PostgreSQL na EXT3/4, XFS, BTRFS a ZFS

FOSDEM PgDay 2016, 28.1.2016, Brussels

Tomáš Vondra
tomas.vondra@2ndquadrant.com
http://blog.pgaddict.com

2ndQuadrant
Professional PostgreSQL

not a filesystem engineer

database engineer

Which file system should we use for PostgreSQL on production systems?

According to our benchmarks from 2003,

the best file system is ...

What does it actually means when a file system is “stable” and “production ready”?

1) reliability

2) consistent performance

3) management & monitoring

DISCLAIMER

I'm not a dedicated fan (or enemy) of any of the file systems discussed in the talk.

SSD

File systems

EXT3, EXT4, XFS, ...

- EXT3/4, XFS, … (and others)
  - traditional design from 90., with journaling and such
  - similar goals / concepts / implementatins
  - continuous improvements
  - mature, reliable, proven by time and production deployments
- basic history
  - 2001 - EXT3
  - 2002 - XFS (1994 - SGI Irix 5.3, 2000 GPL, 2002 Linux)
  - 2008 - EXT4

EXT3, EXT4, XFS, ...

- evolution, not revolution
  - new features (e.g. TRIM, write barriers, ...)
  - scalability improvements (metadata, ...)
  - bug fixes
- conceived at the times of rotational storage
  - mostly work on SSD drives
  - stop-gap for future storage types (NVRAM, ...)
- mostly no support for
  - volume management, multiple drives, snapshots
  - addressed by LVM and/or RAID (hw/sw) – sometimes issues

BTRFS, ZFS

- basic idea
  - integrate all the layers (LVM + dm + ...)
  - designed for consumer-level hardware (expect failures)
  - designed for large data volumes
- that will (hopefully) give us ...
  - flexible management
  - built-in snapshostting
  - compression, deduplication
  - checksums

BTRFS, ZFS

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

Generic “mount options”

Generic “mount options”

- TRIM (discard)
  - enables TRIM commands (sent from kernel to SSD)
  - impacts internal cleanup (block erasure) / wear leveling
  - not entirely necessary, but may help SSD with “garbage collection”
- write barriers
  - prevents controller from reordering writes (e.g. journal x data)
  - ensures consistency of file system, does not prevent data loss
  - write cache + battery => write barriers may be disabled (really?)
- SSD alignment

Specific “mount options”

BTRFS

- nodatacow
  - disables “copy on write” (CoW), enables when snapshotting
  - also disables checksums (require “full” CoW)
  - probably also eliminates “torn-page resiliency” (full_page_writes=on)
- ssd
  - should enable SSD-related optimizations (but not sure which)
- compress=lzo/zlib
  - speculative compression

ZFS

- recordsize=8kB
  - standard ZFS page has 128kB (PostgreSQL uses 8kB pages)
  - makes ARC cache inefficient (smaller number of “slots”)
- logbias=throughput [latency]
  - influences access to ZIL
  - prioritizes latency vs. throughput
- zfs_arc_max
  - limits size of ARC cache (50% RAM by default)
  - should be freed automatically, but external module ...

Benchmark

pgbench (TPC-B)

- transactional benchmark (TPC-B) / stress-test
  - many tiny queries (access through PK, ...)
  - mix of different I/O types (read/write, random/sequential)
- two variants
  - read-only (SELECT)
  - read-write (SELECT + INSERT + UPDATE)
- three data volume categories
  - small (~200MB)
  - medium (~50% RAM)
  - large (~200% RAM)

Hardware

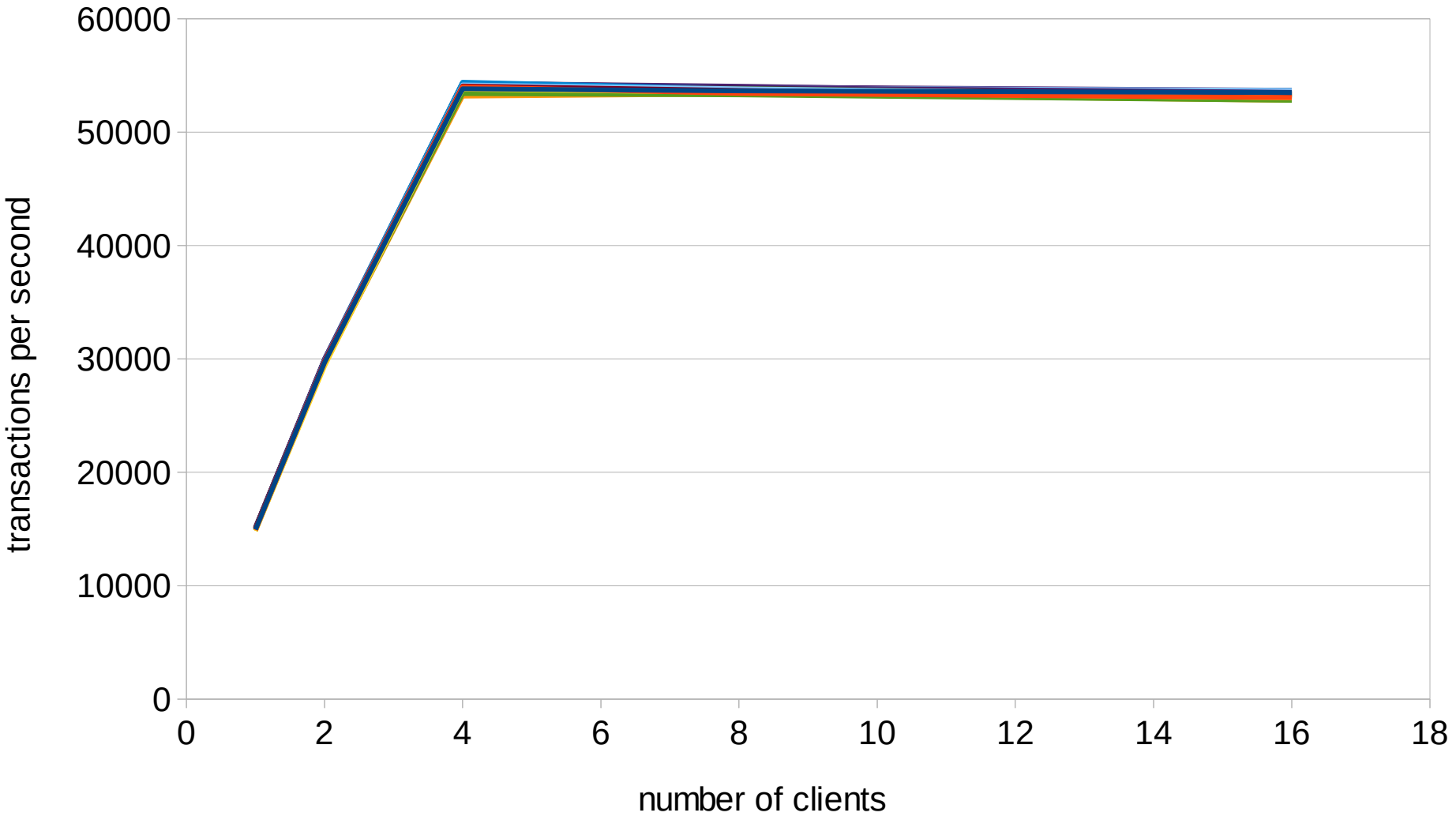
- CPU: Intel i5-2500k
  - 4 cores @ 3.3 GHz (3.7GHz)
  - 6MB cache
  - 2011-2013
- 8GB RAM (DDR3 1333)
- SSD Intel S3700 100GB (SATA3)
- Gentoo + kernel 4.0.4
- PostgreSQL 9.4

Hardware (Cosium)

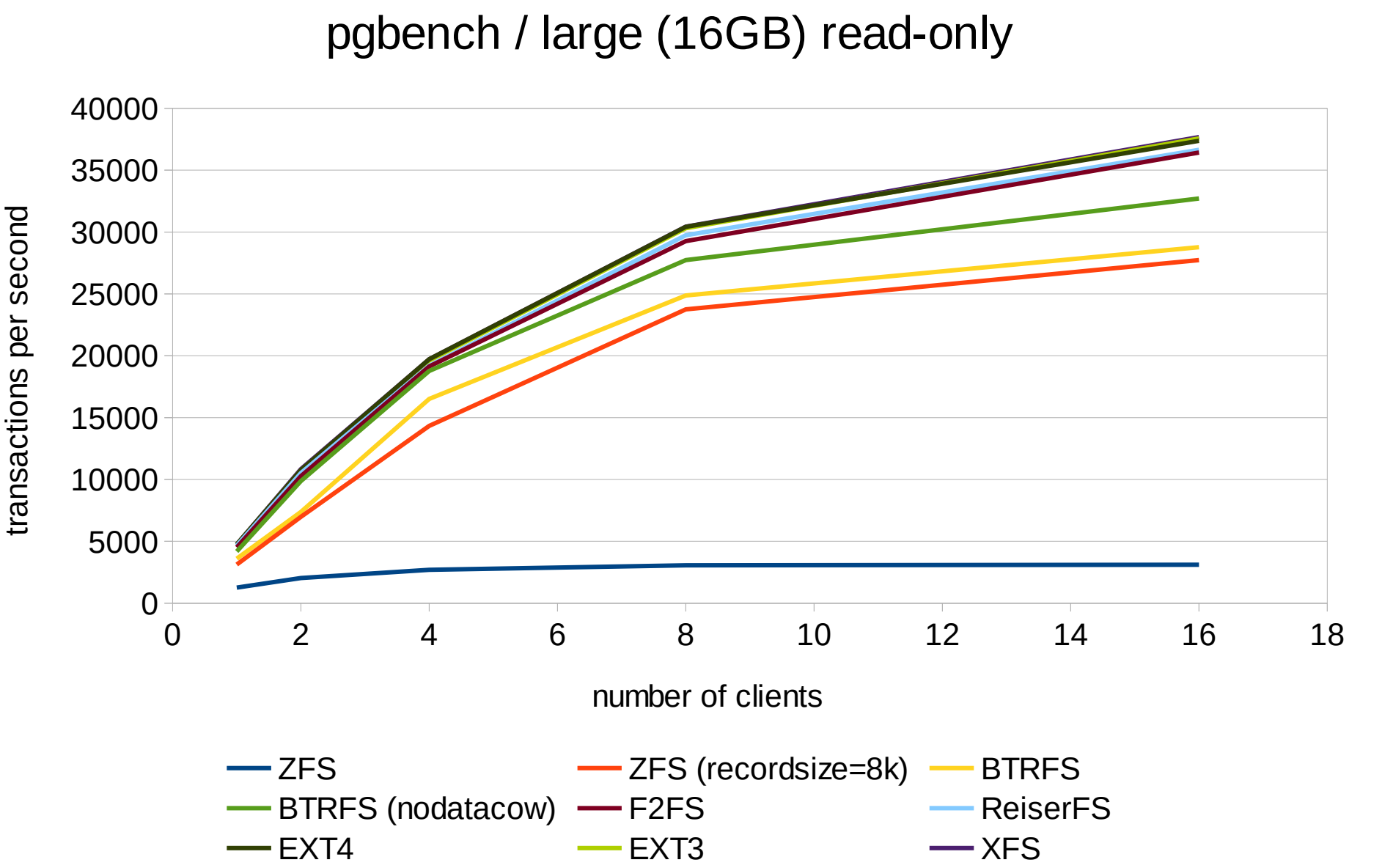
- CPU 2x Intel Xeon E5-2687W v3, 3,1GHz, Cache 25Mo, 9,60GT/s QPI, Turbo, HT, 10C/20T (160W)
- RAM256GB RAM (16x DUAL IN-LINE MEMORY MODULE, 16GB, 2133, 2RX4, 4G, DDR4, R)
- storage A 2x Samsung XS1715 NVME SSD 1.6 TB
- storage B 2x 300GB SAS 10k RPM drive
- storage C 4x 1.2TB SAS 10k RPM drive
- RAID Dell Perc H330 (no write cache)

But that is not representative!

pgbench read-only

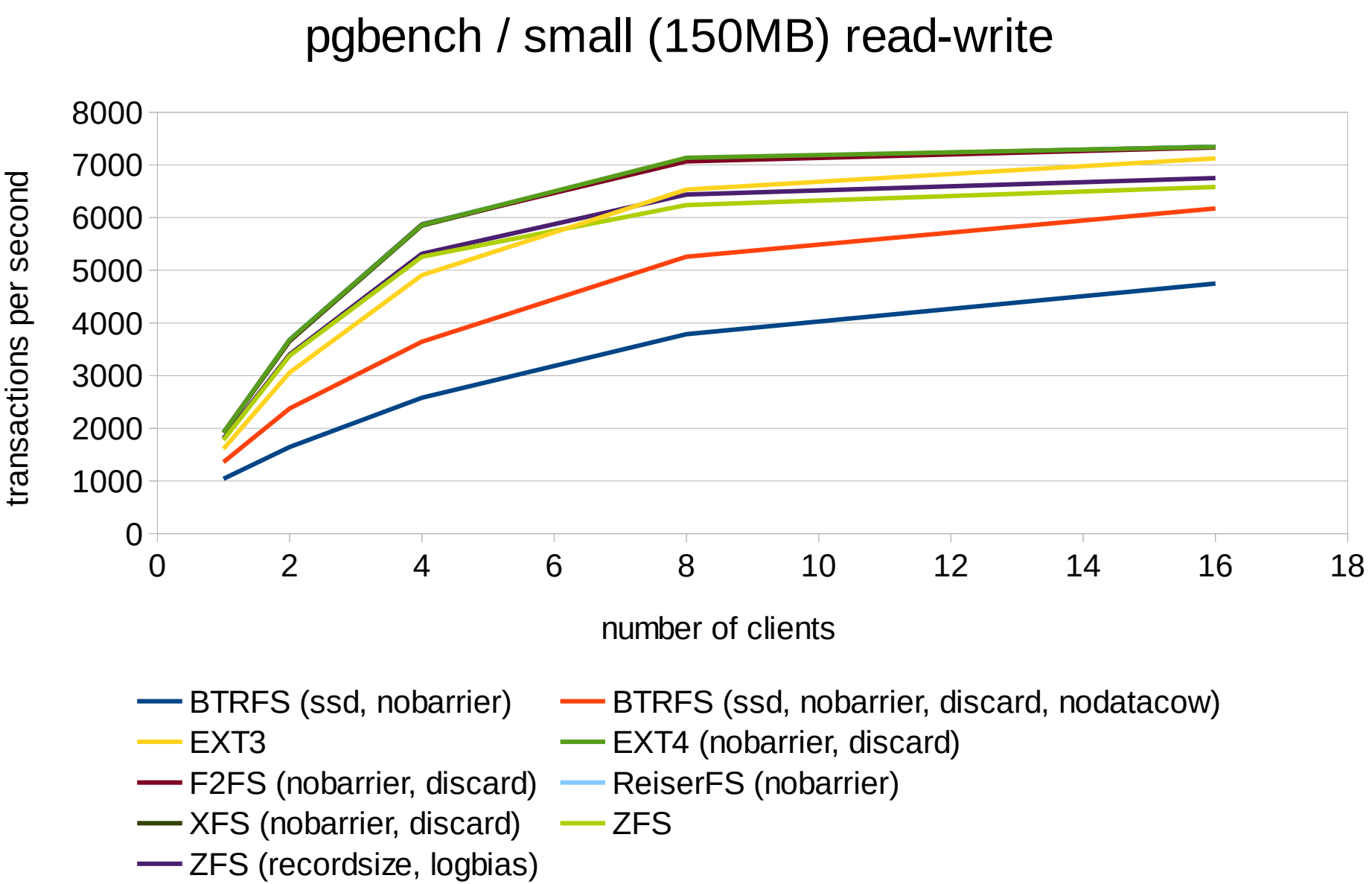


2ndQuadrant
Professional PostgreSQL

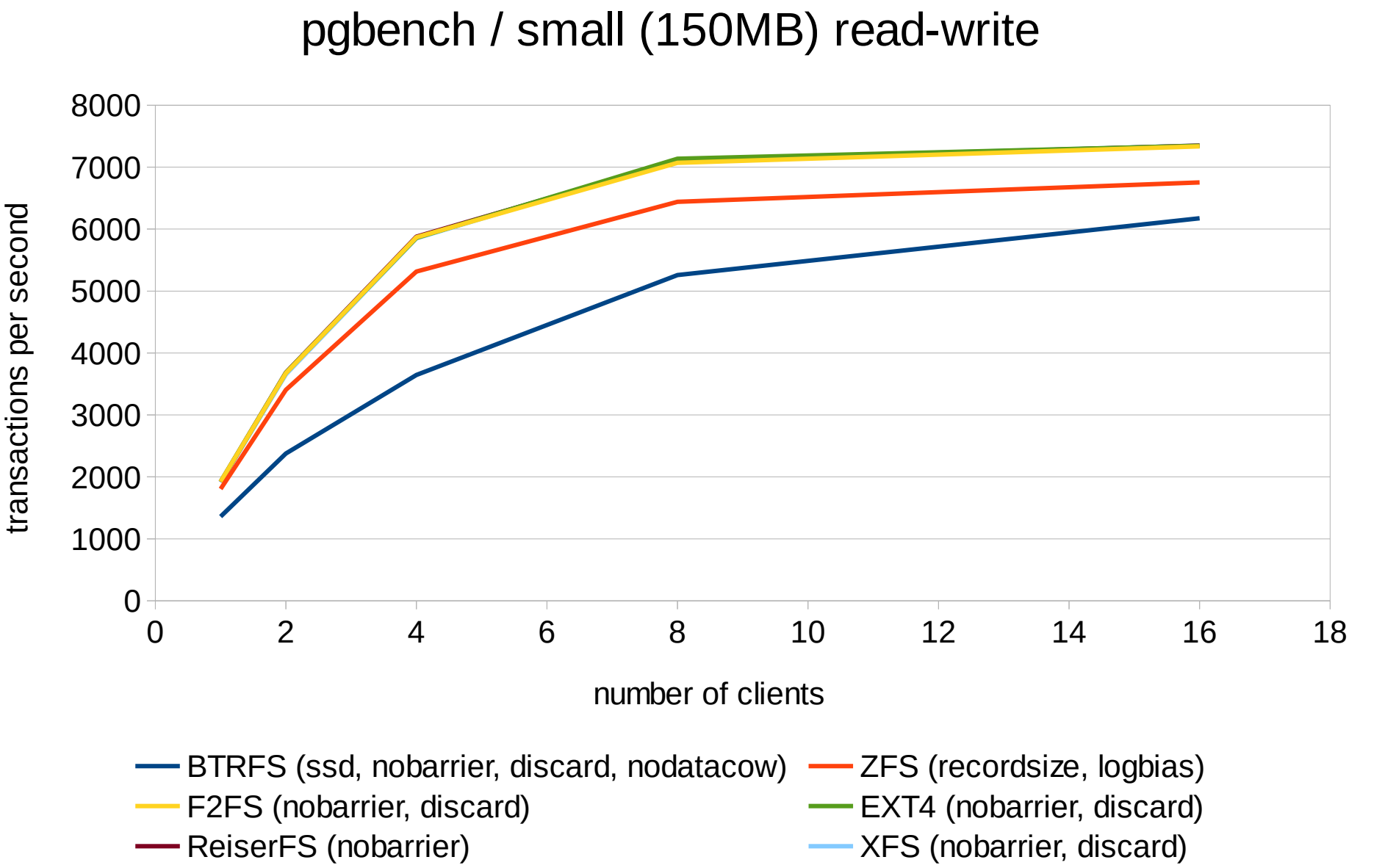


2ndQuadrant
Professional PostgreSQL

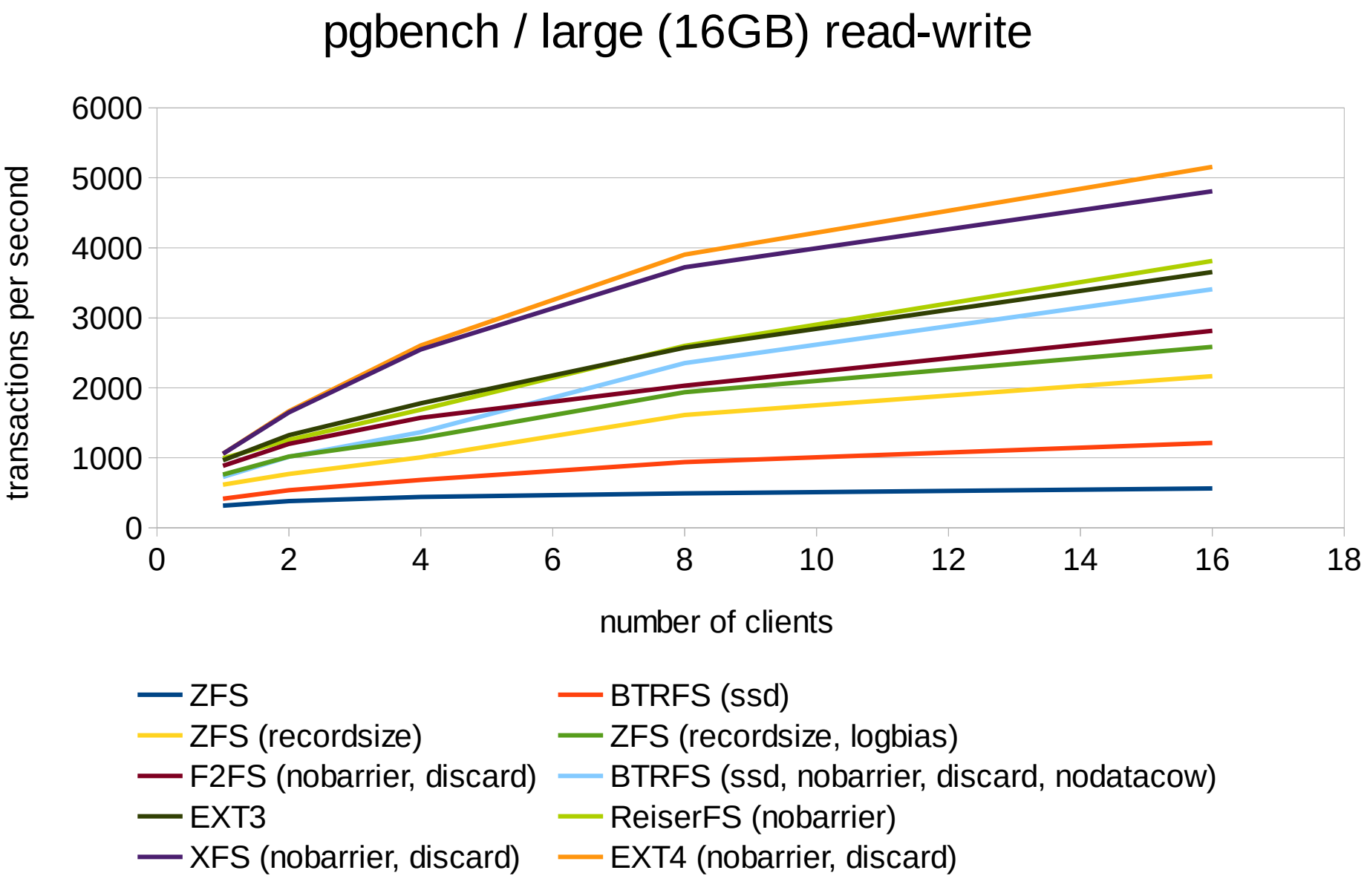
pgbench read-write



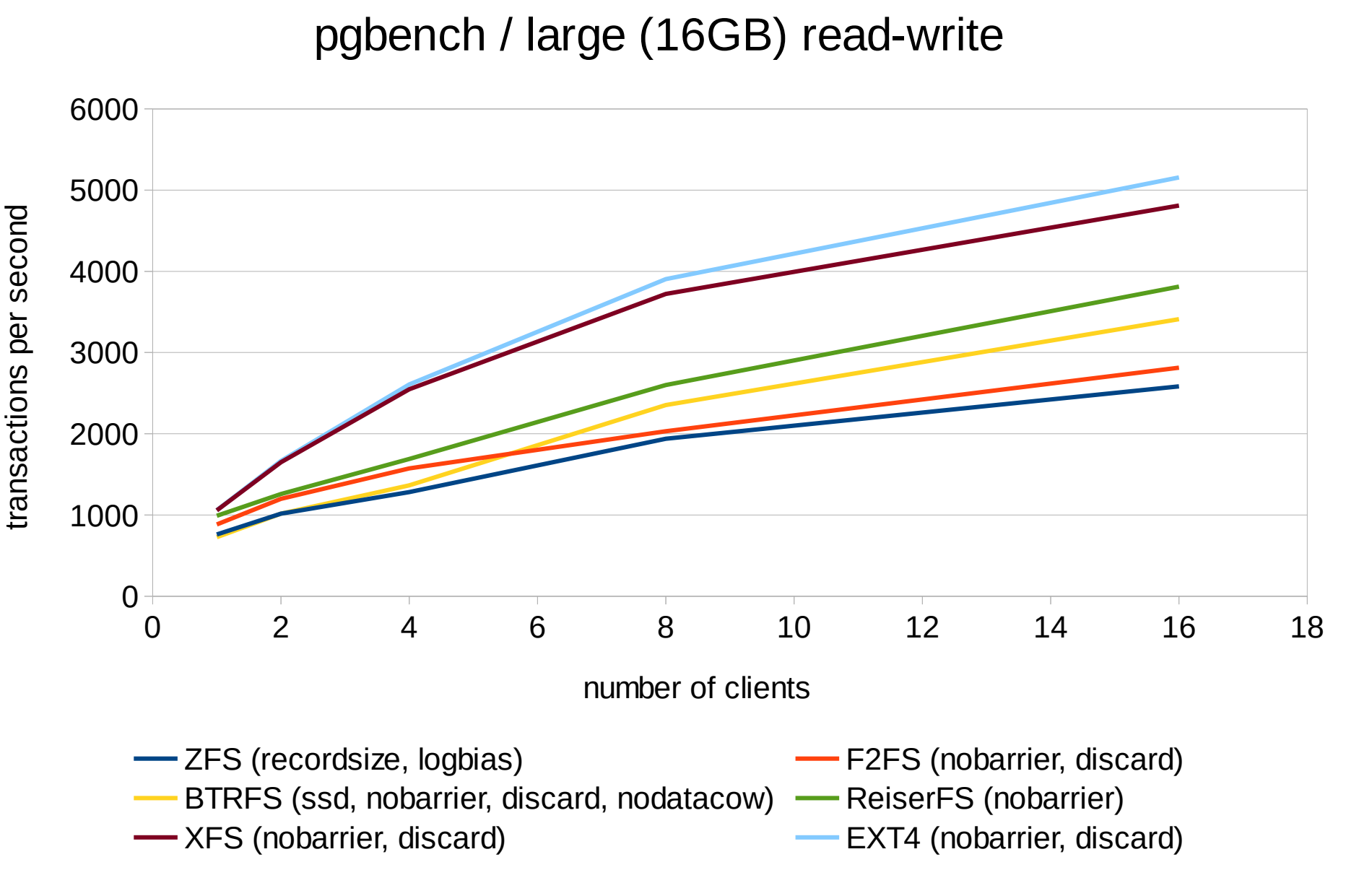
2ndQuadrant
Professional PostgreSQL



2ndQuadrant
Professional PostgreSQL

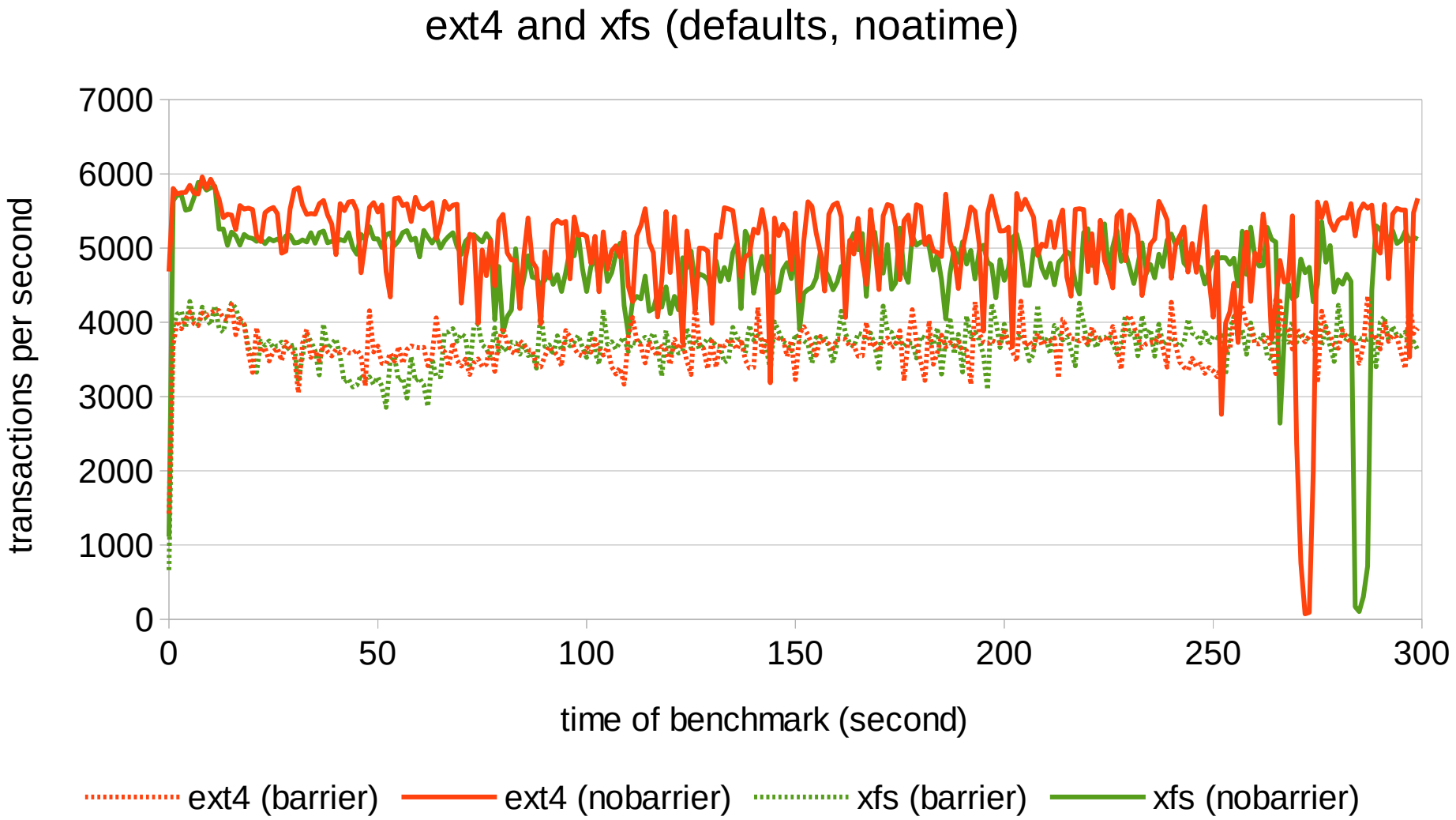


2ndQuadrant
Professional PostgreSQL



2ndQuadrant
Professional PostgreSQL

Write barriers

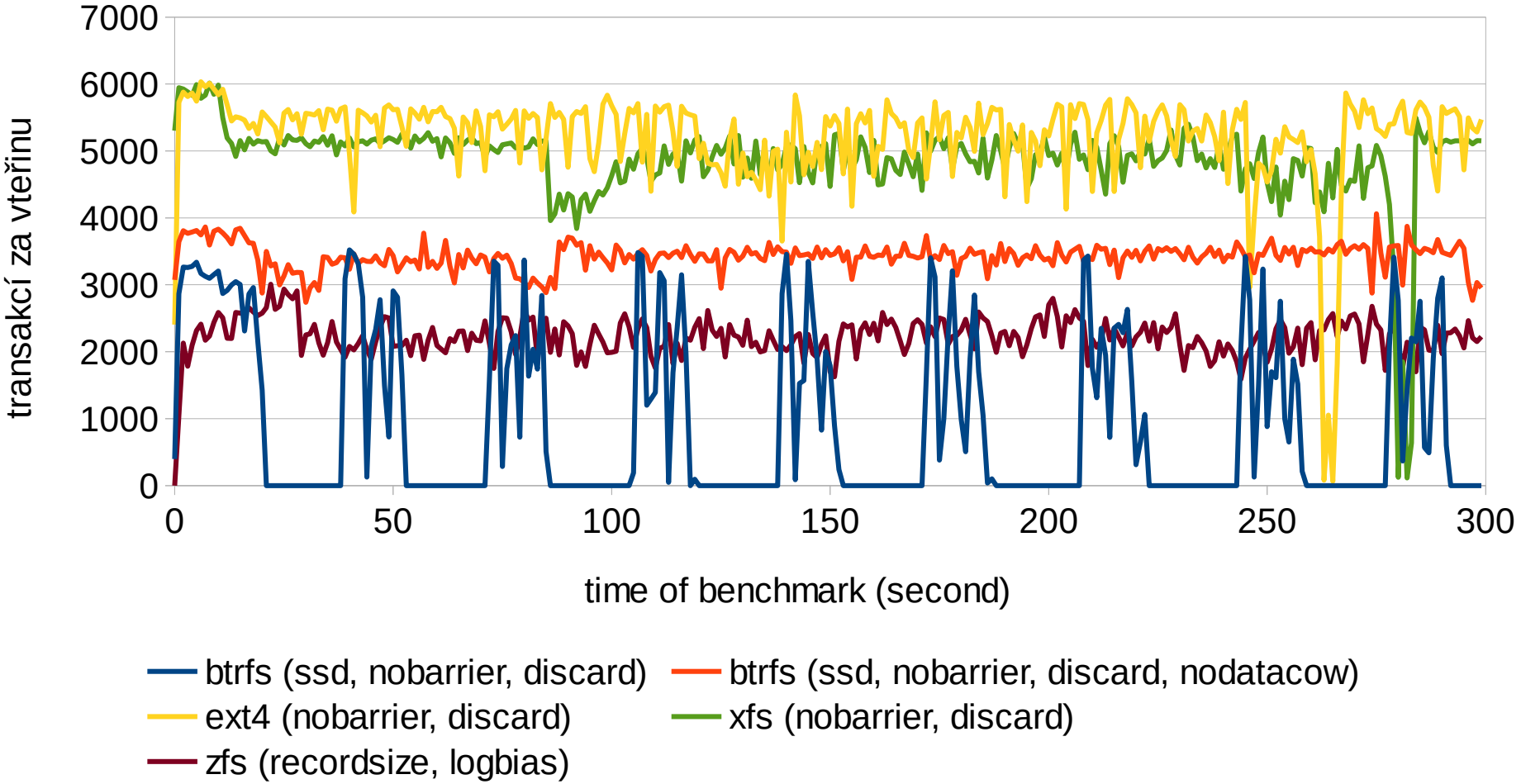


2ndQuadrant
Professional PostgreSQL

Performance variability

pgbench / large (16GB) read-write

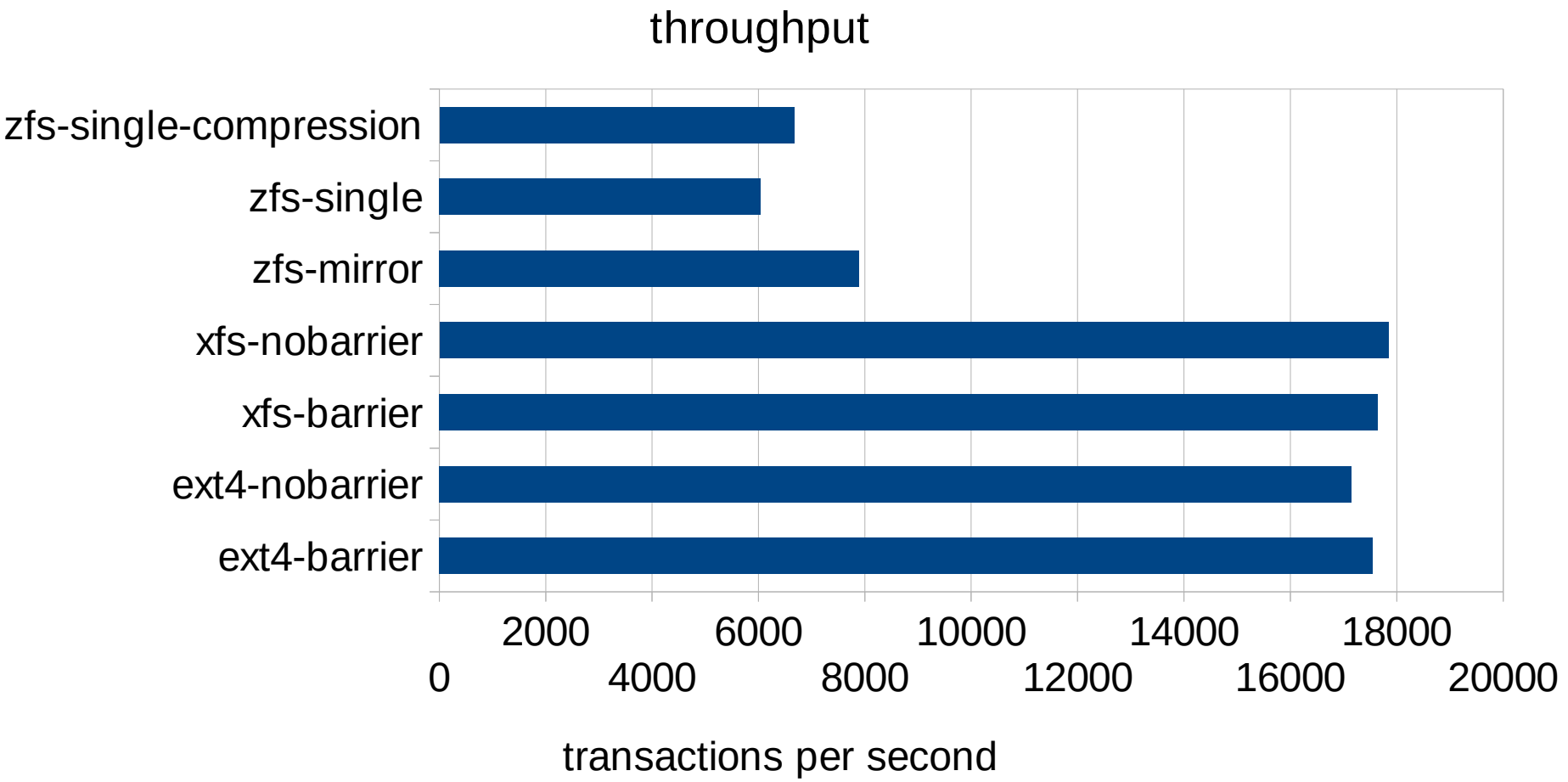
number of transactions per second over time



2ndQuadrant
Professional PostgreSQL

NVME drives

pgbench / large, 60 clients on NVME

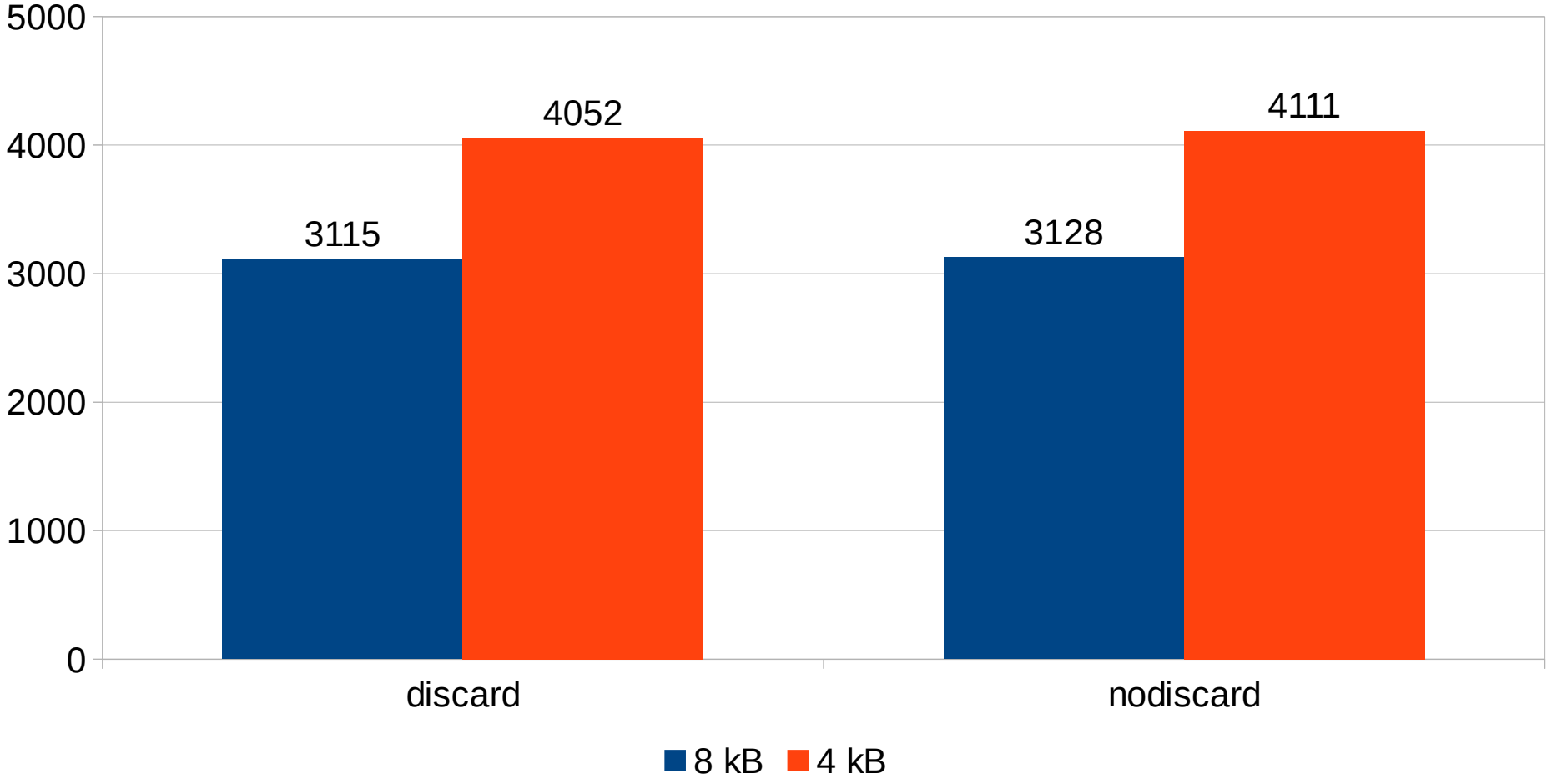


2ndQuadrant
Professional PostgreSQL

4kB vs. 8kB

PostgreSQL se 4kB a 8kB pages

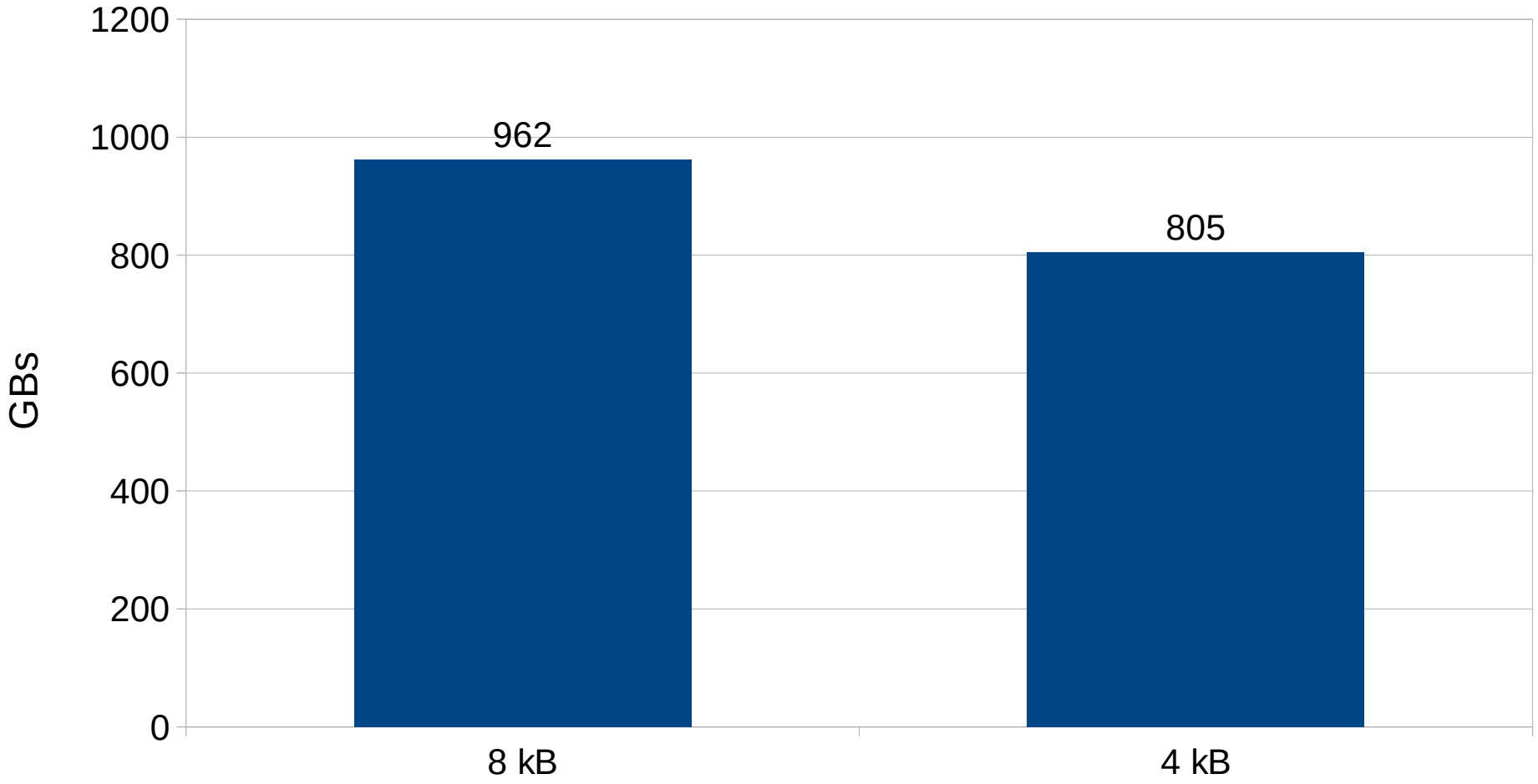
pgbench read-write, 16 clients, scale 5000 (~80GB)



2ndQuadrant
Professional PostgreSQL

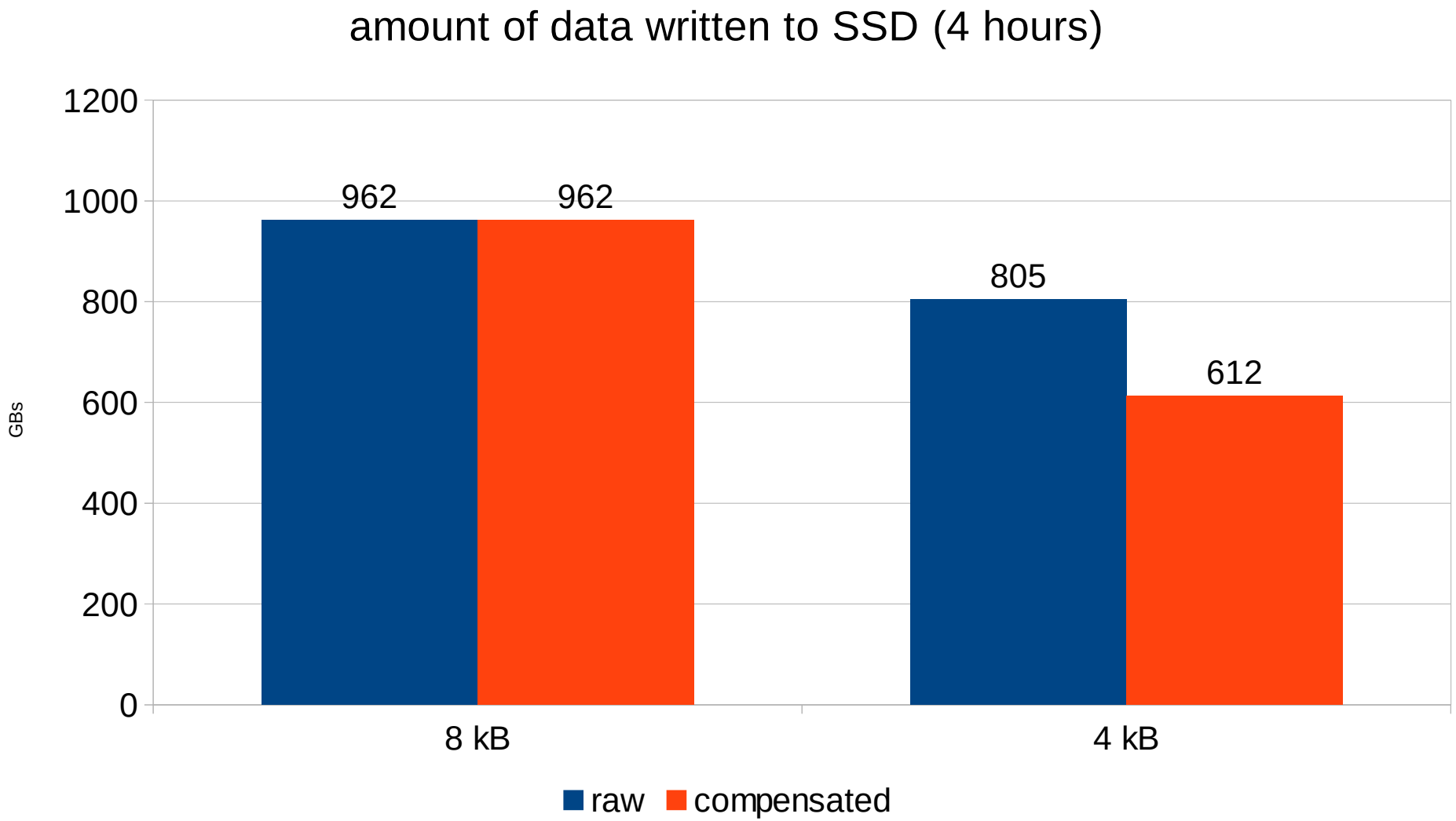
Host_Writes_32MB vs. 4kB/8kB pages

amount of data written to SSD (4 hours)



2ndQuadrant
Professional PostgreSQL

Host_Writes_32MB vs. 4kB/8kB pages



2ndQuadrant
Professional PostgreSQL

EXT / XFS

- similar behavior
  - mostly compromise between throughput and latency
  - EXT4 – higher throughput, more jitter
  - XFS – lower throughput, less jitter
- significant impact of “write barriers”
  - requires reliable drives / RAID controller with BBU
- minimal TRIM impact
  - depends on SSD model (different over-provisioning etc.)
  - depends on how full the SSD is
  - benchmark does not delete (over-writes pages)

BTRFS, ZFS

- significant price for features (based on CoW)
  - about 50% reduction of performance when writing data
- BTRFS
  - most problems I've ran into were na on BTRFS
  - good: no data corruption bugs (but not tested)
  - bad: unstable and inconsistent behavior, lockups
- ZFS
  - alien in the Linux world, separate ARC cache
  - much more mature than BTRFS, nice stable behavior
  - ZFSonLinux actively developed (current 0.6.5, tested 0.6.3)

Conclusion

Conclusion

- if traditional file system is sufficient
  - use EXT4/XFS, depending on your distribution
  - no extreme differences in behavior / performance
  - worth spending some time in tuning
- if you need “advanced” features
  - e.g. snapshotting, multi-device support ...
  - ZFS is good choice (maybe consider FreeBSD)
  - BTRFS (now) definitely not recommended

Questions?

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/