

# Performance, Scalability on the Server Side

John VanDyk

Presented at Des Moines Web Geeks 9/21/2009

Who is this guy?

# History

- Apple //
- Macintosh
- Windows 3.1- Server 2008R2
- Digital Unix (Tru64)
- Linux (primarily RHEL)
- FreeBSD

Systems I've worked with over the years.

# Languages

- Perl
- Userland Frontier™
- Python
- Java
- Ruby
- PHP

Languages I've worked with over the years (Userland Frontier™'s integrated language is UserTalk™)

# UserLand Frontier™



Web CMS, object database, scripting, tools, HTTP server, Windows & Mac.

[Home](#)

[UserLand  
Manila](#)

[Open Source  
Site](#)

[Discussion  
Forum](#)

[User's Guide](#)

[Support](#)

[Directory](#)

[Search](#)

## Announcing the Metadata Plugin for

**Author:** [John VanDyk](#)  
**Posted:** 6/1/2000; 2:36:26 PM  
**Topic:** [Announcing the Metadata Plugin for Manila](#)  
**Msg #:** [4246](#) (top msg in thread)  
**Prev/Next:** [4245/4247](#)  
**Reads:** 11128  
**Enclosure:**

I have released the Metadata Plugin for Manila. This plugin metadata with stories in a Manila site. Metadata can be indexed use these indices to build tables of contents, keyword search whatever you can dream up.

Open source developer  
since 2000



# Drupal

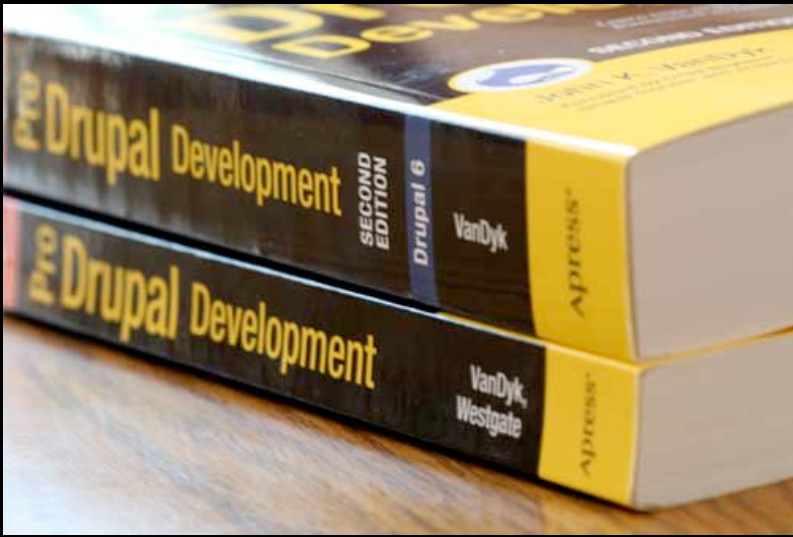
Home » Download » Drupal project » Drupal

## CVS messages for *Drupal*

September 19, 2009

- Commit **#264840** by **Dries** at 05:07
  - Drupal: **/CHANGELOG.txt 1.342**
  - Drupal: **/includes/actions.inc 1.31**
  - Drupal: **/modules/comment/comment.module 1.772**
  - Drupal: **/modules/node/node.module 1.1128**
  - Drupal: **/modules/simpletest/tests/actions.test 1.9**
  - Drupal: **/modules/simpletest/tests/actions\_loop\_test.module**
  - Drupal: **/modules/system/system.admin.inc 1.203**
  - Drupal: **/modules/system/system.api.php 1.76**
  - Drupal: **/modules/system/system.install 1.387**
  - Drupal: **/modules/system/system.module 1.792**
  - Drupal: **/modules/taxonomy/taxonomy.module 1.511**
  - Drupal: **/modules/trigger/tests/trigger\_test.module 1.3**
  - Drupal: **/modules/trigger/trigger.admin.inc 1.18**
  - Drupal: **/modules/trigger/trigger.api.php 1.5**
  - Drupal: **/modules/trigger/trigger.install 1.13**
  - Drupal: **/modules/trigger/trigger.module 1.48**
  - Drupal: **/modules/trigger/trigger.test 1.18**
  - Drupal: **/modules/user/user.module 1.1048**

- Patch **#525540** by **jvandyk, sun, jhodgdon, fago | webchick, TH**



Perl/Python/PHP

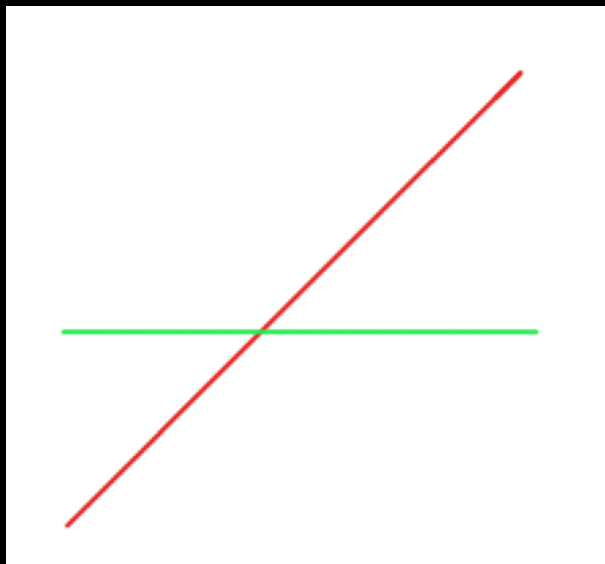
MySQL

Apache

Linux

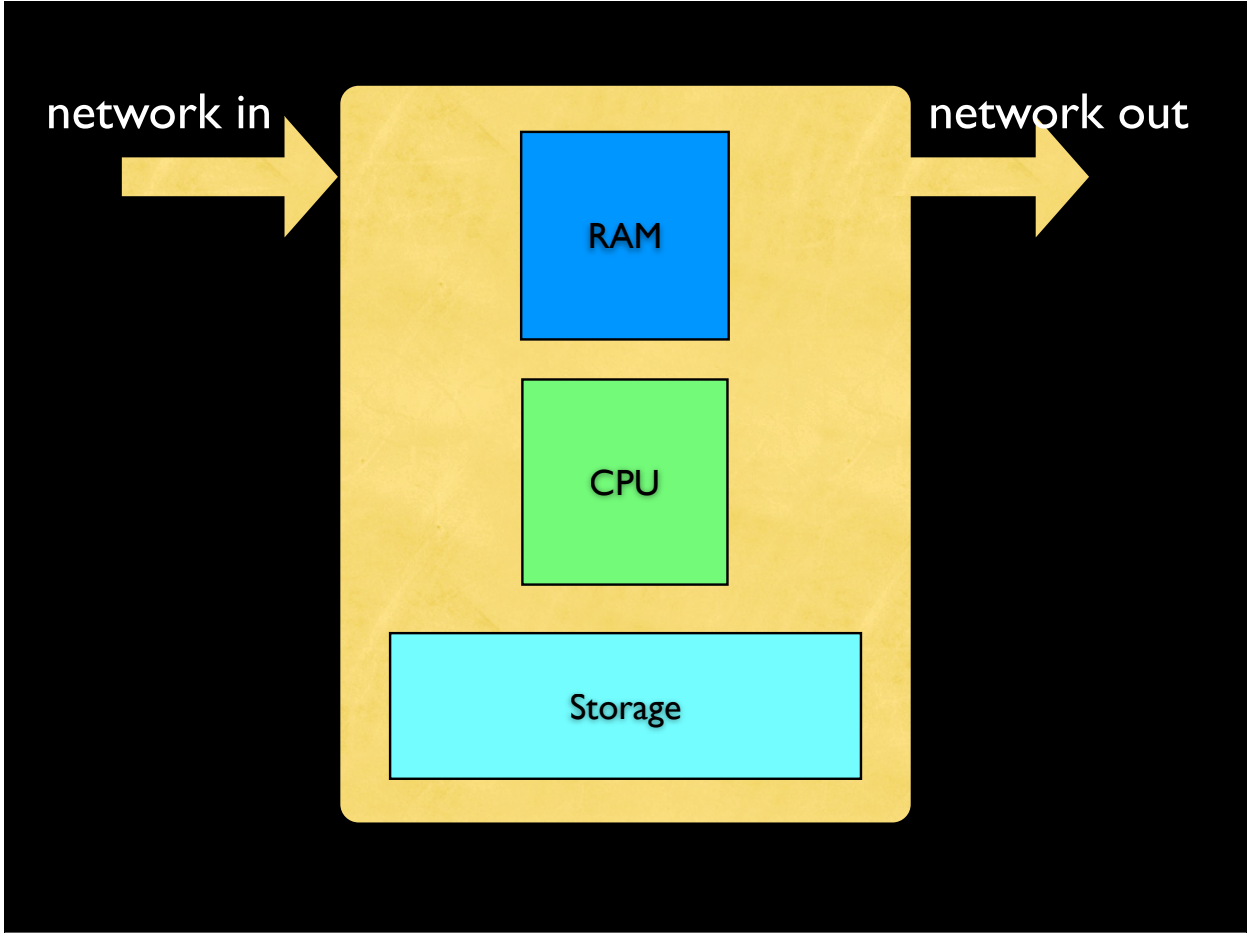
The LAMP stack.

Time  
to Serve  
Request



Number of Clients

Performance vs. scalability.



These are the basic laws of physics. All bottlenecks are caused by one of these four resources.


# Disk-bound

- Tools
  - iostat
  - vmstat

Determine if you are disk-bound by measuring throughput.

# vmstat (BSD)

procs			memory			page			disk			faults		cpu			
r	b	w	avm	fre	flt	re	pi	po	fr	sr	tw	in	sy	cs	us	sy	id
0	2	0	799M	842M	27	0	0	0	12	0	23	344	2906	1549	1	1	98
3	3	0	869M	789M	5045	0	0	0	406	0	10	1311	17200	5301	12	4	84
3	5	0	923M	794M	5219	0	0	0	5178	0	27	1825	21496	6903	35	8	57
1	2	0	931M	784M	909	0	0	0	146	0	12	955	9157	3570	8	4	88



blocked  
processes



plenty of RAM,  
no swapping



idle  
CPUs

A disk-bound FreeBSD machine.

b = blocked for resources

fr = pages freed/sec

cs = context switches

avm = active virtual pages

in = interrupts

flt = memory page faults

sy = system calls per interval

# vmstat (RHEL5)

```
# vmstat -S M 5 25
```

```
procs -----memory----- --swap- ---io--- --system- -----cpu-----
 r  b swpd  free  buff  cache  si  so   bi  bo   in  cs  us  sy  id  wa  st
 1  0    0  1301   194  5531   0   0    0  29 1454 2256 24 20 56  0  0
 3  0    0  1257   194  5531   0   0    0  40 2087 2336 34 27 39  0  0
 2  0    0  1183   194  5531   0   0    0  53 1658 2763 33 28 39  0  0
 0  0    0  1344   194  5531   0   0    0  34 1807 2125 29 19 52  0  0
```



no blocked  
processes

busy but not  
overloaded CPU

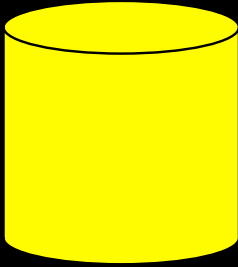
in = interrupts/sec  
cs = context switches/sec  
wa = time waiting for I/O

# Solving disk bottlenecks

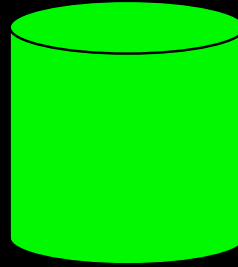
- Separate spindles (logs and databases)
- Get rid of atime updates!
- Minimize writes
- Move temp writes to /dev/shm

Overview of what we're about to dive into.

# Separate spindles



OS  
htdocs  
databases



OS logs  
MySQL binary logs  
Apache logs  
journal

The second spindle can just do sequential writes.  
Also possible to use syslog to pipe the log to a logging server.

# On atime updates

“It's also perhaps the most stupid Unix design idea of all times. Unix is really nice and well done, but think about this a bit:

'For every file that is read from the disk, lets do a ... write to the disk! And, for every file that is already cached and which we read from the cache ... do a write to the disk!’”

- Ingo Molnar, Linux kernel developer

<http://kerneltrap.org/node/14148>

# /etc/fstab (RHEL5)

```
/dev/VolGroup00/LogVol00 /          ext3      defaults,noatime 1 1
LABEL=/boot                /boot     ext3      defaults      1 2
tmpfs                      /dev/shm  tmpfs     defaults      0 0
devpts                    /dev/pts  devpts    gid=5,mode=620 0 0
sysfs                     /sys      sysfs     defaults      0 0
proc                      /proc     proc      defaults      0 0
/dev/VolGroup00/LogVol01  swap      swap      defaults      0 0
```

Note that Ubuntu uses relatime, relative atime. "Access time is only updated if the previous access time was earlier than the current modify or change time."

# /etc/fstab (BSD)

#	Device	Mountpoint	FStype	Options	Dump	Pass#
	/dev/twed0s1b	none	swap	sw	0	0
	/dev/twed0s1a	/	ufs	rw,noatime	1	1
	/dev/twed0s1e	/tmp	ufs	rw,noatime	2	2
	/dev/twed0s1f	/usr	ufs	rw,noatime	2	2
	/dev/twed0s1d	/var	ufs	rw,noatime	2	2
	/dev/acd0	/cdrom	cd9660	ro,noauto	0	0

# In /etc/my.cnf

```
[mysqld]  
...  
tmpdir=/dev/shm/mysqltmp
```

Don't do this for slave servers. Just for master.

Make sure tmpdir is big enough. Can specify tmpdir=foo:bar but MySQL uses them roundrobin.

It's better to mount your own tmpfs filesystem rather than using /dev/shm.

# In /etc/init.d/mysql

```
# Move tmpdir to /dev/shm
if [ ! -d /dev/shm/mysqltmp ] ; then
    mkdir /dev/shm/mysqltmp
fi
chown mysql:mysql /dev/shm/mysqltmp

# If you're running SELinux,
# you also need to set the
# security context.
chcon -R -t tmp_t /dev/shm/mysqltmp
```

# Is it working?

```
# while true; do lsof -c mysql | grep shm; sleep 1; done  
mysqld 3252 mysql 75u REG 0,19      1024 6257755 /dev/  
shm/mysqltmp/#sql_cb4_0.MYI  
mysqld 3252 mysql 248u REG 0,19 17825792 6257756 /dev/  
shm/mysqltmp/#sql_cb4_0.MYD
```



17 MB temp database

# Is it working?

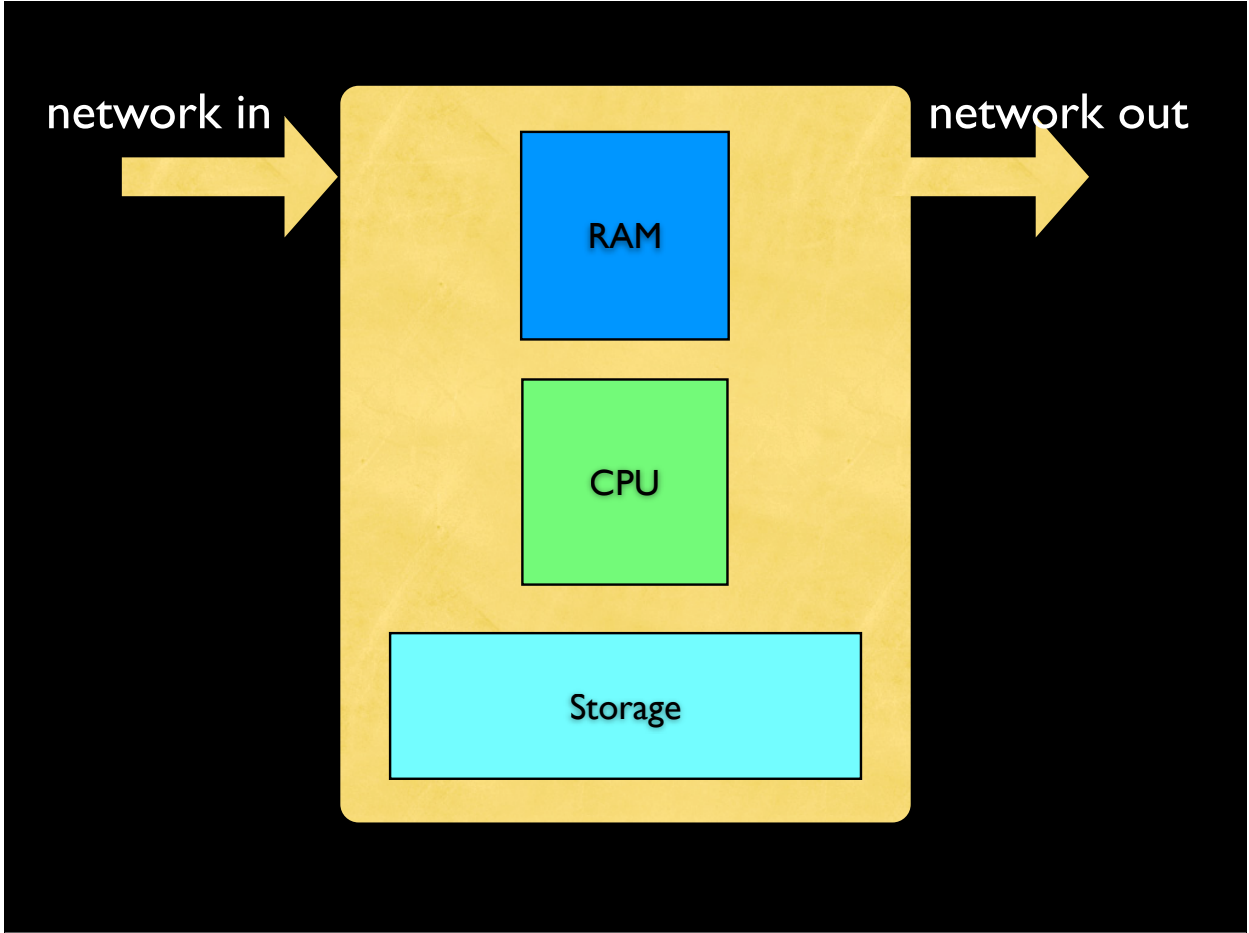
```
# df -h
```

```
Filesystem          Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                    89G   19G   66G   23% /
/dev/sda1           99M   25M   69M   27% /boot
tmpfs               3.9G   0     3.9G   0% /dev/shm
```

# Is it working?

```
# while true; do df -h | grep ^tmpfs | grep -v ' 0%';  
done
```

tmpfs	3.9G	4.0K	3.9G	1%	/dev/shm
tmpfs	3.9G	2.1M	3.9G	1%	/dev/shm
tmpfs	3.9G	6.1M	3.9G	1%	/dev/shm
tmpfs	3.9G	9.1M	3.9G	1%	/dev/shm
tmpfs	3.9G	12M	3.9G	1%	/dev/shm
tmpfs	3.9G	15M	3.9G	1%	/dev/shm
tmpfs	3.9G	17M	3.9G	1%	/dev/shm



Now let's talk about network bottlenecks.

# Network-bound

- Tools
  - netstat
  - systat (sar)
  - sysctl

netstat tells you what's going on with your network.

systat records metrics on your system over time.

sysctl shows you tunable kernel parameters.

# net.core.netdev\_max\_backlog

```
# sysctl net.core.netdev_max_backlog  
net.core.netdev_max_backlog = 1000
```

Add to `/etc/sysctl.conf`:

```
net.core.netdev_max_backlog=30000
```

O'Reilly *Understanding Linux Network Internals* section 10.5.2.

Incoming ethernet frame queue length for “bottom half” processing. Packets coming in faster than the kernel can process them land here. Queue too small = dropped frames. The queue is per CPU. Longer queue helps manage burst of traffic. Not a silver bullet. Not for NAPI drivers.

# Accept queue

```
# sysctl net.core.somaxconn  
net.core.somaxconn: 128
```

Add to `/etc/sysctl.conf`:

```
net.core.somaxconn=1536
```

<http://www.freebsd.org/doc/en/books/handbook/configtuning-kernel-limits.html>

On Linux, socket max connections waiting to get accepted; the `listen()` backlog.  
On BSD, limit of the size of the listen queue for accepting new TCP connections.

# Equivalent on BSD

```
# sysctl kern.ipc.somaxconn  
kern.ipc.somaxconn: 128
```

Add to `/etc/sysctl.conf`:

```
kern.ipc.somaxconn=1536
```

<http://www.freebsd.org/doc/en/books/handbook/configtuning-kernel-limits.html>

On Linux, socket max connections waiting to get accepted; the `listen()` backlog.  
On BSD, limit of the size of the listen queue for accepting new TCP connections.

# net.ipv4.tcp\_max\_tw\_buckets

```
# sysctl net.ipv4.tcp_max_tw_buckets  
net.ipv4.tcp_max_tw_buckets = 18000
```

Add to /etc/sysctl.conf:

```
net.ipv4.tcp_max_tw_buckets=30000
```

<http://kbase.redhat.com/faq/docs/DOC-1253>

Time-wait buckets. May run out of buckets if creating TCP connections very quickly.  
“Number of sockets to be held simultaneously in TIME-WAIT.”

# Autotuning

```
# sysctl net.ipv4.tcp_moderate_rcvbuf  
net.ipv4.tcp_moderate_rcvbuf = 1
```

If this returns 1 TCP receiver “autotuning” in effect; autotuned for each connection.  
Receiver autotuning is “new”; sender autotuning has been around a long time.

# Network buffer sizes

```
# sysctl net.ipv4.tcp.rmem_default  
net.ipv4.tcp_rmem = 4096 87380 4194304
```

```
# sysctl net.ipv4.tcp.wmem_default  
net.ipv4.tcp_wmem = 4096 16384 4194304
```

Add to `/etc/sysctl.conf`:

```
net.ipv4.tcp_rmem=4096 87380 4194304  
net.ipv4.tcp_wmem=32768 436600 873200
```

<http://www.cl.cam.ac.uk/~pes20/Netsem/linuxnet.pdf>  
IBM Press, *Performance Tuning for Linux Servers*

“TCP socket buffer sizes are controlled by their own parameters rather than the core kernel buffer size parameters.” -IBM

Careful with these. How many connections are you expecting? How much memory do you have?

# Network buffer maximums

```
# sysctl net.core.rmem_max  
net.core.rmem_max = 131071
```

```
# sysctl net.core.wmem_max  
net.core.wmem_max = 131071
```

These are in 4k pages, so the above are OK. Kernel sets memory limit to twice the requested value.

<http://www.psc.edu/networking/projects/tcptune/historical.php>

Debian: 131071

Ubuntu: 131071

Careful with these. How many connections are you expecting? How much memory do you have?

# Equivalent on BSD

```
# sysctl net.inet.tcp.recvspace  
net.inet.tcp.recvspace: 65536
```

```
# sysctl net.inet.tcp.sendspace  
net.inet.tcp.sendspace: 32768
```

Add to /etc/sysctl.conf:

```
net.inet.tcp.sendspace=65536  
net.inet.tcp.recvspace=32768 ← HTTP GET
```

We switch the values around because HTTP requests are tiny but the send file is large.

# Disable TCP selective acknowledgements

```
# sysctl -a | grep sack  
net.ipv4.tcp_sack = 1  
net.ipv4.tcp_dsack = 1  
net.ipv4.tcp_fack = 1
```

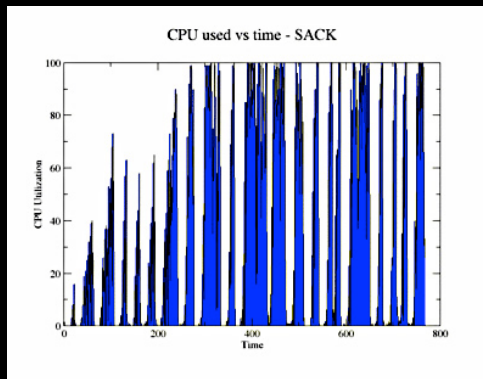
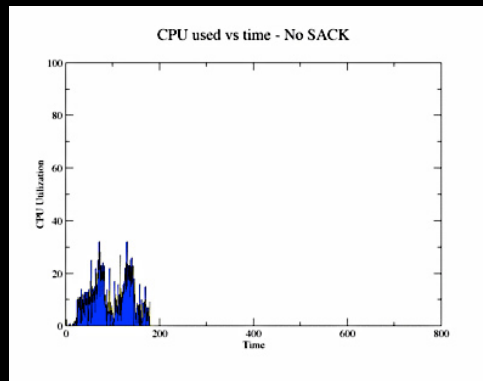
Add to /etc/sysctl.conf:

```
net.ipv4.tcp_sack = 0  
net.ipv4.tcp_dsack = 0  
net.ipv4.tcp_fack
```

<http://www.faqs.org/rfcs/rfc2018.html> (also 2883, 3517)

<http://www.ibm.com/developerworks/linux/library/l-tcp-sack/index.html>

SACK lets a host receiving packet 0, 1, 3, 4, 5, 6 request retransmission of only packet 2. dsack is duplicate sack. It's important for high-latency networks. How important are your high-latency users? Fack is "forwarding acknowledgement", a refinement of sack.



<http://www.ibm.com/developerworks/linux/library/l-tcp-sack/index.html>

Kernel must scan entire retransmission queue to find the packet; this takes time.

# Decrease TCP retries

```
# sysctl -a | grep retries
...
net.ipv4.tcp_retries2 = 15 ←
net.ipv4.tcp_retries1 = 3
net.ipv4.tcp_synack_retries = 5 ←
net.ipv4.tcp_syn_retries = 5
```

retries2: most important; attempts back off exponentially

retries1: minimum is 3 per RFC 793. After this many times, alert the network layer that something is wrong.

synack: resend response to incoming SYN/ACK segment (TIME\_WAIT connection). Each attempt: 35 seconds.

syn: resend response on an active connection. Initiated by server. Not as important.

# TCP Keepalive

```
# sysctl -a | grep keepalive  
net.ipv4.tcp_keepalive_time = 7200  
net.ipv4.tcp_keepalive_probes = 9  
net.ipv4.tcp_keepalive_intvl = 75
```

Start probing after 7200 seconds (2 hours).

Send 9 probes.

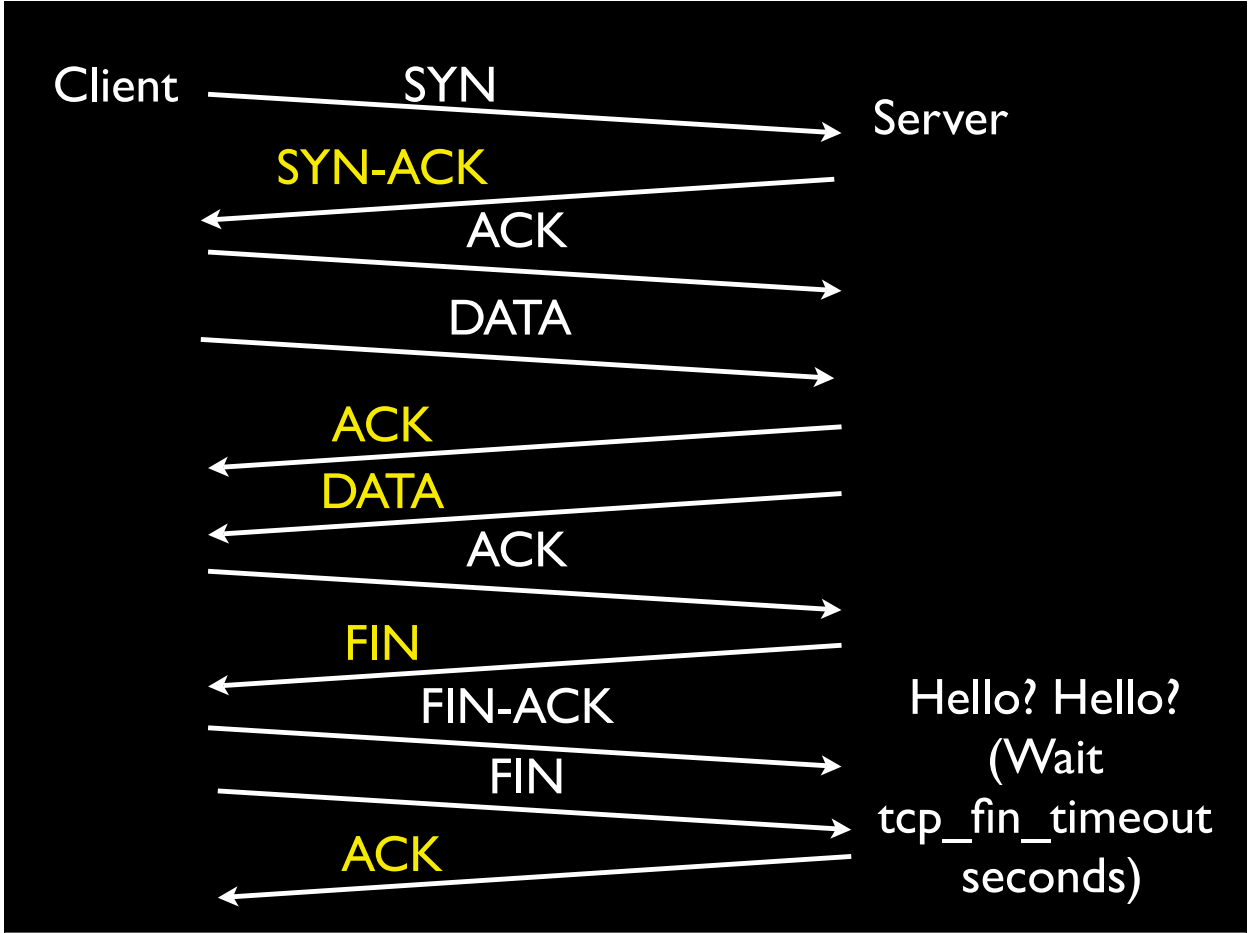
Wait 75 seconds between probes.

More reasonable: after two minutes, send 3 probes 30 seconds apart.  
But YMMV; who is your audience? Are they in Africa?

# FIN Timeout

```
# sysctl net.ipv4.tcp_fin_timeout  
net.ipv4.tcp_fin_timeout = 60
```

Wait for 60 seconds for the client to acknowledge our statement that we are done with this connection.

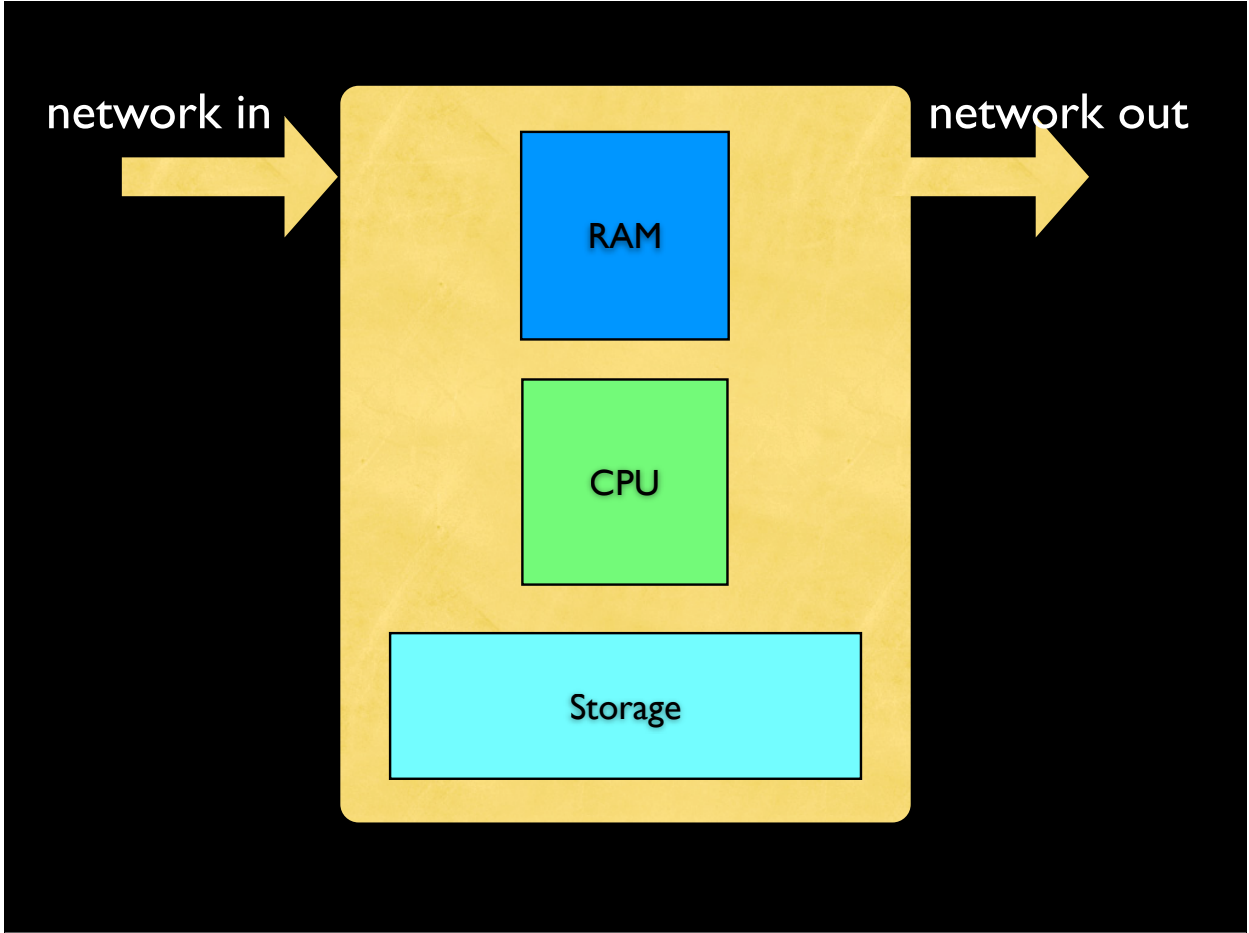


After telling the client we're closing, how long to wait for a FIN, ACK?

# Orphan Retries

```
# sysctl net.ipv4.tcp_orphan_retries  
net.ipv4.tcp_orphan_retries = 0
```

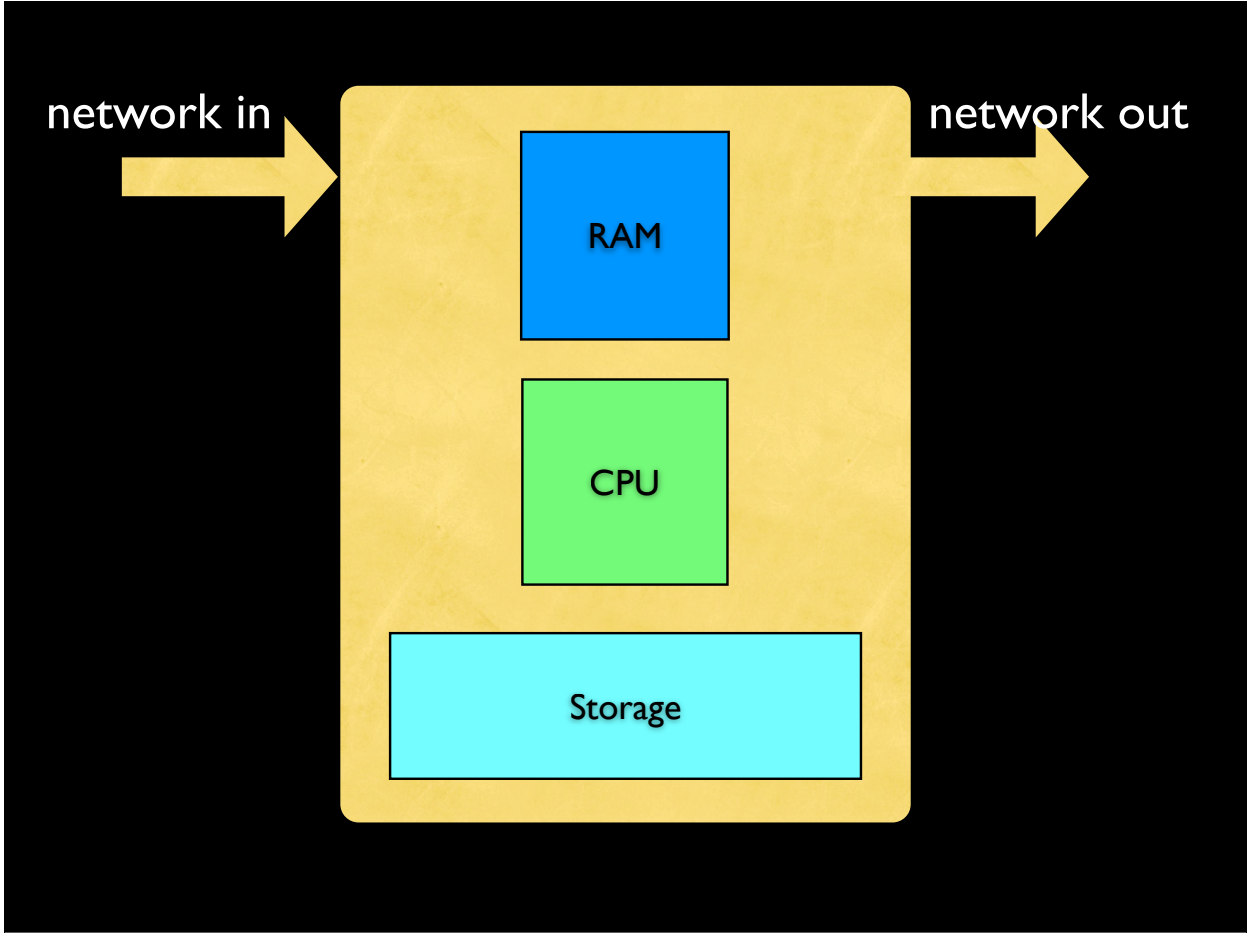
man tcp 7: “The maximum number of attempts made to probe the other end of a connection which has been closed by our end.”



Now let's talk about memory bottlenecks.

# RAM is cheap!

- Add more!
- To take advantage of >4G, you need 64-bit
- Allocate RAM strategically
  - buffers
  - Apache MaxClients (httpd.conf)
  - PHP memory\_limit (php.ini)



Now let's talk about CPU bottlenecks.

# What is your CPU doing?

- It should be executing your code
- It should be serving data
- It should NOT be doing something stupid
  - Like compiling your code on the fly 18 bazillion times

1. Read file from disk

2. Compile into opcode

3. Store in memory

4. Execute

5. Goto 1

PHP without an opcode cache.

1. Read file from disk

2. Compile into opcode

3. Store in memory

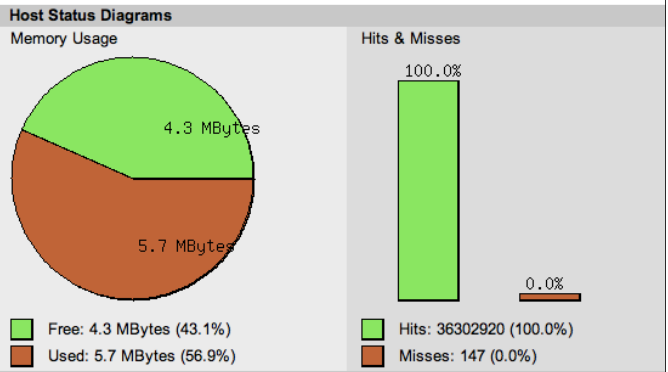
4. Execute

5. Goto 3

PHP with an opcode cache.

General Cache Information	
Cache Version	3.0.19
PHP Version	5.2.10
Cache Host	argiope.ent.iastate.edu
Server Software	Apache
Allocated Memory	1 Segment(s) with 10.0 MBytes (mmap memory, file locking)
Start Time	2009/09/17 12:31:49
Uptime	2 days, 10 hours and 1 minute
Cache Upload Support	1

Detailed Cache Information	
Cache Size	63 ( 4.9 MBytes)
Cache Hits	36302920
Cache Misses	147
Request Rate (hits, misses)	173.80 cache requests/second
Cache Hit Rate	173.80 cache requests/second



### Detailed Memory Usage and Fragmentation

# APC settings

```
apc.shm_size = 10  
apc.stat = 0
```

apc.shm\_size is the size of the cache.

apc.stat = 0 means don't stat the file every time. Must restart webserver if a file is changed.

# Apache tuning

- Calculating MaxClients for your server is critical!
- Get rid of .htaccess files
- Test KeepAlive performance

<http://httpd.apache.org/docs/2.0/misc/perf-tuning.html>

A few Apache tips.

# Calculating MaxClients

PID	USERNAME	THR	PRI	NICE	SIZE	RES	STATE	C	TIME	WCPU	COMMAND
27803	www	1	46	0	72440K	47020K	select	2	0:03	3.66%	httpd
28057	www	1	45	0	38648K	13192K	select	2	0:02	2.39%	httpd
27539	www	1	46	0	38648K	14456K	select	2	0:07	1.95%	httpd
27878	www	1	45	0	71416K	46884K	select	2	0:04	1.95%	httpd
28031	www	1	46	0	38648K	13240K	select	1	0:01	1.95%	httpd
27467	www	1	44	0	64248K	35272K	select	2	0:07	1.66%	httpd
27938	www	1	46	0	38648K	13008K	select	2	0:01	1.46%	httpd
24575	www	1	51	0	73464K	48196K	select	2	0:45	1.17%	httpd
27503	www	1	4	0	72440K	48028K	sbwait	0	0:05	0.78%	httpd
28023	www	1	45	0	38648K	12692K	select	0	0:01	0.78%	httpd
28042	www	1	45	0	38648K	12600K	select	2	0:00	0.78%	httpd
26640	www	1	45	0	72440K	47128K	select	2	0:17	0.68%	httpd
27906	www	1	44	0	38648K	13112K	select	2	0:01	0.59%	httpd



50 MB

Find out how much memory Apache is using.

# Calculating MaxClients

```
# free -m
          total      used      free      shared    buffers    cached
Mem:      2025      1216      809         0         205        718
-/+ buffers/cache:      291      1733
Swap:      4031         0      4031
```

$$\frac{2025 \text{ MB total RAM}}{50 \text{ MB RAM per process}} = 40.5 \text{ processes}$$

MaxClients = 36

Calculating MaxClients on a box with 2G of RAM. Allow some breathing room for the OS.

Questions?