HEAnet

Scaling Apache 2.x > 20,000 concurrent downloads

ApacheCon

Europe 05

http://www.stdlib.net/~colmmacc/Apachecon-EU2005/

Material

- Introduction
- Benchmarking
- Tuning Apache
- Tuning an Operating System
- Design of ftp.heanet.ie
- Future directions

A warning

“1000 httpd processes per CPU is close to the limit”

- Sander Temme, 12:01 yesterday, this stage

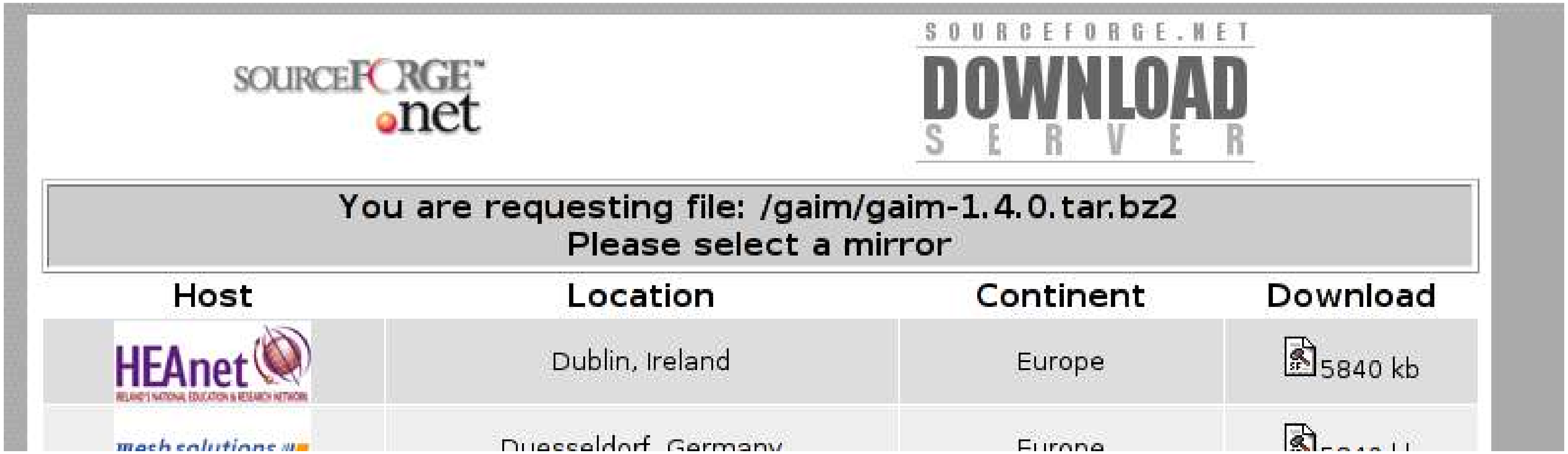
ftp.heanet.ie

ftp.heanet.ie

- National Mirror Server for Ireland
  - http://ftp.heanet.ie/about/
  - http://ftp.heanet.ie/status/
- Also used for Network/Systems development
  - IPv6
  - Apache 2.0/2.1/2.2
- Give back to OpenSource community
  - And get free T-Shirts
- Relatively small budget (50k Euro Vs 400k Euro)

ftp.heanet.ie

- Mirror for;
  - Apache, Sourceforge, Debian, FreeBSD, RedHat, Fedora, Slackware, Ubuntu, NASA Worldwinds, Mandrake, SuSe, Gentoo, Linux, OpenBSD, NetBSD ... and so on



You are requesting file: /gaim/gaim-1.4.0.tar.bz2
Please select a mirror

Host	Location	Continent	Download
HEAnet IRELAND'S NATIONAL EDUCATION & RESEARCH NETWORK	Dublin, Ireland	Europe	5840 kb
mesh solutions	Duesseldorf, Germany	Europe	5840 kb

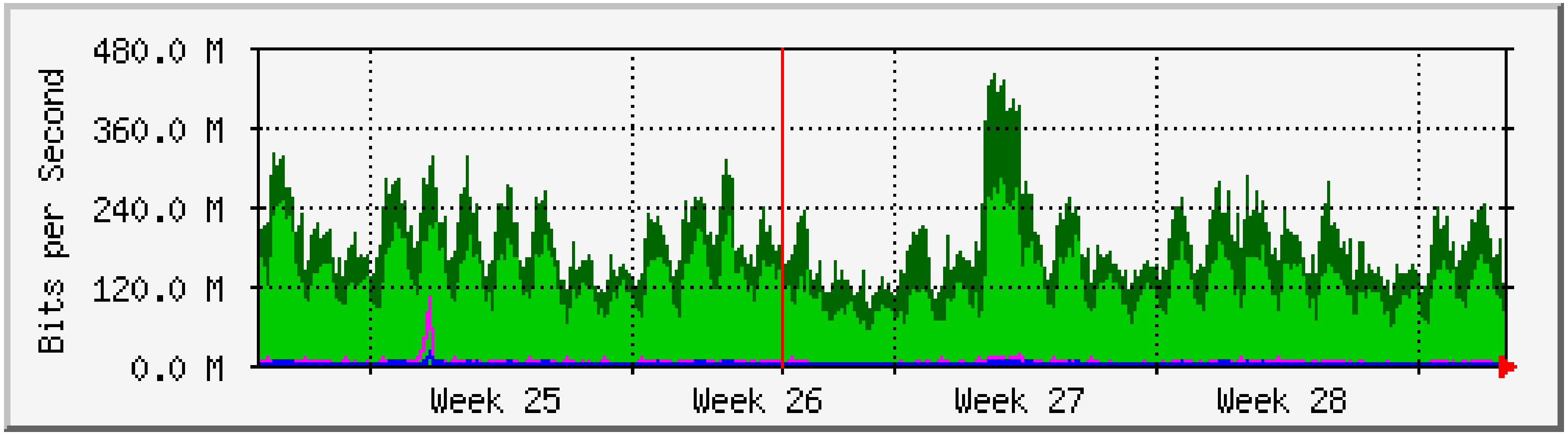
The Numbers

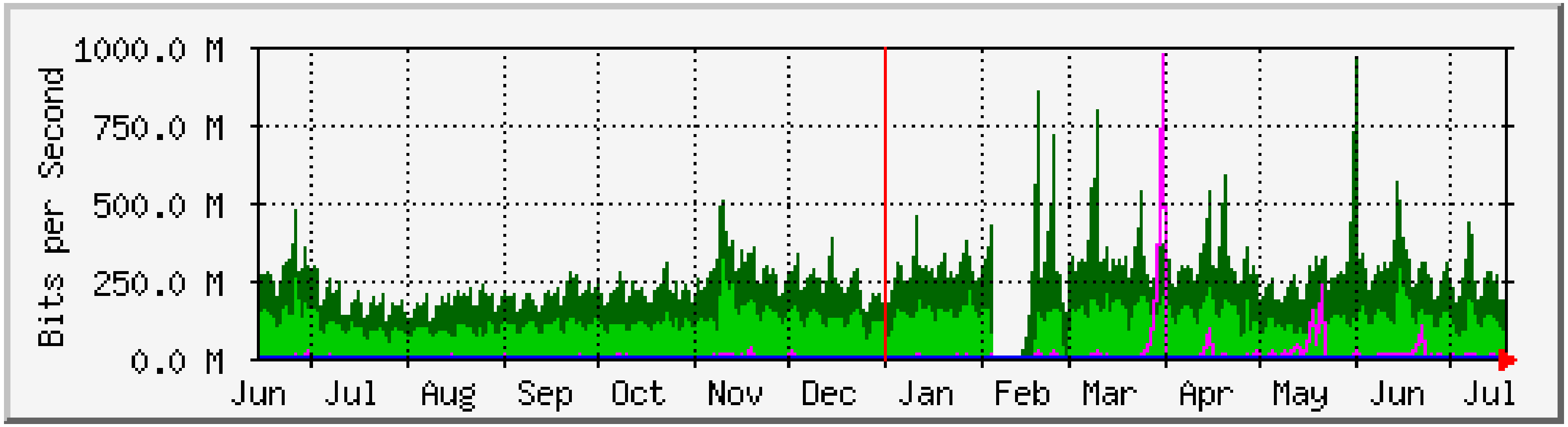
- > 27,000 concurrent downloads from 1 webserver, in production
- 984Mbit/sec, in production. 4Gbit/sec in testing.
- Roughly 80% of all Sourceforge downloads from April ’03 to April ’04
- Usually 4 times busier than ftp.kernel.org
- 7 Free T-Shirts (RedHat and Sourceforge)

The Numbers: a day

- 10,753,084 files stored
- 4.53 Terabytes
- 3,011,067 downloads
- 3.4TB shipped

The Numbers


Bits per Second
480.0 M
360.0 M
240.0 M
120.0 M
0.0 M
Week 25
Week 26
Week 27
Week 28


Bits per Second
1000.0 M
750.0 M
500.0 M
250.0 M
0.0 M
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul

Resources

- http://www.kegel.com/c10k.html
- http://httpd.apache.org/
- http://www.csn.ul.ie/~mel/projects/vm/
- Kernel sources
- Tuning/NFS/high-availability HOWTO’s

Method

- Research the principles behind the software involved
- Configure, test, benchmark, repeat

- Configure, test, benchmark, repeat

Benchmarking

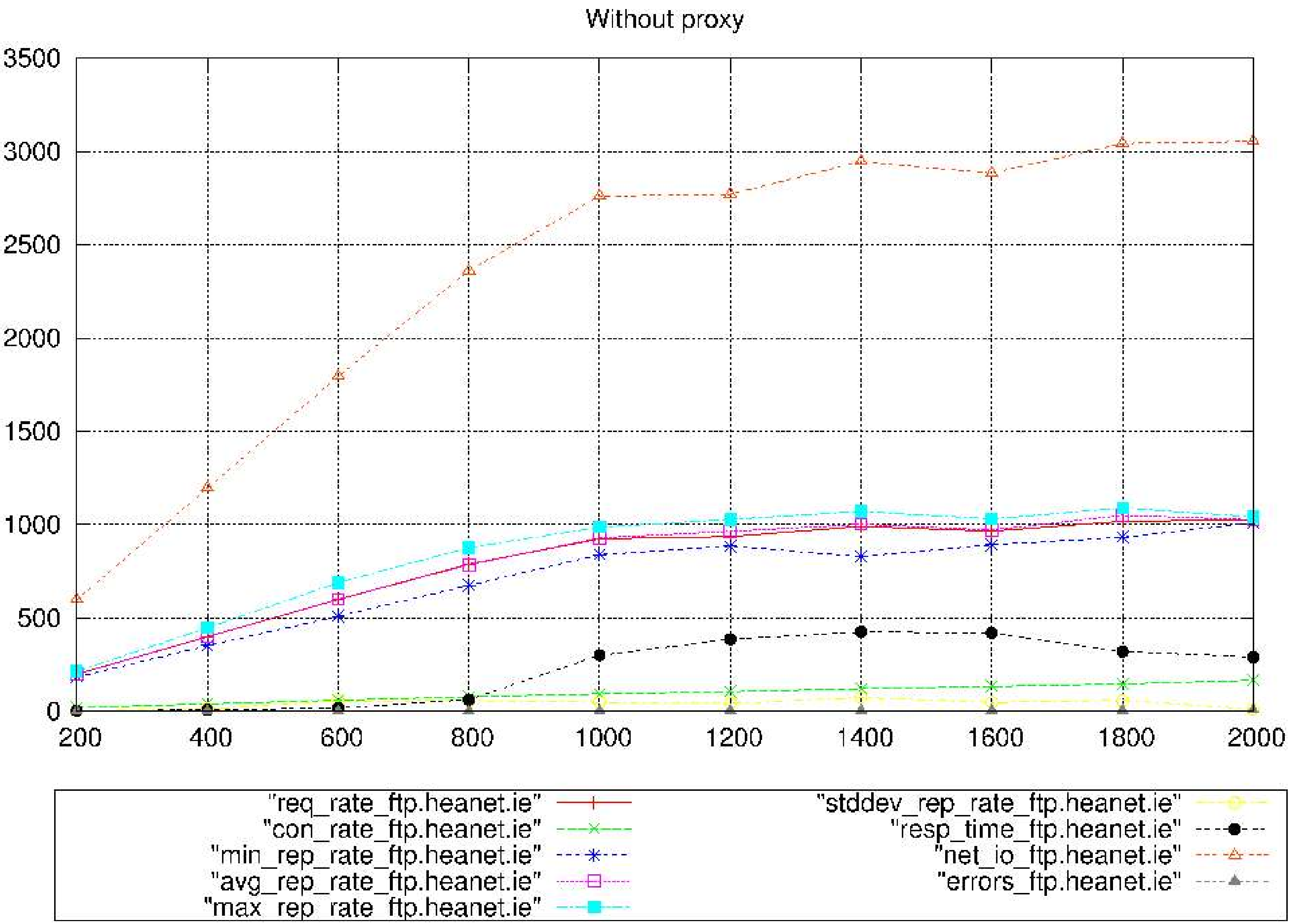
- Webserver benchmarking:
  - apachebench, httperf, autobench
  - most important benchmark and can be used for measuring any system changes.

- Use common files for benchmarking
  - /pub/heanet/100.txt
  - /pub/heanet/1000.txt
  - /pub/heanet/10000.txt

Benchmarking

- ab gives a good quick overview of current server performance.

- httperf + autobench stress-tests the webserver to determine maximum response rate, detect any errors and so on. Produces useful graphs.


Without proxy
3500
3000
2500
2000
1500
1000
500
0
200
400
600
800
1000
1200
1400
1600
1800
2000
"req_rate_ftp.heanet.ie"
"con_rate_ftp.heanet.ie"
"min_rep_rate_ftp.heanet.ie"
"avg_rep_rate_ftp.heanet.ie"
"max_rep_rate_ftp.heanet.ie"
"stddev_rep_rate_ftp.heanet.ie"
"resp_time_ftp.heanet.ie"
"net_io_ftp.heanet.ie"
"errors_ftp.heanet.ie"

Benchmarking Filesystems

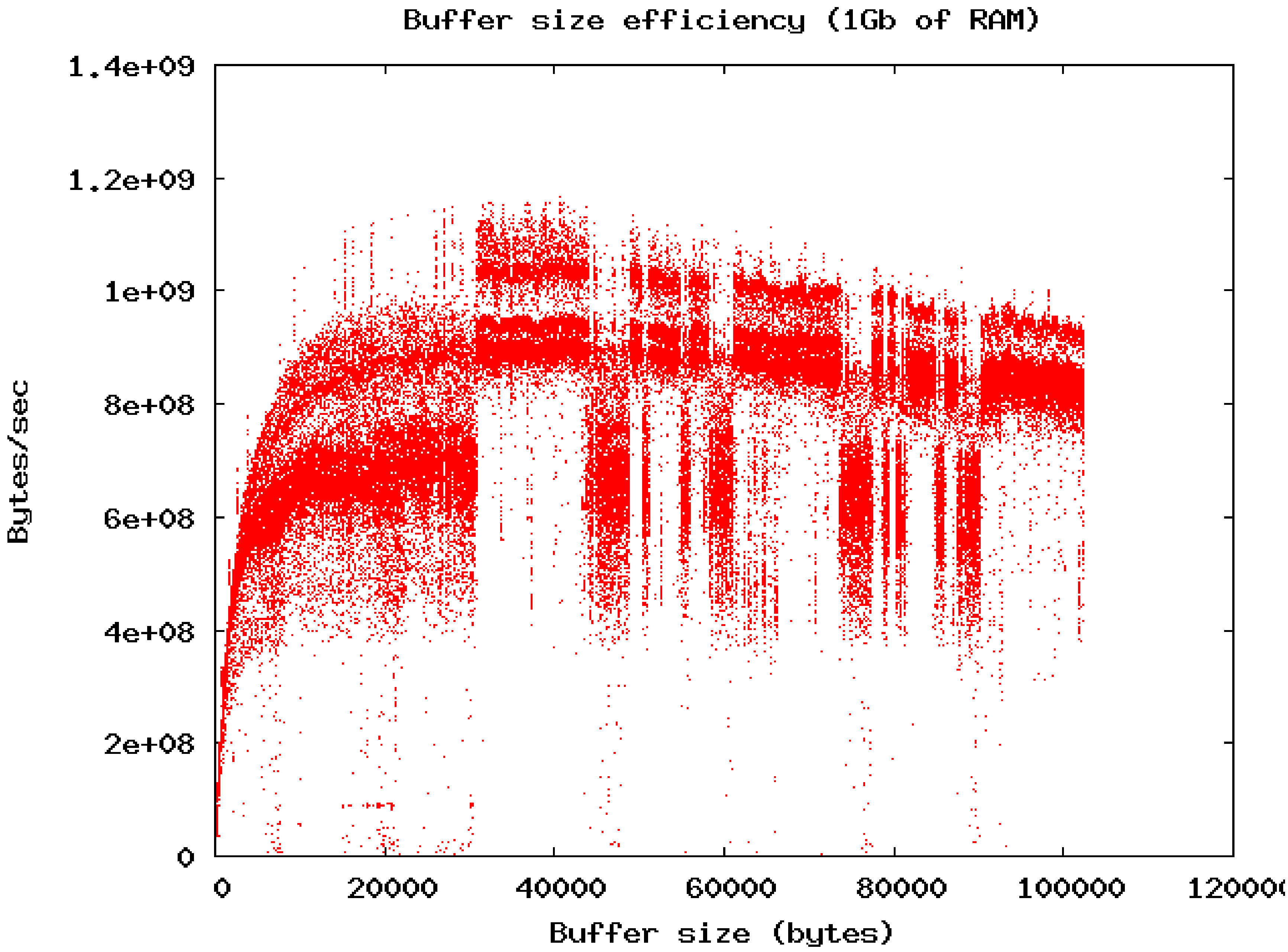
- IOzone, postmark, bonnie++
- Postmark aimed at simulating mail-server load. May be suitable for some webservers, but unlikely.
- IOzone is extensive and thorough
- bonnie++ is simple to understand and sufficient for most needs

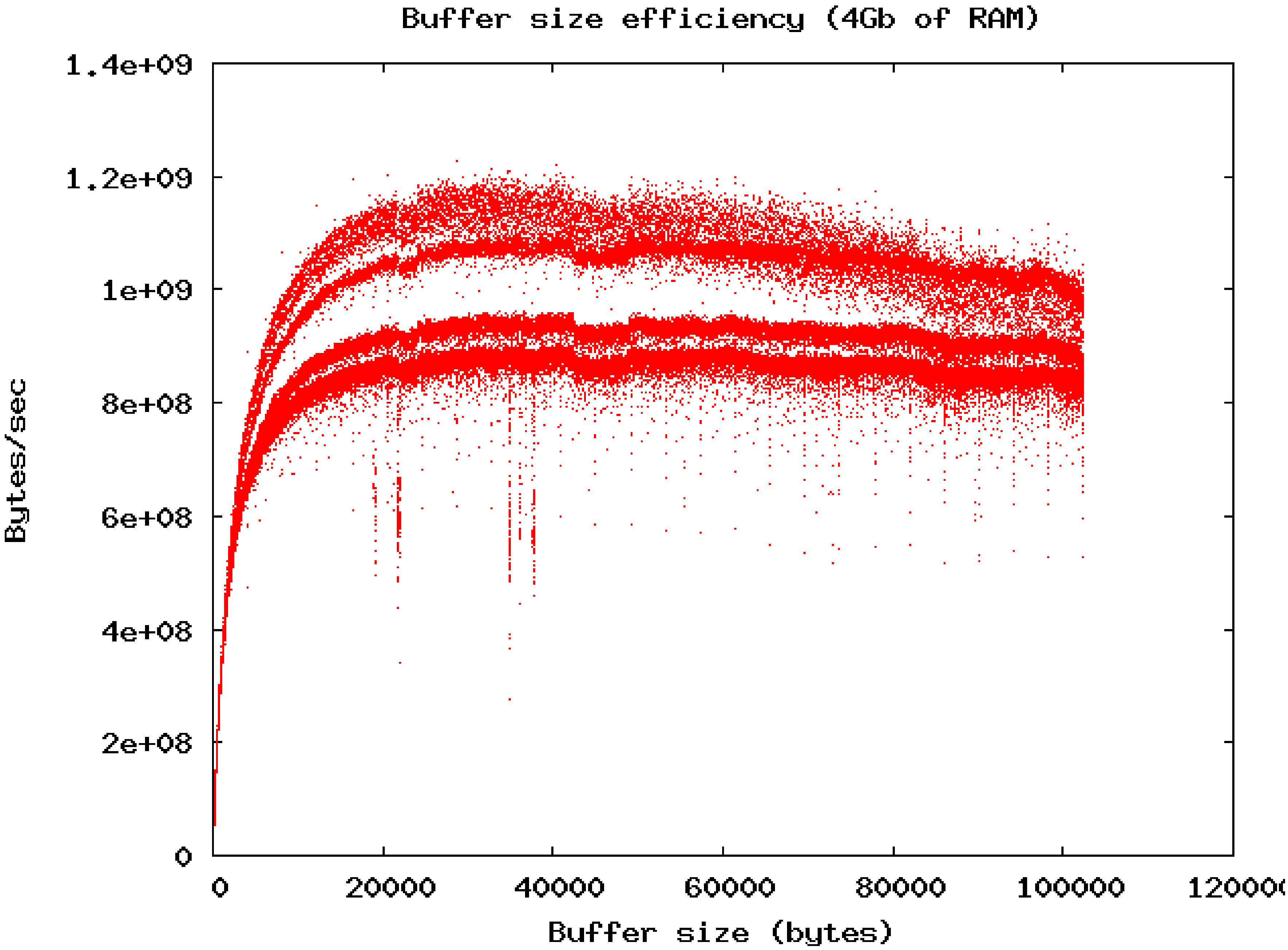
VM and scheduler benchmarking

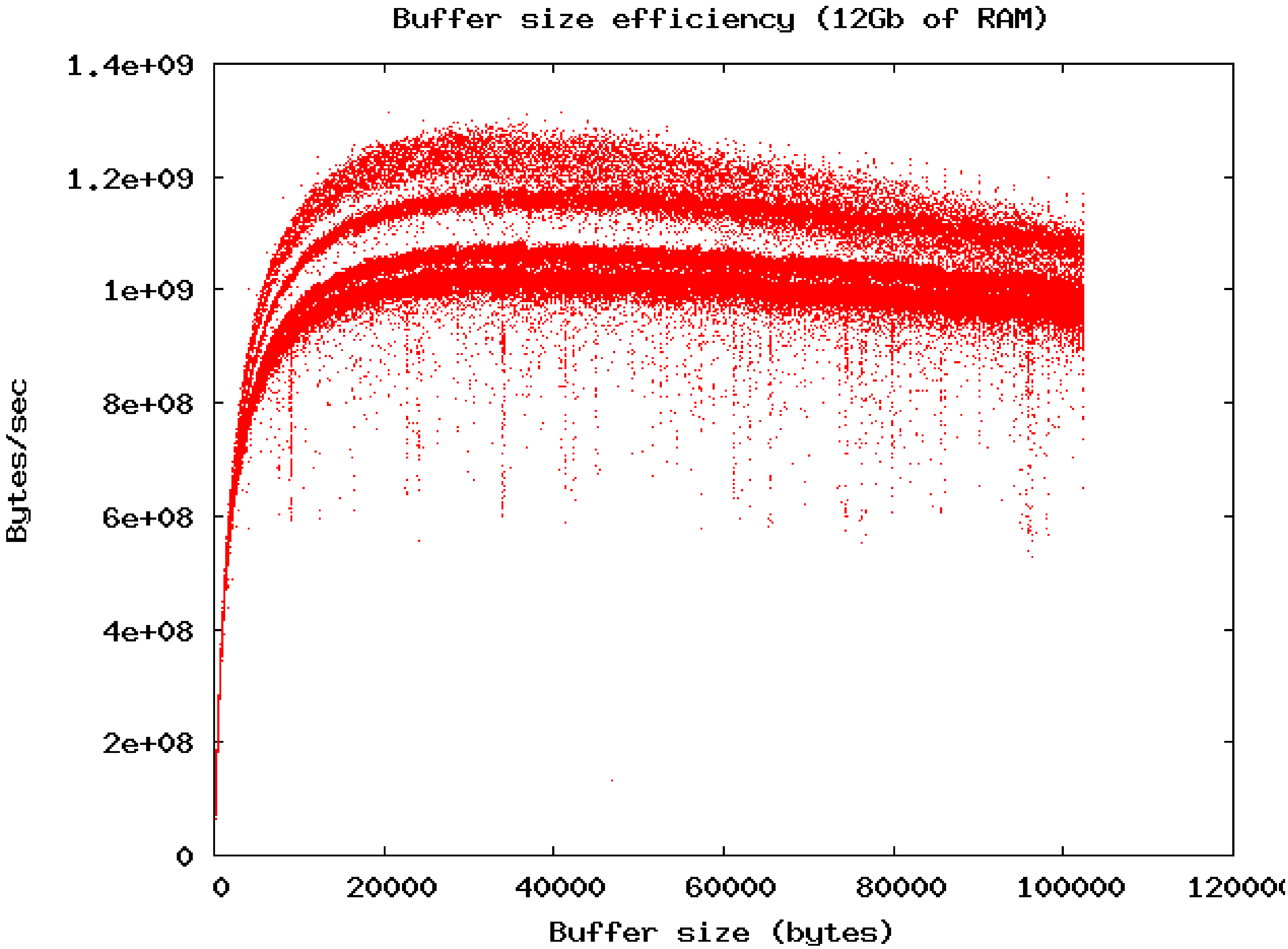
- No generic tools for benchmarking schedulers and memory managers
- Benchmarks usually consist of compiling a kernel, benchmarking a webserver, etc
- To judge the effect of the VM and scheduler on I/O, we use dder.sh

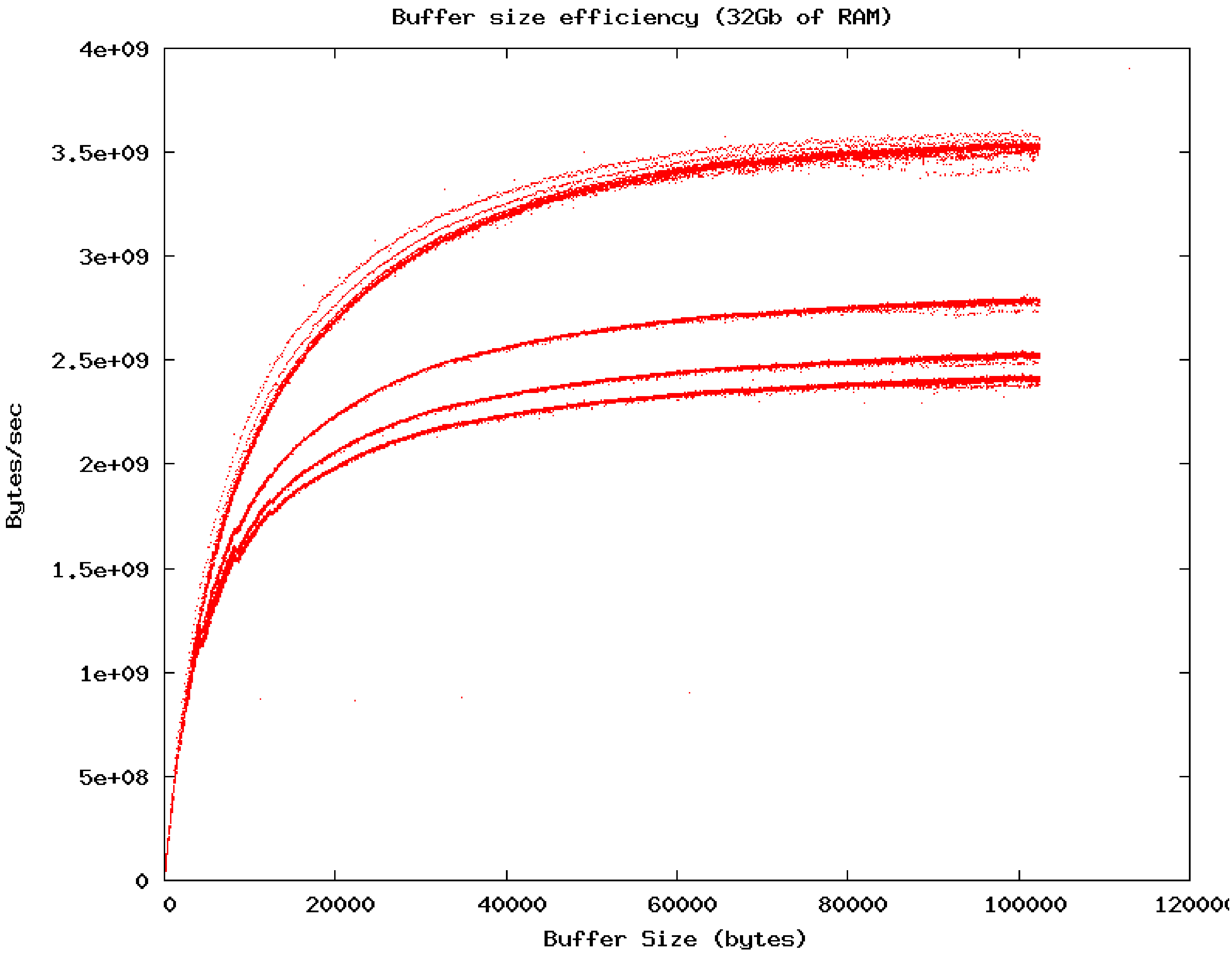
Scheduler and VM

```
#!/bin/sh

STARTNUM="1"
ENDNUM="102400"

# create a 100 MB file
dd bs=1024 count=102400 if=/dev/zero of=local.tmp

# Clear the record
rm -f record

# Find the most efficient size
for size in `seq $STARTNUM $ENDNUM`; do
        dd bs=$size if=local.tmp of=/dev/null                2>> record
done

# get rid of junk
grep  "transf" record | awk '{ print $7 }' | cut -b 2- | cat -n | \
while read number result ; do
        echo -n $(( $number + $STARTNUM - 1 ))
        echo " " $result
done > record.sane
```


Buffer size efficiency (1Gb of RAM)
1.4e+09
1.2e+09
1e+09
8e+08
6e+08
4e+08
2e+08
0
Bytes/sec
0
20000
40000
60000
80000
100000
12000
Buffer size (bytes)


Buffer size efficiency (4Gb of RAM)
Bytes/sec
1.4e+09
1.2e+09
1e+09
8e+08
6e+08
4e+08
2e+08
0
0
20000
40000
60000
80000
100000
12000
Buffer size (bytes)


Buffer size efficiency (12Gb of RAM)
Bytes/sec
1.4e+09
1.2e+09
1e+09
8e+08
6e+08
4e+08
2e+08
0
0
20000
40000
60000
80000
100000
12000
Buffer size (bytes)


Buffer size efficiency (32Gb of RAM)
4e+09
3.5e+09
3e+09
2.5e+09
2e+09
1.5e+09
1e+09
5e+08
0
Bytes/sec
0
20000
40000
60000
80000
100000
12000
Buffer Size (bytes)

Tuning Apache 2.x

- Choosing an MPM
  - Run with various different ones, measure with benchmark utilities. For our load, the prefork MPM came out on top by a margin of 20%

- Static Vs DSO
  - Very small difference (0.2%) in favour of compiled-in static modules.

Tuning Apache 2.x

```
<IfModule prefork.c>
        StartServers                    100
        MinSpareServers                 10
        MaxSpareServers                 10
        ServerLimit                     50000
        MaxClients                      50000
        MaxRequestsPerChild             2000
</IfModule>
```

Tuning Apache 2.x

```
<Directory "/ftp/">
        Options Indexes FollowSymLinks
        AllowOverride None
        Order allow,deny
        Allow from all
        IndexOptions NameWidth=* +FancyIndexing \
                    +SuppressHTMLPreamble +XHTML
</Directory>
```

Tuning Apache 2.x

- Sendfile
  - Enabled if found, however broken on Linux with IPv6 (checksum offloading bug).
- Mmap
  - Next best thing, allows Apache to treat files as contiguous memory, kernel handle's reading.

Tuning Apache 2.x

- mod_cache
  - Experimental in 2.0, occasional bugs in 2.1, not for everyone, but very useful nevertheless.
  - Not just for proxies, allows webserver to cache files as they are sent for the first time.
  - Thus many reads from a slow filesystem can be avoided

Tuning Apache 2.x

- mod_mem_cache
  - can use memory to cache file content, however on Linux the VM caches aggressively anyway
- mod_disk_cache
  - Can use filesystem directory as cache.
  - By using 4x 36Gb 15K RPM SCSI disks in a RAID0 configuration we can speed up read() speed very much.

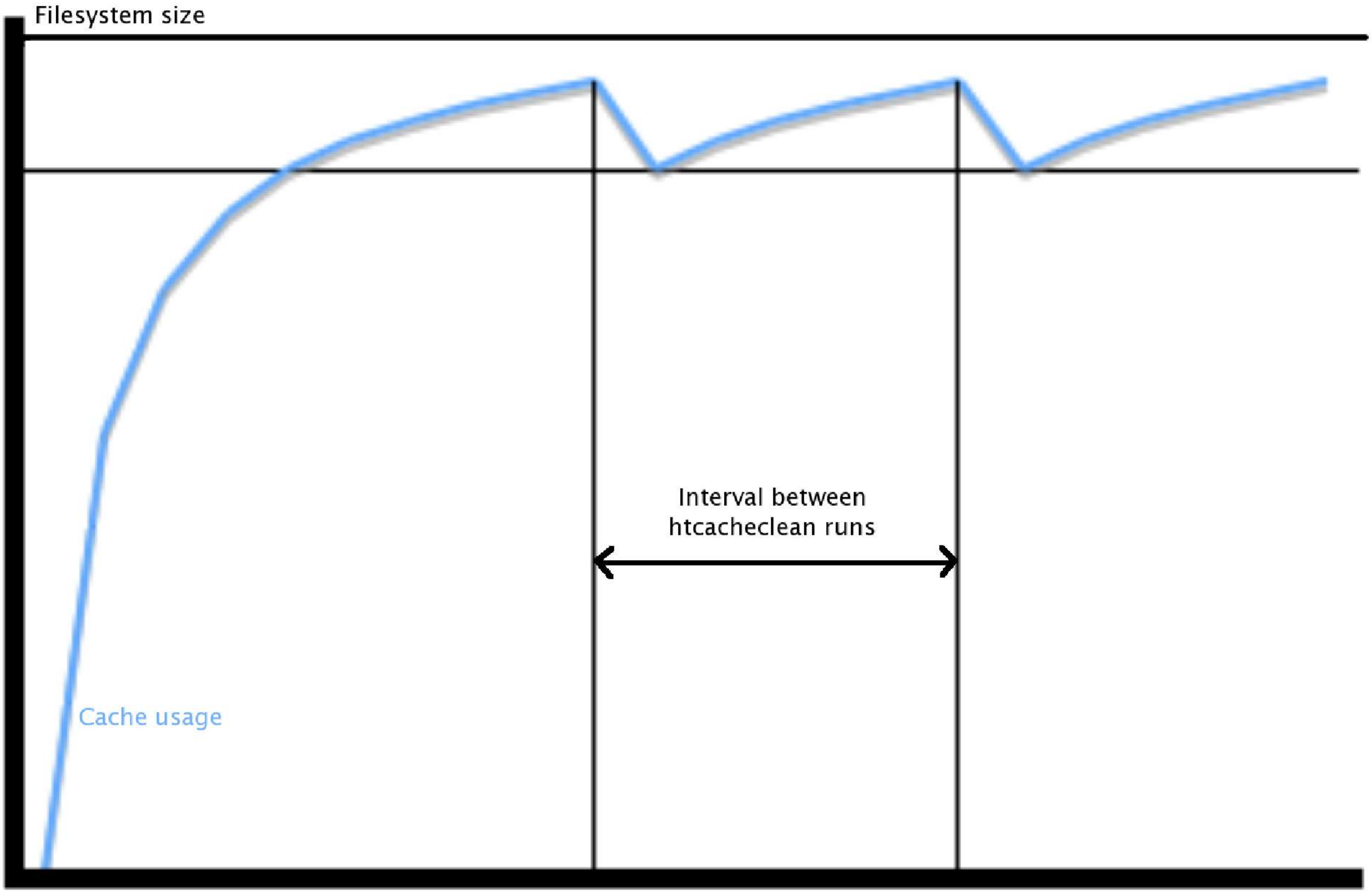
Tuning Apache 2.x

```
<IfModule mod_cache.c>
        <IfModule mod_disk_cache.c>
          CacheRoot /usr/local/apache2/cache/
          CacheEnable disk /
          CacheDirLevels 5
          CacheDirLength 3
        </IfModule>
</IfModule>
```

Tuning Apache 2.x

- Cache cleaning doesn't work in 2.0.
    - Brutal combination of find, xargs and rm is one option.
    - Use htcacheclean from 2.1

- htcacheclean runs periodically and prunes down to a target size. Important to ensure there is “grow” room


Filesystem size
Interval between
htcacheclean runs
Cache usage

htcacheclean

- Deletes files somewhat arbitrarily
- noatime is a valueable mount option
- Hack:
  - Second filesystem (ramfs) with atime
  - mod_disk_cache hack to create 0-byte files there also
  - find | xargs ls -u | sort -rn | head | rm

Tuning the Operating System

- Choose a kernel:
  - 2.6 Vs 2.6-mm? Vs 2.4
  - 2.6 kernel is MUCH better, allows > 20,000 processess in production
  - 2.4 limits at about 11,000
  - 2.6-mm was needed for a while, but most patches in now. 2.6.11 found be most stable yet.

Tuning the Operating System

- Tuning a filesystem
  - Always mount with noatime, can double read speed.
  - XFS: use logbufs=8, ihashsize=65567 mount options
  - Ext3: set blocksize to 4096, use dir_index build option

Tuning the Operating System

- Tuning NFS
  - Use jumboframes if possible, increase rsize and wsize accordingly. Increase the number of NFS threads available on the server side.
  - Use nolock mount option on the clients if they will not be doing any writing.

Tuning the Operating System

- Tuning the VM
  - Linux VM uses similar approach to mod_disk_cache for freeing space.
  - Allocate memory to processes genourously and prune back to target level periodically
  - The VM also caches file data aggresively.
  - If a lot of files are being served quickly, easy to fill memory and generate OOM

Tuning the Operating System

- Tuning the VM

```
vm/min_free_kbytes = 204800
vm/lower_zone_protection = 1024
vm/page-cluster = 20
vm/swappiness = 200
vm/vm_vfs_scan_ratio = 2
```

- Other sysctls:

```
fs/file-max=5049800
```

Tuning the Operating System

- Tuning the Networking stack

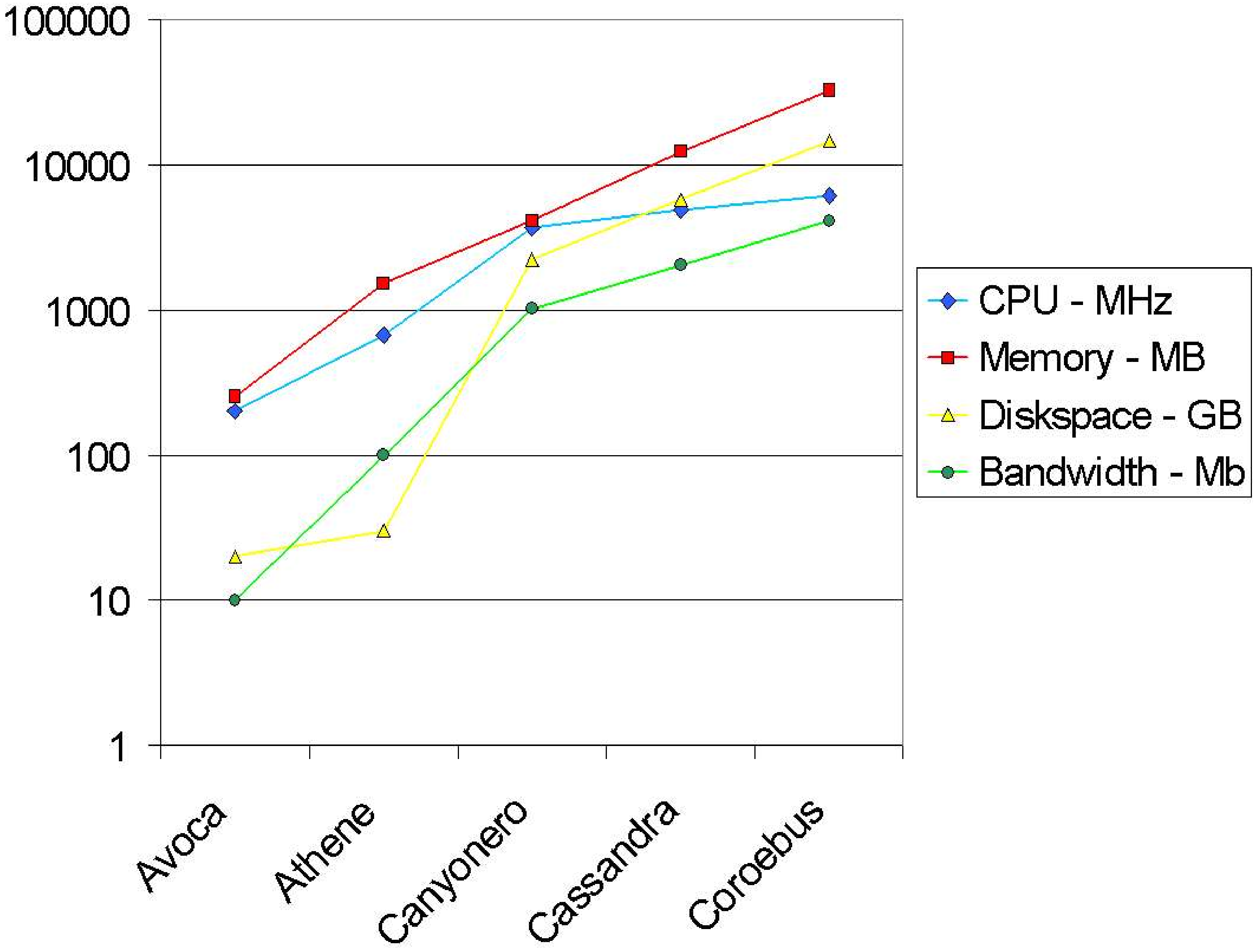
```
net/ipv4/tcp_rfc1337=1
net/ipv4/tcp_syncookies=1
net/ipv4/tcp_keepalive_time = 300
net/ipv4/tcp_max_orphans=1000
sys/net/core/rmem_default=262144
sys/net/core/rmem_max=262144
```

System Design

- RAM intensive, buy lots
- Bounce buffering and PAE means CPU hit, buy lots
- Fast (15k RPM) system disks for intermediary caching

System Design

Machine	Model	CPU	Memory	Storage	Network
Avoca	Alphaserver	200Mhz	256Mb	20Gb	10Mbit
Athene	Alphaserver DS20E	667Mhz	1.5Gb	30Gb	100Mbit
Canyonero	Dell 2650	2x 1.8Ghz	4Gb	2.2Tb	1Gb
Cassandra	Dell 2650	2x 2.4Ghz	12Gb	5.6Tb	2Gb
Coroebus	Dell 7250	2x 1.5Ghz	32Gb	14.2Tb	4Gb


100000
10000
1000
100
10
1
Avoca
Athene
Canyonero
Cassandra
Coroebus
CPU - MHz
Memory - MB
Diskspace - GB
Bandwidth - Mb

Future directions

- Multicast services
- Jumboframes
- mod_ftp(d) and reverse proxies
- Itanium platform
- mod_bittorrent?

Other talks

- TH14: What's new in HTTPD 2.2
- TH17: Caching, Tips for Improving Performance
- FR09: Clustering and Load-balancing using mod_proxy

Questions?

?