PostgreSQL Portland Performance Practice Project

Database Test 2 (DBT-2)

Characterizing Filesystems

Mark Wong
markwkm@postgresql.org

Portland State University

April 9, 2009

Review

From last time:

- DBT-2 live demo.

This is:

- A collection of reason why you should test your own system.
- A guide for how to understand the i/o behavior of your own system.
- A model of simple i/o behavior based on PostgreSQL

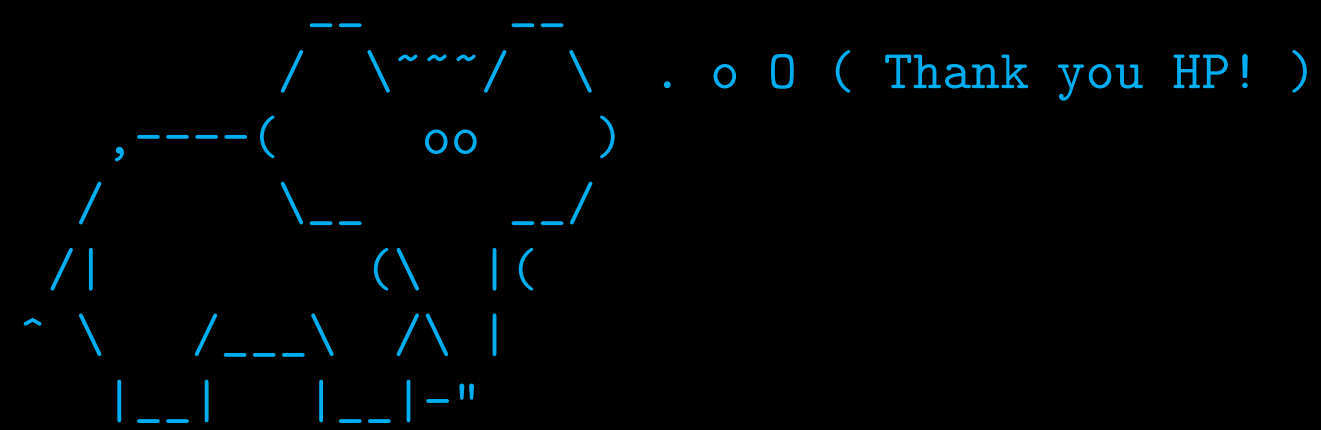
This is not:

- A comprehensive Linux filesystem evaluation.
- A filesystem tuning guide (mkfs and mount options).
- Representative of your system (unless you have the exact same hardware.)
- A test of reliability or durability.

Contents

Some hardware details

- HP DL380 G5
- 2 x Quad Core Intel(R) Xeon(R) E5405
- HP 32GB Fully Buffered DIMM PC2-5300 8x4GB DR LP Memory
- HP Smart Array P800 Controller
- MSA70 with 25 x HP 72GB Hot Plug 2.5 SAS 15,000 rpm Hard Drives



Criteria

PostgreSQL dictates the following (even though some of them are actually configurable:

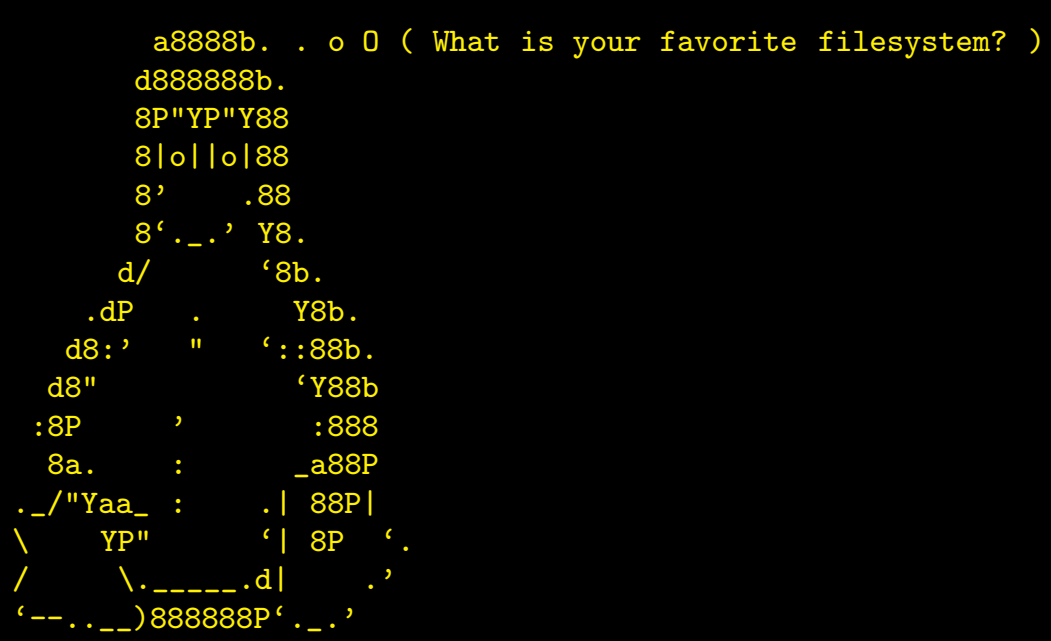
- 8KB block size
- No use of posix_fadvise
- No use of direct i/o
- No use of async i/o
- Uses processes, not threads.

Based on the hardware we determine these criteria:

- 64GB data set - To minimize any caching in the 32GB of system memory.
- 8 processes performing i/o - One fio process per core.

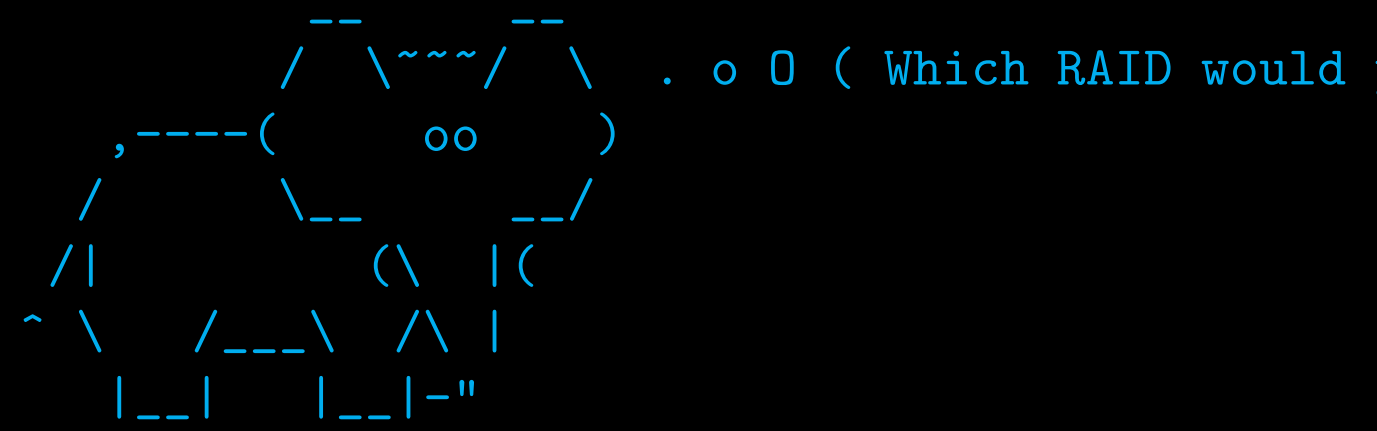
Linux Filesystems Tested

- ext2
- ext3
- ext4
- jfs
- reiserfs
- xfs

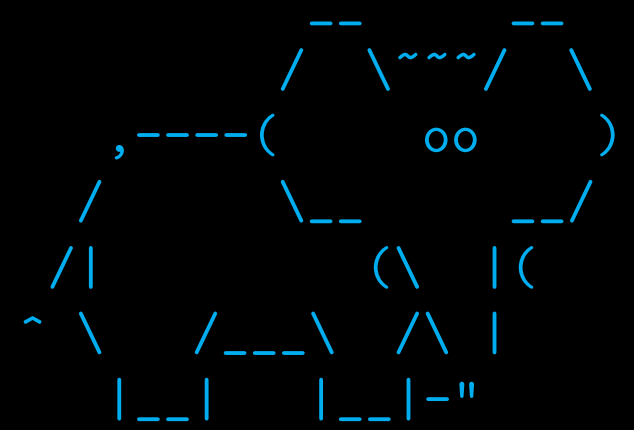


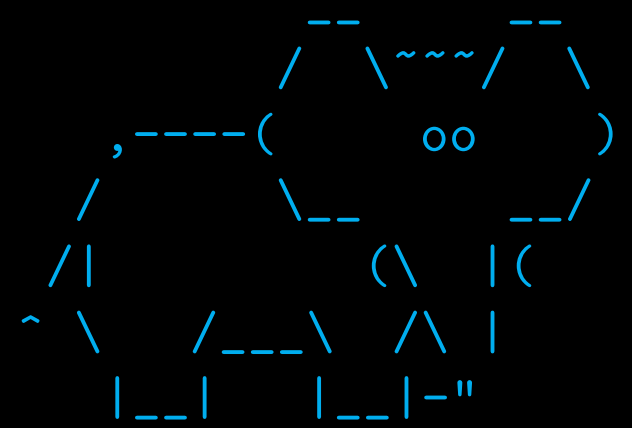
Hardware RAID Configurations

- RAID 0
- RAID 1
- RAID 1+0
- RAID 5
- RAID 6

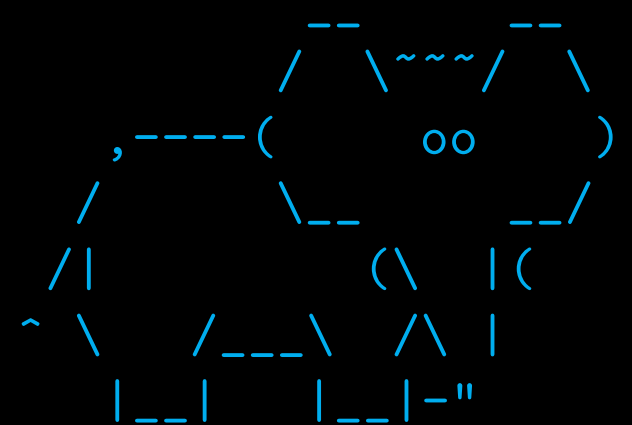


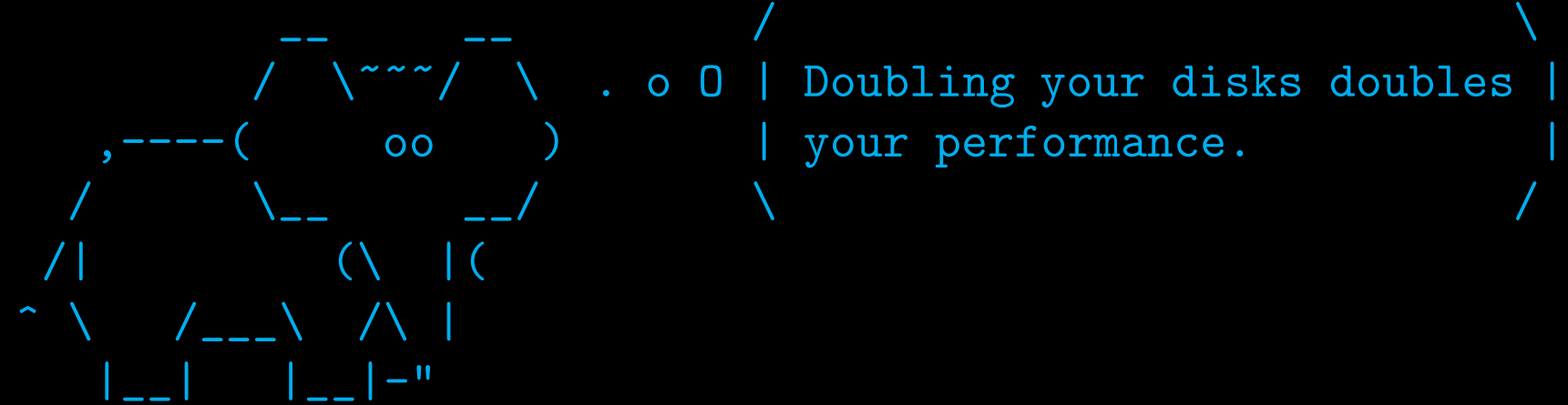
Preconceptions

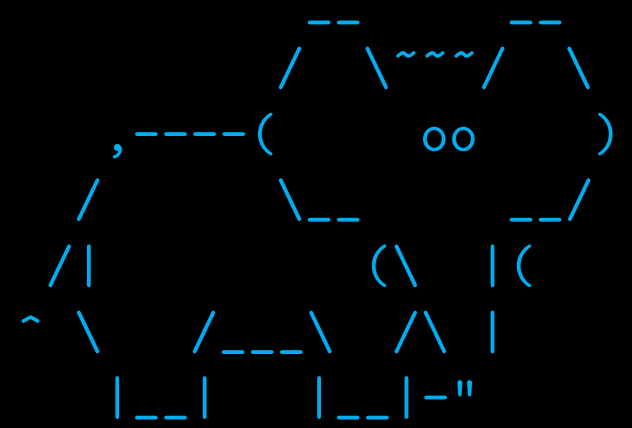




. o O (atime slows us down a lot.)

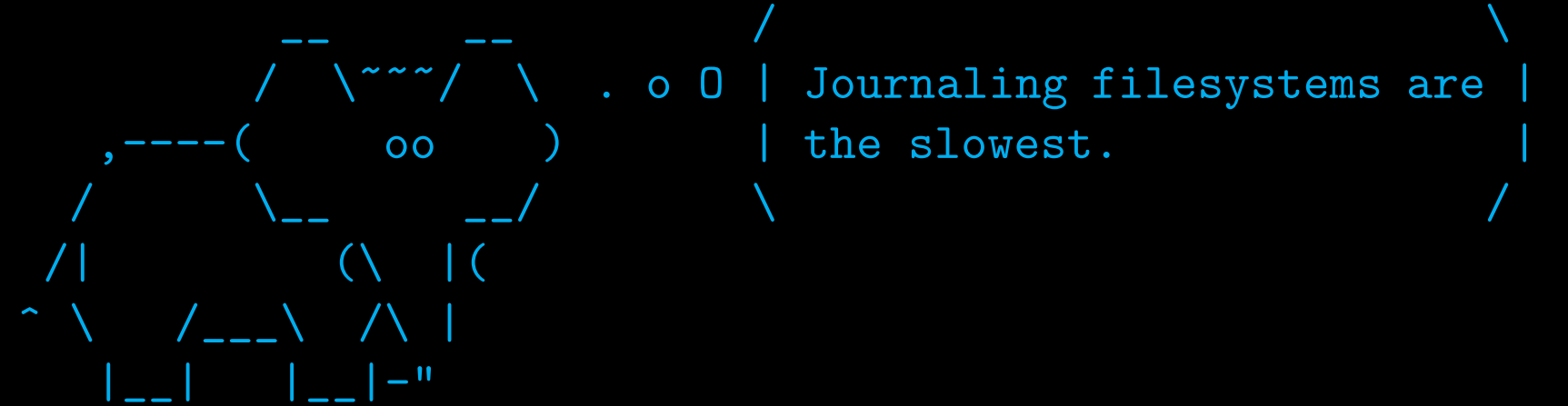

. o O (Partition alignment is huge.)


Doubling your disks doubles
your performance.



. o O (RAID-5 is awful at writes.)

. o O (RAID-6 improves on RAID-5.)


Journaling filesystems are
the slowest.

fio[1]

Flexible I/O by Jens Axboe.

- Sequential Read
- Sequential Write
- Random Read
- Random Write
- Random Read/Write

[1] http://brick.kernel.dk/snaps/

Sequential Read

```
[seq-read]
rw=read
size=8g
directory=/test
fadvise_hint=0
blocksize=8k
direct=0
numjobs=8
nrfiles=1
runtime=1h
exec_prerun=echo 3 > /proc/sys/vm/drop_caches
```

Sequential Write

```
[seq-write]
rw=write
size=8g
directory=/test
fadvise_hint=0
blocksize=8k
direct=0
numjobs=8
nrfiles=1
runtime=1h
exec_prerun=echo 3 > /proc/sys/vm/drop_caches
```

Random Read

```
[random-read]
rw=randread
size=8g
directory=/test
fadvise_hint=0
blocksize=8k
direct=0
numjobs=8
nrfiles=1
runtime=1h
exec_prerun=echo 3 > /proc/sys/vm/drop_caches
```

Random Write

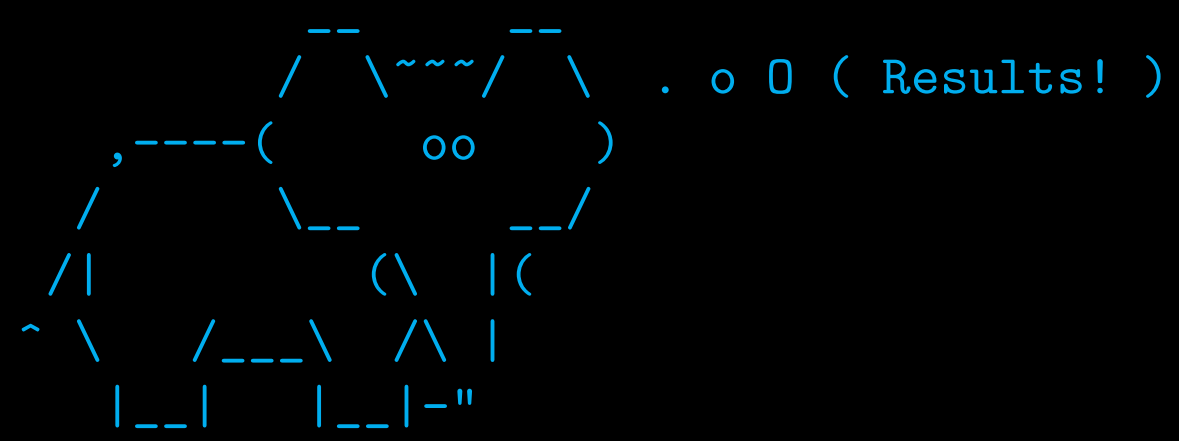
```
[random-write]
rw=randwrite
size=8g
directory=/test
fadvise_hint=0
blocksize=8k
direct=0
numjobs=8
nrfiles=1
runtime=1h
exec_prerun=echo 3 > /proc/sys/vm/drop_caches
```

Random Read/Write

```
[read-write]
rw=rw
rwmixread=50
size=8g
directory=/test
fadvise_hint=0
blocksize=8k
direct=0
numjobs=8
nrfiles=1
runtime=1h
exec_prerun=echo 3 > /proc/sys/vm/drop_caches
```

Versions Tested

Linux 2.6.25-gentoo-r6
fio v1.22
and
Linux 2.6.28-gentoo
fio v1.23



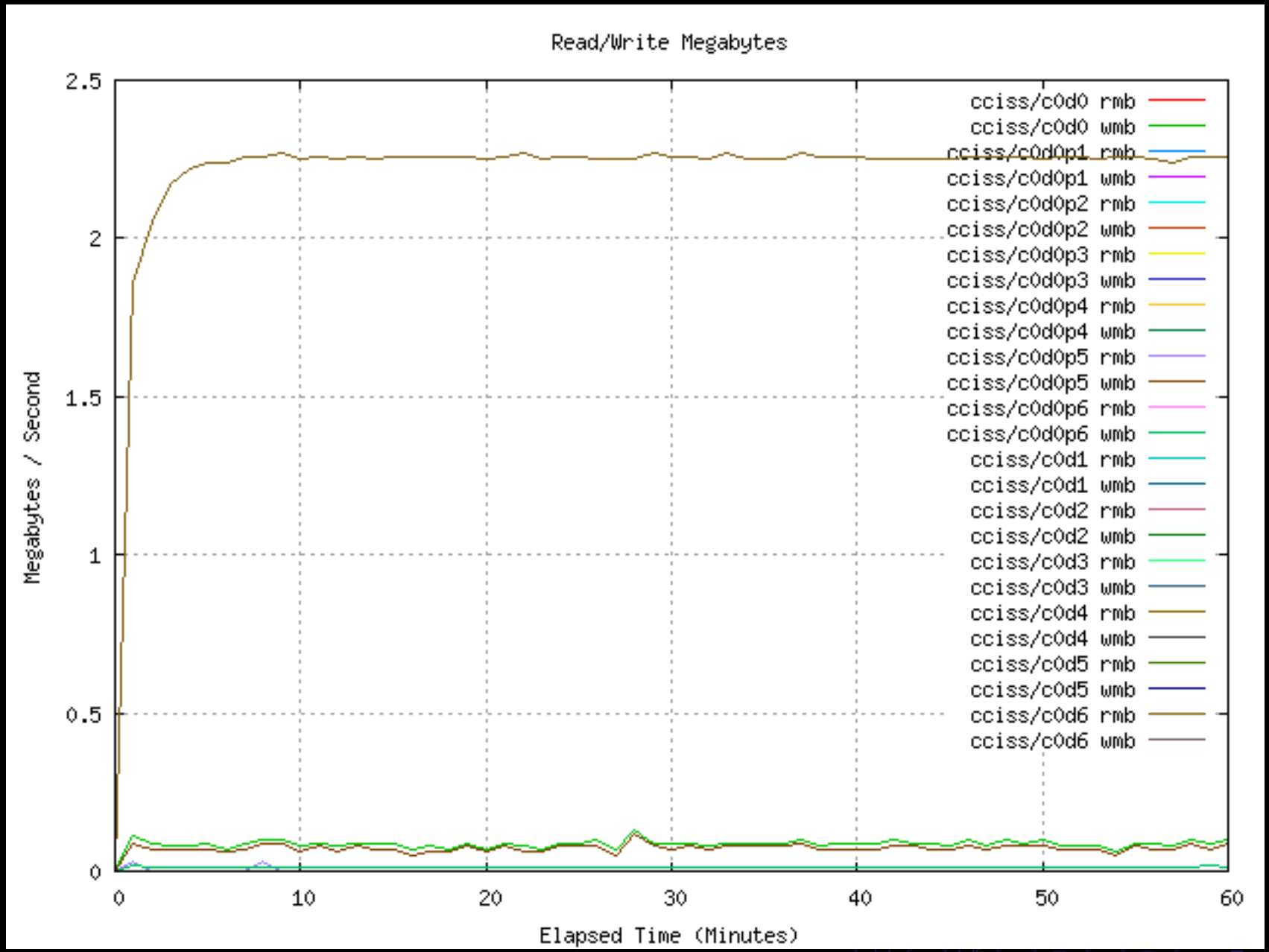
There is more information than what is in this presentation, raw data is at: `http://207.173.203.223/~markwkm/community6/fio/2.6.28/` For example other things of interest may be:

- Number of i/o operations.
- Processor utilization.

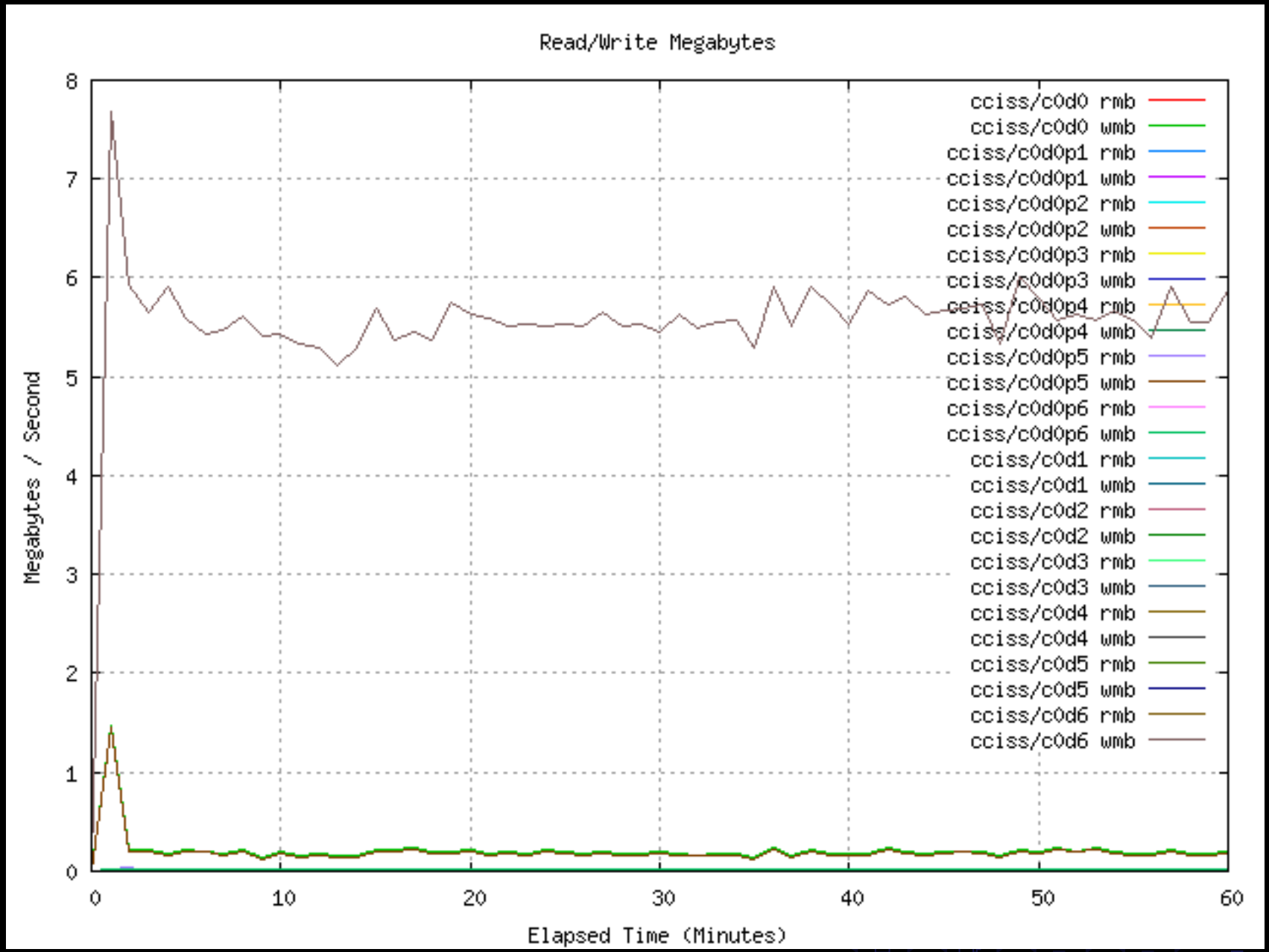
A few words about statistics...

Not enough data to calculate good statistics.
Examine the following charts for a feel of how reliable the results may be.

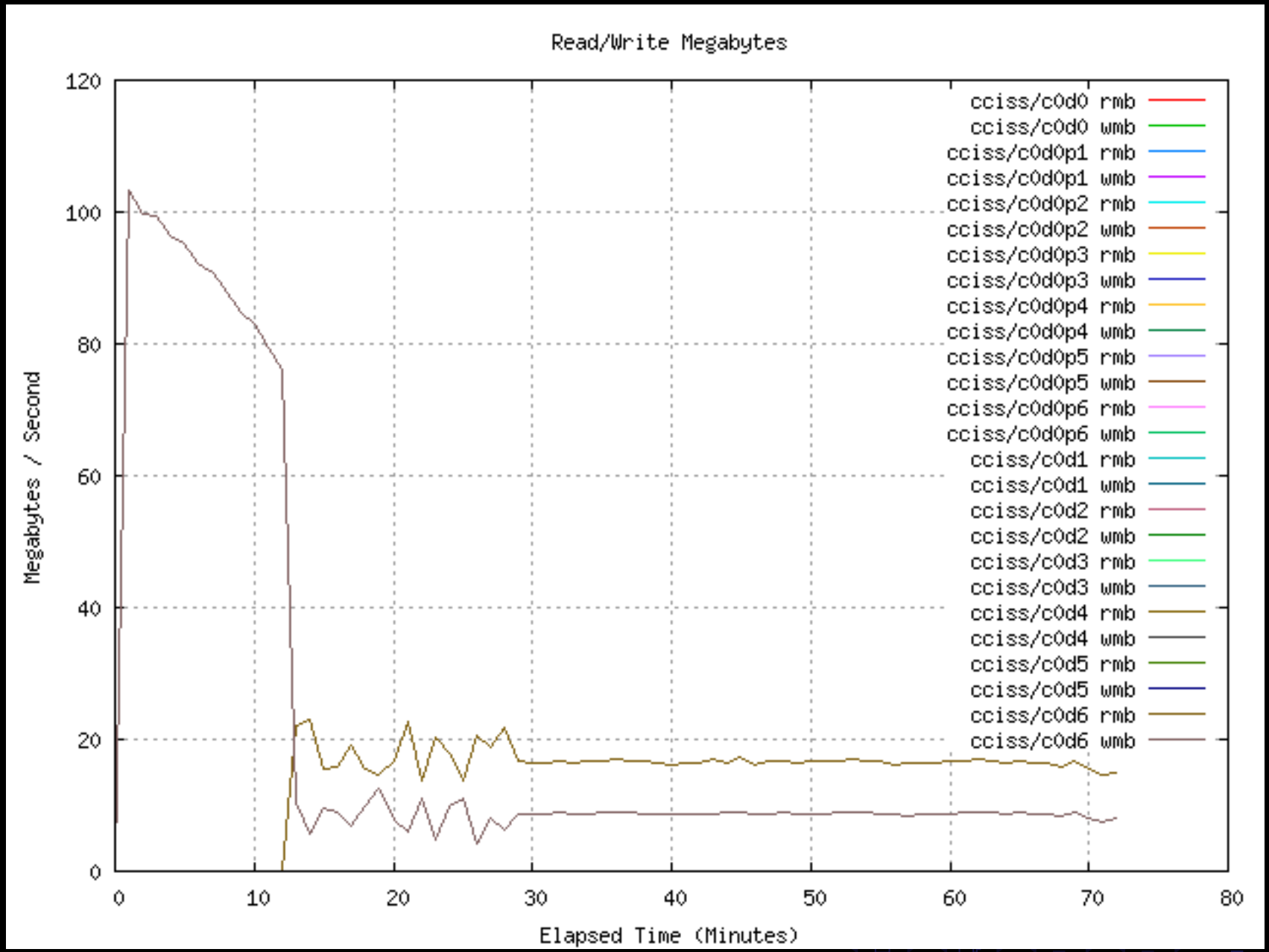
Random Read with Linux 2.6.28-gentoo ext2 on 1 Disk



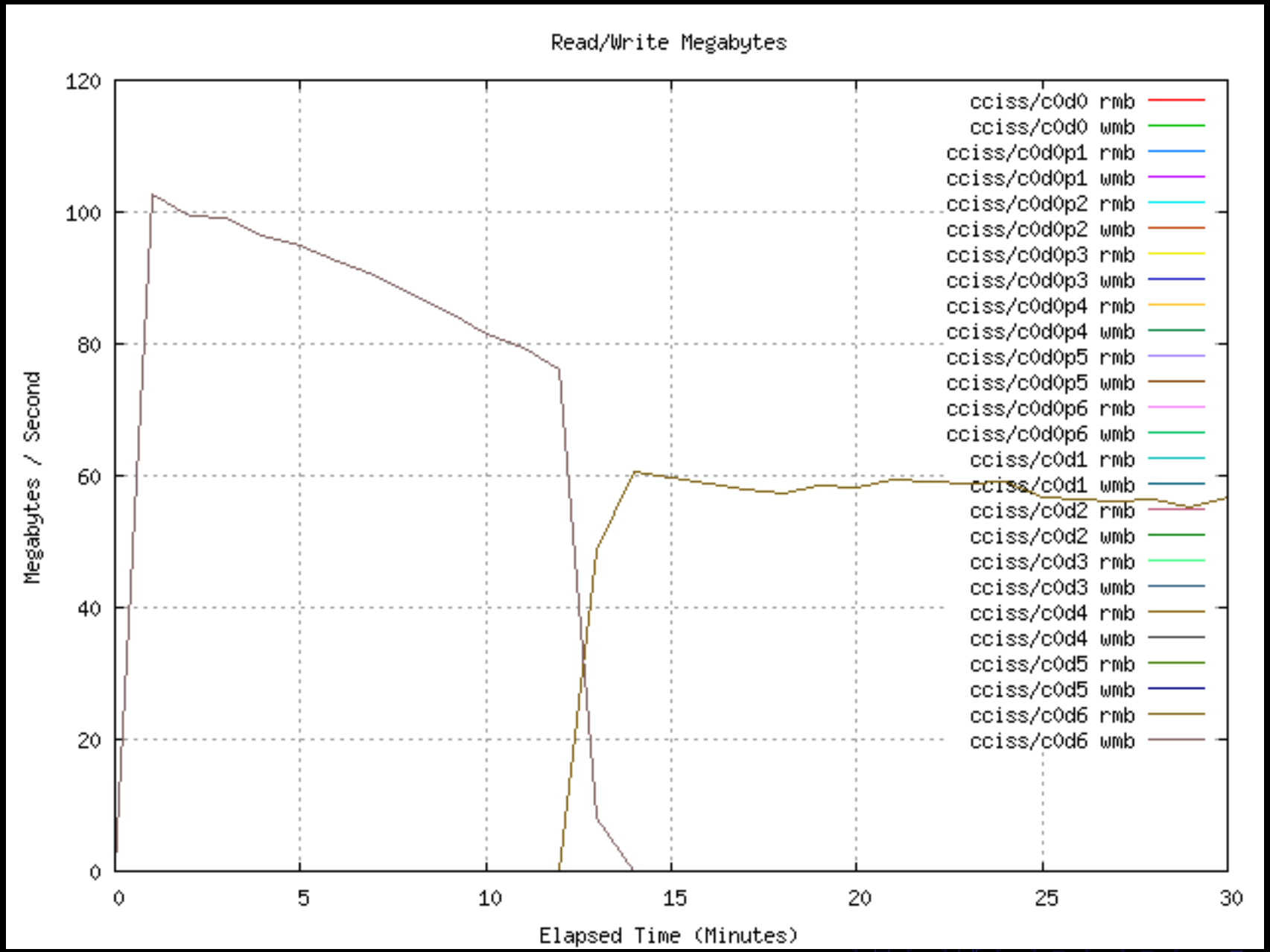
Random Write with Linux 2.6.28-gentoo ext2 on 1 Disk



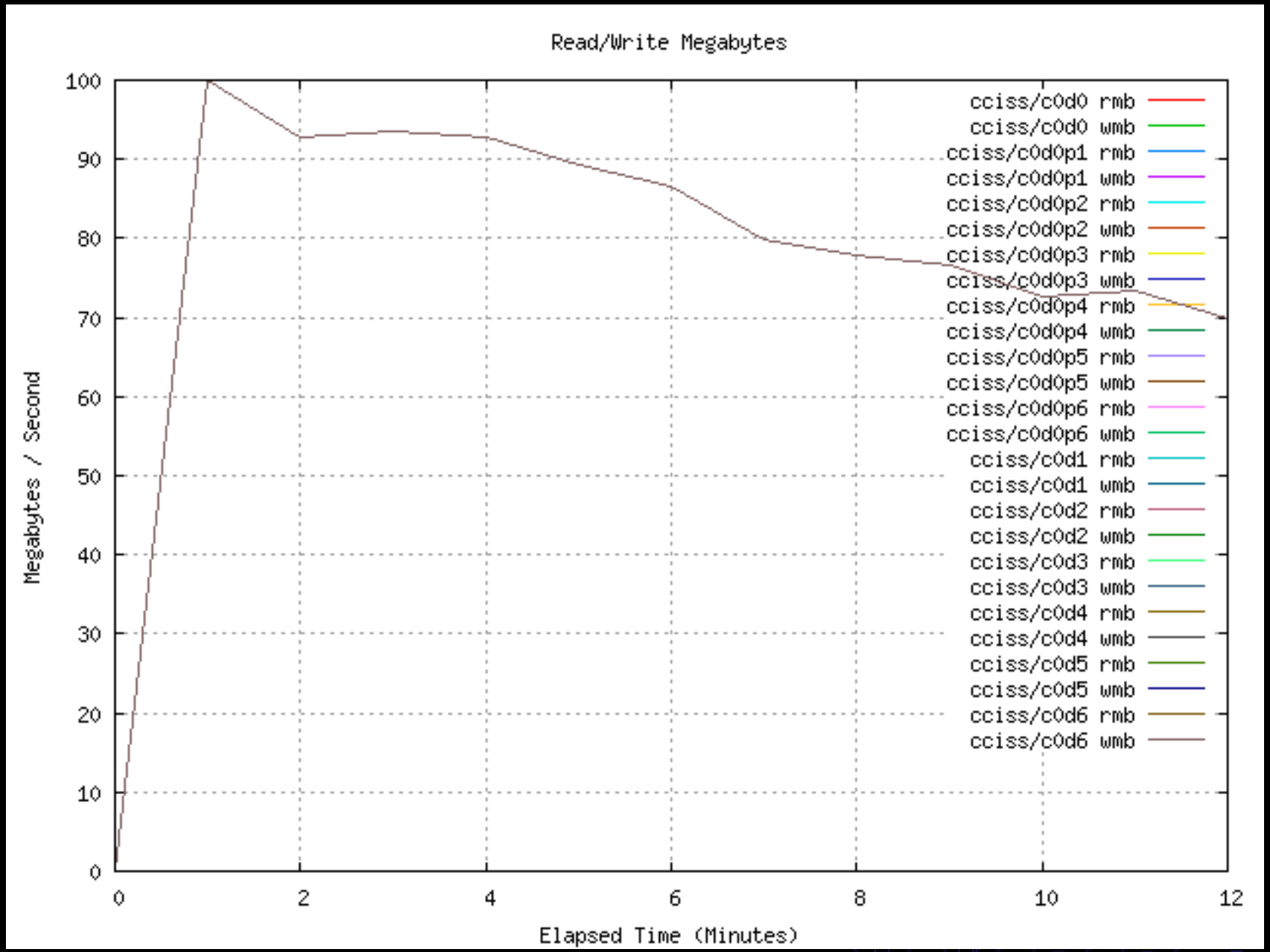
Read/Write Mix with Linux 2.6.28-gentoo ext2 on 1 Disk



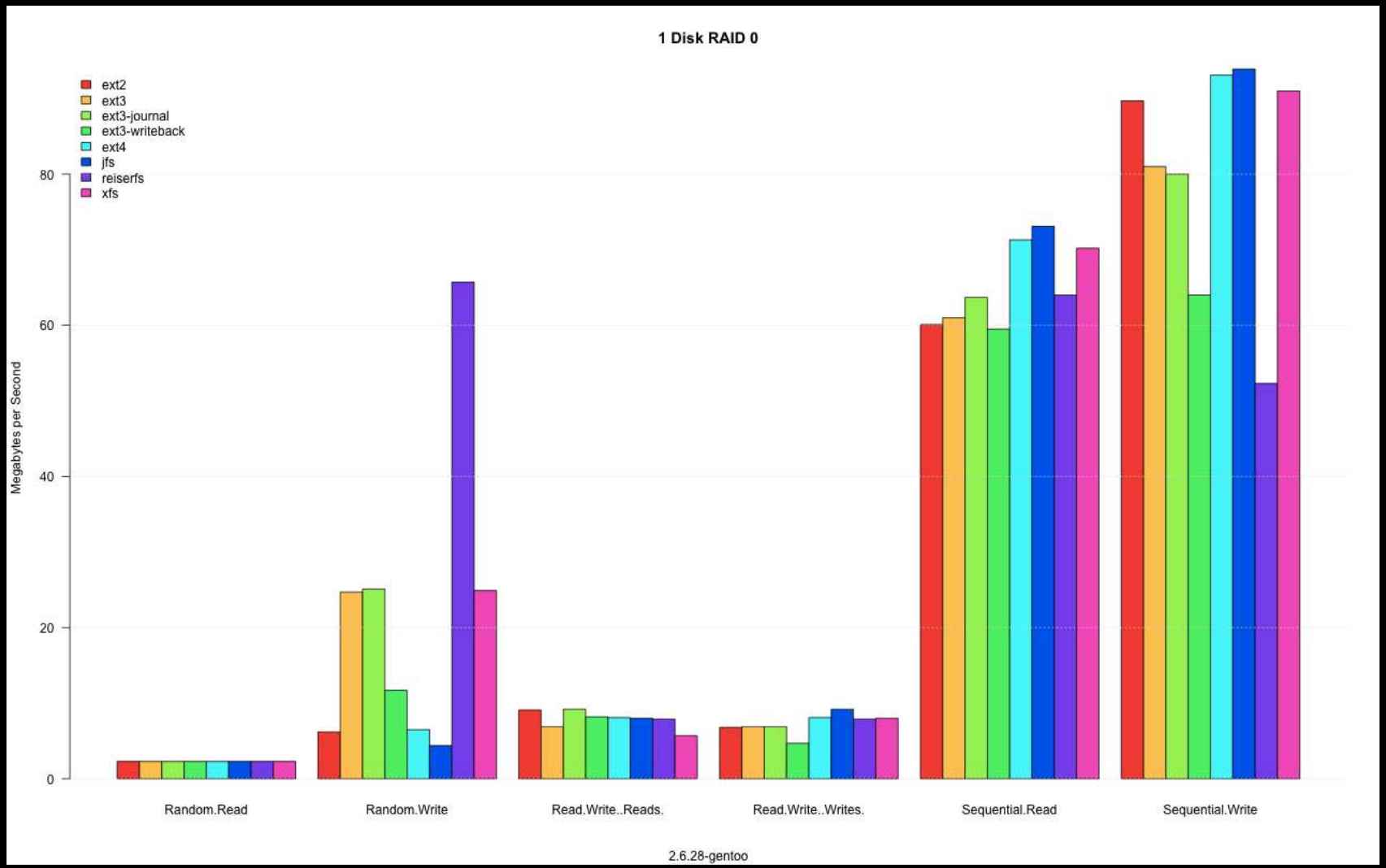
Sequential Read with Linux 2.6.28-gentoo ext2 on 1 Disk



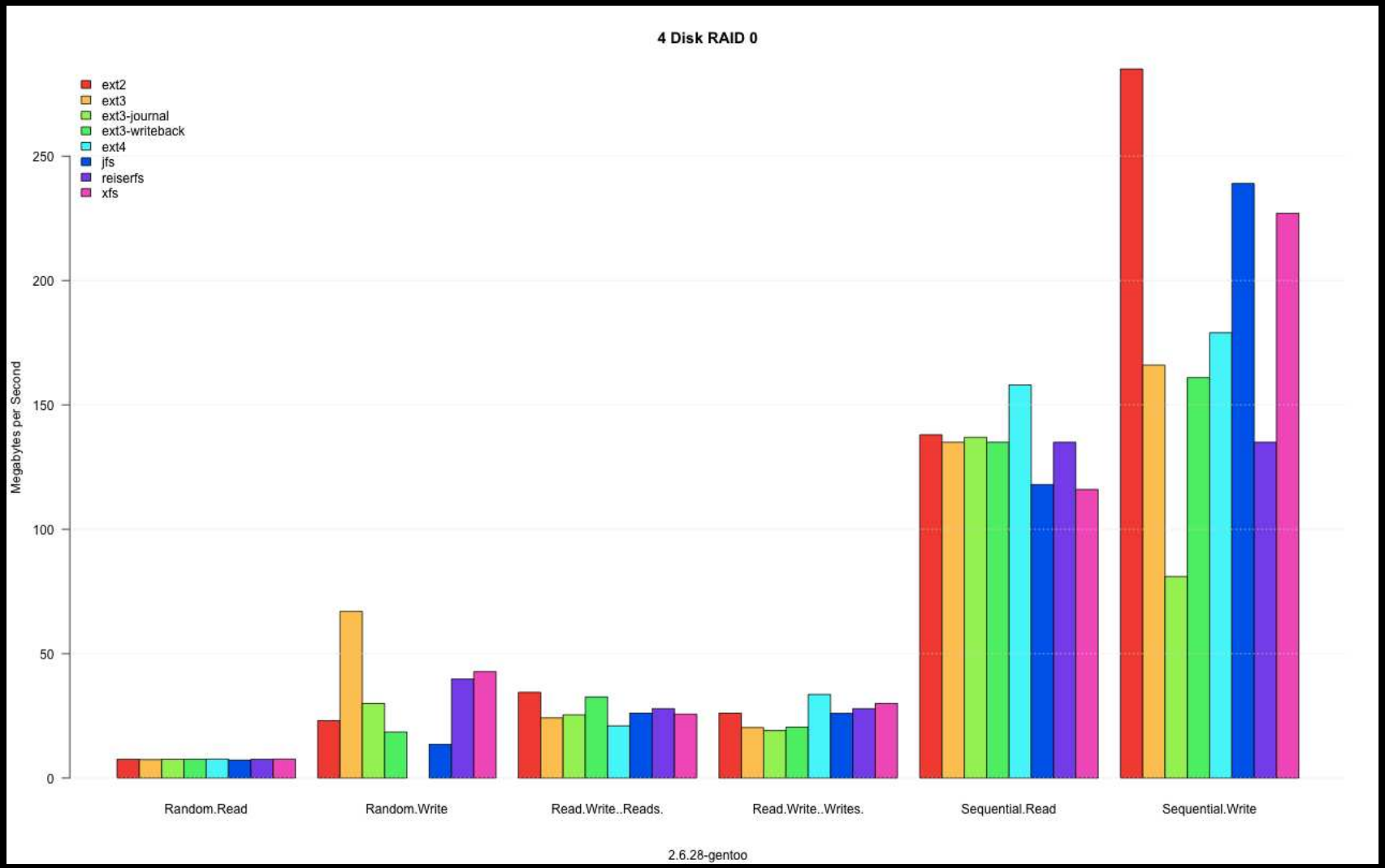
Sequential Write with Linux 2.6.28-gentoo ext2 on 1 Disk



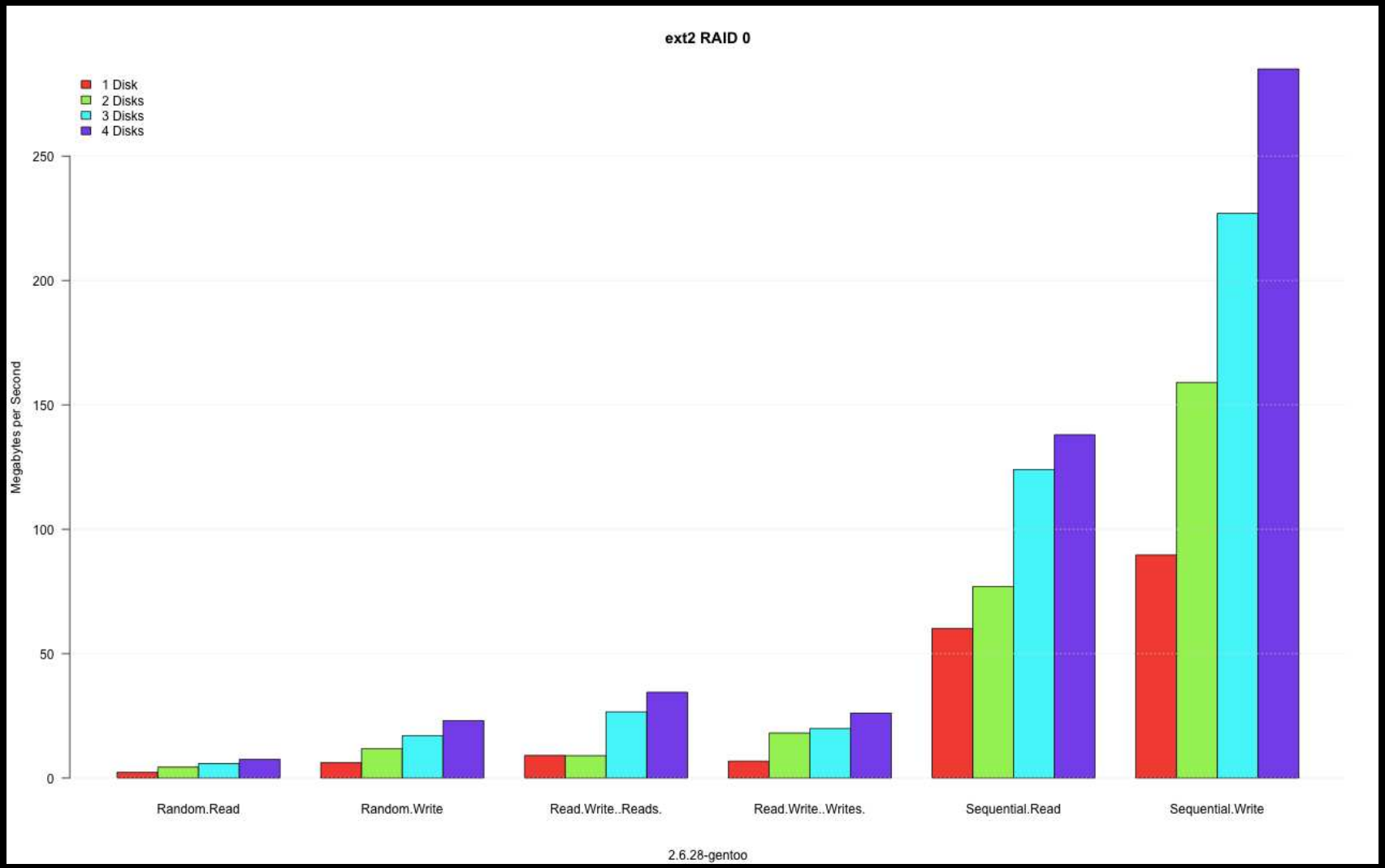
Results for all the filesystems when using only a single disk...
(If you see missing data, it means the test didn't complete.)

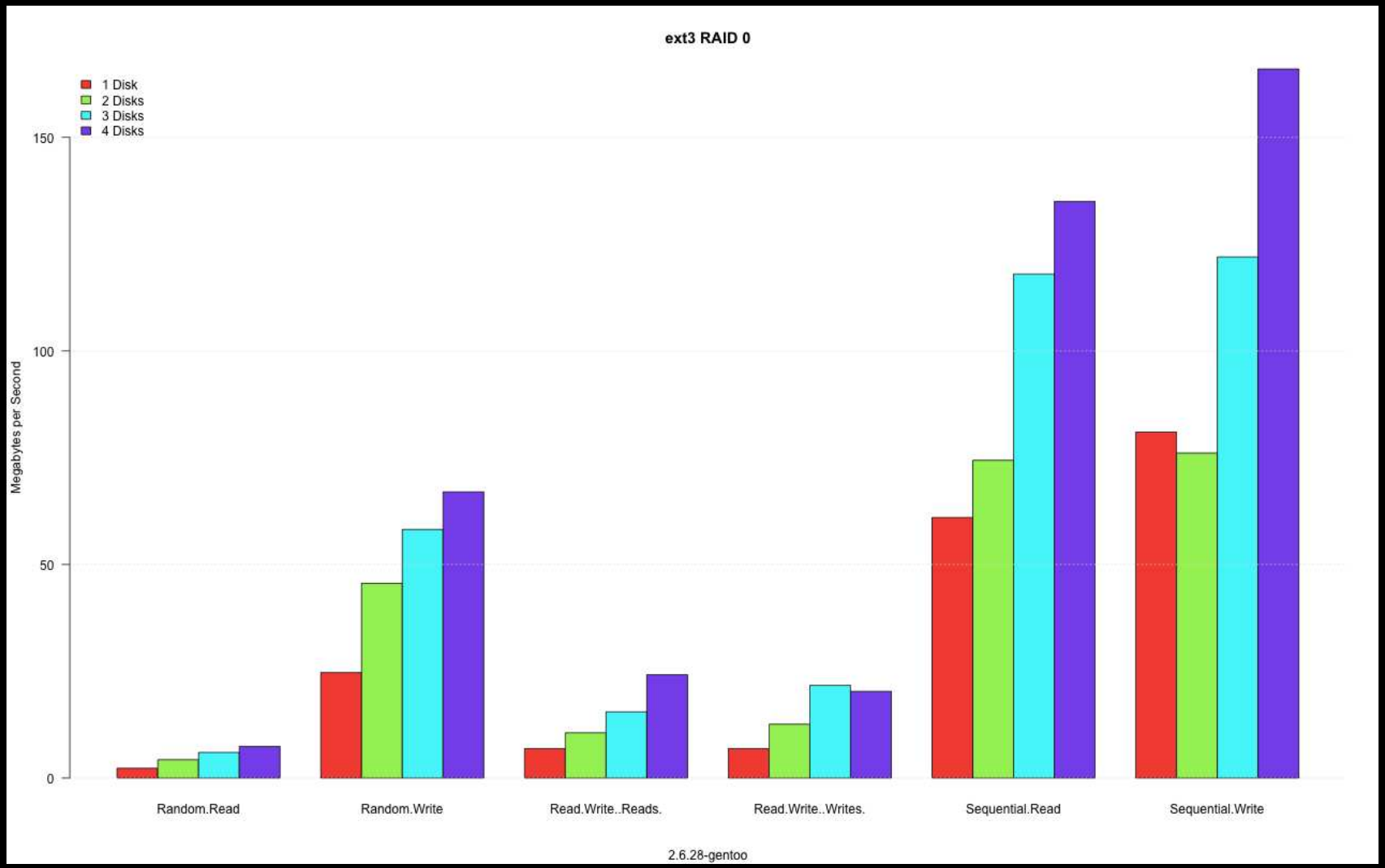

1 Disk RAID 0
ext2
ext3
ext3-journal
ext3-writeback
ext4
jfs
reiserfs
xfs
Megabytes per Second
0
20
40
60
80
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo

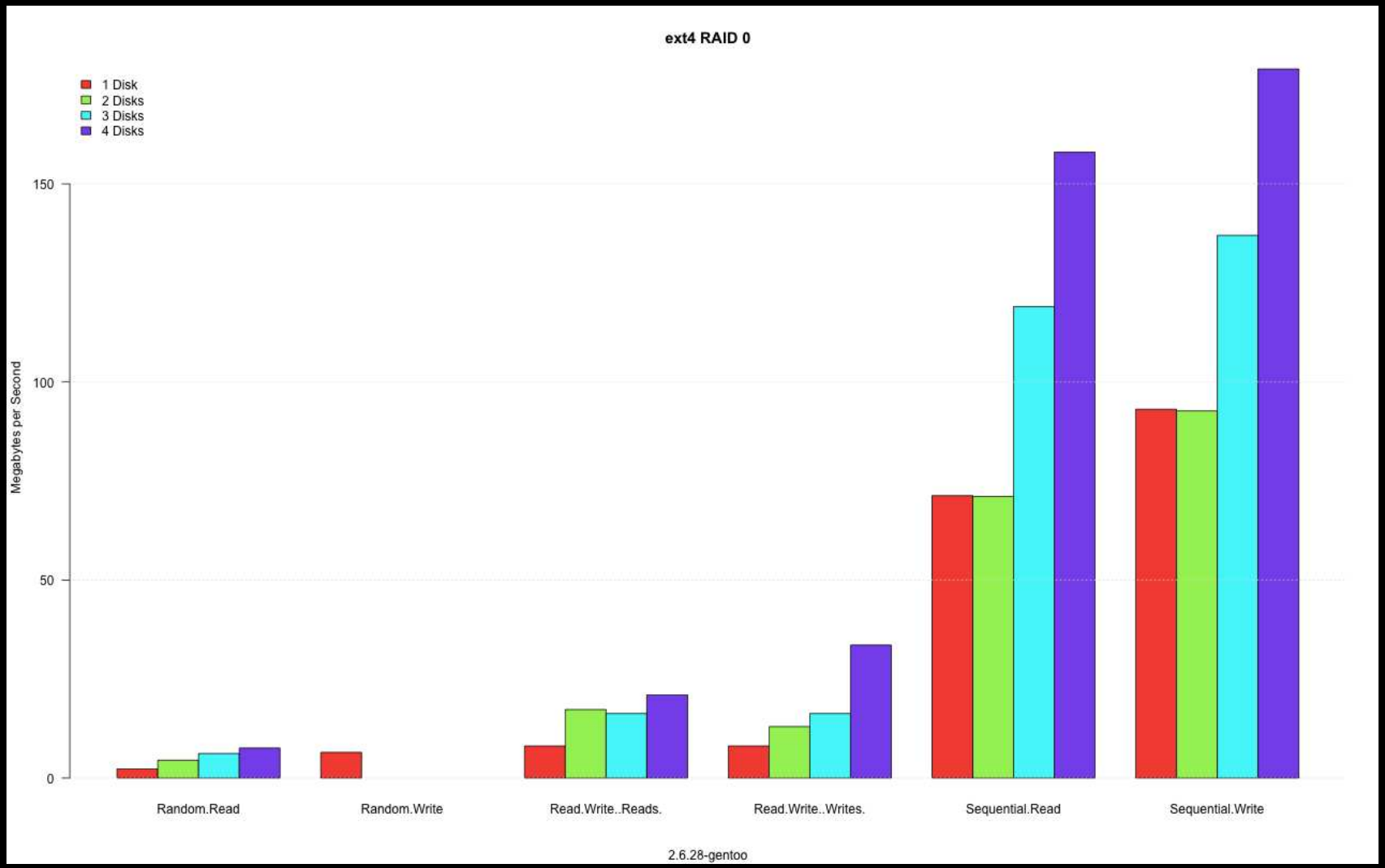
Results for all the filesystems when striping across four disks...

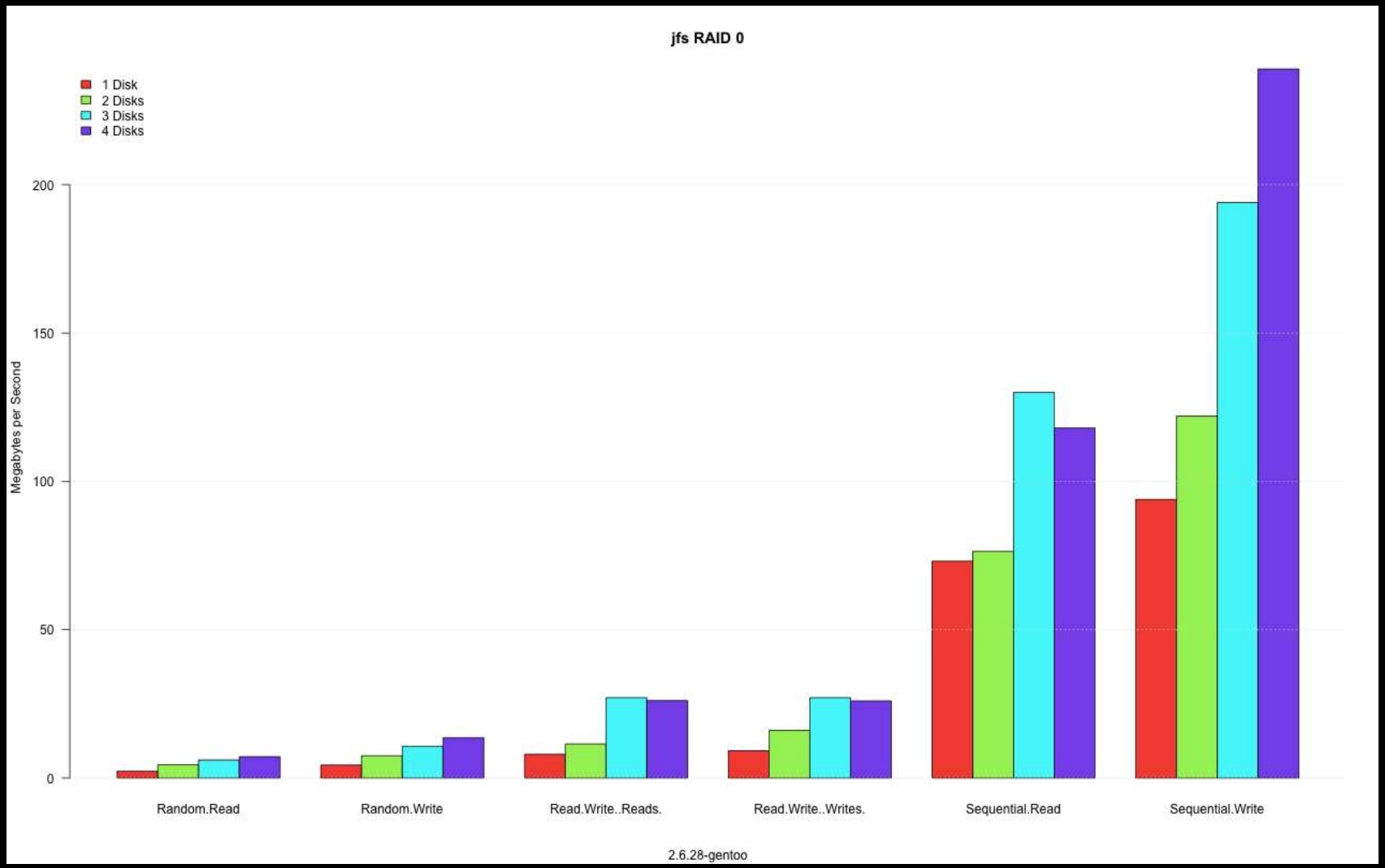

4 Disk RAID 0
ext2
ext3
ext3-journal
ext3-writeback
ext4
jfs
reiserfs
xfs
Megabytes per Second
0
50
100
150
200
250
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo

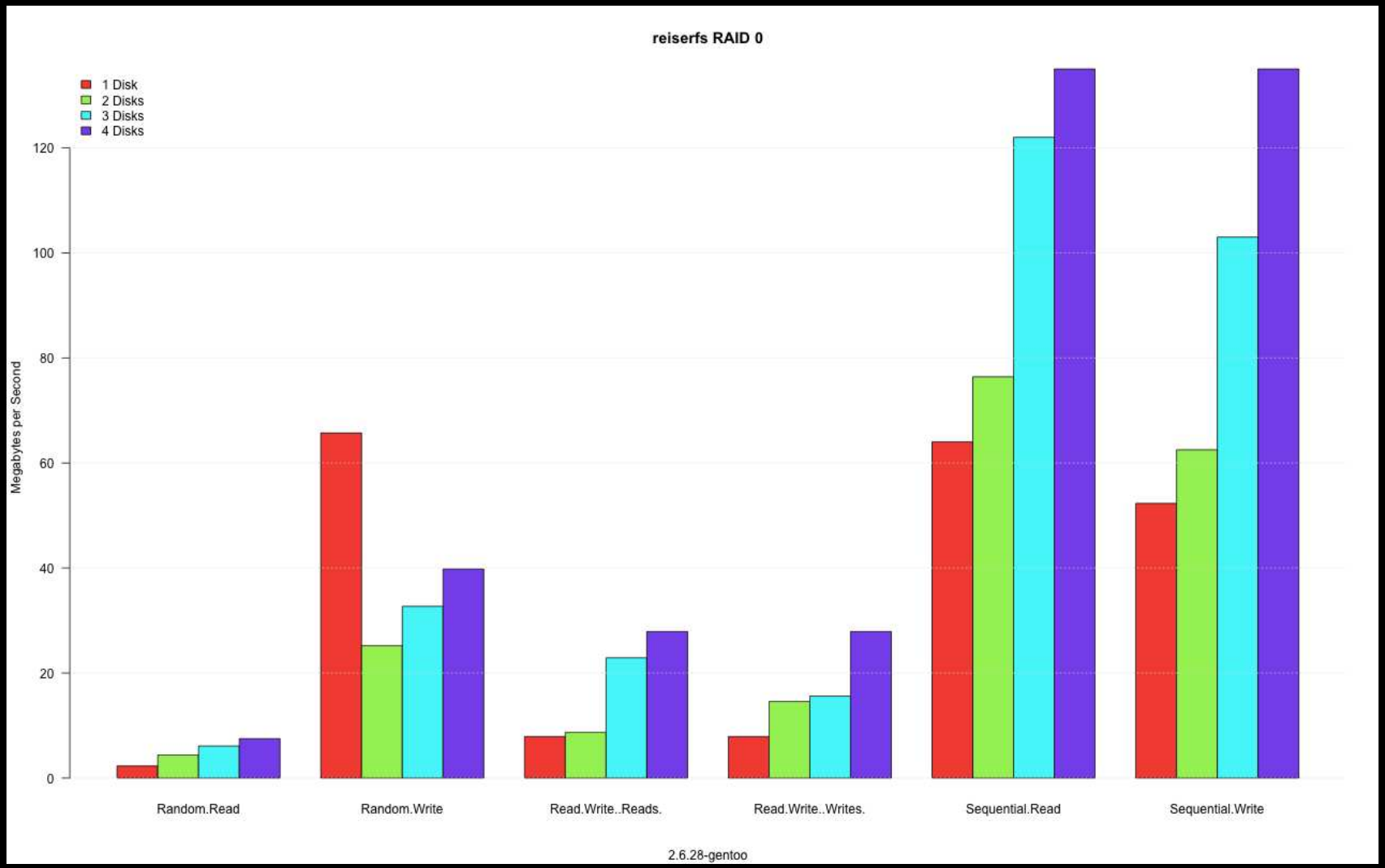
Let's look at that step by step for RAID 0.

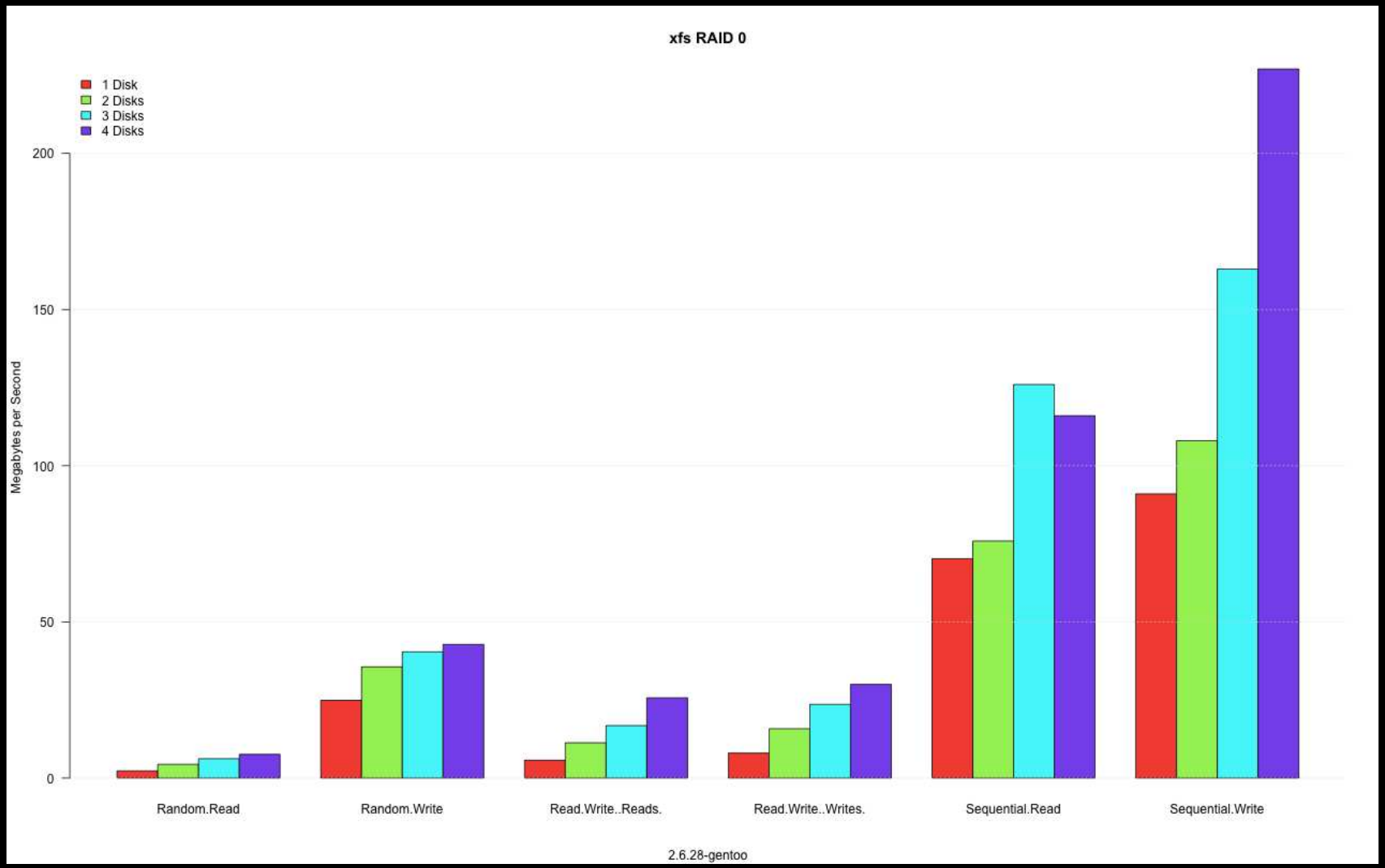

ext2 RAID 0
1 Disk
2 Disks
3 Disks
4 Disks
Megabytes per Second
250
200
150
100
50
0
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo


ext3 RAID 0
1 Disk
2 Disks
3 Disks
4 Disks
Megabytes per Second
150
100
50
0
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo


ext4 RAID 0
1 Disk
2 Disks
3 Disks
4 Disks
Megabytes per Second
150
100
50
0
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo

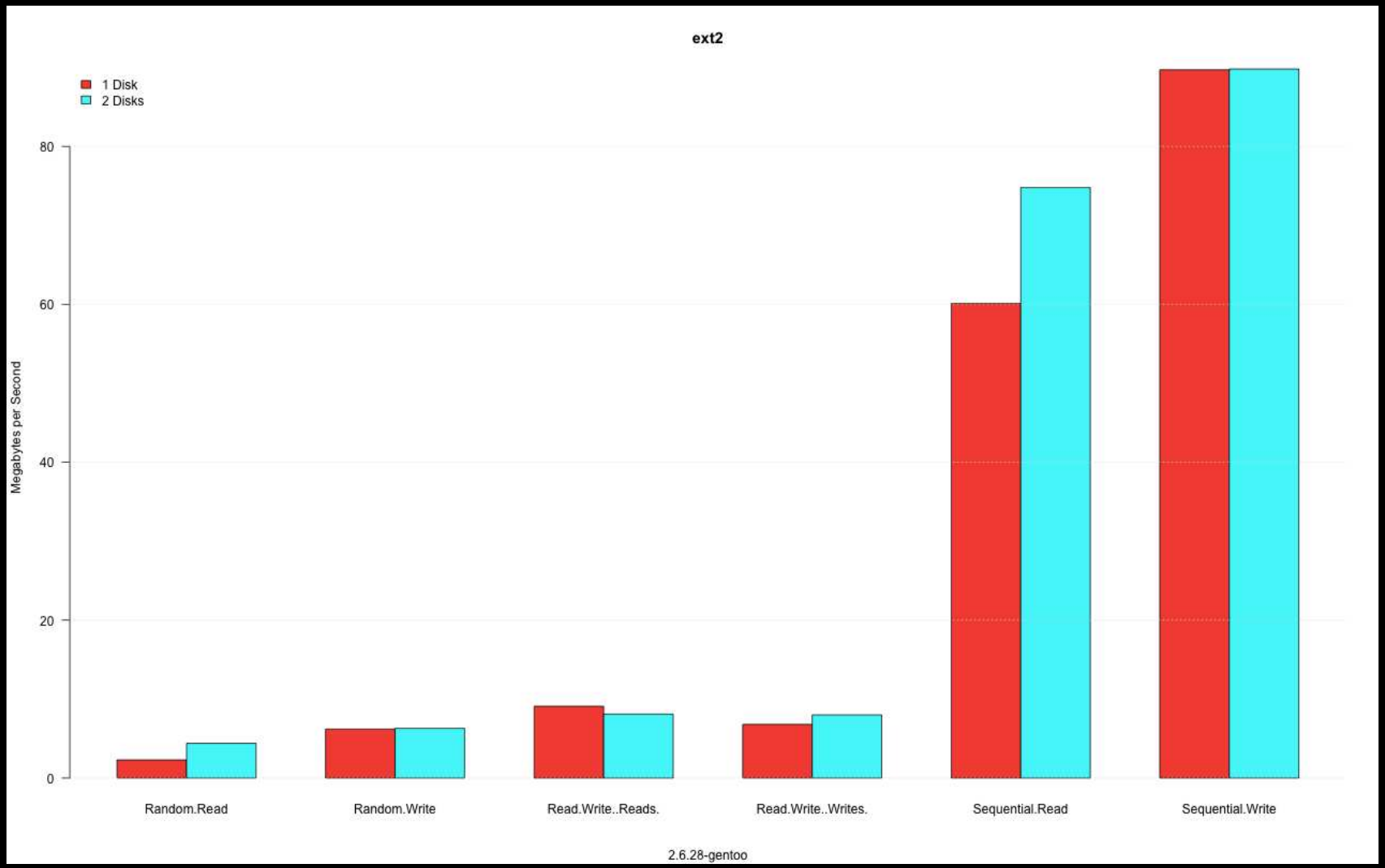

jfs RAID 0
1 Disk
2 Disks
3 Disks
4 Disks
Megabytes per Second
0
50
100
150
200
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo


reiserfs RAID 0
1 Disk
2 Disks
3 Disks
4 Disks
Megabytes per Second
0
20
40
60
80
100
120
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo


xfs RAID 0
1 Disk
2 Disks
3 Disks
4 Disks
Megabytes per Second
200
150
100
50
0
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo

Note trends might not be the same when adding disks to other RAID configurations.

What benefits are there in mirroring?


ext2
1 Disk
2 Disks
Megabytes per Second
0
20
40
60
80
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo

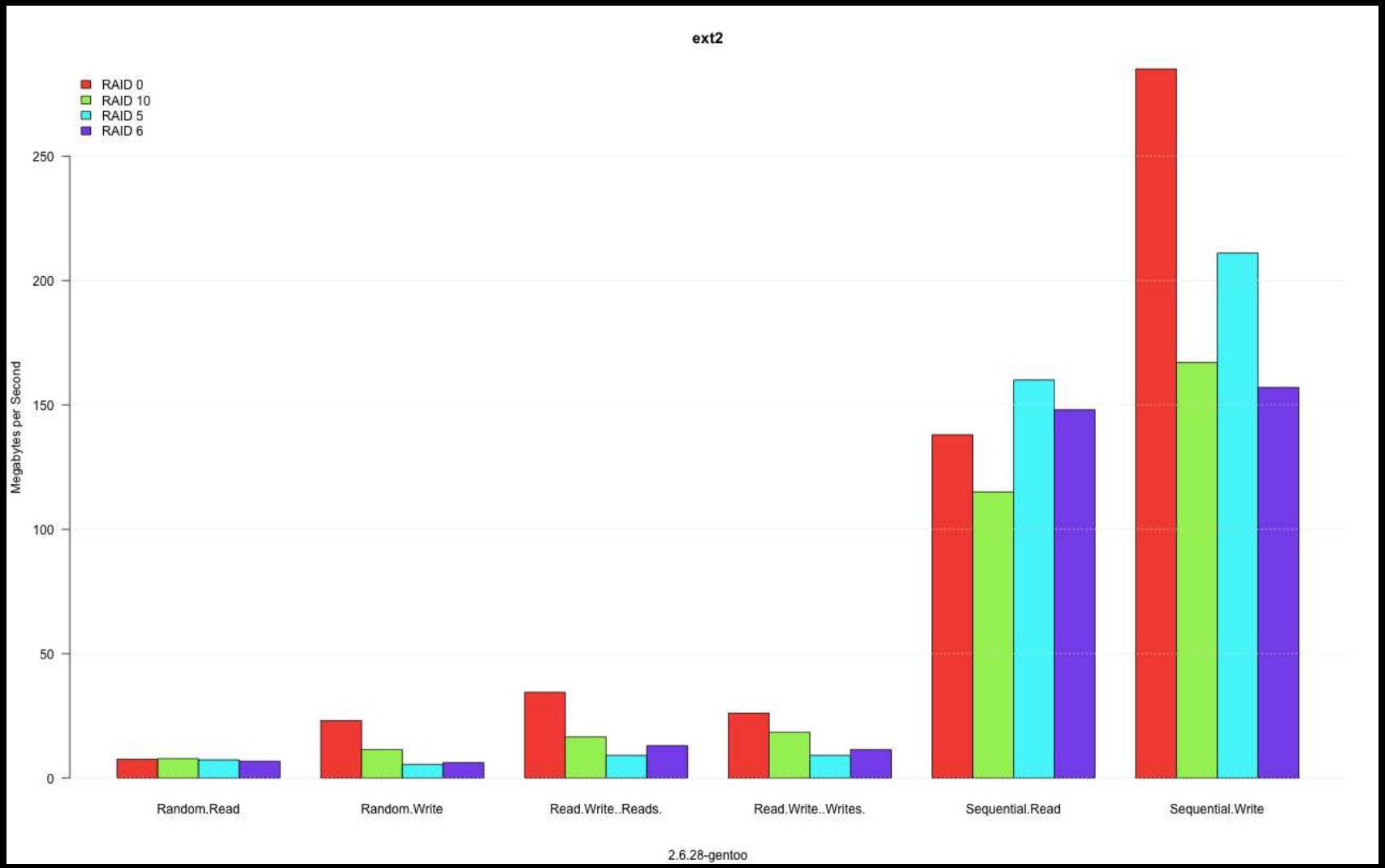
- Should have same write performance as a single disk.
- May be able to take advantage of using two disks for reading.

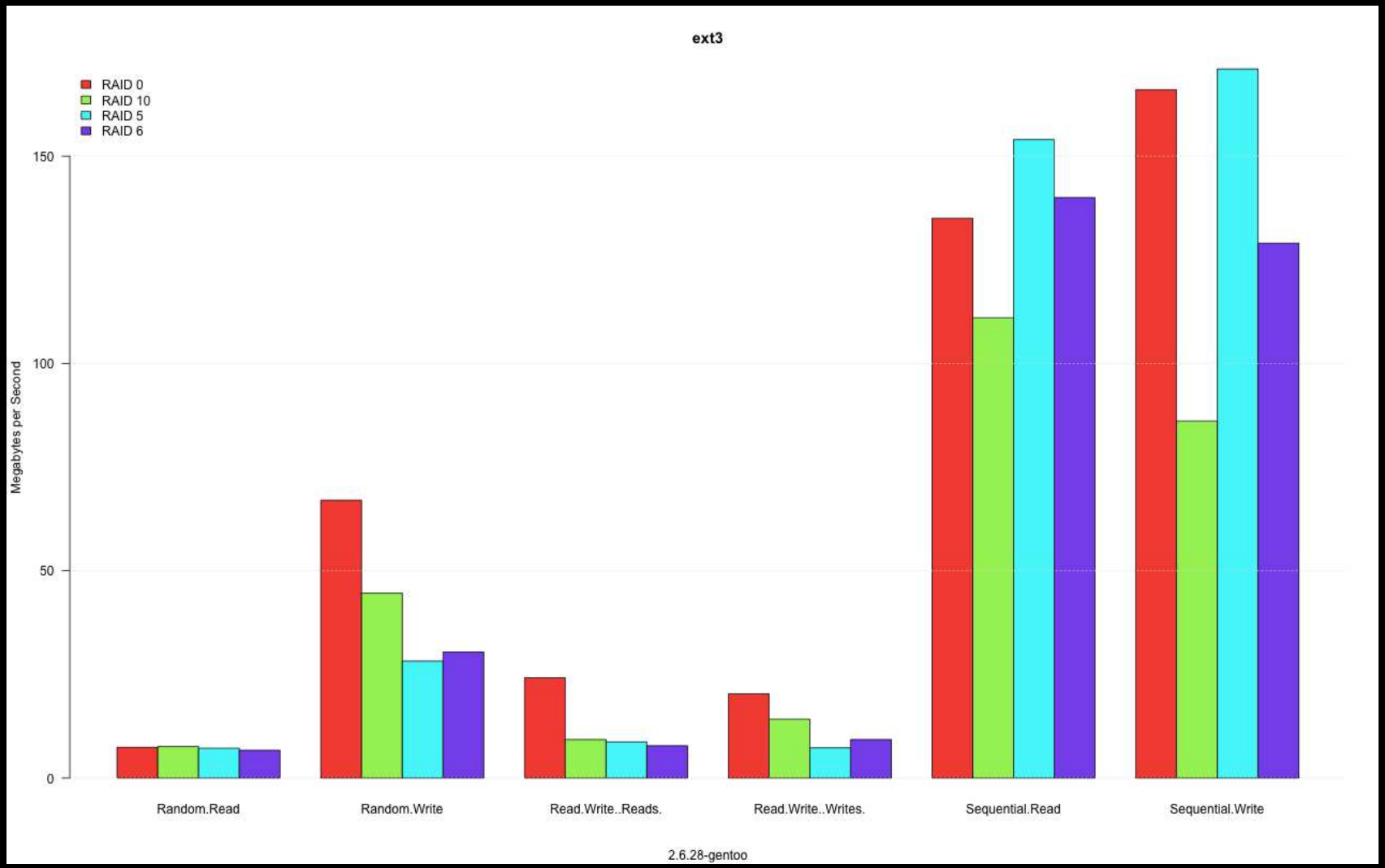
What is the best use of 4 disks?
Is RAID-6 and improvement over RAID-5?[2]

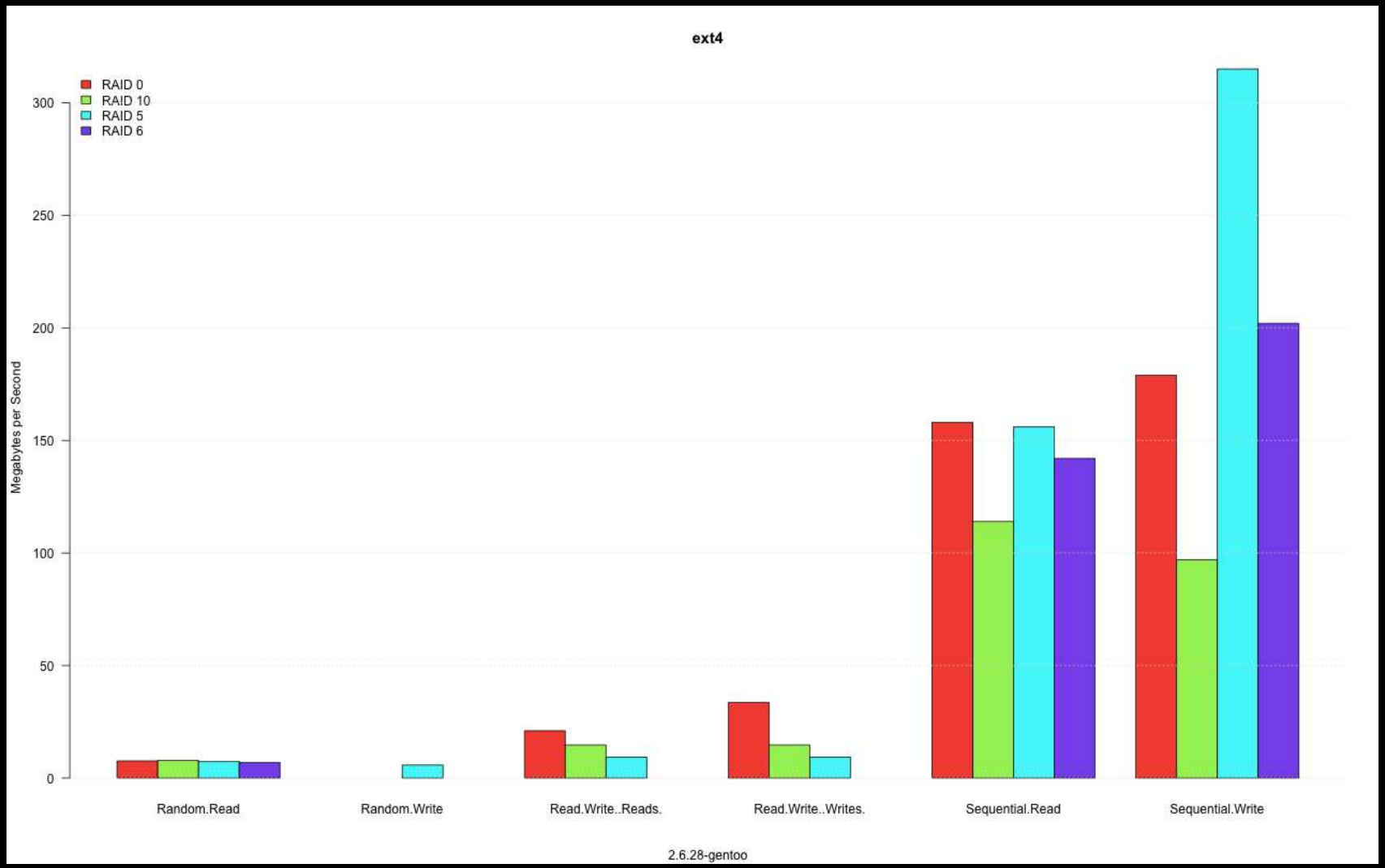
- RAID 5 (striped disks with parity) combines three or more disks in a way that protects data against loss of any one disk; the storage capacity of the array is reduced by one disk.
- RAID 6 (striped disks with dual parity) (less common) can recover from the loss of two disks.

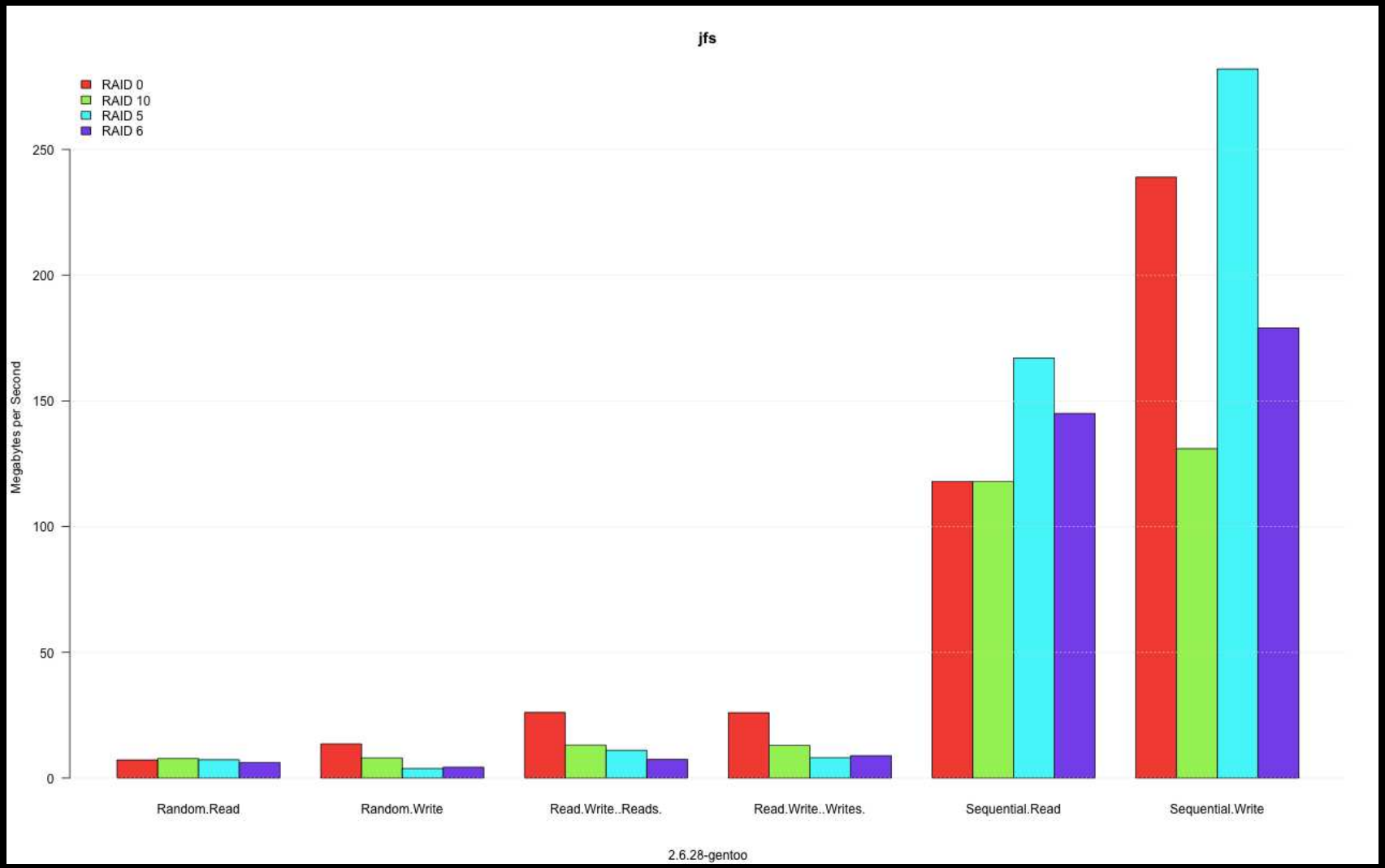
Do the answers depend on what filesystem is used?

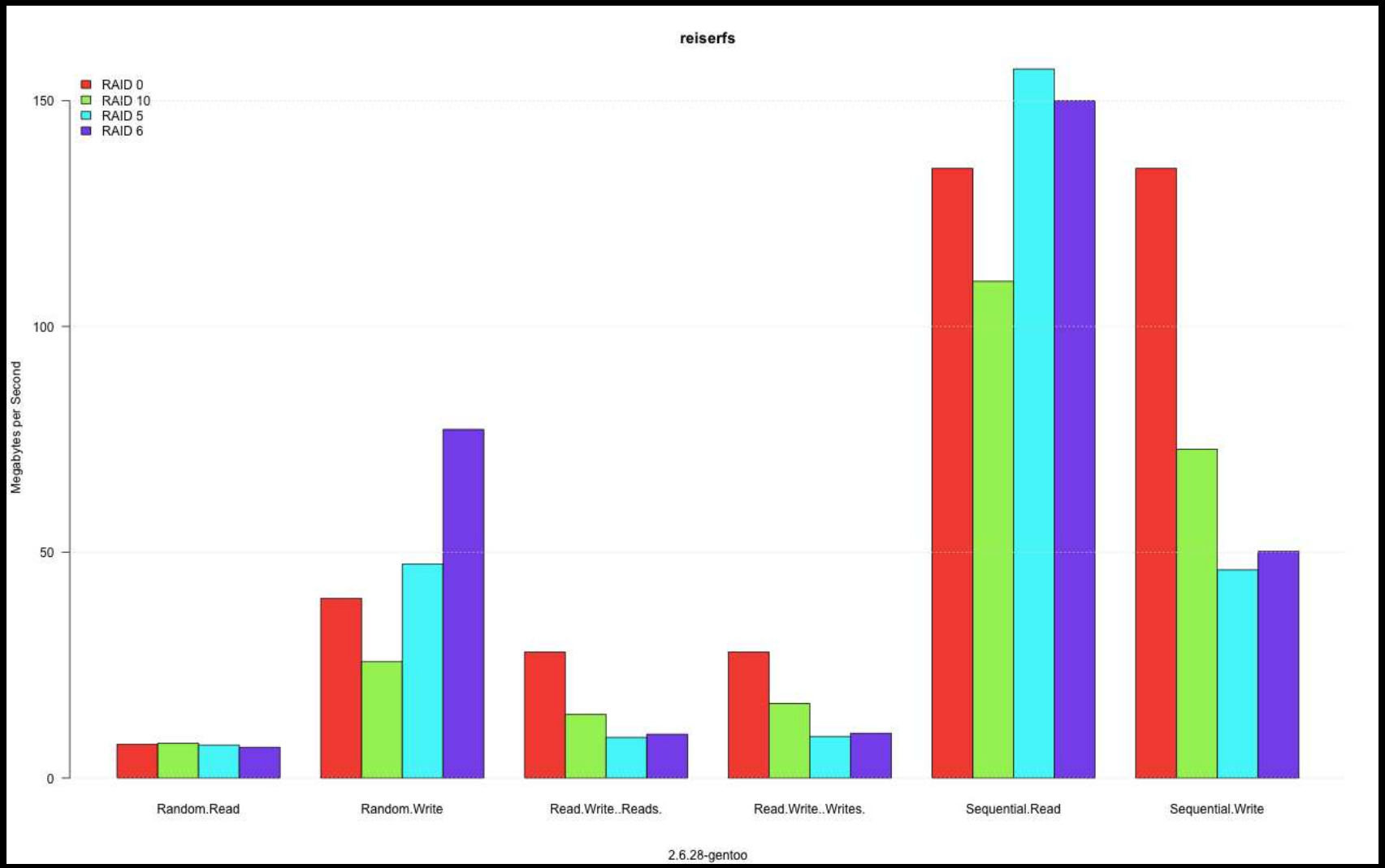
[2] http://en.wikipedia.org/wiki/RAID

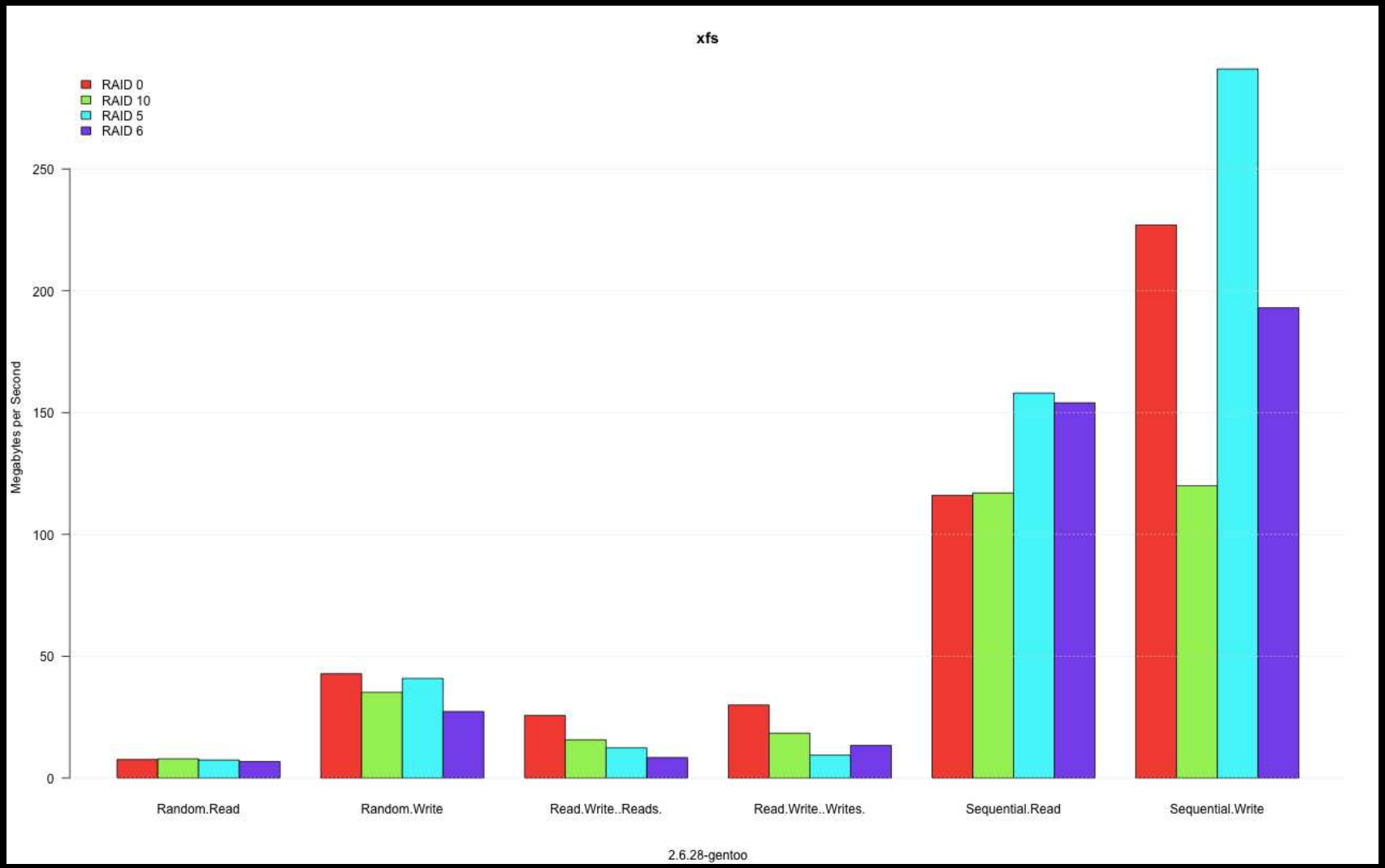

ext2
RAID 0
RAID 10
RAID 5
RAID 6
Megabytes per Second
250
200
150
100
50
0
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo


ext3
RAID 0
RAID 10
RAID 5
RAID 6
Megabytes per Second
150
100
50
0
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo


ext4
RAID 0
RAID 10
RAID 5
RAID 6
Megabytes per Second
300
250
200
150
100
50
0
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo


jfs
RAID 0
RAID 10
RAID 5
RAID 6
Megabytes per Second
250
200
150
100
50
0
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo


reiserfs
RAID 0
RAID 10
RAID 5
RAID 6
150
100
50
0
Megabytes per Second
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo


xfs
RAID 0
RAID 10
RAID 5
RAID 6
Megabytes per Second
250
200
150
100
50
0
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
2.6.28-gentoo

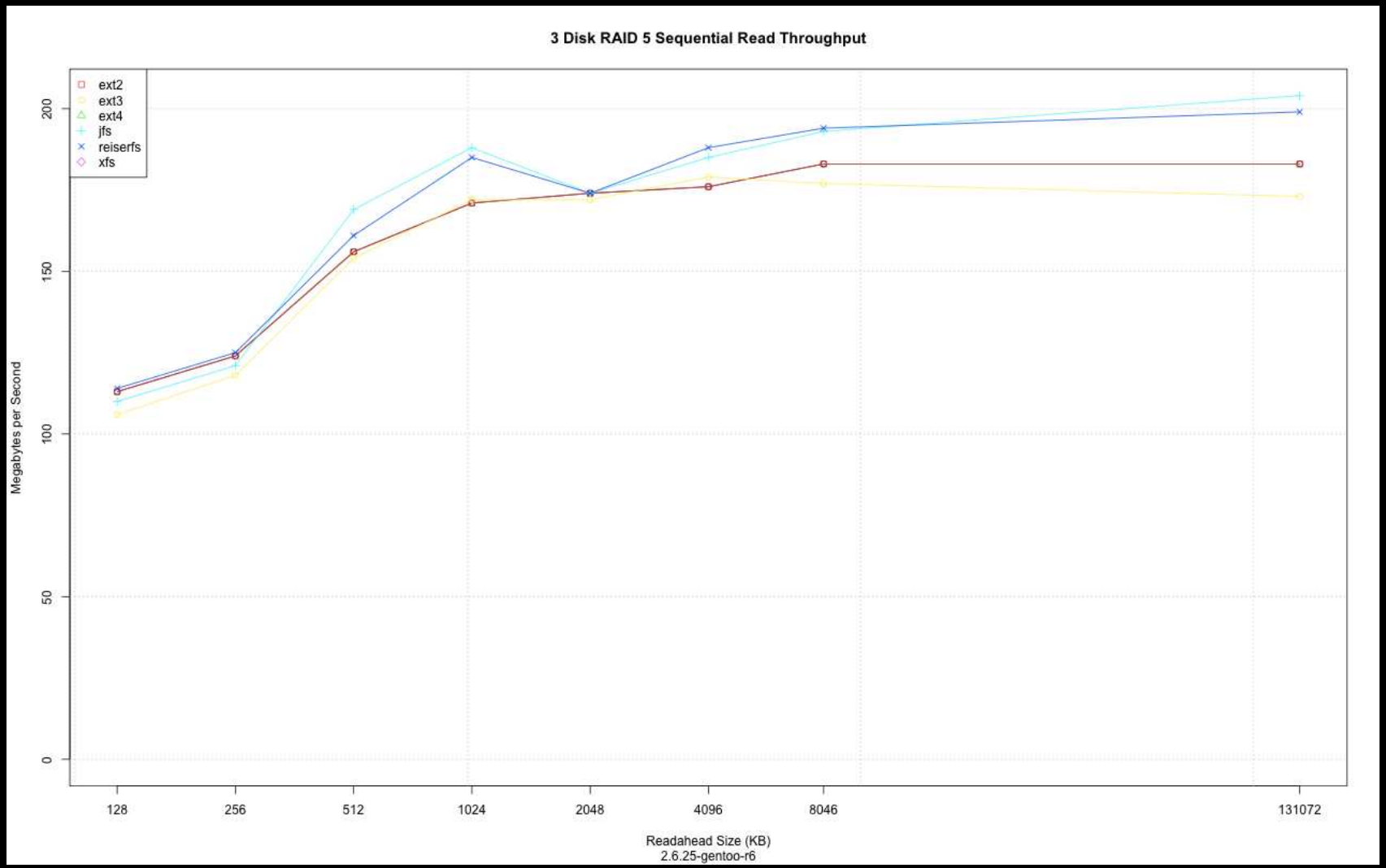
Filesystem Readahead

Values are in number of 512-byte segments, set per device.
To get the current readahead size:

```
$ blockdev --getra /dev/sda
256
```

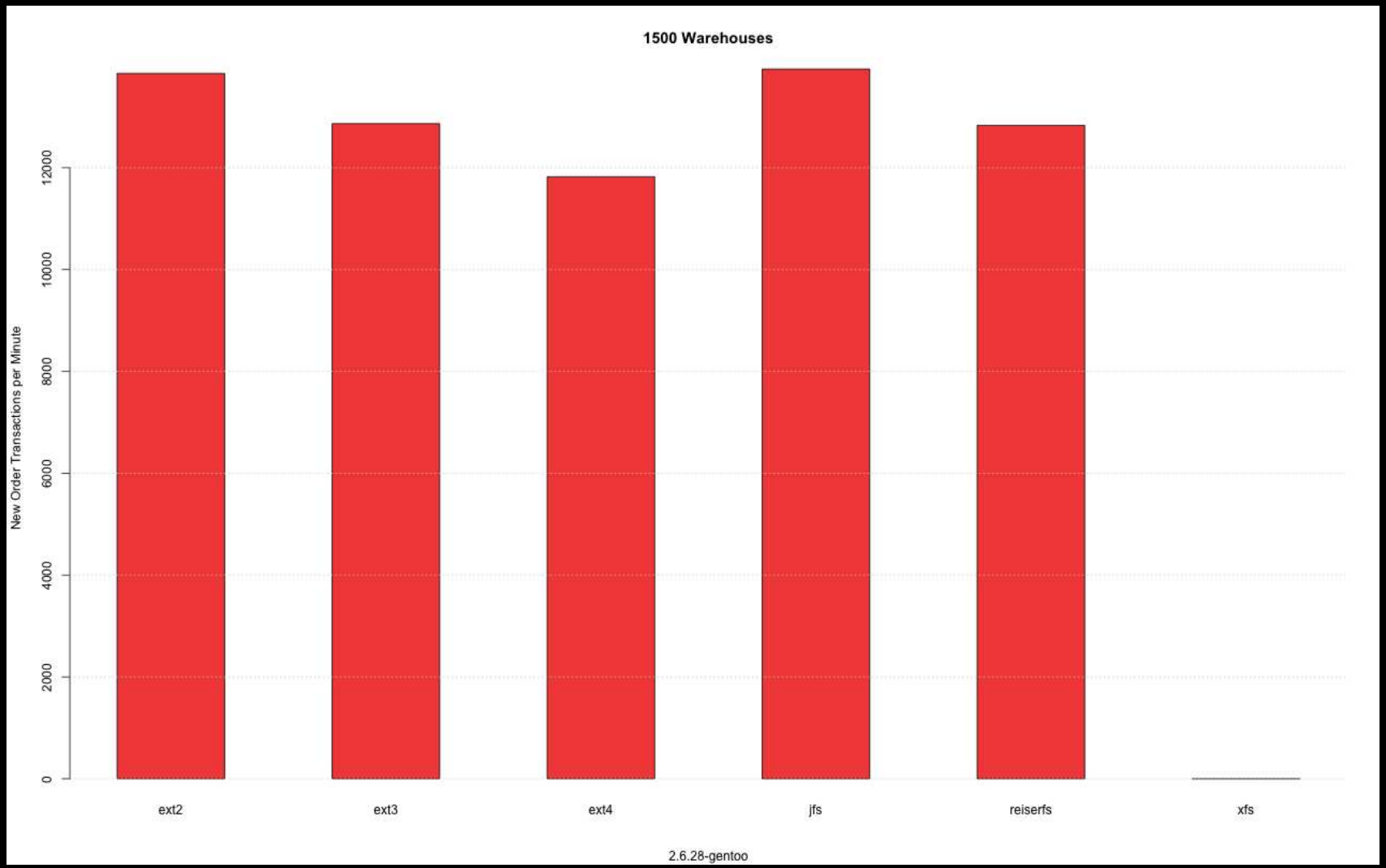
To set the readahead size to 8 MB:

```
$ blockdev --setra 16384 /dev/sda
```


3 Disk RAID 5 Sequential Read Throughput
ext2
ext3
ext4
jfs
reiserfs
xfs
Megabytes per Second
0
50
100
150
200
128
256
512
1024
2048
4096
8046
131072
Readahead Size (KB)
2.6.25-gentoo-r6

What does this all mean for my database system?

It depends... Let's look at one sample of just using difference filesystems.


1500 Warehouses
New Order Transactions per Minute
0
2000
4000
6000
8000
10000
12000
ext2
ext3
ext4
jfs
reiserfs
xfs
2.6.28-gentoo

Raw data for comparing filesystems

Initial inspection of results suggests we are close to being bound by processor and storage performance:

- http://207.173.203.223/~markwkm/community6/dbt2/m-fs/ext2.1500.2/
- http://207.173.203.223/~markwkm/community6/dbt2/m-fs/ext3.1500.2/
- http://207.173.203.223/~markwkm/community6/dbt2/m-fs/ext4.1500.2/
- http://207.173.203.223/~markwkm/community6/dbt2/m-fs/jfs.1500.2/
- http://207.173.203.223/~markwkm/community6/dbt2/m-fs/reiserfs.1500.2/
- http://207.173.203.223/~markwkm/community6/dbt2/m-fs/xfs.1500.2/

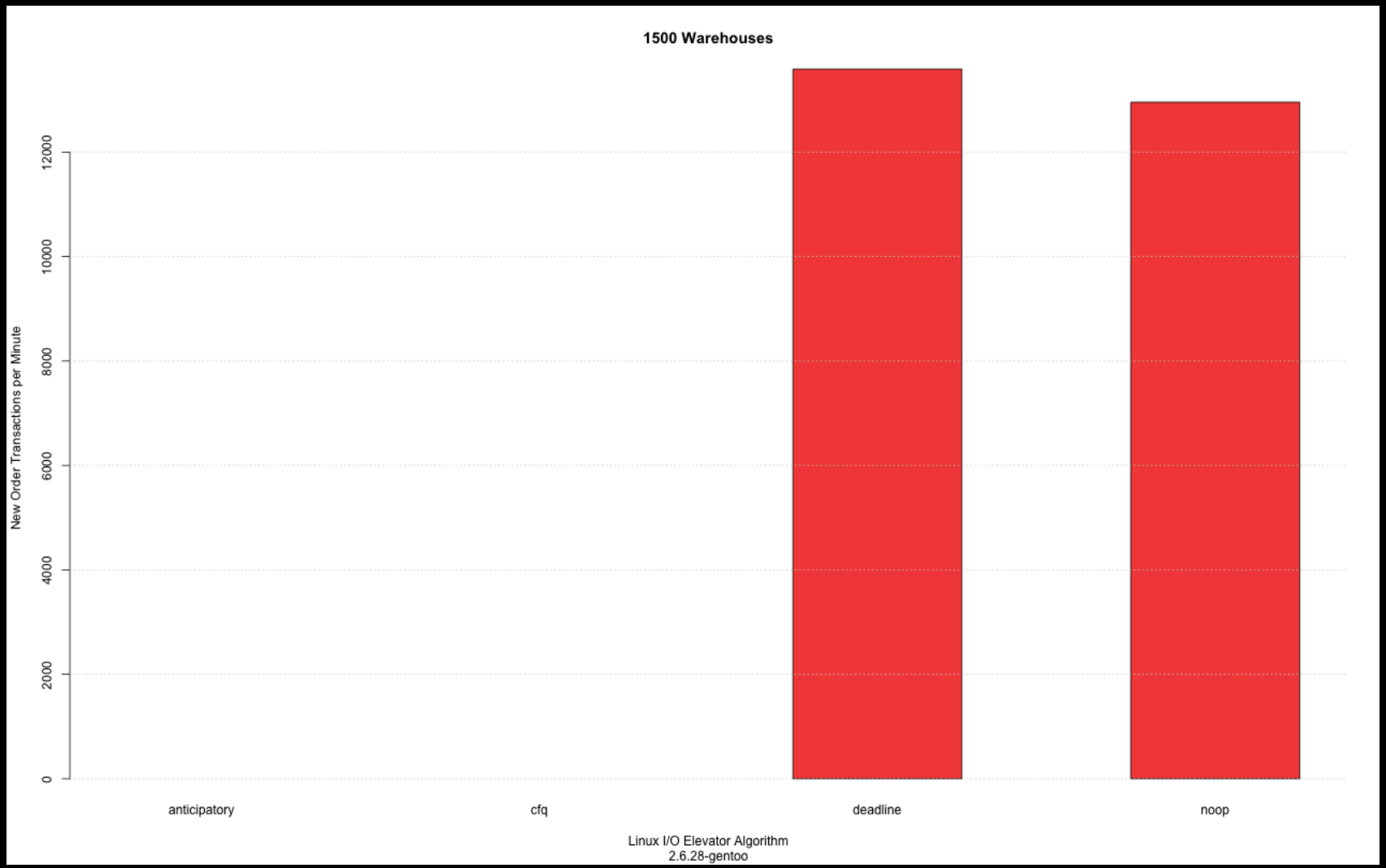
Linux I/O Elevator Algorithm

To get current algorithm in []'s:

```
$ cat /sys/block/sda/queue/scheduler
noop anticipatory [deadline] cfq
```

To set a different algorithm:

```
$ echo cfg > /sys/block/sda/queue/scheduler
noop anticipatory [deadline] cfq
```


1500 Warehouses
New Order Transactions per Minute
0
2000
4000
6000
8000
10000
12000
anticipatory
cfq
deadline
noop
Linux I/O Elevator Algorithm
2.6.28-gentoo

Raw data for comparing I/O elevator algorithms

Initial inspection of results suggests we are close to being bound by processor and storage performance:

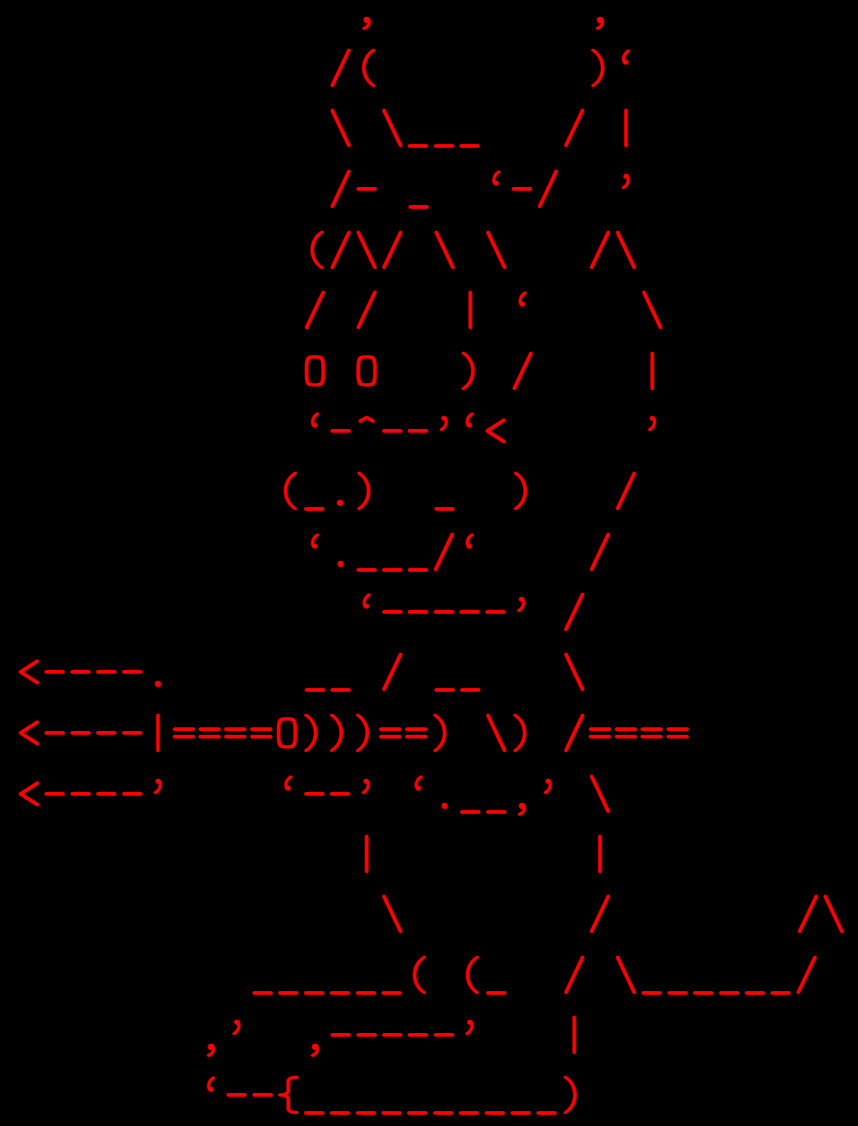
- http://207.173.203.223/~markwkm/community6/dbt2/m-elevator/ext2.anticipatory.1/
- http://207.173.203.223/~markwkm/community6/dbt2/m-elevator/ext2.cfs.1/
- http://207.173.203.223/~markwkm/community6/dbt2/m-elevator/ext2.deadline.1/
- http://207.173.203.223/~markwkm/community6/dbt2/m-elevator/ext2.noop.1/

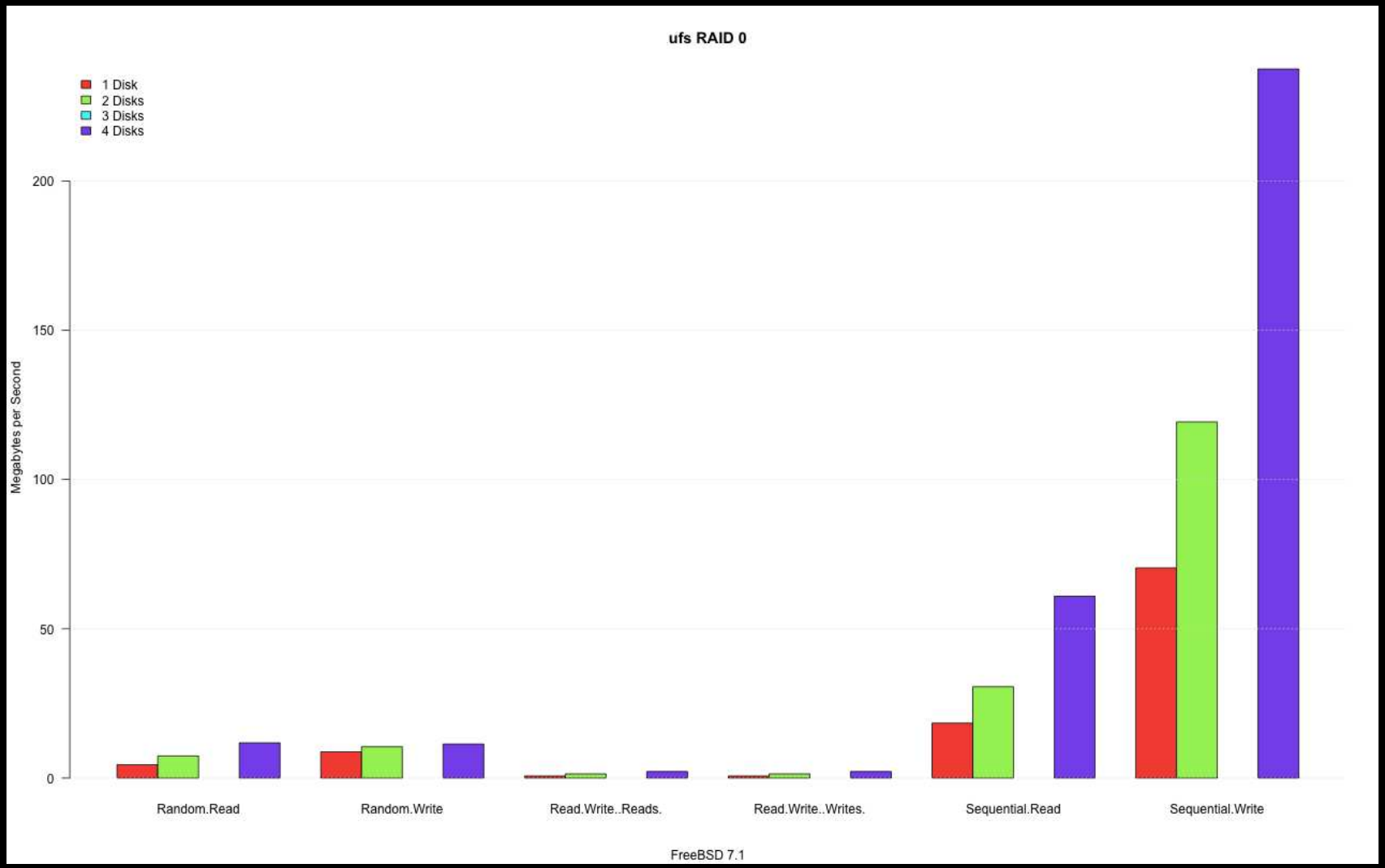
Why use deadline?[3]

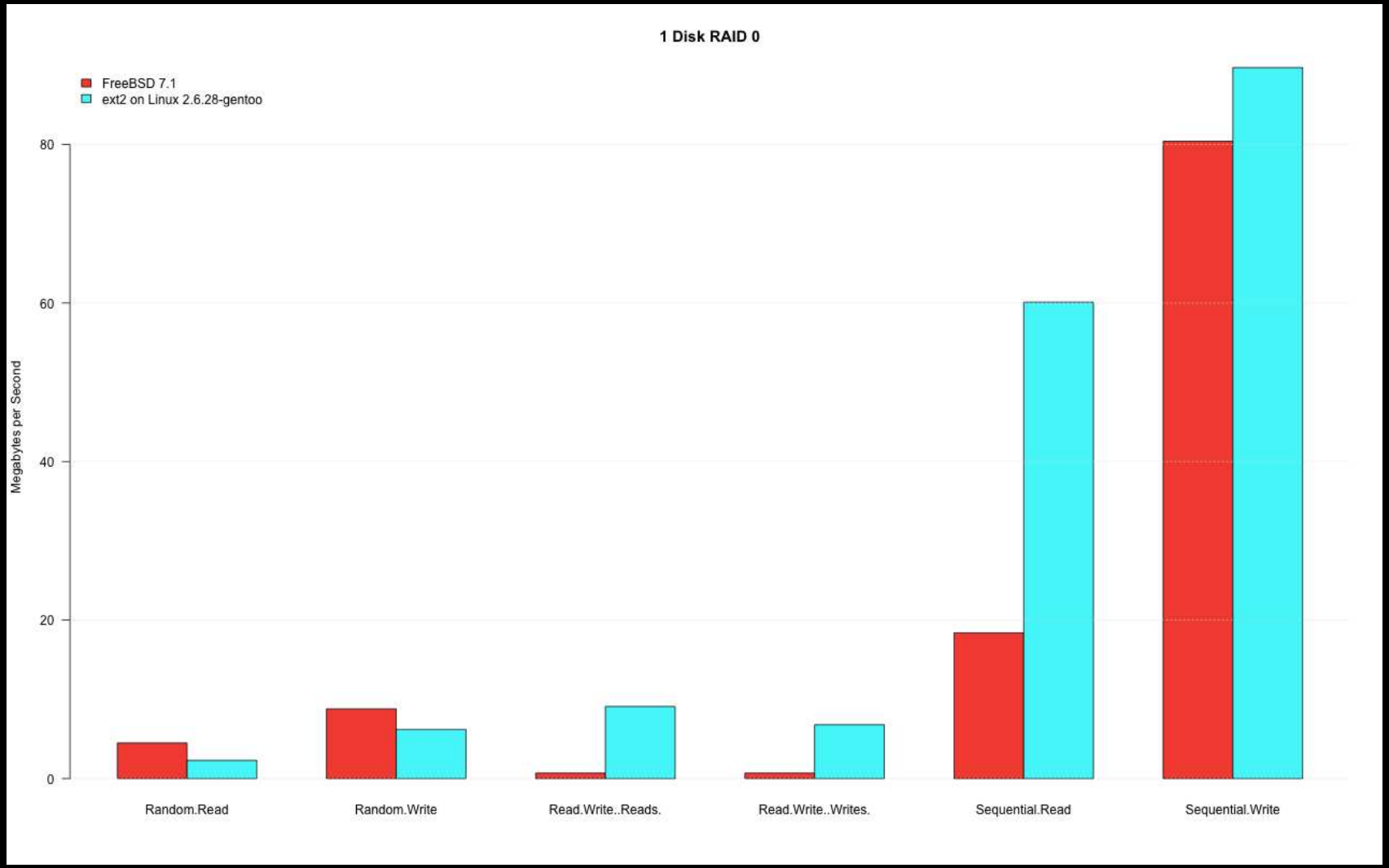
- Attention! Database servers, especially those using "TCQ" disks should investigate performance with the 'deadline' IO scheduler. Any system with high disk performance requirements should do so, in fact.
- Also, users with hardware RAID controllers, doing striping, may find highly variable performance results with using the as-iosched. The as-iosched anticipatory implementation is based on the notion that a disk device has only one physical seeking head. A striped RAID controller actually has a head for each physical device in the logical RAID device.

[3] Linux source code: Documentation/block/as-iosched.txt

Bonus Data - Preview of FreeBSD




ufs RAID 0
1 Disk
2 Disks
3 Disks
4 Disks
Megabytes per Second
200
150
100
50
0
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write
FreeBSD 7.1


1 Disk RAID 0
FreeBSD 7.1
ext2 on Linux 2.6.28-gentoo
Megabytes per Second
0
20
40
60
80
Random.Read
Random.Write
Read.Write..Reads.
Read.Write..Writes.
Sequential.Read
Sequential.Write

Final Notes

- You have to test your own system to see what it can do.

Materials Are Freely Available - In a new location!

PDF

- `http://www.slideshare.net/markwkm`

LaTeX Beamer (source)

- `http://git.postgresql.org/git/performance-tuning.git`

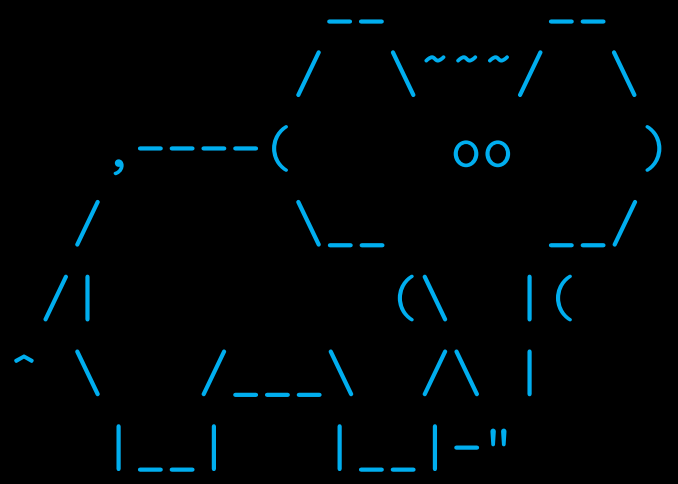
Time and Location

When: 2nd Thursday of the month
Location: Portland State University
Room: FAB 86-01 (Fourth Avenue Building)
Map: `http://www.pdx.edu/map.html`

Coming up next time. . . Effects of tuning PostgreSQL Parameters

Acknowledgements

Haley Jane Wakenshaw



License