PostgreSQL on EXT3/4, XFS, BTRFS and ZFS

pgconf.eu 2015, October 27-30, Vienna

Tomas Vondra
tomas.vondra@2ndquadrant.com

2ndQuadrant
Professional PostgreSQL

not a filesystem engineer

database engineer

Which file system should I use for PostgreSQL in production?

According to results of our benchmarks

from 2003 the best file system ...

What does it mean that a file system is “stable” and “production ready”?

I don't hate any of the filesystems!

SSD

File systems

- EXT3/4, XFS, … (and others)
  - traditional design, generally from 90s
  - same goals, different features and tuning options
  - incremental improvements, reasonably “modern”
  - mature, reliable, battle-tested
- BTRFS, ZFS
  - next-gen CoW file systems, new architecture / design
- others (not really discussed in the talk)
  - log-organized, distributed, clustered, ...

a bit about history

EXT3, EXT4, XFS

- EXT3 (2001) / EXT4 (2008)
  - evolution of original Linux file system (ext, ext2, ...)
  - improvements, bugfixes ...
- XFS (2002)
  - originally SGI Irix 5.3 (1994)
  - 2000 - released under GPL
  - 2002 – merged into 2.5.36
- both EXT4 and XFS are
  - reliable file systems with a journal
  - proven by time and many production deployments

EXT3, EXT4, XFS

- conceived in time of rotational devices
  - mostly work on SSDs
  - stop-gap for future storage systems (NVRAM, ...)
- mostly evolution, not revolution
  - adding features (e.g. TRIM, write barriers, ...)
  - scalability improvements (metadata, ...)
  - fixing bugs
- be careful when dealing with
  - obsolete benchmarks and anecdotal “evidence”
  - misleading synthetic benchmarks

EXT3, EXT4, XFS

- traditional design + journal
- not designed for
  - multiple devices
  - volume management
  - snapshots
  - ...
- require additional components to do that
  - hardware RAID
  - software RAID (dm)
  - LVM / LVM2

BTRFS, ZFS

BTRFS, ZFS

- fundamental ideas
  - integrating layers (LVM + dm + ...)
  - aimed at consumer level hardware (failures are common)
  - designed for larger data volumes

- which hopefully gives us …

  more flexible management
  - built-in snapshots
  - compression, deduplication
  - checksums

BTRFS, ZFS

- BTRFS
  - merged in 2009, still “experimental”
  - on-disk format marked as “stable” (1.0)
  - some say it's “stable” or even “production ready” ...
  - default in some distributions
- ZFS
  - originally Sun / Solaris, but “got Oracled” :-(
  - today slightly fragmented development (Illumos, Oracle, ...)
  - available on other BSD systems (FreeBSD)
  - “ZFS on Linux” project (but CDDL vs. GPL and such)

Generic tuning options

Generic tuning options

- TRIM (discard)
  - enable / disable sending TRIM commands to SSDs
  - influences internal cleanup processes / wear leveling
  - not necessary, may help the SSD with “garbage collection”
- write barriers
  - prevent the drive from reordering writes (journal x data)
  - does not protect against data loss (but consistency)
  - write cache + battery => write barriers may be turned off
- SSD alignment

Specific tuning options

BTRFS

- nodatacow
  - disables “copy on write” (CoW), but still done for snapshots
  - also disables checksums (requires “full” CoW)
  - also probably end of “torn-page resiliency” (have to do FPW)
- ssd
  - enables various SSD optimizations (unclear which ones)
- compress=lzo/zlib
  - compression (speculative)

ZFS

- recordsize=8kB
  - standard page 128kB (much larger than 8kB pages in PostgreSQL)
  - problems when caching in ARC (smaller number of “slots”)
- logbias=throughput [latency]
  - impacts work with ZIP (latence vs. throughput optimizations)
- zfs_arc_max
  - limitation of ARC cache size
  - should be modified automatically, but external kernel module ...
- primarycache=metadata
  - prevents double buffering (shared buffers vs. ARC)

ZFS

- recordsize=8kB
  - standard page 128kB (much larger than 8kB pages in PostgreSQL)
  - problems when caching in ARC (smaller number of “slots”)
- logbias=throughput [latency]
  - impacts work with ZIP (latence vs. throughput optimizations)
- zfs_arc_max
  - limitation of ARC cache size
  - should be modified automatically, but external kernel module ...
- ~~primarycache=metadata~~
  - ~~prevents double bufferingu (shared buffers vs. ARC)~~

Benchmark

System

- CPU: Intel i5-2500k
  - 4 cores @ 3.3 GHz (3.7GHz)
  - 6MB cache
  - 2011-2013
- 8GB RAM (DDR3 1333)
- SSD Intel S3700 100GB (SATA3)
- Gentoo + kernel 4.0.4
- PostgreSQL 9.4

pgbench (TPC-B)

- transactional benchmark / stress-test
  - small queries (access using PK, ...)
  - mix different typs of I/O (reads/writes, random/sequential)
- variants
  - read-write (SELECT + INSERT + UPDATE)
  - read-only (SELECT)
- data set sizes
  - small (~200MB)
  - medium (~50% RAM)
  - large (~200% RAM)

But it's not representative!

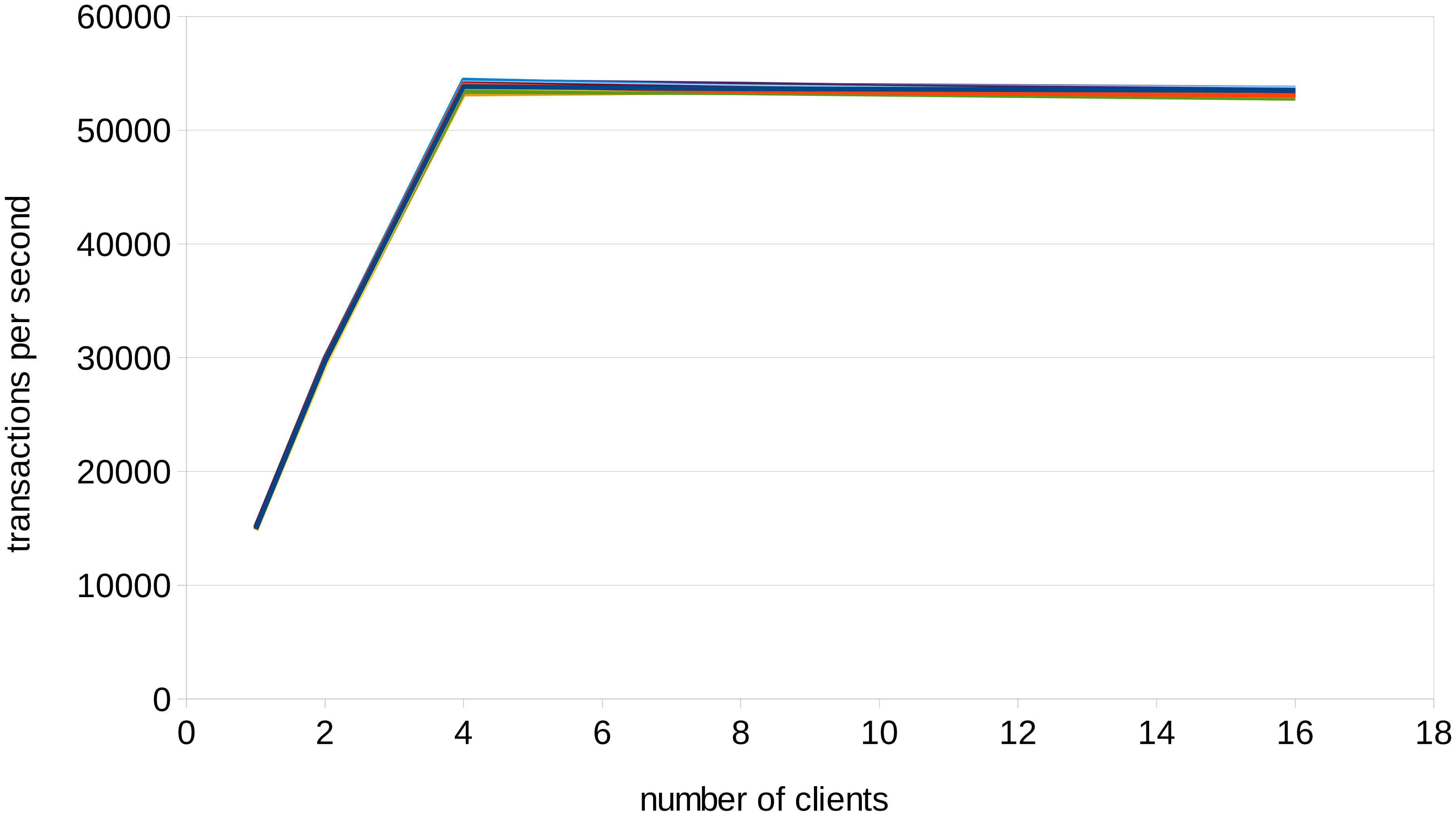
Results

- more than 40 combinations tested
- every test runs >4 days

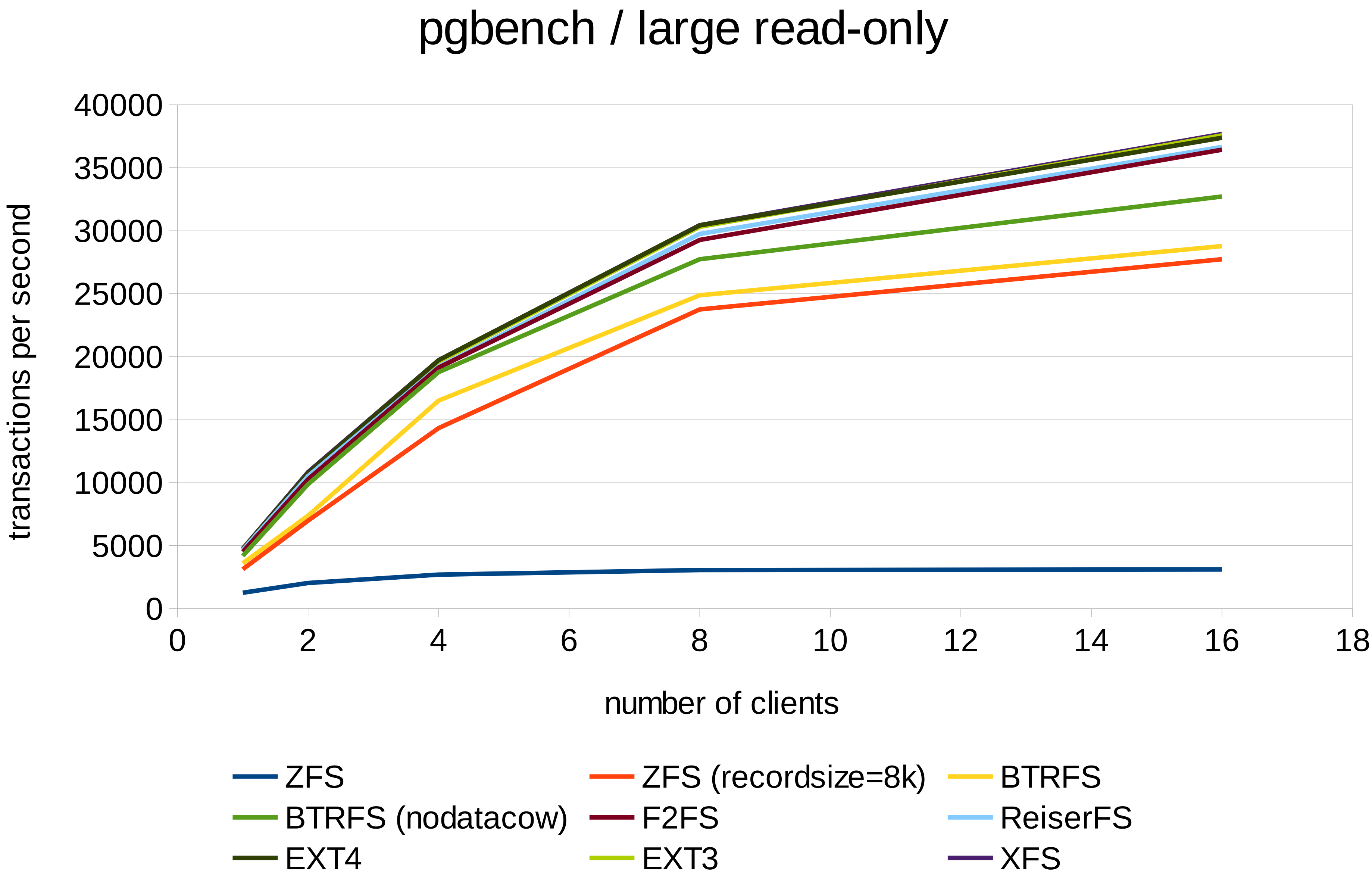
https://bitbucket.org/tvondra/fsbench-i5

pgbench read-only

pgbench / small read-only

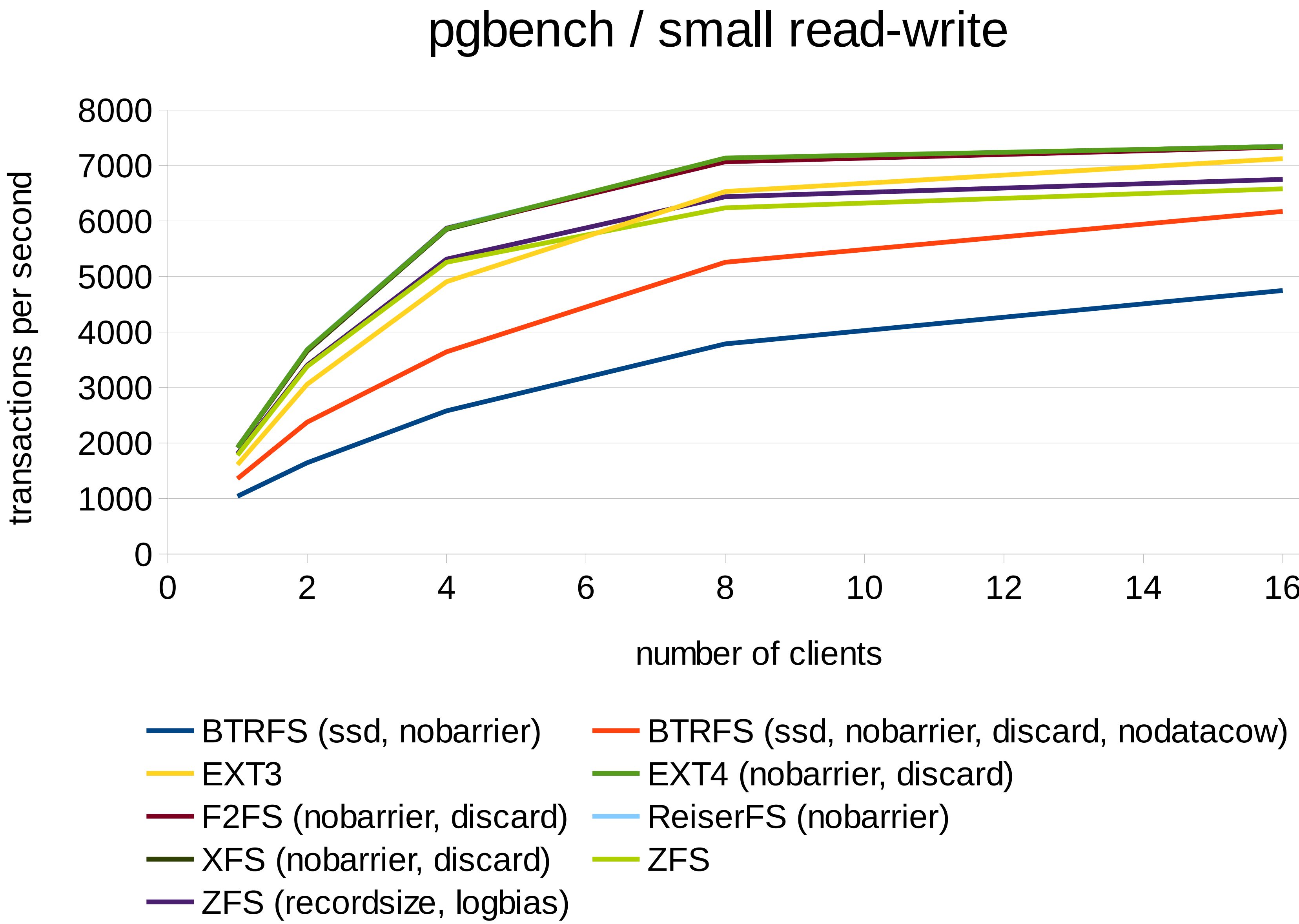


2ndQuadrant
Professional PostgreSQL



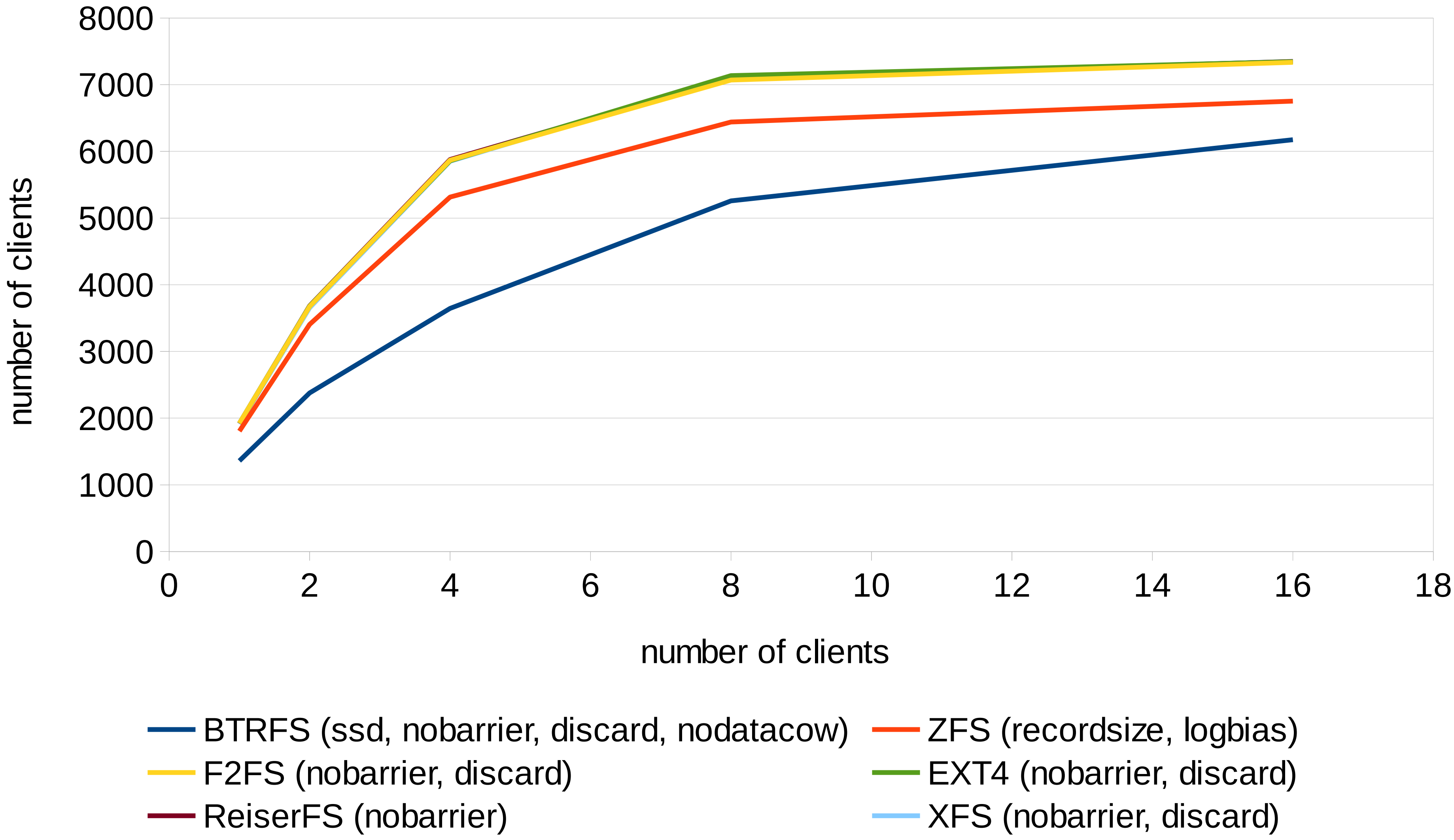
2ndQuadrant
Professional PostgreSQL

pgbench read-write

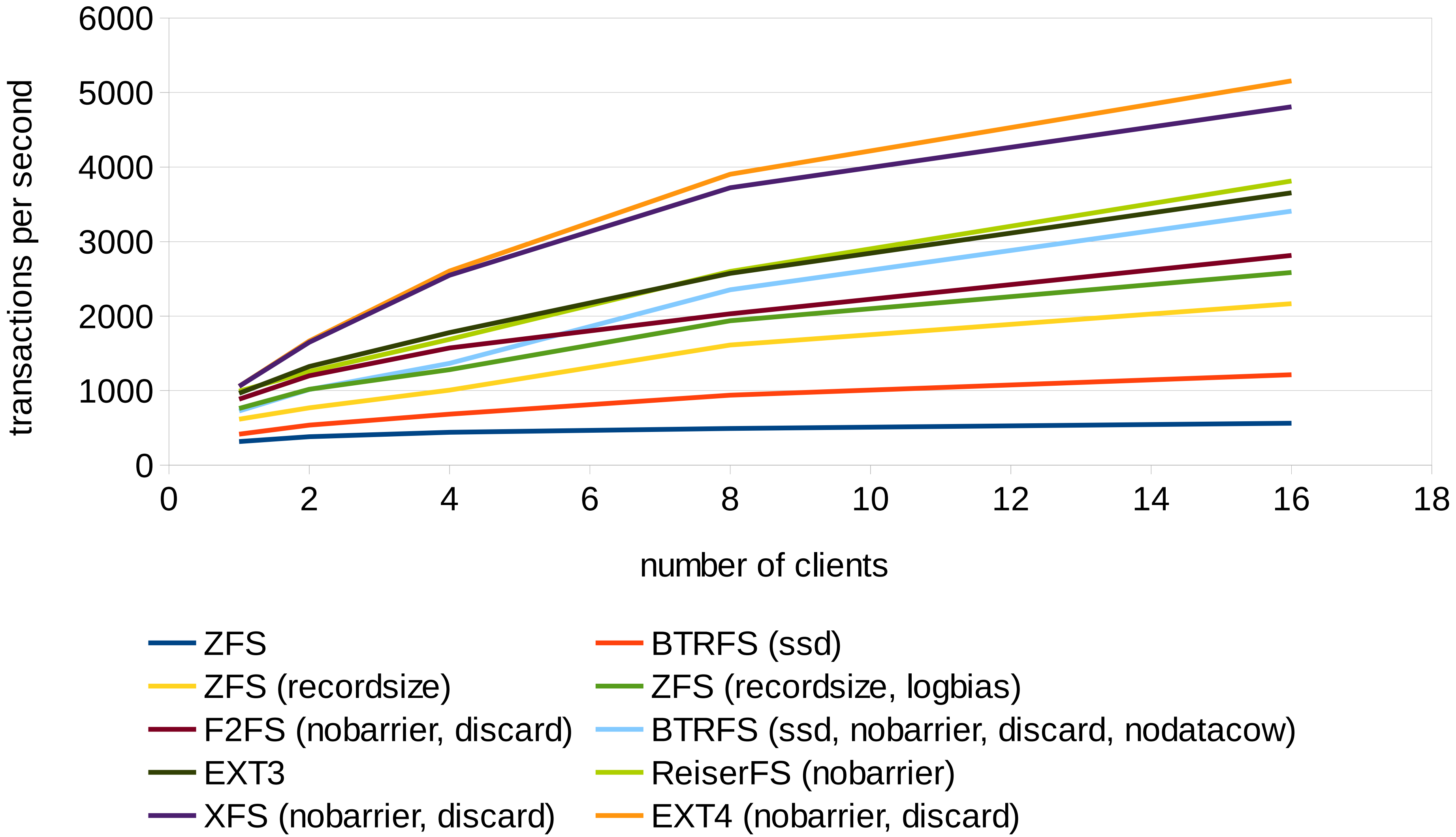


2ndQuadrant
Professional PostgreSQL

pgbench / small read-write

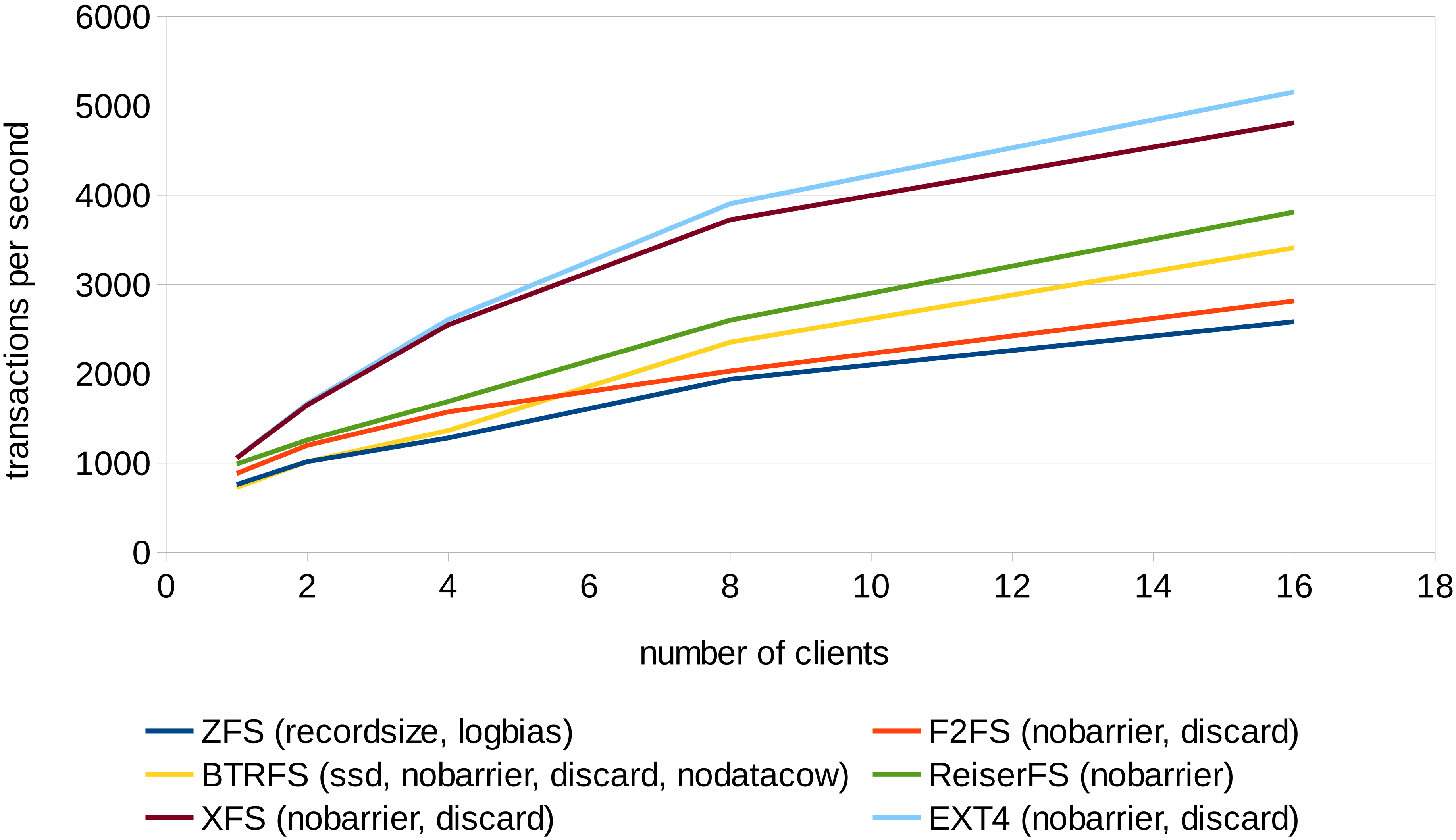


pgbench / large read-write



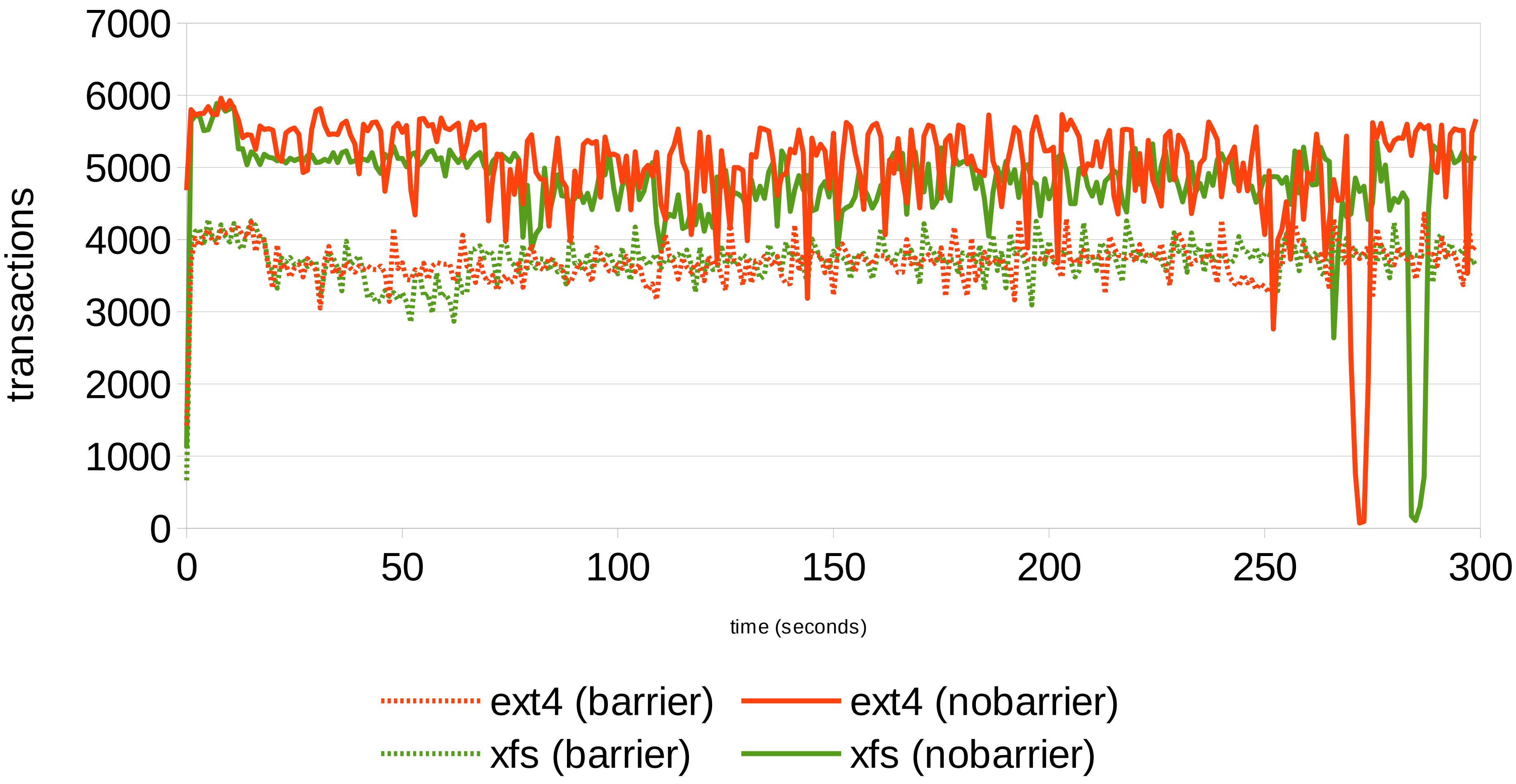
2ndQuadrant
Professional PostgreSQL

pgbench / large read-write

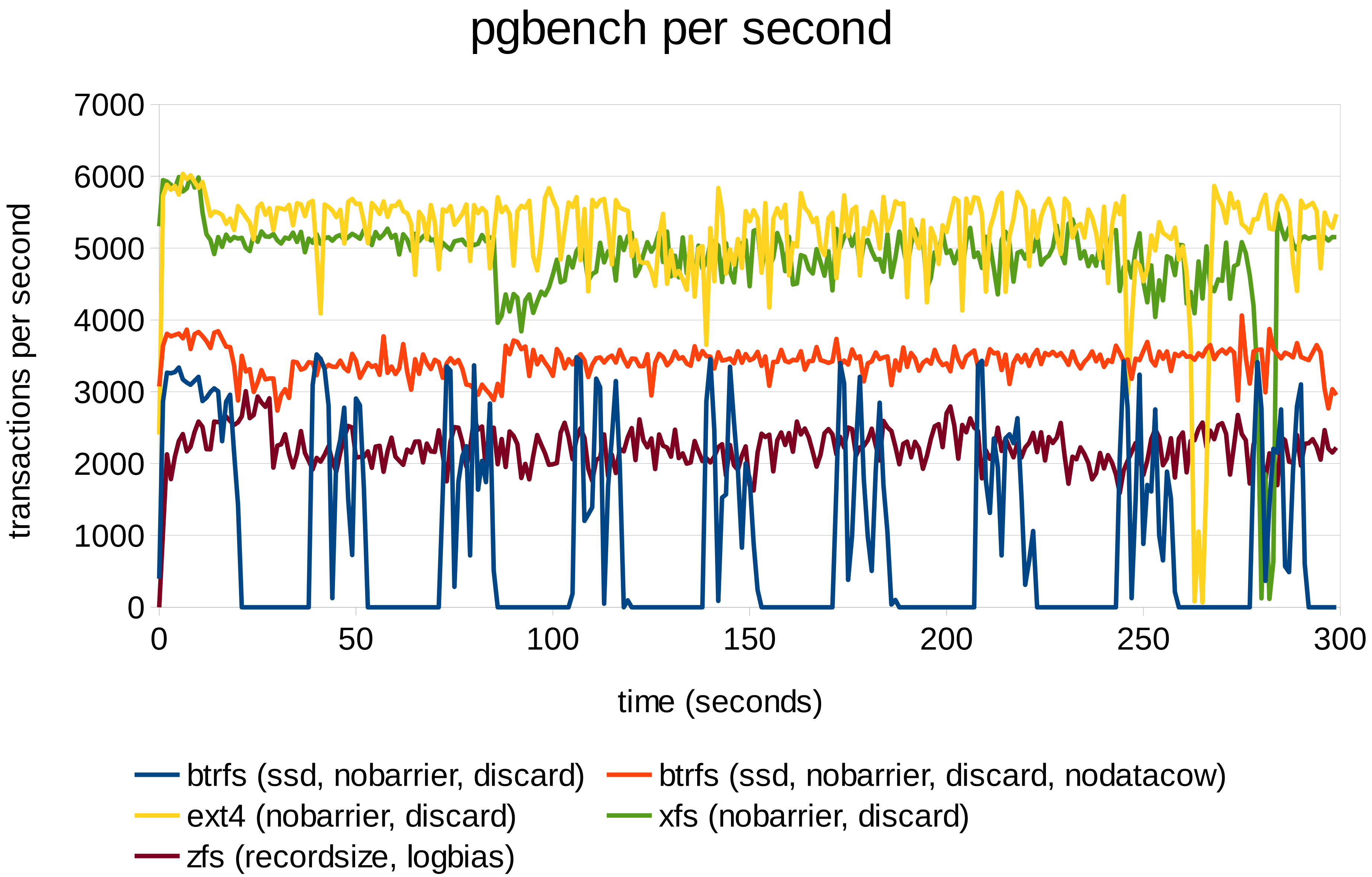


Write barriers

ext4 and xfs (defaults, noatime)



variability



2ndQuadrant
Professional PostgreSQL

EXT / XFS

- mostly the same behavior
  - EXT4 – higher throughput but more jitter
  - XFS – lower throughput, less jitter
- significant impact of “write barriers”
  - reliable drives / RAID controller needed
- small impact of TRIM
  - depends on SSD model (over-provisioning etc.)
  - depends on how “full” the SSD is

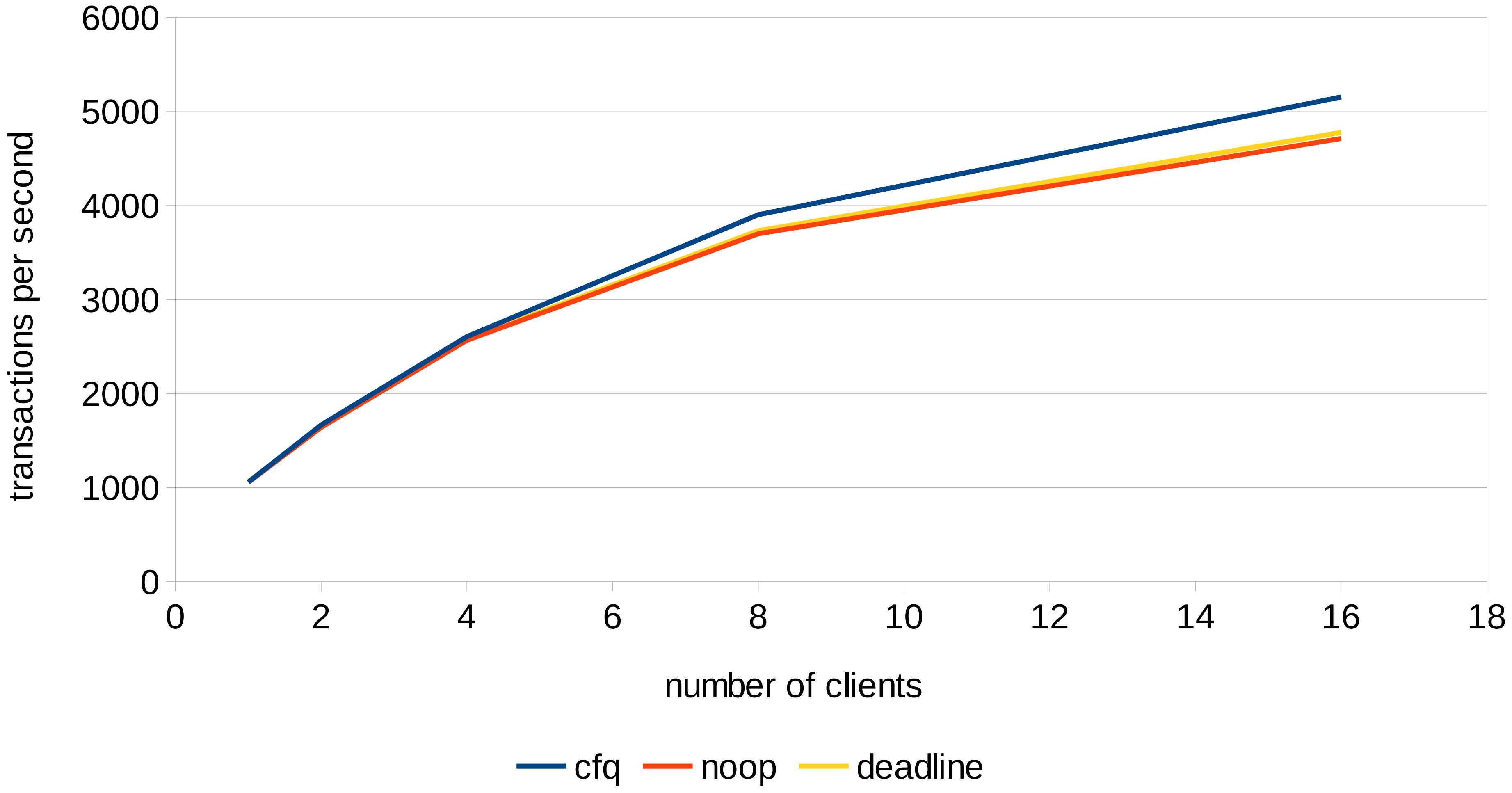
BTRFS, ZFS

- significant price for CoW (but features)
  - about 50% performance reduction in writes
- BTRFS
  - all the problems I had while testing were with BTRFS
  - good: no data corruption bugs
  - bad: rather unstable and inconsistent behavior
- ZFS
  - a bit alien in Linux world
  - much more mature than BTRFS, nice behavior
  - the ZFSonLinux still heavily developed

Questions?

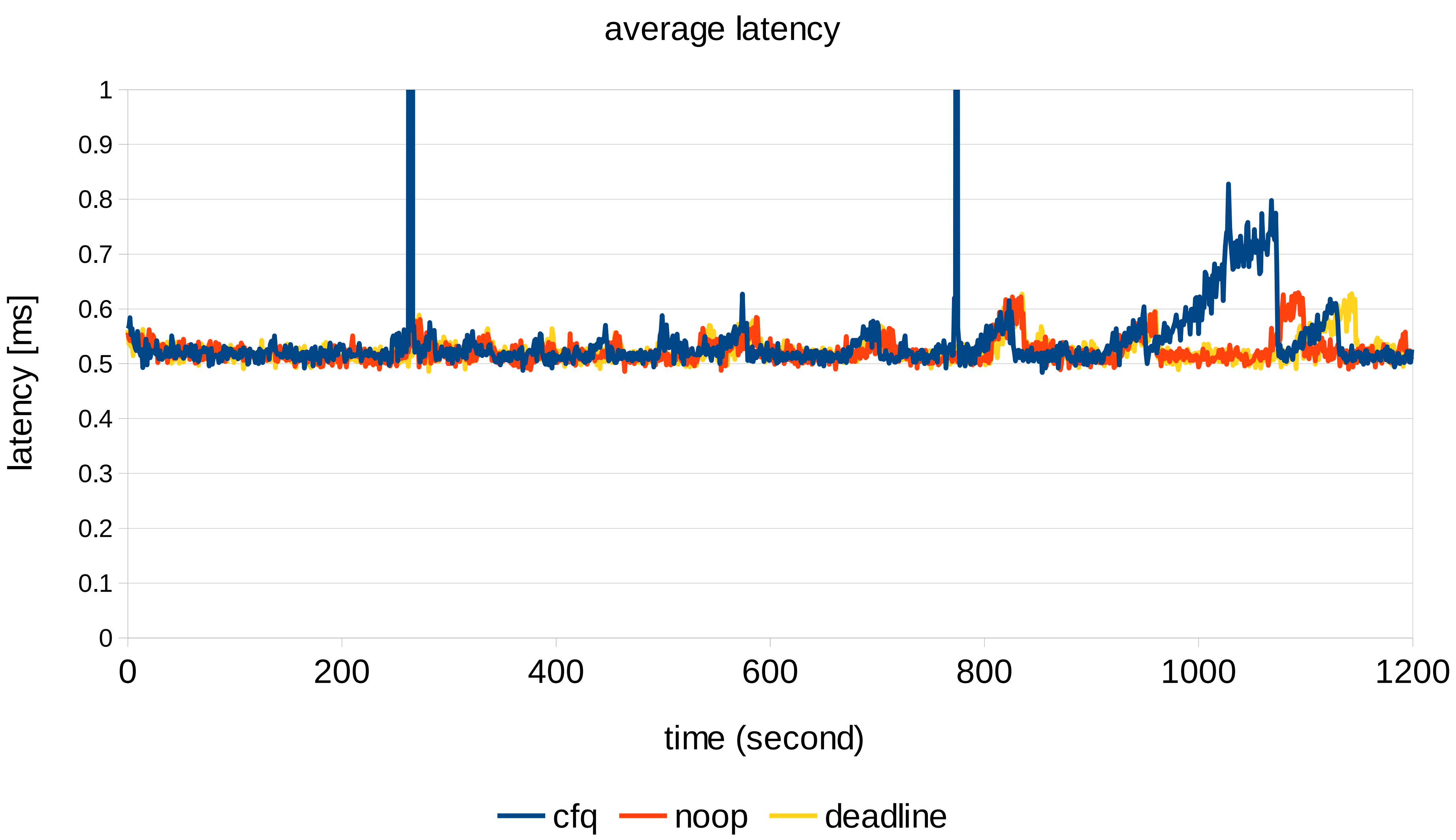
pgbench / large read-write

ext4 (noatime, discard, nobarrier)



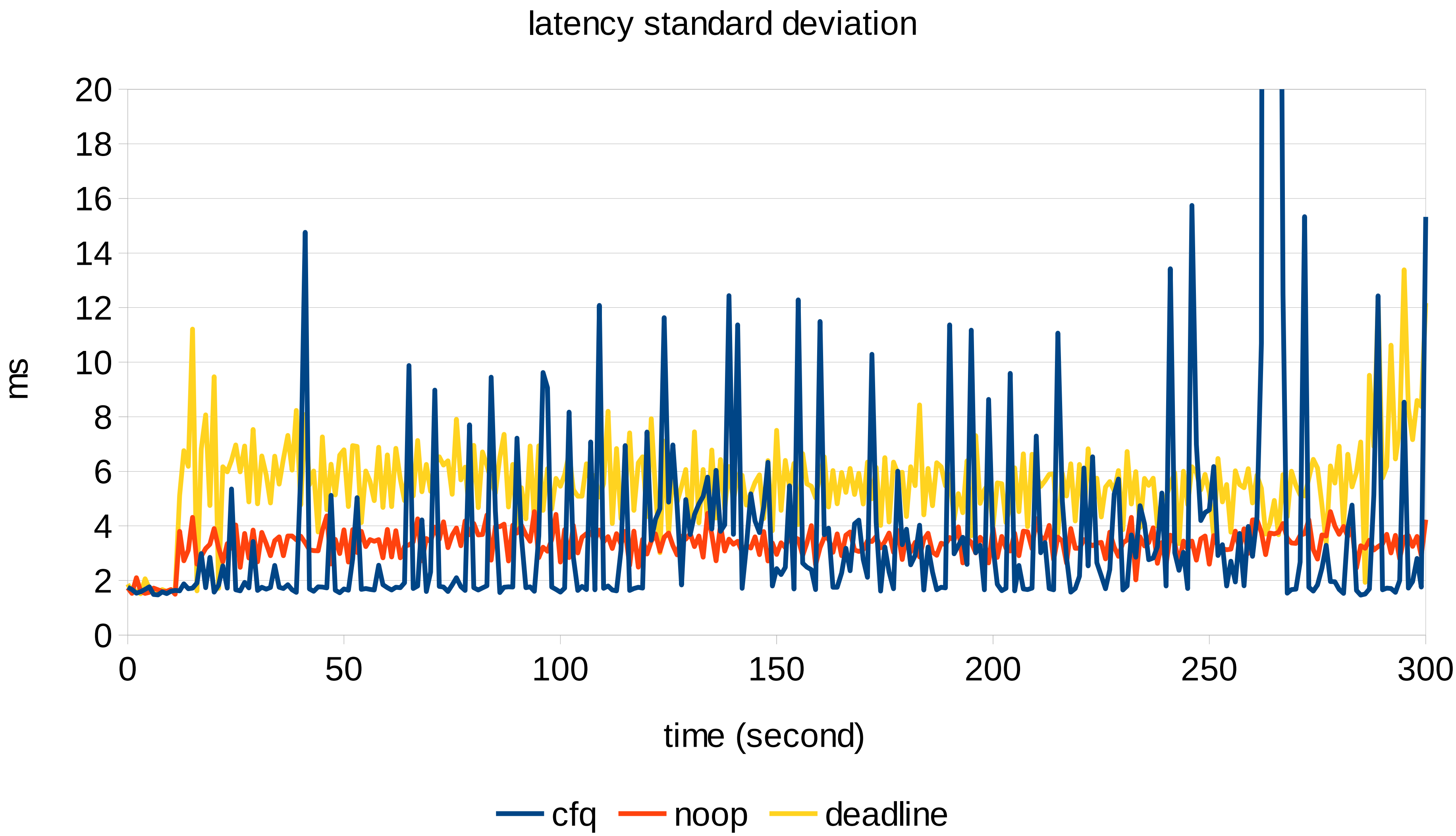
2ndQuadrant
Professional PostgreSQL

pgbench / large read-write (16 clients)



2ndQuadrant
Professional PostgreSQL

pgbench / large read-write (16 clients)



2ndQuadrant

Professional PostgreSQL

BTRFS, ZFS

```
Tasks: 215 total,   2 running, 213 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us, 12.6%sy,  0.0%ni, 87.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  16432096k total, 16154512k used,   277584k free,     9712k buffers
Swap:  2047996k total,    22228k used,  2025768k free, 15233824k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
24402 root      20   0     0    0    0 R 99.7  0.0   2:28.09 kworker/u16:2
24051 root      20   0     0    0    0 S  0.3  0.0   0:02.91 kworker/5:0
    1 root      20   0 19416  608  508 S  0.0  0.0   0:01.02 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:09.10 kthreadd
    ...

Samples: 59K of event 'cpu-clock', Event count (approx.): 10269077465
Overhead  Shared Object        Symbol
  37.47%  [kernel]             [k] btrfs_bitmap_cluster
  30.59%  [kernel]             [k] find_next_zero_bit
  26.74%  [kernel]             [k] find_next_bit
   1.59%  [kernel]             [k] _raw_spin_unlock_irqrestore
   0.41%  [kernel]             [k] rb_next
   0.33%  [kernel]             [k] tick_nohz_idle_
   ...
```

2ndQuadrant
Professional PostgreSQL

BTRFS, ZFS

```
$ df /mnt/ssd-s3700/
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/sda1       97684992 71625072  23391064  76% /mnt/ssd-s3700

$ btrfs filesystem df /mnt/ssd-s3700
Data: total=88.13GB, used=65.82GB
System, DUP: total=8.00MB, used=16.00KB
System: total=4.00MB, used=0.00
Metadata, DUP: total=2.50GB, used=2.00GB    <= full (0.5GB for btrfs)
Metadata: total=8.00MB, used=0.00
: total=364.00MB, used=0.00

$ btrfs balance start -dusage=10 /mnt/ssd-s3700
```

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

2ndQuadrant Professional PostgreSQL

EXT3/4, XFS

- Linux Filesystems: Where did they come from?
  (Dave Chinner @ linux.conf.au 2014)
  https://www.youtube.com/watch?v=SMcVdZk7wV8
- Ted Ts'o on the ext4 Filesystem
  (Ted Ts'o, NYLUG, 2013)
  https://www.youtube.com/watch?v=2mYDFr5T4tY
- XFS: There and Back … and There Again?
  (Dave Chinner @ Vault 2015)
  https://lwn.net/Articles/638546/
- XFS: Recent and Future Adventures in Filesystem Scalability
  (Dave Chinner, linux.conf.au 2012)
  https://www.youtube.com/watch?v=FegjLbCnoBw
- XFS: the filesystem of the future?
  (Jonathan Corbet, Dave Chinner, LWN, 2012)
  http://lwn.net/Articles/476263/