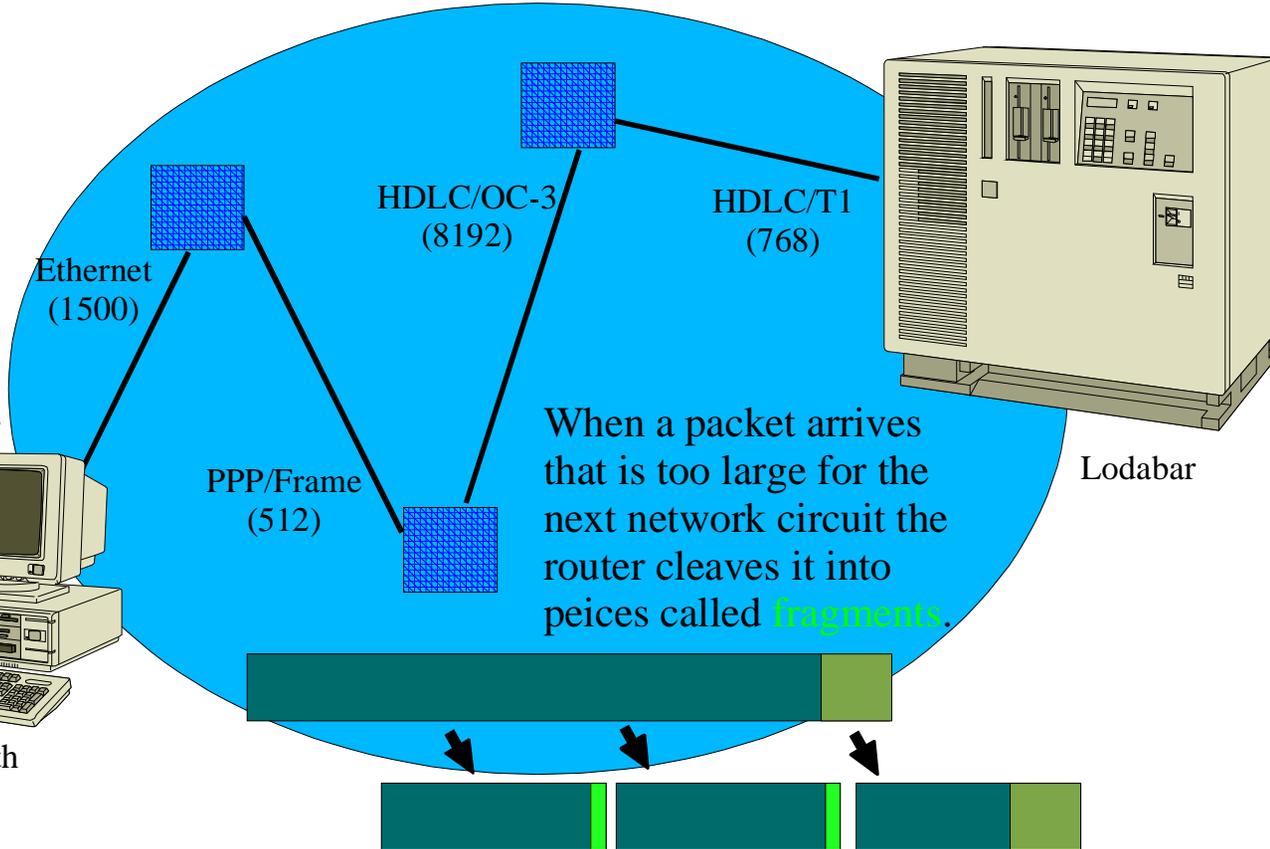
This parameter can have a significant effect on services that generate numerous outbound connections, such as the **squid** proxy server.

Fragmentation

MTU
(Maximum
Transmission
Unit) The
largest data
segment
permitted
on a circuit.



Mephiboseth



Fragment Timeout

The sysctl variable `net.ipv4.ipfrag_time` determines how long a fragment can remain in the fragment buffer awaiting the other fragments of its packet.

The default value is 30 seconds which should be more than sufficient unless you have extremely long-haul, high-latency circuits in your network.

This value is adjustable with the sysctl command -

```
~ $ /sbin/sysctl -w net.ipv4.ipfrag_time=15  
net.ipv4.ipfrag_time = 15
```

If you don't want to increase the fragment buffer to a greater value, but are still experiencing reassembly failures, you can try reducing the amount of time fragments live in the buffer.

Fragment Overflow

The kernel will buffer IP packet fragments for reassembly until the buffer size reaches `net.ipv4.ipfrag_high_thresh`. Once that limit has been reached, the kernel will throw away additional packet fragments until the buffer size has fallen to `net.ipv4.ipfrag_low_thresh`. This can behave like bursts of lost packets.

The defaults are 256k and 192k respectively. On NFS servers, or servers processing large requests from WAN clients, these levels may be too low.

You can check to see if your system is experiencing problems reassembling packets by checking the SNMP value `ReasmFails`:

```
cat /proc/net/snmp | grep "^Ip:" | cut -f17 -d" "
```

The values of the fragmentation buffer limits can be adjusted with the `sysctl` command:

```
/sbin/sysctl -w net.ipv4.ipfrag_high_thresh=524288  
/sbin/sysctl -w net.ipv4.ipfrag_low_thresh=393216
```

TCP Close



Both sides of a TCP connection must close their end of the connection independently.

When a Linux host sends a **FIN** to close its end of a connection, it waits `net.ipv4.tcp_fin_timeout` seconds for the **FINACK** before assuming the remote is dead and closes its end anyway. The number of connections waiting for a **FINACK** can be seen in the output of `netstat -a`, having a status of **FIN_WAIT**.

The Network Filesystem (NFS)

rsize and wsize

The NFS mount options **rsize** and **wsize** determine the block size with which the clients and server will interact.

The default size for both **rsize** and **wsize** is 1024, but they can be any multiple of 512 up to 8192 for NFSv2 or 32768 for NFS v3. Larger operation size *typically* results in greater throughput.

By default, NFS uses UDP/IP. UDP/IP does not perform MRU/MTU discovery. So, the presence of WAN links (with low MRU/MTU values) and the ability/willingness of the NFS server to reassemble packet fragments can reduce the effectiveness of increasing the operation size.

UDP/IP is generally faster than TCP/IP and has less overhead. In some cases, however, TCP/IP may be preferable. (For instance, you are attached to a large WAN.) You can request the use of TCP/IP using the **tcp** option when mounting the NFS filesystem, but not all NFS servers support NFS over TCP.

nfsd instances

The number of nfsd instances (representing NFS server threads) determines how many NFS operations the system can process simultaneously.

The default is 8 instances.

The number of instances started is determined by a parameter to the `rpc.nfsd` command usually called from the system startup scripts.

On Redhat distributions, NFS processes are started by the `/etc/rc.d/init.d/nfs` script and the number of instances is set using the `RPCNFSDCOUNT` variable defined in the script.

Timeouts

The default timeout for NFS RPC calls is seven-tenths of a second. Calls that cannot be completed in that time are either re-issued or assumed to fail. On a slow network, or when using a habitually slow NFS server, it may help to increase this timeout value.

The NFS mount parameter to set the RPC timeout is `timeo={int}`, where *int* is the timeout in tenths of a second.

The `acregmax/acregmin` and `acdirmax/acdirmin` NFS mount parameters establish the duration that NFS will cache file and directory attributes respectively. This value is in seconds. The default for `acregmax` is 3, and 20 for `acdirmax`.

The `nocto` parameter prevents NFS from automatically fetching the attributes on new files, which is the default behavior.

The Virtual Machine

vm.freepages

The vm.freepages parameter is used to protect the system from out of memory conditions. This parameter contains three values.

Value#1 – This number of pages is reserved for use by the kernel. When the free pages in a system drops to this point, only the kernel can allocate additional memory.

Value #2 – When the free pages of a system reach this point, the kernel begins to swap aggressively in an attempt to free pages.

Value #3 – When the free pages of a system drop to this point, the kernel begins to swap. The kernel will aim to keep, at minimum, this number of pages free.

```
~ $ sysctl vm.freepages
```

```
vm.freepages = 383      766   1149
```

```
~ $ sysctl -w vm.free-pages="512    768   1024"
```

Raising the amount of memory reserved for the kernel can help guard against “**memory squeeze**” conditions.



bdflush

bdflush is the kernel process whose job is to tell the kernel to write modified pages from the buffer to disk. **bdflush** is started during system startup by the `/sbin/update` command.

The purpose of **bdflush** is to make the scheduling of I/O operations more intelligent, as earlier versions of Linux would simply write all dirty buffer pages out to disk periodically (more or less).

By increasing the amount of dirty pages allowed in the cache, and the maximum duration those pages may remain in the cache disk, I/O becomes more batched. On systems with fast disks and intelligent controllers capable of re-ordering disk operations, this can improve performance.

On systems with slow disks that may not be able to unmask interrupts during disk operations, writing data out to disk in frequent small operations is preferable.

vm.bdflush

~ \$ /sbin/sysctl vm.bdflush

```
vm.bdflush = 30 64 64 256 600 3000 60 0 0
```

Value #1 – Maximum number of modified pages in buffer

Value #2 – No meaning

Value #3 – No meaning

Value #4 – No meaning

Value #5 – How many jiffies before the buffer is processed

Value #6 – How many jiffies a modified page may remain in the buffer

Value #7 – Maximum percentage of modified pages in the buffer

Value #8 – No meaning

Value #9 – No meaning

A jiffy is a system clock tick. Intel (and Intel like) systems have roughly 100 clock ticks per second.

Alpha systems have 1024 clock ticks per second.

You can change the bdflush parameters with the sysctl command.

```
~ $ /sbin/sysctl -w vm.bdflush="60 0 0 0 600 3000 60 0 0"
```

The value of dummy entries will remain unchanged.

kswapd

```
~ $ /sbin/sysctl vm.kswapd  
vm.kswapd = 768 32 8
```

Value#1 - Maximum number of pages kswapd tries to free in one round*.

Value#2 – Minimum kswapd tries to free in one round (the do something value).

Value#3 – Maximum number of pages kswapd writes in one round.

*This number is divide by 4 or 8.

Increasing the first value on a system that swaps heavily may improve performance.

A system with good disk I/O throughput may merit an increase of the third value.

```
~ $ /sbin/sysctl -w vm.kswapd="1024 32 64"  
vm.kswapd = 1024 32 64
```

Page Clusters

The Linux kernel groups swap pages into clusters of 32 pages referred to, appropriately enough, as *page clusters*.

```
~ $ /sbin/sysctl vm.page-clusters  
vm.page-clusters = 4
```

When a page must be read in from swap, the kernel will always read in $2^{\text{vm.page-clusters}}$ pages at a time. A value greater than **5** will serve no purpose as the kernel will be reading into a different cluster probably not containing any pages it will want.

Reading in multiple pages at once avoids the latency of disk seek operations, under the assumption that other pages in the same cluster will be requested soon.

Shared Memory

Shared memory is an IPC mechanism where multiple processes can acquire access to the same memory pages, and thus, work on the same data set. It is most commonly used by server applications such as Relation Database Management Systems (Informix, PostgreSQL, etc...).

The Linux kernel supports shared memory segments of up to 1GB.

The amount of shared memory that can be allocated is controlled by a set of kernel parameters.

- [kernel.shmmax](#) - The maximum shared memory segment size
- [kernel.shmmni](#) - The minimum shared memory segment size
- [kernel.shmall](#) - The maximum number of shared memory segments

[kernel.shmmax](#) and [kernel.shmmni](#) are expressed in bytes.

Adjusting the amount of allowed shared memory does not effect system performance, but may allow you to further scale applications that require shared memory.

Misc....

ISA & Interrupts

The ISA bus on Intel x86 machines has a prioritized interrupt structure dating back to 1977. All IRQ's have a priority from 0 (highest) to 15 (lowest), with zero taken by the system timer.

Devices, like serial ports, by default have a priority between 11 and 13, some of the lowest in the system. While at the same time, they have the smallest buffer (16 bytes, and the interrupt is not thrown until the buffer is half full).

Thus, a 33.6 modem conversation, resulting in 3700 bytes per second (a byte every .0002 seconds), can result in over 400 interrupts per second. Any lost bytes require the entire packet (296-1500 bytes) to be retransmitted.

IRQ priorities can be adjusted with the [irqtune](#) utility.

<http://www.best.com/~cae/irqtune/>

PPPD (Dial Up)

Make sure your serial port is set to the maximum speed.

Try Van Jacobsen header compression both on and off ([novj](#)). While VJ saves quite a bit of bandwidth, if your ISP's port servers are over-taxed, it will cause very high latency.

Experiment with the MTU/MRU, or find out what your ISP recommends. Common recommendations are 296 and 576. Larger packets can help with things like FTP and HTTP, while smaller packets are faster for interactive services such as telnet, ssh, irc, etc...

syslogd and fsync()

The default behaviour of syslogd is to call fsync(). After every write to a log file, fsync() forces the system to commit the write to disk immediately instead of maintaining it in the buffer until the most opportune time. This can cause a constant background “chirping” of disk activity.

You can disable fsync() for a given logfile by prefixing its name with a hyphen in /etc/syslog.conf.

For example:

mail.* /var/log/maillog becomes

mail.* -/var/log/maillog

This can have a significant effect on the performance of busy mail or LDAP servers especially. It also can extend the battery life of laptop systems.

Warning: Disabling fsync() may result in the loss of some log information in the event of a system crash.

PostgreSQL: Buffers and fsync()

By default, the PostgreSQL server processes call `fsync()` after every transaction, forcing buffered writes to be flushed to disk. By starting `postmaster` with the `-F` option, the administrator can disable the calls to `fsync()`, allowing the kernel to schedule the writes in the most opportune manner.

Warning: Disabling `fsync()` can result in data loss or a corrupted database in the event of system crash.

The `postmaster`'s `-B` option can be used to adjust the number of shared memory buffers allocated. The default is 64 buffers, each buffer being 8k. The processes/threads forked by the `postmaster` to handle client connections use the shared memory buffers to store pages of tables and indexes in process and maintain lock tables.

The value of `-B` must be *at least* twice the number as `-N` (maximum number of clients, default 32).