Linux Filesystems and MySQL

Ammon Sutherland

April 23, 2013

PERCONA LIVE

Preface...

"Who is it?" said Arthur.

"Well," said Ford, "if we're lucky it's just the Vogons come to throw us into space."

"And if we're unlucky?"

"If we're unlucky," said Ford grimly, "the captain might be serious in his threat that he's going to read us some of his poetry first ..."

Background

- Long-time Linux System Administrator turned DBA
  - University systems
  - Managed Hosting
  - Online Auctions
  - E-commerce, SEO, marketing, data-mining

A bit of an optimization junkie…

Once in a while I share: http://shamallu.blogspot.com/

Agenda

- Basic Theory
  - Directory structure
  - LVM
  - RAID
  - SSD
  - Filesystem concepts
- Filesystem choices
- MySQL Tuning
- Benchmarks
  - IO tests
  - FS maintenance
  - OLTP
- AWS EC2
- Conclusions

Basic Theory

deadlock detected
we rollback transaction two
err one two one three

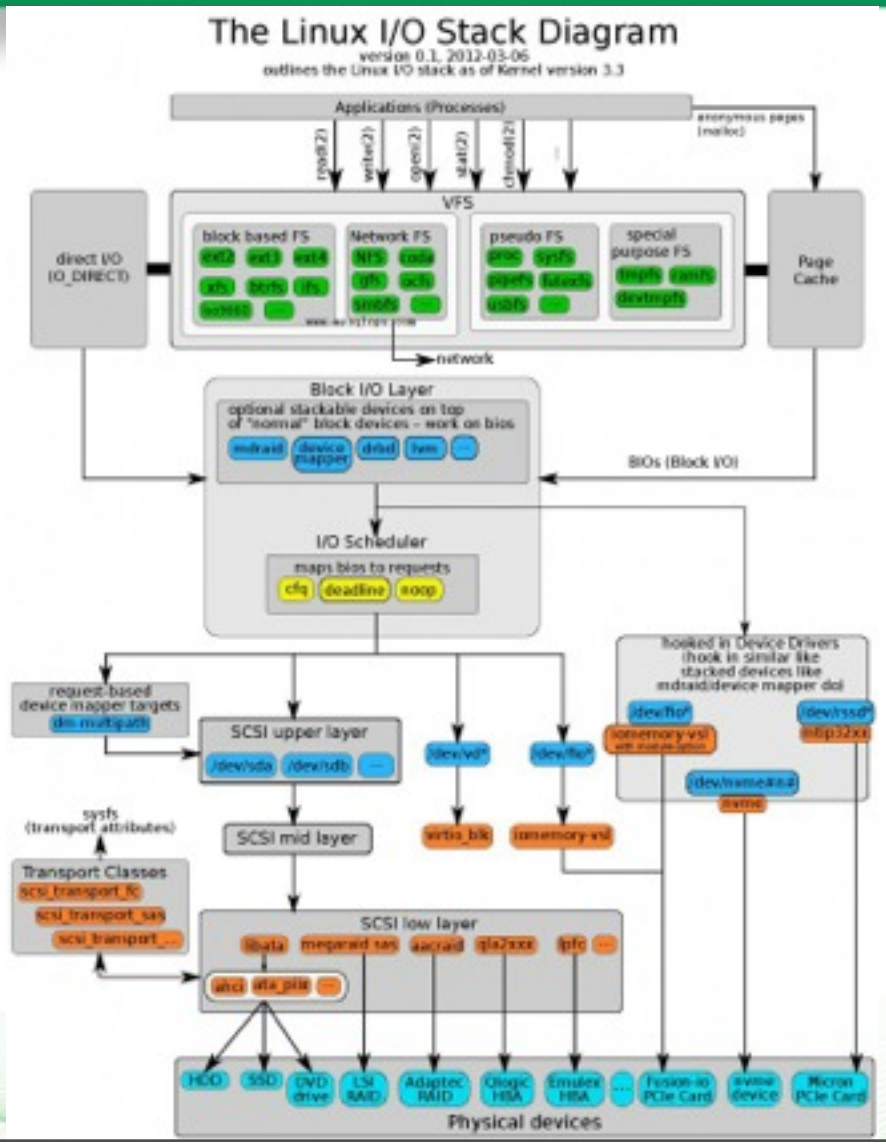
- A MySQL Haiku -

Directory Structure

Things that must be stored on disk

- Data files (.ibd or .MYD and .MYI) – Random IO
- Main InnoDB data file (ibdata1) – Random IO
- InnoDB Log files (ib_logfile0, ib_logfile1) – Sequential IO (one at a time)
- Binary logs and relay logs – Sequential IO
- General query log and Slow query log – Sequential IO
- Master.info – technically Random IO
- Error log – Infrequent Sequential IO

Linux IO Sub-System



Hard Drives

- Rotating platters
- SAS vs. SATA
  - SAS 6gb/s connectors can handle SATA 3gb/s drives
  - SAS typically cost more (much more for larger size)
  - SAS often will do higher rpm rates (10k, 15k rpm)
  - SAS has more logic on the drives
  - SAS has more data consistency and error reporting logic vs. SATA S.M.A.R.T.
  - SAS uses higher voltages allowing for external arrays with longer signal runs
  - SAS does TCQ vs. SATA NCQ (provides some similar effect)
  - Both do 8b10b encoding (25% parity overhead)

SSD

- Pros:
  - Very fast random reads and writes
  - Handle high concurrency very well
- Cons:
  - Cost per GB
  - Lifespan and performance depend on write-cycles. Beware write amplification
  - Requires care with RAID cards

RAID

Typical RAID Modes:

- RAID-0: Data striped, no redundancy (2+ disks)
- RAID-1: Data mirrored, 1:1 redundancy (2+ disks)
- RAID-5: Data striped with parity (3+ disks)
- RAID-6: Data striped with double parity (4+ disks)
- RAID-10: Data striped and mirrored (4+ disks)
- RAID-50: RAID-0 striping of multiple RAID-5 groups (6+ disks)

RAID (cont.)

Typical RAID Benefits and risks:

- RAID-0 - Scales reads and writes, multiplies space (risky, no disks can fail)
- RAID-1 - Scales reads not writes, no additional space gain (data intact with only one disk and rebuilt)
- RAID-5 - Scales reads and some writes (parity penalty, can survive one disk failure and rebuild)
- RAID-6 - Scales reads and less writes than RAID-5 (double parity penalty, can survive 2 disk failures and rebuild)
- RAID-10 - Scales 2x reads vs writes, (can lose up to two disks in particular combinations)
- RAID-50 - Scales reads and writes (can lose one disk per RAID-5 group and still rebuild)

RAID Cards

- Purpose:
  - Offload RAID calculations from CPU, including parity
  - Routine disk consistency checks
  - Cache
- Tips:
  - Controller Cache is best mostly for writes
  - Write-back cache is good - Beware of “learn cycles”
  - Disk Cache - best disabled on SAS drives. SATA drives frequently use for NCQ
  - Stripe size - should be at least the size of the basic block being accessed. Bigger usually better for larger files
  - Read ahead - depends on access patterns

LVM

Why use it?

- Ability to easily expand disk
- Snapshots (easy for dev, proof of concept, backups)

Cost?

- Straight usage usually 2-3% performance penalty
- With 1 snapshot 40-80% penalty
- Additional snapshots are only 1-2% additional penalty each

IO Scheduler

Goal - minimize seeks, prioritize process io

- CFQ - multiple queues, priorities, sync and async
- Anticipatory - anticipatory pauses after reads, not useful with RAID or TCQ
- Deadline - "deadline" contract for starting all requests, best with many disk RAID or TCQ
- Noop - tries to not interfere, simple FIFO, recommended for VM's and SSD's

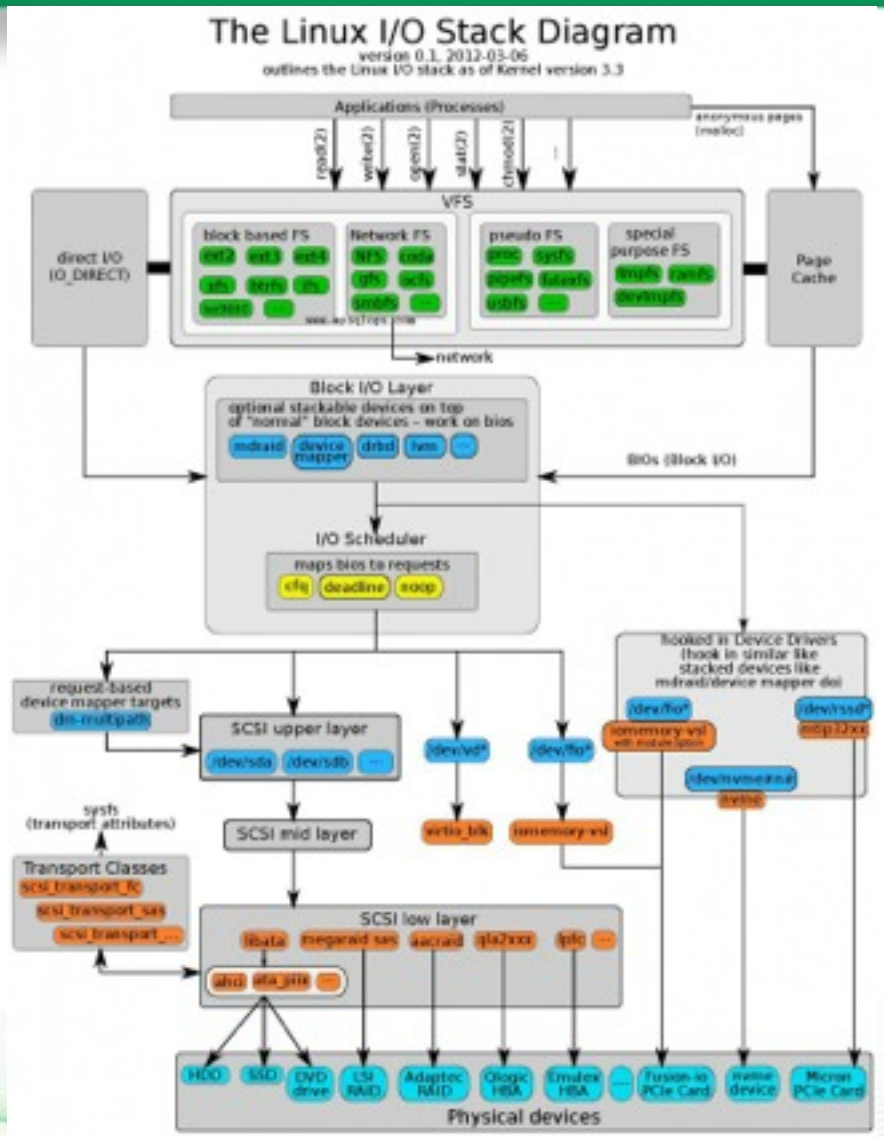
Filesystem Concepts

- Inode - stores, block pointers and metadata of a file or directory
- Block - stores data
- Superblock - stores filesystem metadata
- Extent - contiguous "chunk" of free blocks
- Journal - record of pending and completed writes
- Barrier - safety mechanism when dealing with RAID or disk caches
- fsck - filesystem check

VFS Layer

- API layer between system calls and filesystems, similar to MySQL storage engine API layer

Linux IO Sub-System



Filesystem Choices

In the style of Edgar Allan Poe's "The Raven"…

Once upon a SQL query
While I joked with Apple's Siri
Formatting many a logical volume on my quad core
Suddenly there came an alert by email
as of some threshold starting to wail
wailing like my SMS tone
"Tis just Nagios" I muttered,
"sending alerts unto my phone,
Only this - I might have known."

Ext filesystems

- ext2 - no journal
- ext3 - adds journal, some enhancements like directory hashes, online resizing
- ext4 - adds extents, barriers, journal checksum, removes inode locking
- common features - block groups, reserved blocks

- ex2/3 max FS size=32 TiB, max file size=2 TiB
- ext4 max FS size=1 EiB, max file size=16 TiB

XFS

- extents, data=writeback style journaling, barriers, delayed allocation, dynamic inode creation, online growth, cannot be shrunk

- max FS size=16 EiB, max file size 8 EiB

Btrfs

- extents, data and metadata checksums, compression, subvolumes, snapshots, online b-tree rebalancing and defrag, SSD TRIM support

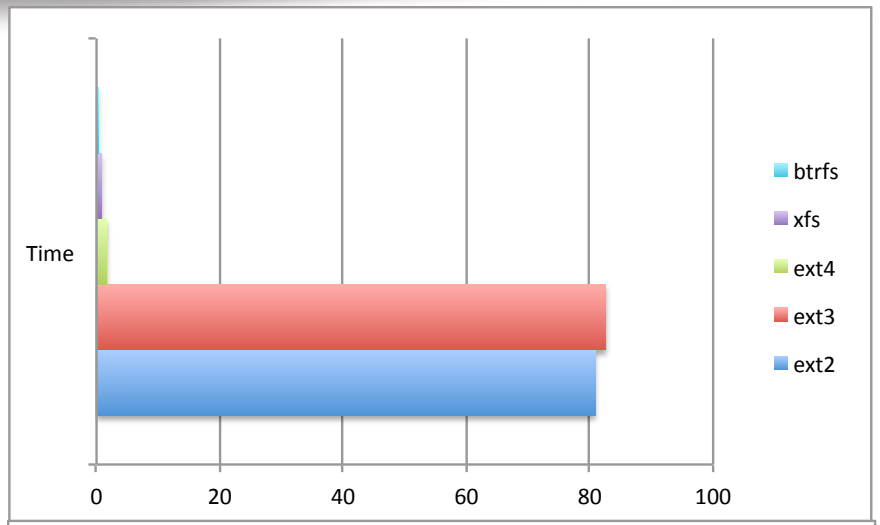
- max FS size=16 EiB, max file size 16 EiB

ZFS*

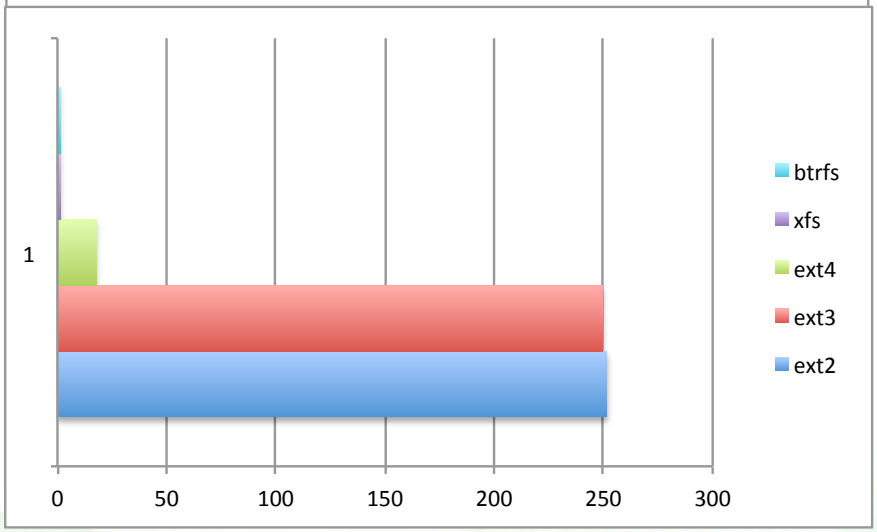
- volume management, RAID-Z, continuous integrity checking, extents, data and metadata checksums, compression, subvolumes, snapshots, encryption, ARC cache, transactional writes, deduplication
- max FS size=16 EiB, max file size 16
- * note that not all these features are yet supported natively on Linux

Filesystem Maintenance

- FS Creation (732GB)
  - Less is better



- FSCK
  - Less is better



MySQL Tuning Options

Continuing in the style of "The Raven"…

Ah distinctly I remember
as I documented for each member
of the team just last Movember
in the wiki that we keep
write and keep and nothing more…
When my query thus completed
Fourteen duplicate rows deleted
All my replicas then repeated
repeated the changes as before
I dumped it all to a shared disk
kept as a backup forever more.

MySQL Tuning Options for IO

- innodb_flush_logs_at_trx_commit
- innodb_flush_method
- innodb_buffer_pool_size
- innodb_io_capacity
- Innodb_adaptive_flushing
- Innodb_change_buffering
- Innodb_log_buffer_size
- Innodb_log_file_size
- innodb_max_dirty_pages_pct
- innodb_max_purge_lag
- innodb_open_files
- table_open_cache
- innodb_page_size
- innodb_random_read_ahead
- innodb_read_ahead_threshold
- innodb_read_io_threads
- innodb_write_io_threads
- sync_binlog
- general_log
- slow_log
- tmp_table_size, max_heap_table_size

InnoDB Flush Method

- Applies to InnoDB Log and Data file writes
- O_DIRECT - "Try to minimize cache effects of the I/O to and from this file. In general this will degrade performance, but it is useful in special situations, such as when applications do their own caching. File I/O is done directly to/from user space buffers." - Applies to log and data files, follows up with fsync, eliminates need for doublewrite buffer
- DSYNC - "Write I/O operations on the file descriptor shall complete as defined by synchronized I/O data integrity completion." - Applies to log files, data files get fsync
- fdatasync - (deprecated option in 5.6) Default mode. fdatasync on every write to log or disk
- O_DIRECT_NO_FSYNC - (5.6 only) O_DIRECT without fsync (not suitable for XFS)
- fsync - flush all data and metadata for a file to disk before returning
- fdatasync - flush all data and only metadata necessary to read the file properly to disk before returning

InnoDB Flush Method - Notes

- O_DIRECT - "The thing that has always disturbed me about O_DIRECT is that the whole interface is just stupid, and was probably designed by a deranged monkey on some serious mind-controlling substances." --Linus Torvalds
- O_DIRECT - "The behaviour of O_DIRECT with NFS will differ from local file systems. Older kernels, or kernels configured in certain ways, may not support this combination. The NFS protocol does not support passing the flag to the server, so O_DIRECT I/O will only bypass the page cache on the client; the server may still cache the I/O. The client asks the server to make the I/O synchronous to preserve the synchronous semantics of O_DIRECT. Some servers will perform poorly under these circumstances, especially if the I/O size is small. Some servers may also be configured to lie to clients about the I/O having reached stable storage; this will avoid the performance penalty at some risk to data integrity in the event of server power failure. The Linux NFS client places no alignment restrictions on O_DIRECT I/O."
- DSYNC - "POSIX provides for three different variants of synchronized I/O, corresponding to the flags O_SYNC, O_DSYNC, and O_RSYNC. Currently (2.6.31), Linux only implements O_SYNC, but glibc maps O_DSYNC and O_RSYNC to the same numerical value as O_SYNC. Most Linux file systems don't actually implement the POSIX O_SYNC semantics, which require all metadata updates of a write to be on disk on returning to user space, but only the O_DSYNC semantics, which require only actual file data and metadata necessary to retrieve it to be on disk by the time the system call returns."

Benchmarks

There once was a small database program
It had InnoDB and MyISAM
One did transactions well,
and one would crash like hell
Between the two they used all of my RAM

- A database Limerick -

Testing Setup...

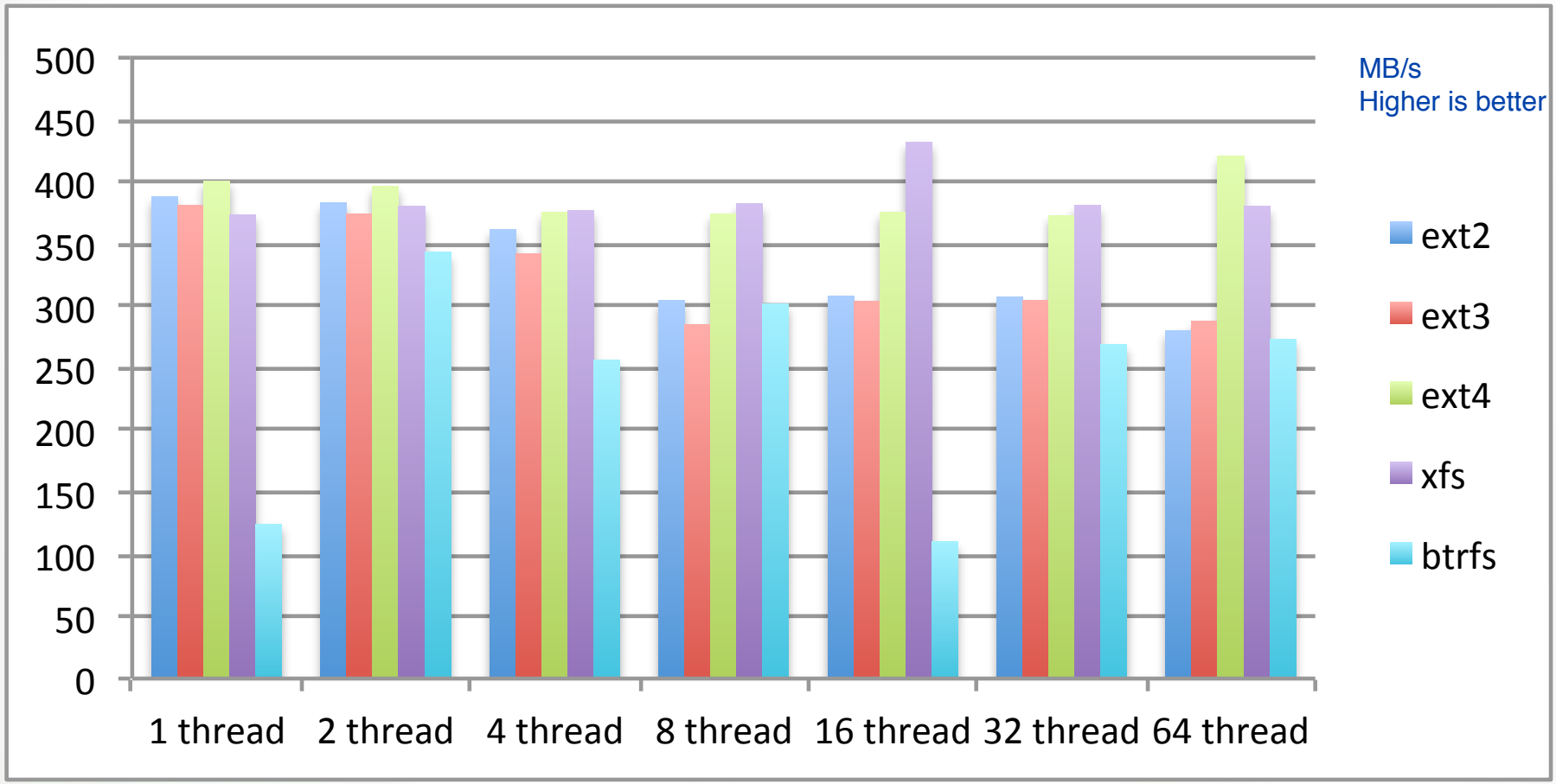
- Dell PowerEdge 1950
  - 2x Quad-core Intel Xeon 5150 @ 2.66 Ghz
  - 16 GB RAM
  - 4 x 300 GB SAS disks at 10k rpm (RAID-5, 64KB stripe size)
  - Dell Perc 6/i RAID Controller with 512MB cache
  - CentOS 6.4 (sysbench io tests done with Ubuntu 12.10)
  - MySQL 5.5.30

Testing Setup (cont)

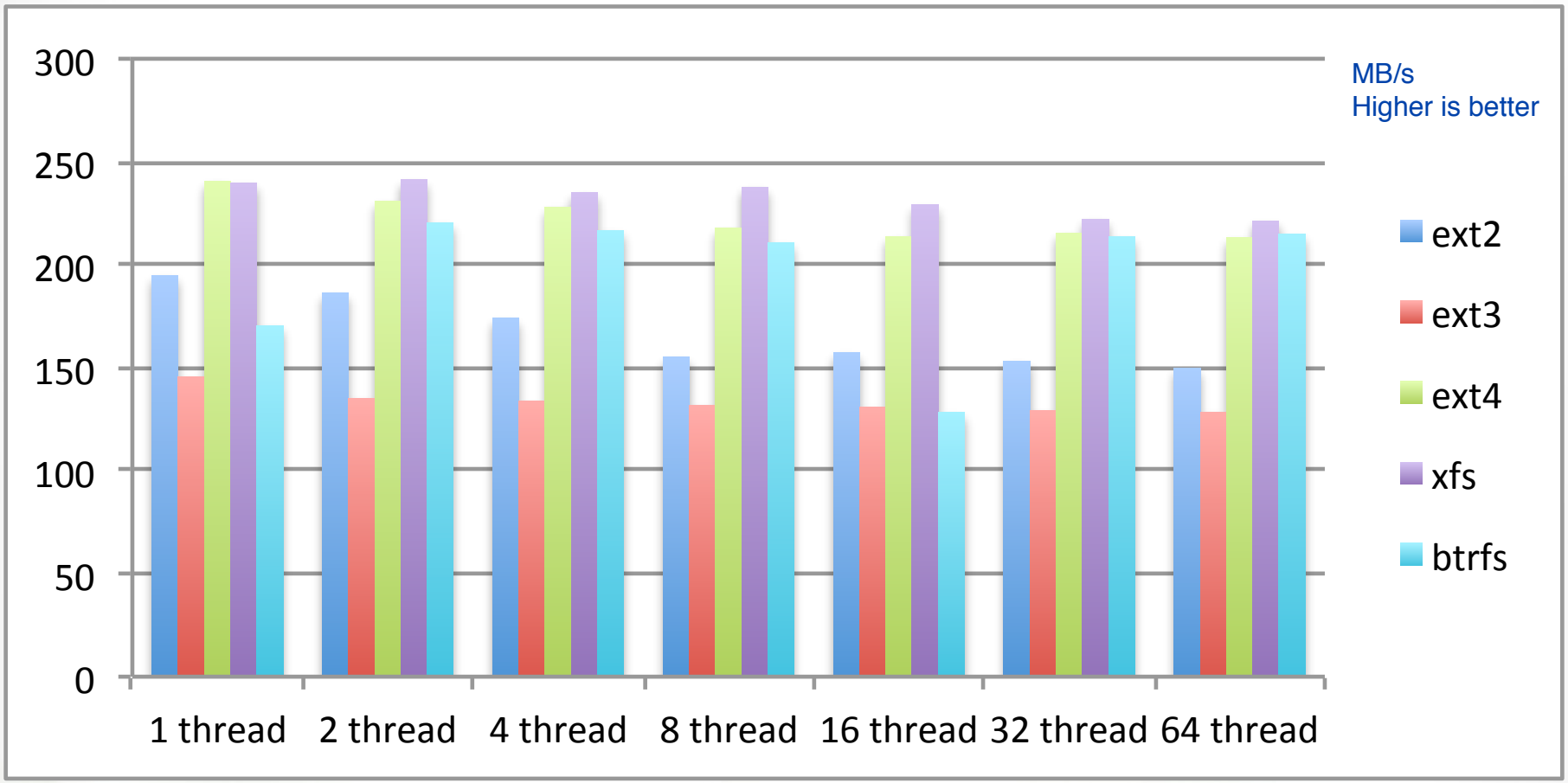
my.cnf settings:

```
log-error
skip-name-resolve
key_buffer = 1G
max_allowed_packet = 1G
query_cache_type=0
query_cache_size=0
slow-query_log=1
long-query-time=1
log-bin=mysql-bin
max_binlog_size=1G
binlog_format=MIXED
innodb_buffer_pool_size = 4G # or 14G, see tests
innodb_additional_mem_pool_size = 16M
innodb_log_file_size    = 1G
innodb_file_per_table   = 1
innodb_flush_method     = O_DIRECT   # Unless specified as fdatasync or O_DSYNC
innodb_flush_log_at_trx_commit = 1
### innodb_doublewrite_buffer=0   # for zfs tests only
```

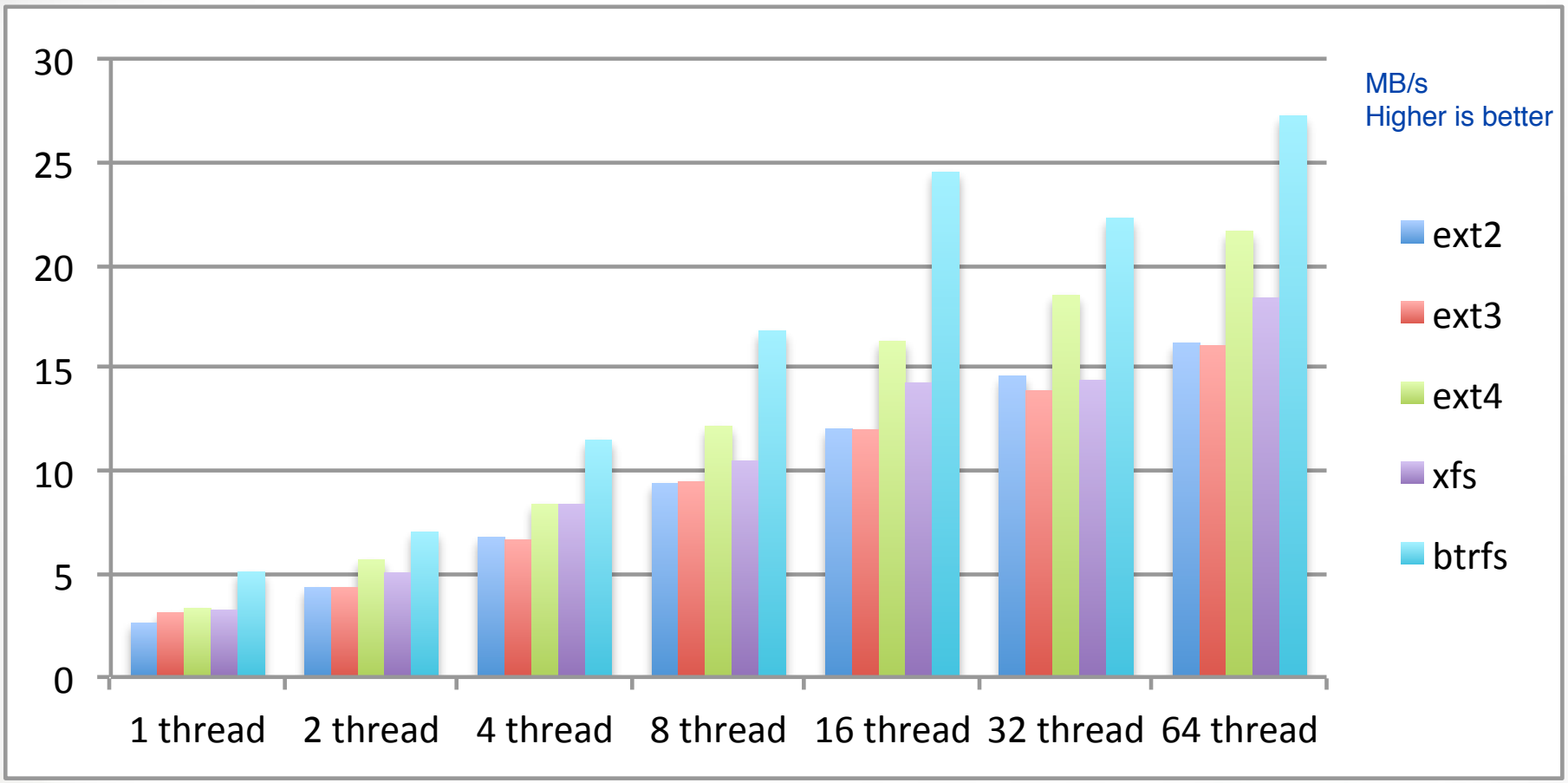
IO Tests - Sysbench - Sequential Reads



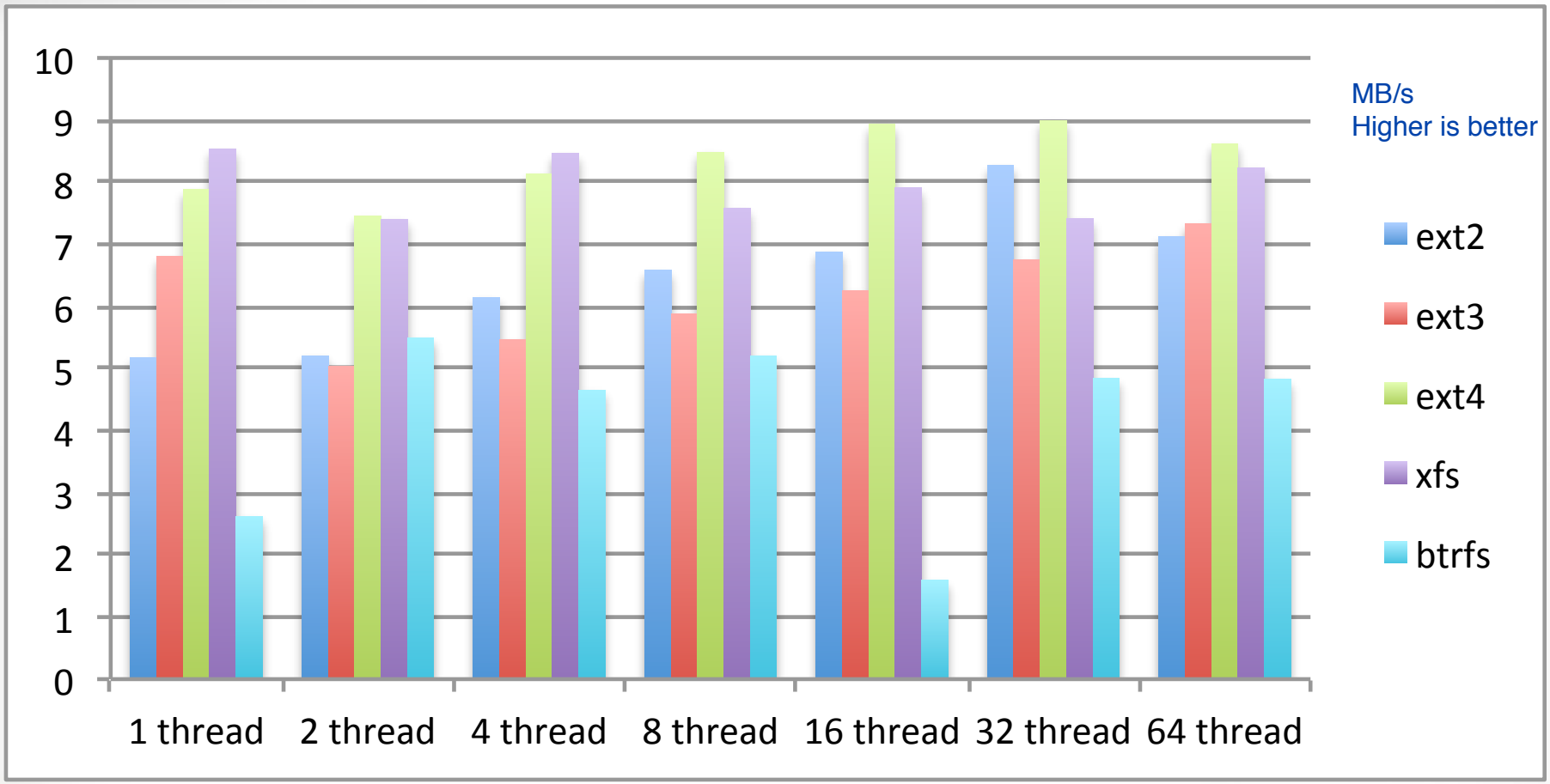
IO Tests - Sysbench - Sequential Writes



IO Tests - Sysbench - Random Reads



IO Tests - Sysbench - Random Writes

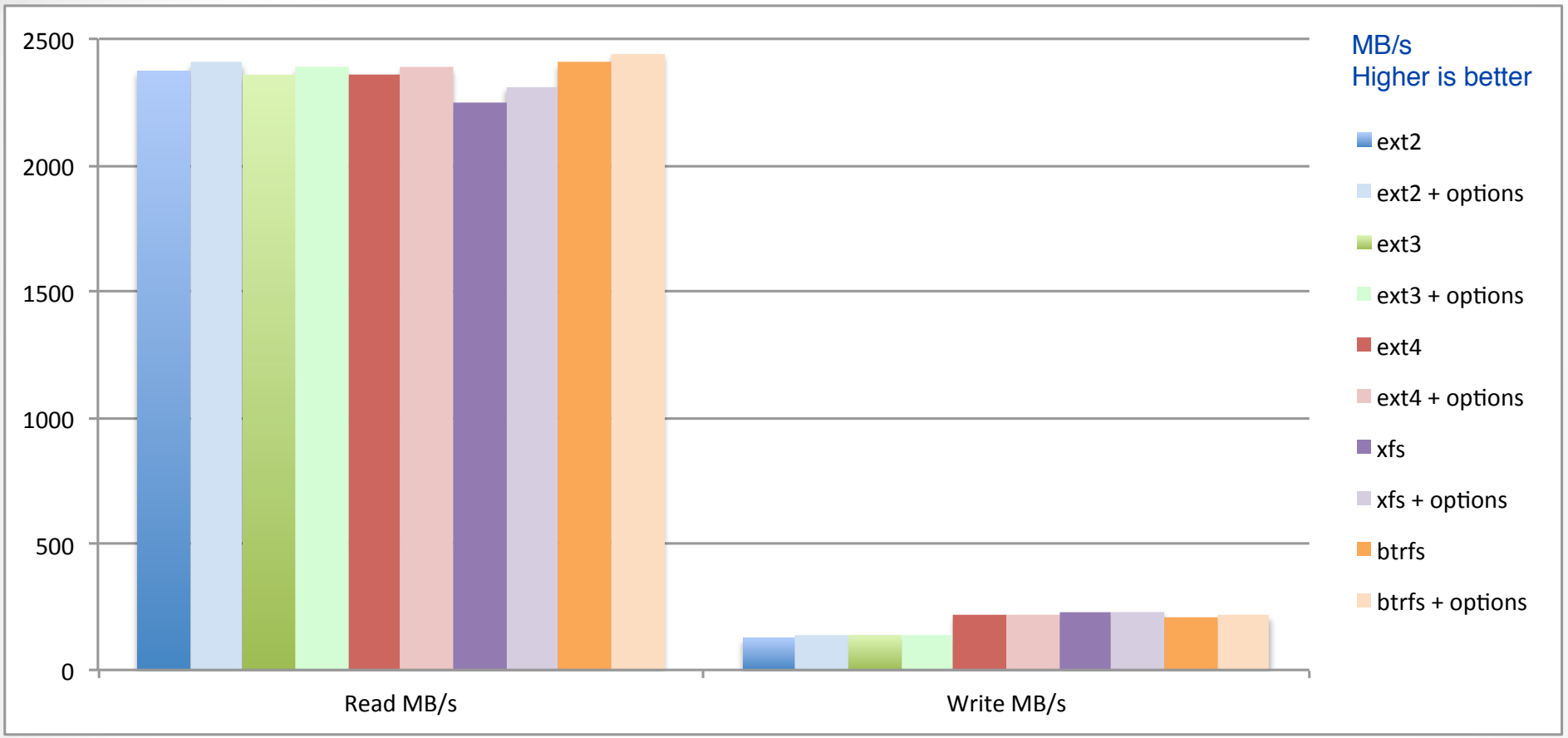


Mount Options

ext2: noatime
ext3: noatime
ext4: noatime,barrier=0
xfs: inode64,nobarrier,noatime,logbufs=8
btrfs: noatime,nodatacow,space_cache
zfs: noatime (recordsize=16k, compression=off, dedup=off)

all - **noatime** - Do not update access times (atime) metadata on files after reading or writing them
ext4 / xfs - **barrier=0 / nobarrier** - Do not use barriers to pause and receive assurance when writing (aka, trust the hardware)
xfs - **inode64** - use 64 bit inode numbering - became default in most recent kernel trees
xfs - **logbufs=8** - Number of in-memory log buffers (between 2 and 8, inclusive)
btrfs - **space_cache** - Btrfs stores the free space data ondisk to make the caching of a block group much quicker (Kernel 2.6.37+). It's a persistent change and is safe to boot into old kernels
btrfs - **nodatacow** - Do not copy-on-write data. datacow is used to ensure the user either has access to the old version of a file, or to the newer version of the file. datacow makes sure we never have partially updated files written to disk. nodatacow gives slight performance boost by directly overwriting data (like ext[234]), at the expense of potentially getting partially updated files on system failures. Performance gain is usually < 5% unless the workload is random writes to large database files, where the difference can become very large
btrfs - **compress=zlib** - Better compression ratio. It's the default and safe for olders kernels
btrfs - **compress=lzo** - Fastest compression. btrfs-progs 0.19 or olders will fail with this option. The default in the kernel 2.6.39 and newer

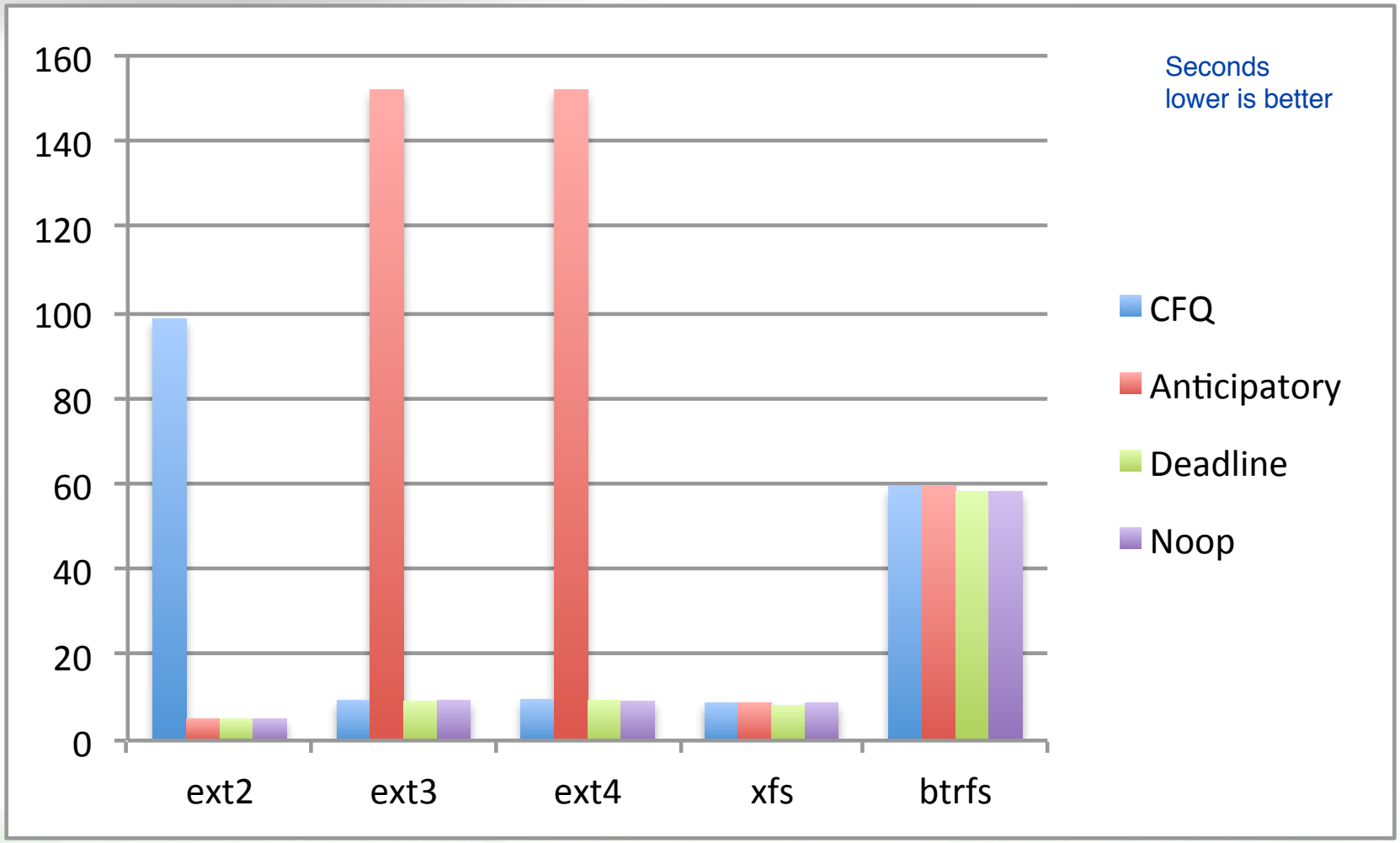
iobench with mount options



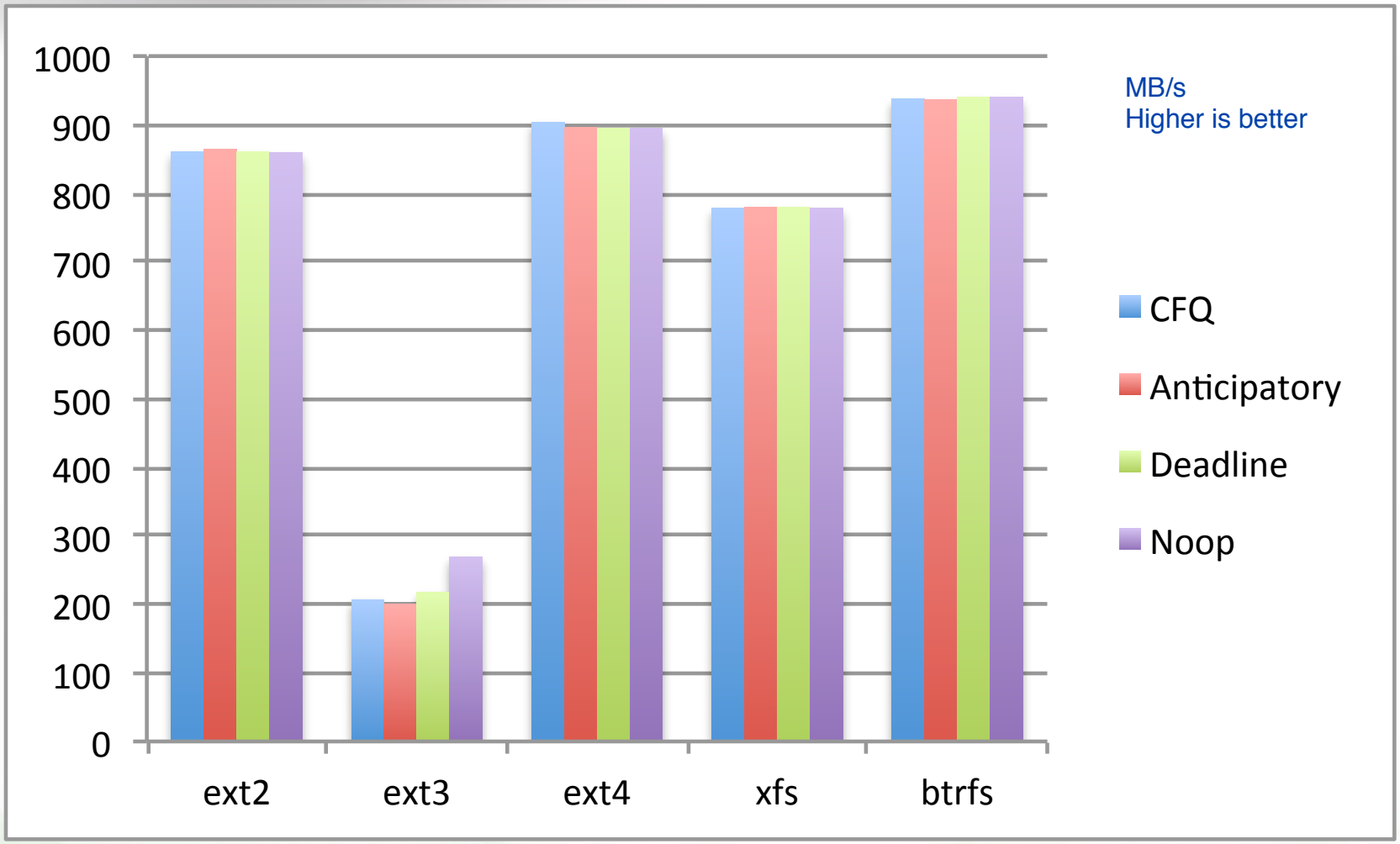
IO Scheduler Choices

Round and round the disk drive spins
but SSD sits still and grins.
It is randomly fast
for data current and past.
My database upgrade begins

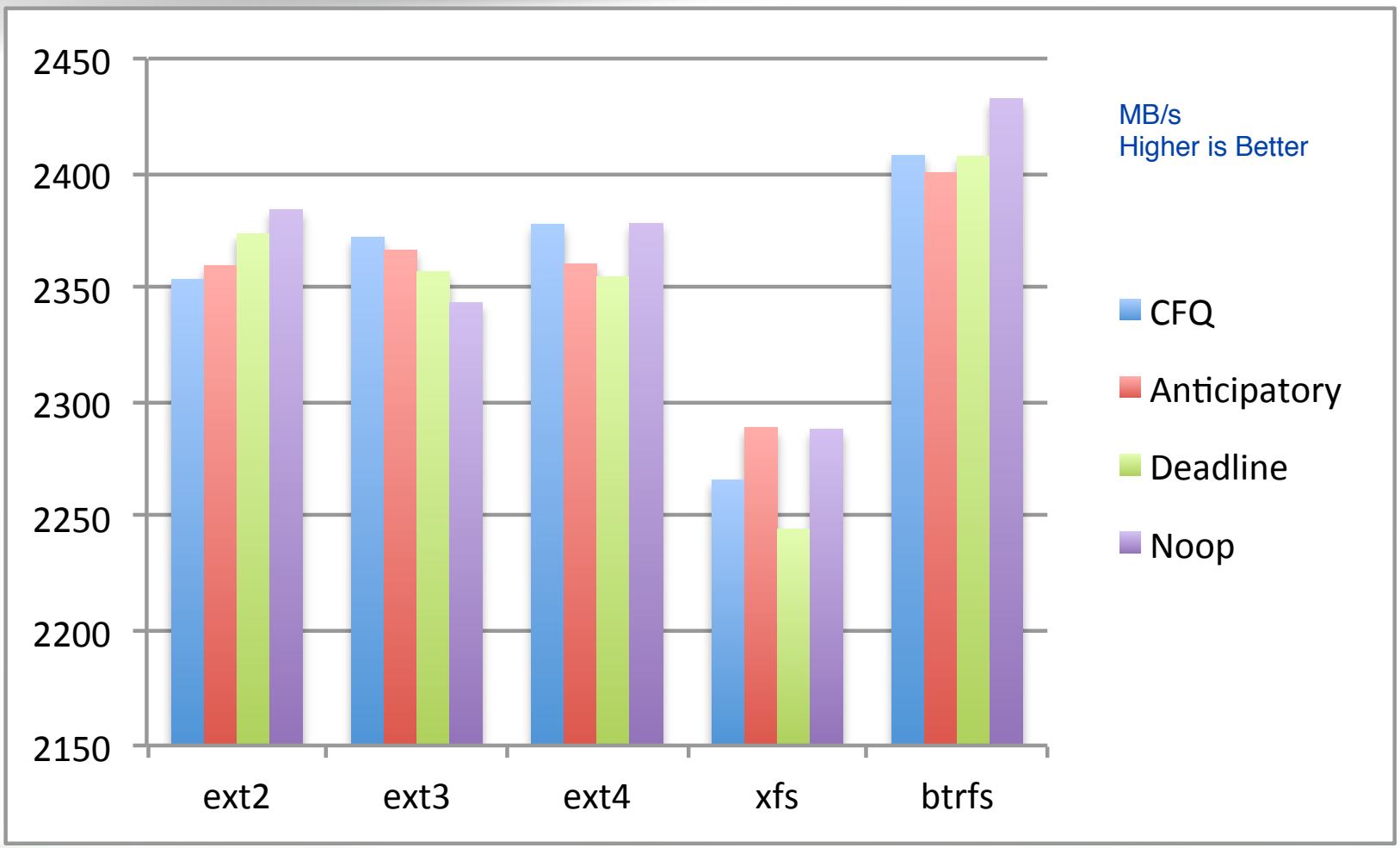
SQLite


160
140
120
100
80
60
40
20
0
ext2
ext3
ext4
xfs
btrfs
Seconds
lower is better
CFQ
Anticipatory
Deadline
Noop

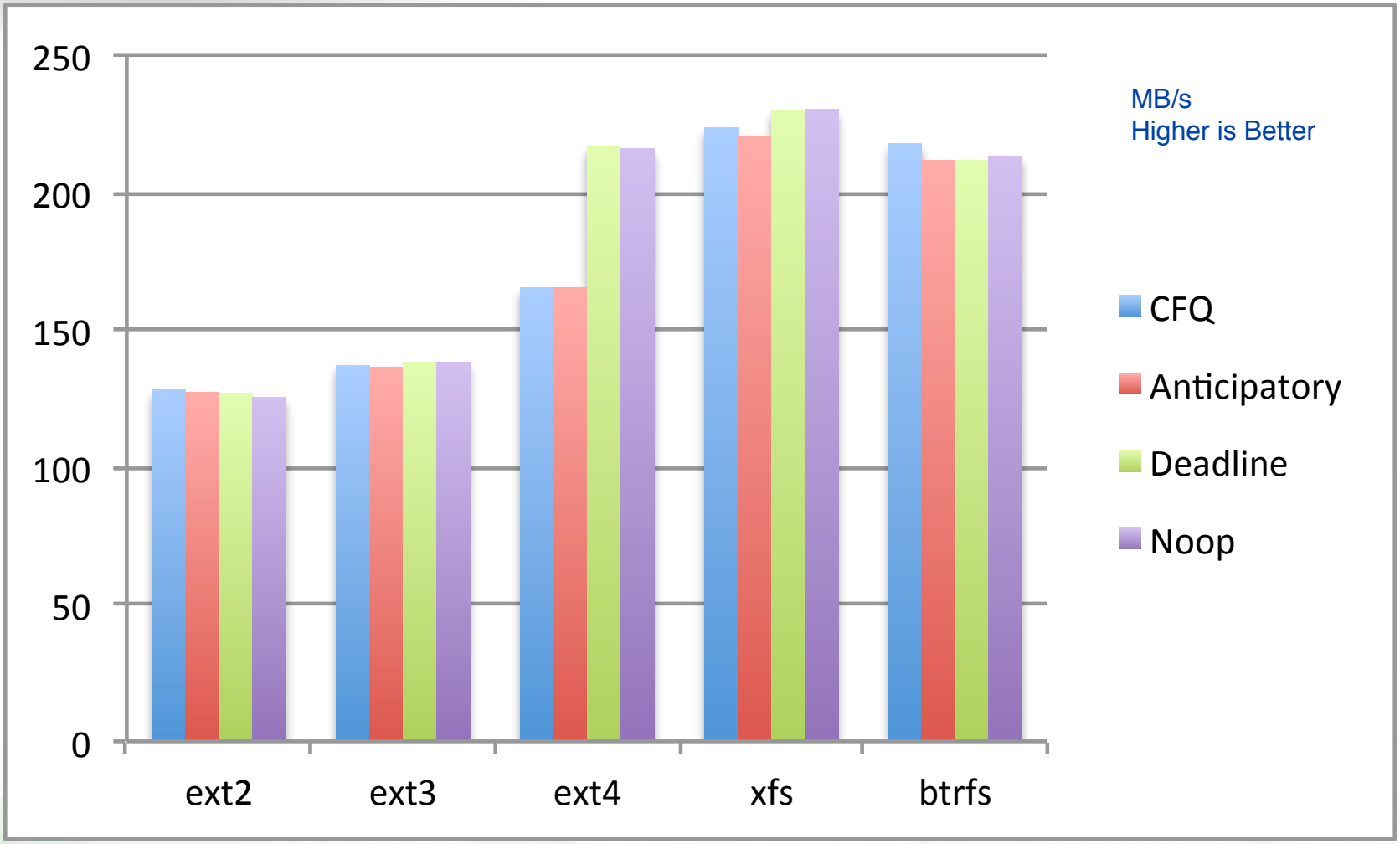
aio-stress


1000
900
800
700
600
500
400
300
200
100
0
ext2
ext3
ext4
xfs
btrfs
MB/s
Higher is better
CFQ
Anticipatory
Deadline
Noop

iozone read


2450
2400
2350
2300
2250
2200
2150
ext2
ext3
ext4
xfs
btrfs
MB/s
Higher is Better
CFQ
Anticipatory
Deadline
Noop

iozone write

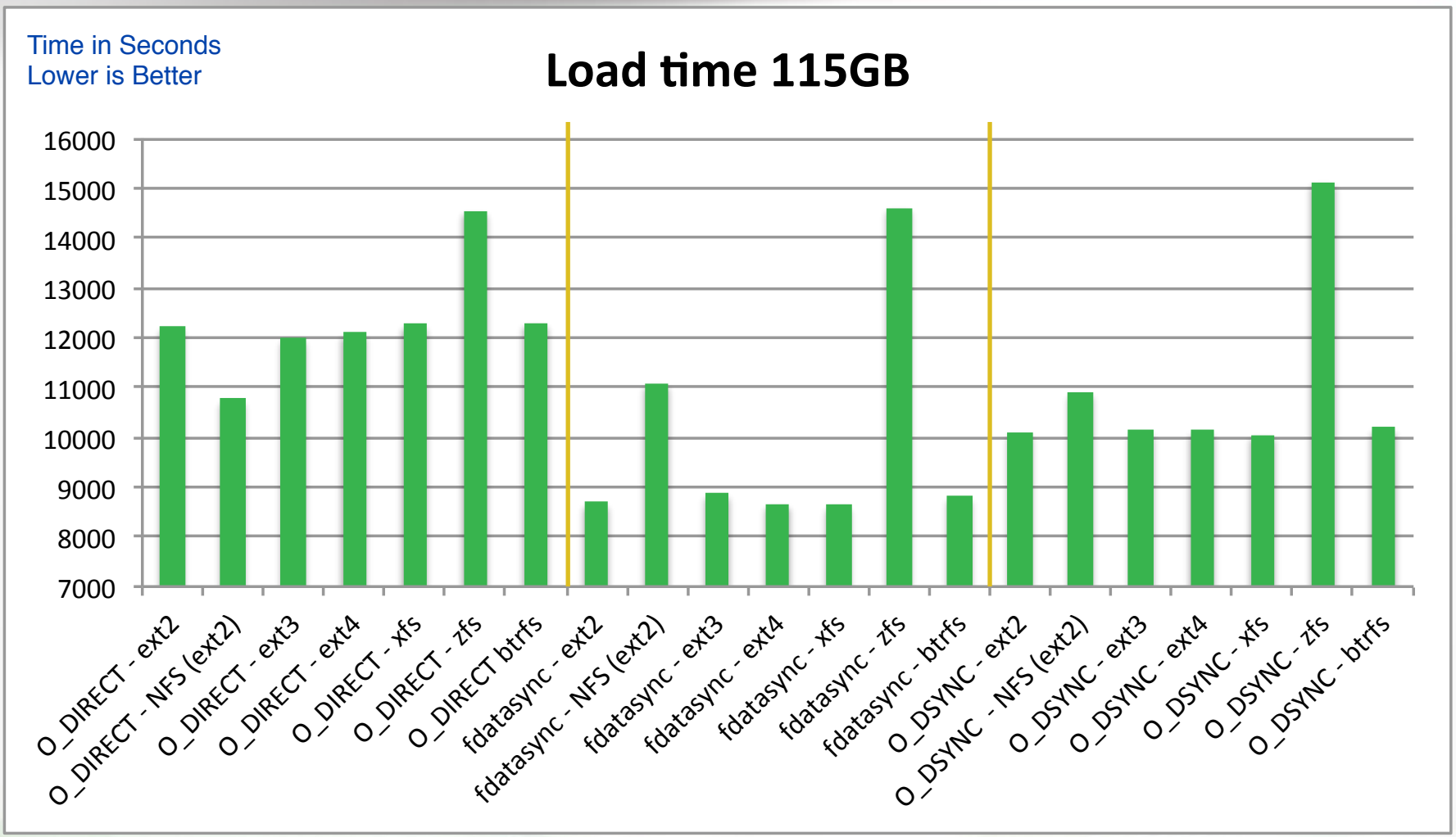


Real World Workloads

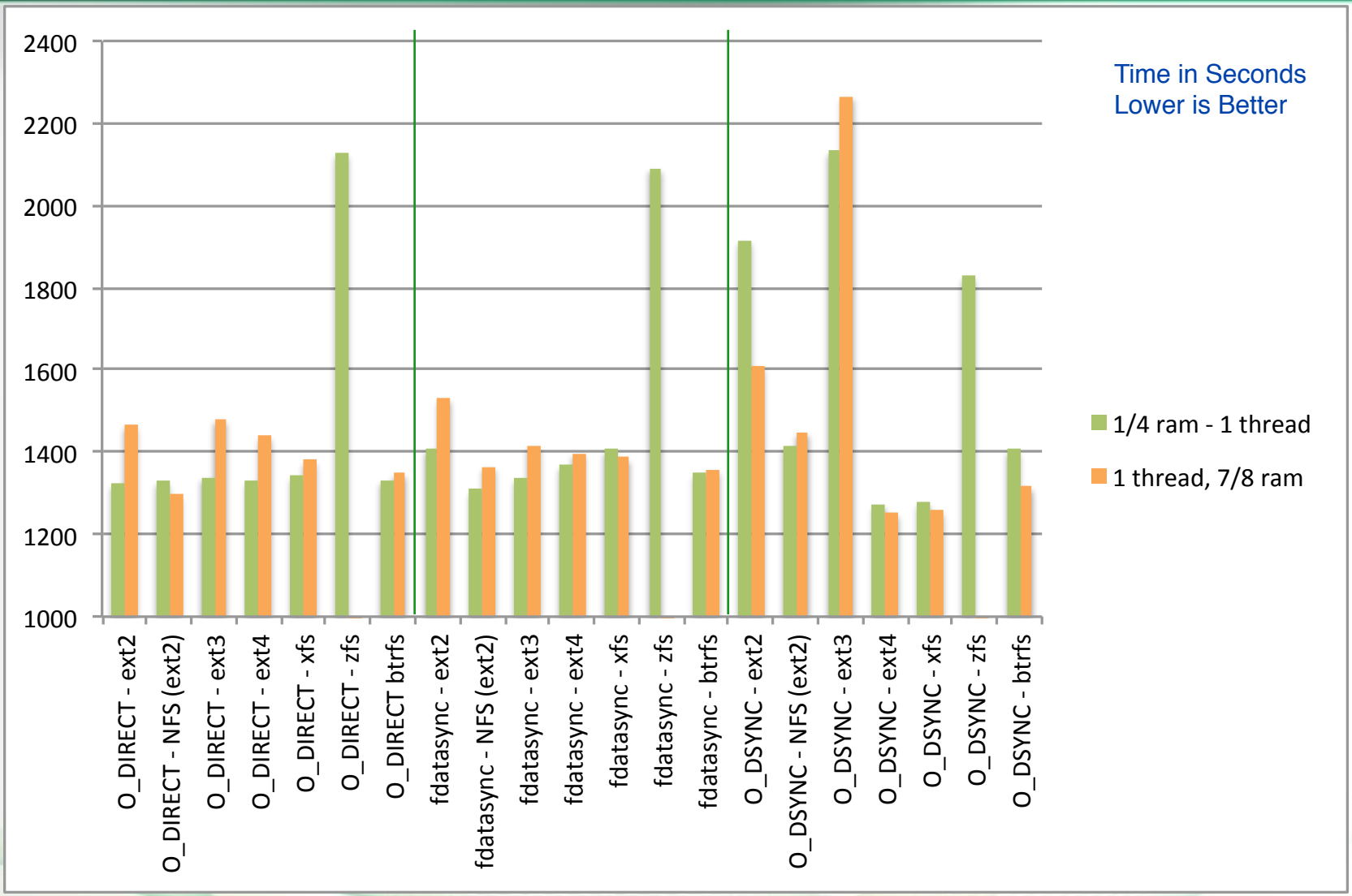
Flush local tables
Make an LVM snapshot
Backup with rsync

- A Haiku on easy backups -

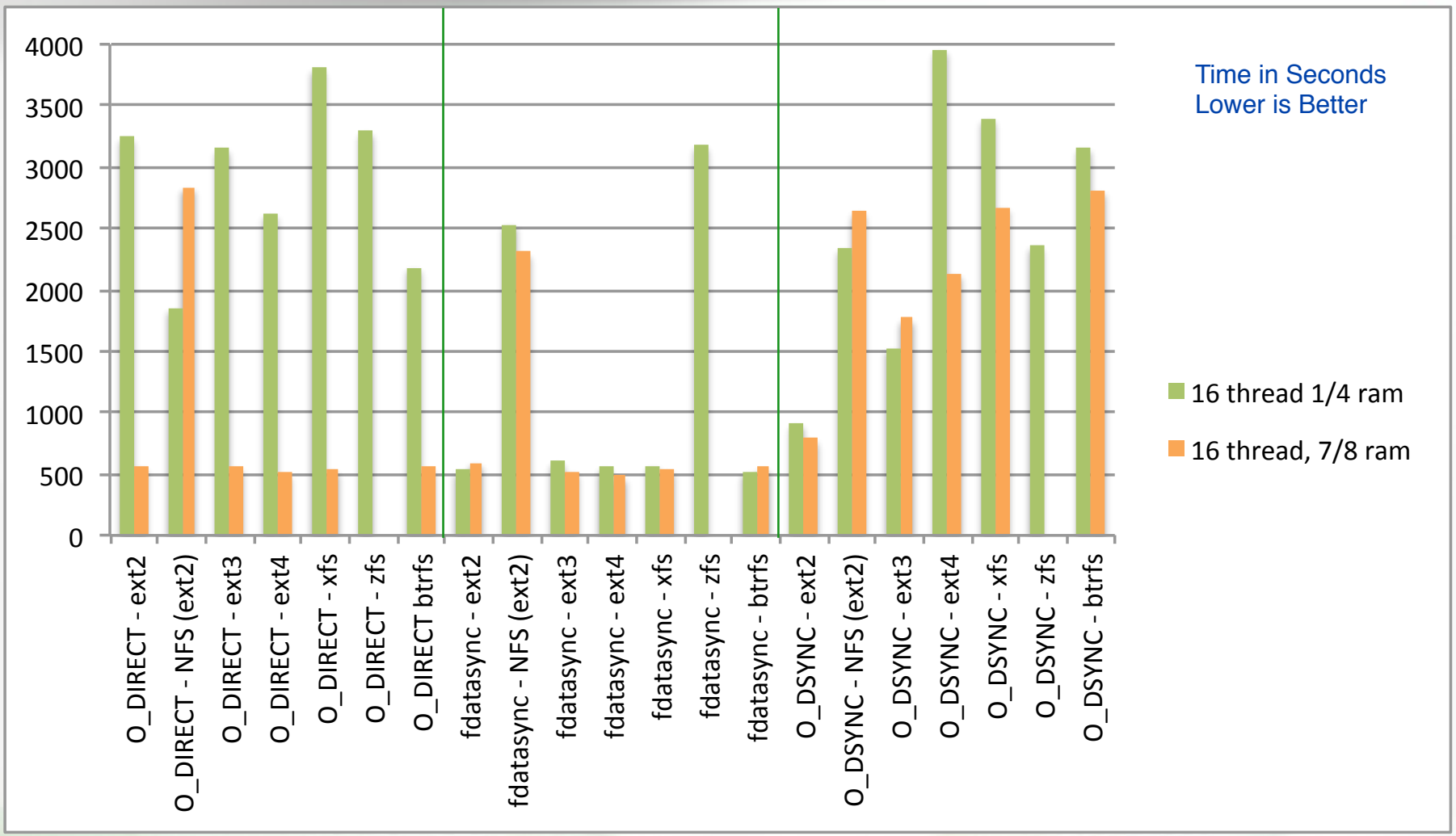
Data Loading Performance



OLTP Performance - 1 thread



OLTP Performance - 16 thread



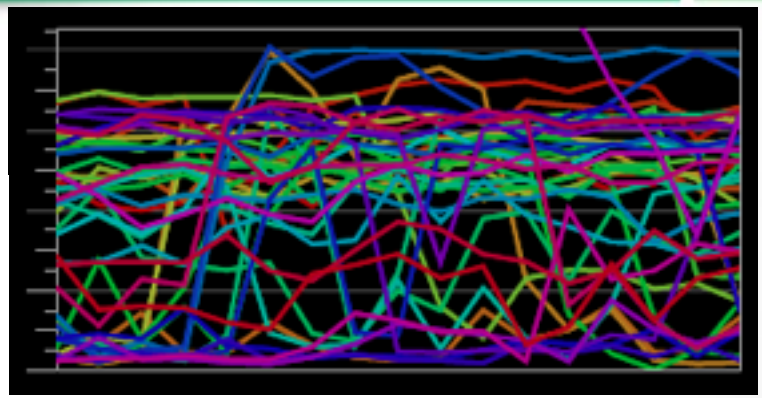
AWS Cloud Options

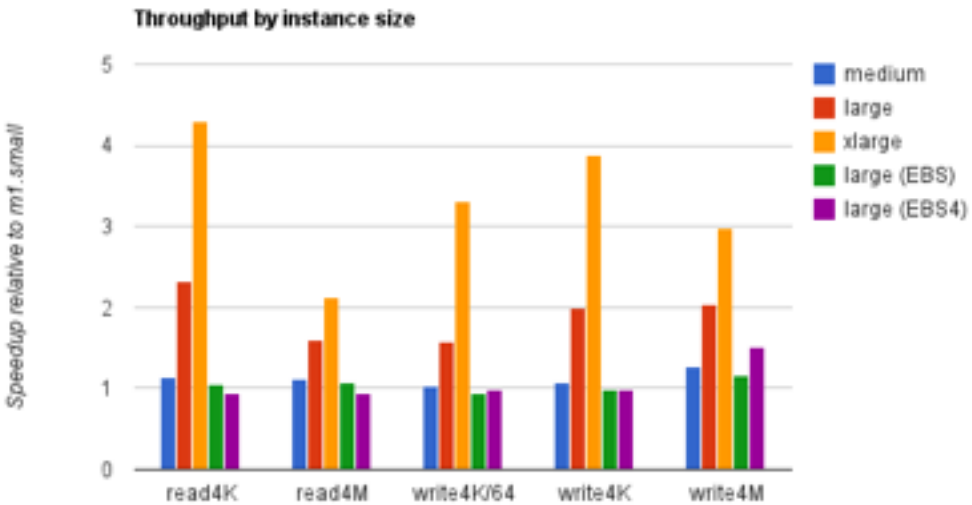
Performance, uptime,
Consistency and scale-up:
No, this is a cloud…

- A haiku on clouds -

Cloud Performance

- EC2 - Slightly unpredictable





- *Note: not my research or graphs. See blog.scalyr.com for benchmarks and writeup

Conclusions

Oracle is Red,
IBM is Blue,
I like stuff for free
MySQL will do.

Conclusions

IO Schedulers - Deadline or Noop

Filesystem - Ext3 is usually slowest. Btrfs not there quite yet but looking better. Linux zfs is cool, but performance is sub-par.

InnoDB Flush Method - O_DIRECT not always best

Filesystem Mount options make a difference

Artificial benchmarks are fun, but like most things comparative speed is very workload dependent

Further Reading...

For more information please see these great resources:

Wikipedia:

http://en.wikipedia.org/wiki/Ext2 and http://en.wikipedia.org/wiki/Ext3 and http://en.wikipedia.org/wiki/Ext4 and http://en.wikipedia.org/wiki/XFS and http://en.wikipedia.org/wiki/Btrfs

MySQL Performance Blog:

http://www.mysqlperformanceblog.com/2009/02/05/disaster-lvm-performance-in-snapshot-mode/

http://www.mysqlperformanceblog.com/2012/05/22/btrfs-probably-not-ready-yet/

http://www.mysqlperformanceblog.com/2013/01/03/is-there-a-room-for-more-mysql-io-optimization/

http://www.mysqlperformanceblog.com/2012/03/15/ext4-vs-xfs-on-ssd/

http://www.mysqlperformanceblog.com/2011/12/16/setting-up-xfs-the-simple-edition/

MySQL at Facebook (and dom.as blog):

http://dom.as/2008/11/03/xfs-write-barriers/

http://www.facebook.com/note.php?note_id=10150210901610933

Dimitrik:

http://dimitrik.free.fr/blog/archives/2012/01/mysql-performance-linux-io.html

http://dimitrik.free.fr/blog/archives/02-01-2013_02-28-2013.html#159

http://dimitrik.free.fr/blog/archives/2011/01/mysql-performance-innodb-double-write-buffer-redo-log-size-impacts-mysql-55.html

...Further Reading

For more information please see these great resources:

Phoronix:

http://www.phoronix.com/scan.php?page=article&item=ubuntu_1204_fs&num=1

http://www.phoronix.com/scan.php?page=article&item=linux_39_fs&num=1

http://www.phoronix.com/scan.php?page=article&item=fedora_15_lvm&num=3

Misc:

http://erikugel.wordpress.com/2011/04/14/the-quest-for-the-fastest-linux-filesystem/

https://raid.wiki.kernel.org/index.php/Performance

http://uclibc.org/~aldot/mkfs_stride.html

http://indico.cern.ch/getFile.py/access?contribId=3&sessionId=0&resId=1&materialId=paper&confId=13797

http://linux.die.net/man/2/open

http://linux.die.net/man/2/fsync

http://blog.scalyr.com/2012/10/16/a-systematic-look-at-ec2-io/

http://docs.openstack.org/trunk/openstack-object-storage/admin/content/filesystem-considerations.html

https://btrfs.wiki.kernel.org/index.php/Main_Page

http://zfsonlinux.org/

https://blogs.oracle.com/realneel/entry/mysql_innodb_zfs_best_practices

Parting thought

Do you like MyISAM?
I do not like it, Sam-I-am.
I do not like MyISAM.

Would you use it here or there?
I would not use it here or there.
I would not use it anywhere.
I do not like MyISAM.
I do not like it, Sam-I-am.

Would you like it in an e-commerce site?
Would you like it with in the middle of the night?
I do not like it for an e-commerce site.
I do not like it in the middle of the night.
I would not use it here or there.
I would not use it anywhere.
I do not like MyISAM.
I do not like it Sam-I-am.

Would you could you for foreign keys?
Use it, use it, just use it please!
You may like it, you will see
Just convert these tables three…
Not for foreign keys, not for those tables three!
I will not use it, you let me be!