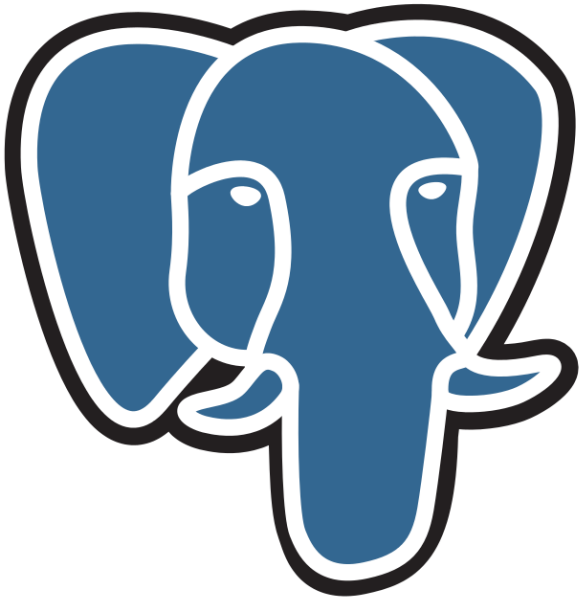


Pohádka o checkpointu

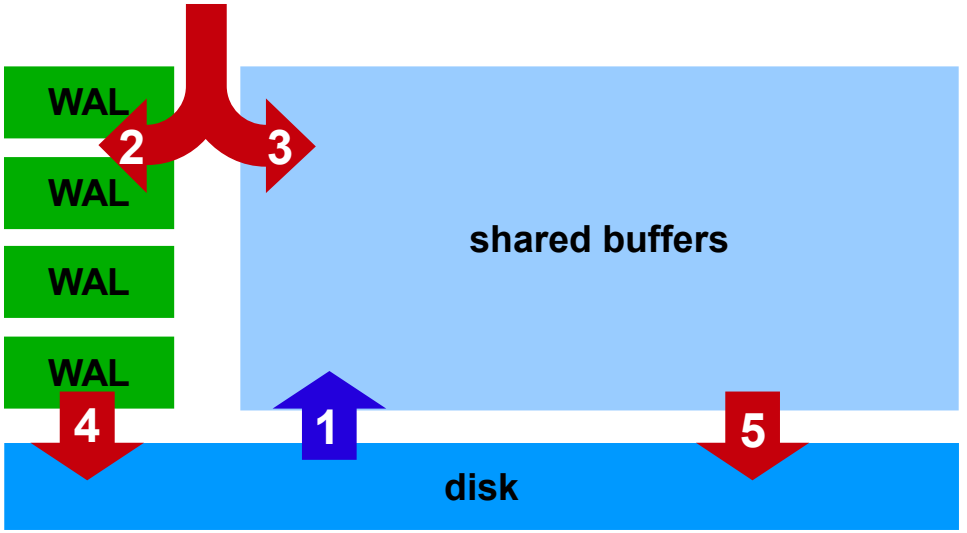
Co se děje při checkpointu, možnosti a ladění.

CSPUG Praha, 22. 11. 2011
Tomáš Vondra, tv@fuzzy.cz

Co je to checkpoint?

- činnost nutná kvůli samostatnému transakčnímu logu (WAL)

1) načtení dat do shared buffers
2) zápis do WAL (disk)
3) úprava shared buffers (RAM)
4) sync dat ve WAL (commit)
5) checkpoint (zápis dat)



- nejčastější problém u databází s netriviálním počtem změn
- ne zcela jednoduchý monitoring a určení příčiny problému
- interakce s operačním systémem (pdflush v Linuxu apod)

Kdy k checkpointu dochází?

- po vyčerpání počtu transakčních logů (xlog)

```
checkpoint_segments = 64
```

- po vypršení časového limitu (timed)

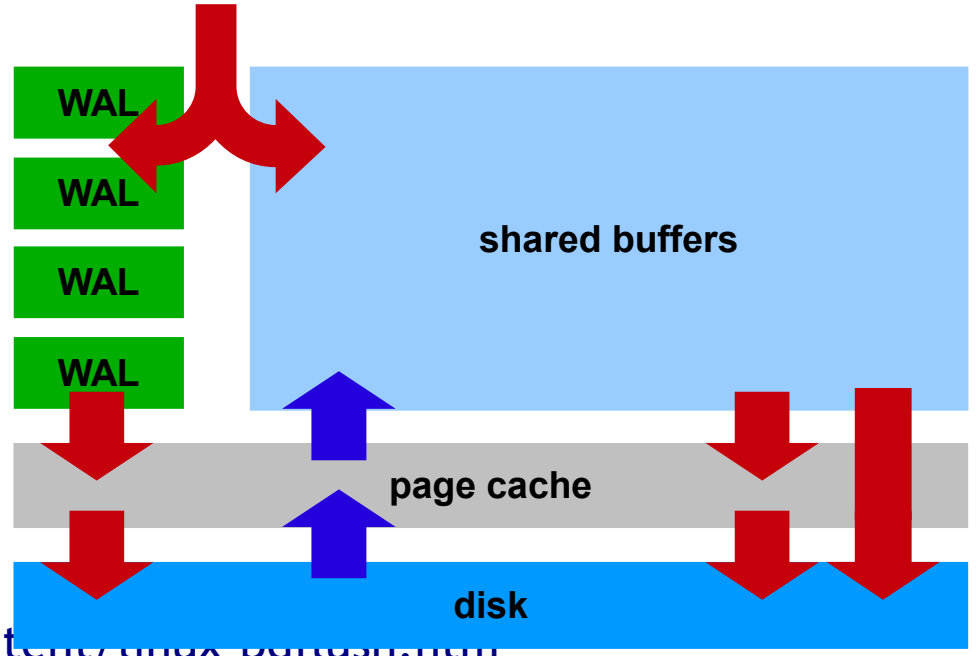
```
checkpoint_timeout = 5m
```

- na vyžádání uživatele

```
CHECKPOINT
```

Interakce s OS (Linux)

- je to trochu složitější kvůli interakci s OS
- operační systémy vesměs mají cache pro disky (page cache)
- výhody cache – opakovaný zápis, řazení zápisů (sekvenční)
- zápis na disk až při fsync



- Linux – pdflush (/proc/sys/vm)
  - dirty_background_ratio
  - dirty_ratio
  - dirty_expire_centisecs

- http://www.westnet.com/~gsmith/content/linux-pdflush.htm

page cache v Linuxu

- **vm.dirty_background_ratio**
  - „dirty“ limit než pdflush začne zapisovat na pozadí
  - bývalo 10%, dnes 5%
- **vm.dirty_ratio**
  - „dirty“ limit než procesy musí zapisovat přímo (bez cache)
  - bývalo 40%, dnes 10%
- **vm.dirty_expire_centisecs**
  - timeout jak dlouho smí být „dirty“ data v cache
  - standardně 30s

interakce s page cache

příklad: server s 16GB paměti (dnes víceméně minimum)

- 4 GB - shared buffers
- 1 GB - OS + různé blbosti
- 11 GB – page cache (maximum)
  - dirty_background_ratio = 5% = 560 MB*
  - dirty_ratio = 10% = 1020 MB*

To by pro slušný řadič / diskové pole neměl být problém, ne?

* počítá se z (MemFree + Cached – Mapped) – viz. /proc/meminfo

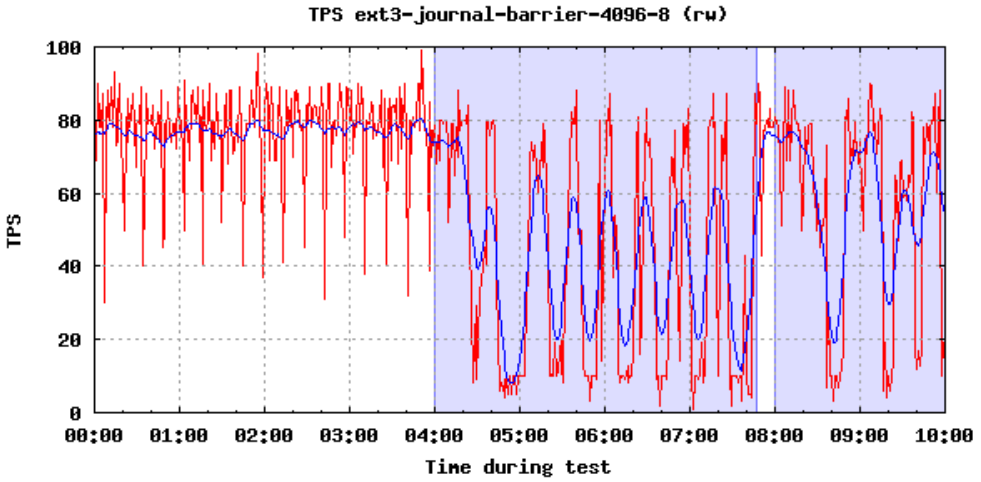
interakce s page cache (2)

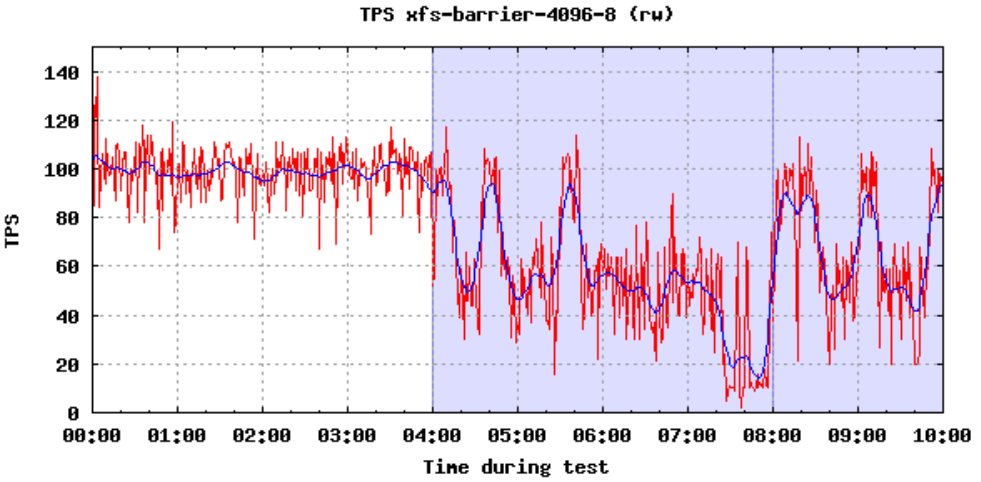
- řekněme že zapíšete 1GB dat …
- ve skutečnosti musíte zapsat > 2GB
  - 1GB data, 1GB pro WAL, overhead (metadata)

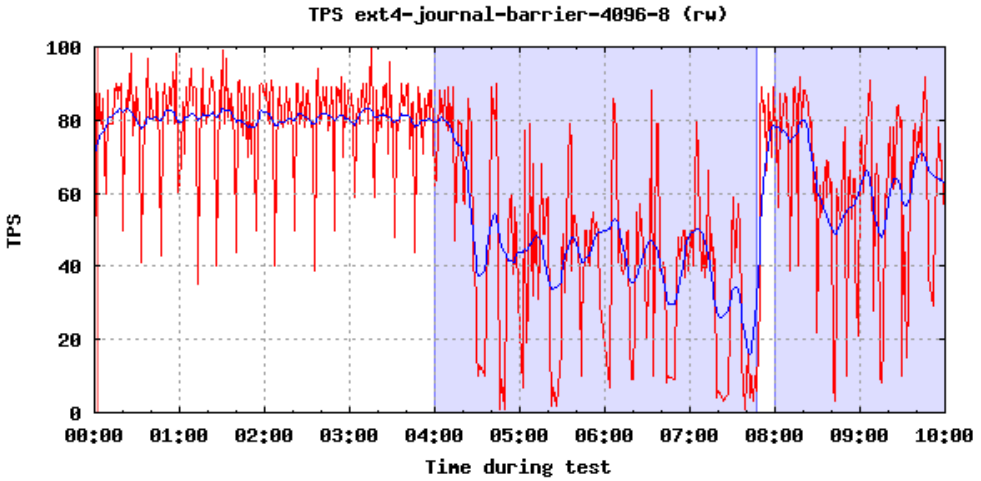
- různá povaha zápisů
  - do WAL sekvenčně
  - datových souborů (CHECKPOINT) mix (náhodně + sekvenčně)

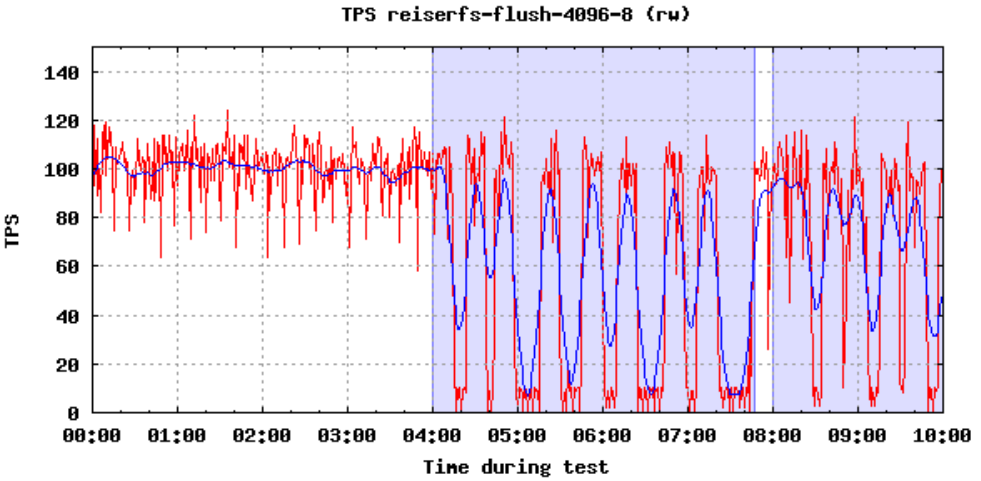
- standardní 7.2k SATA disk zvládne zhruba zápis
  - 60MB/s sekvenčně
  - 0.5MB/s náhodně

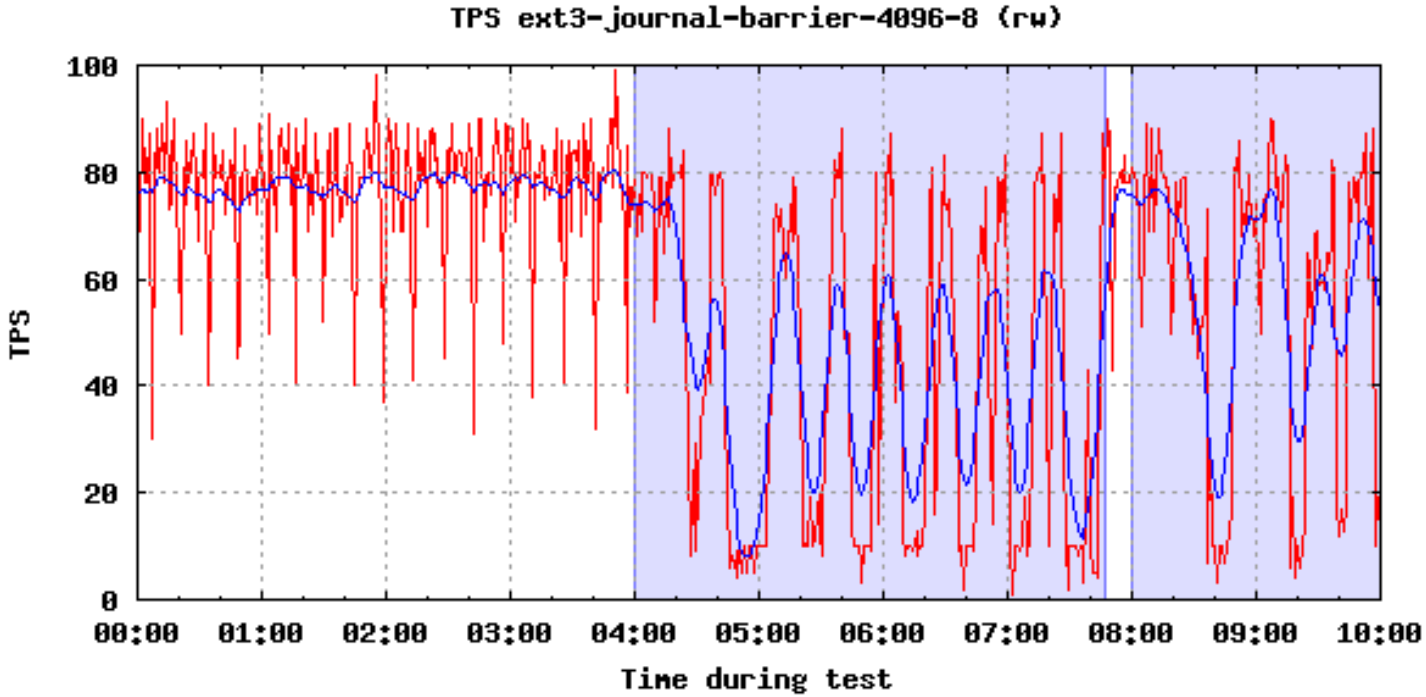
dopad checkpointu (tps)

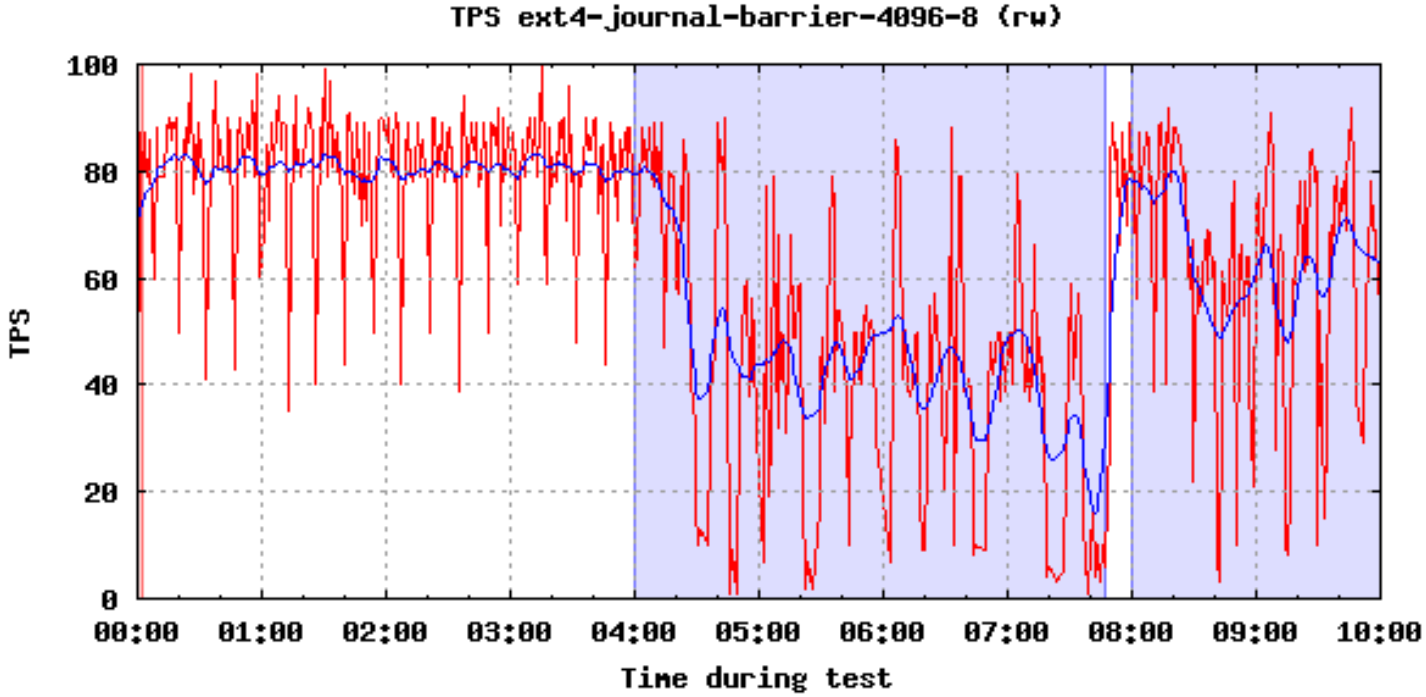


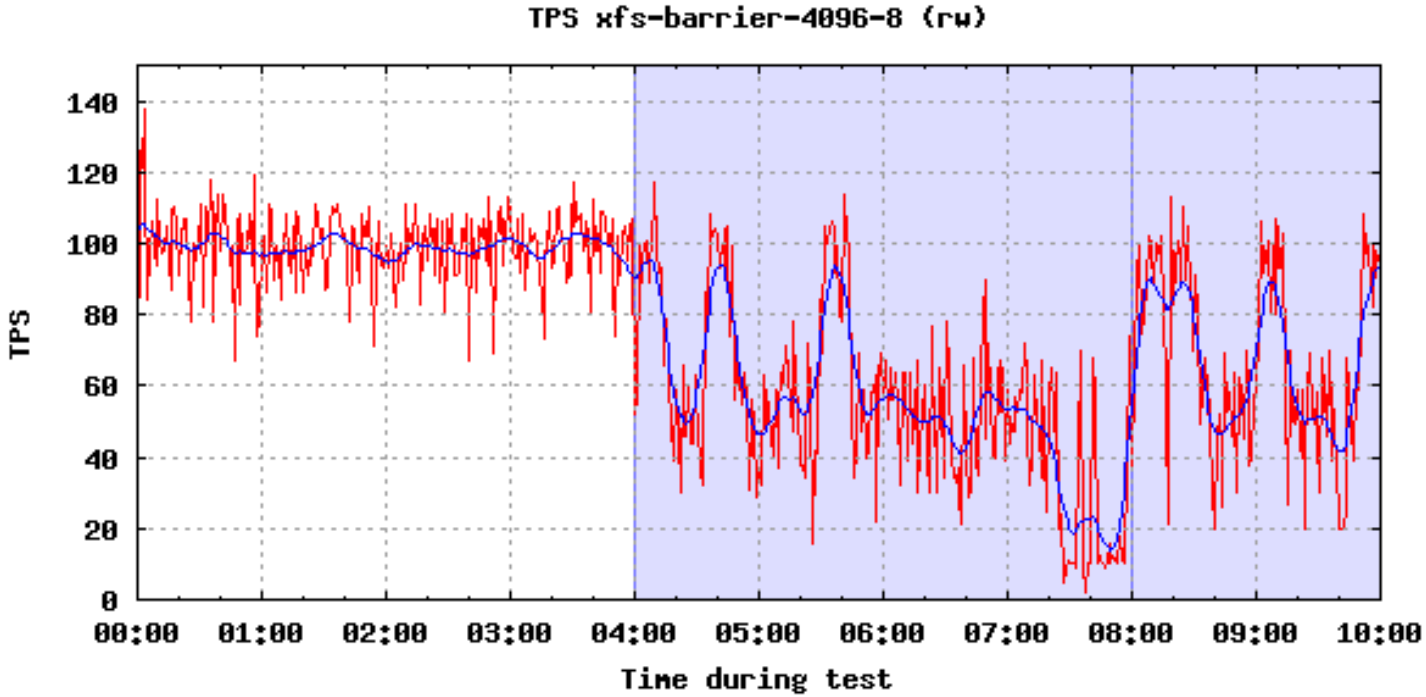


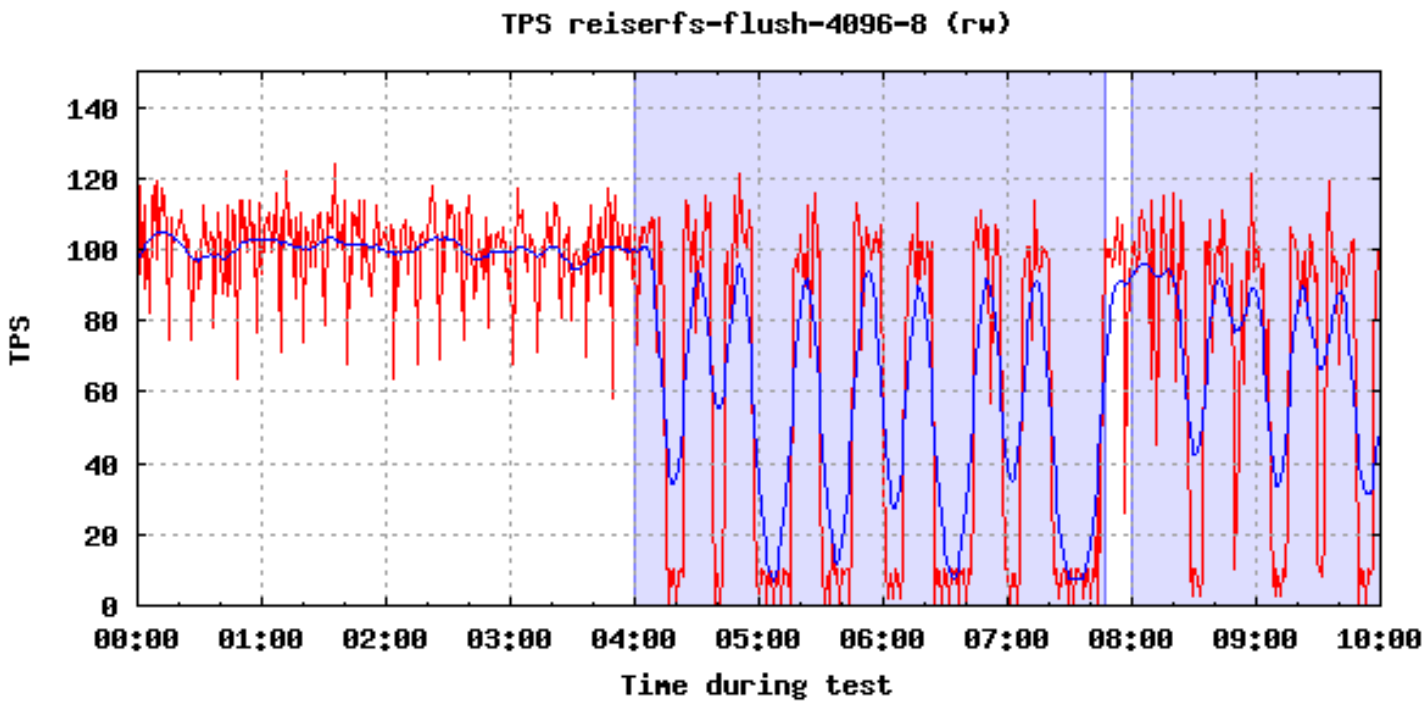




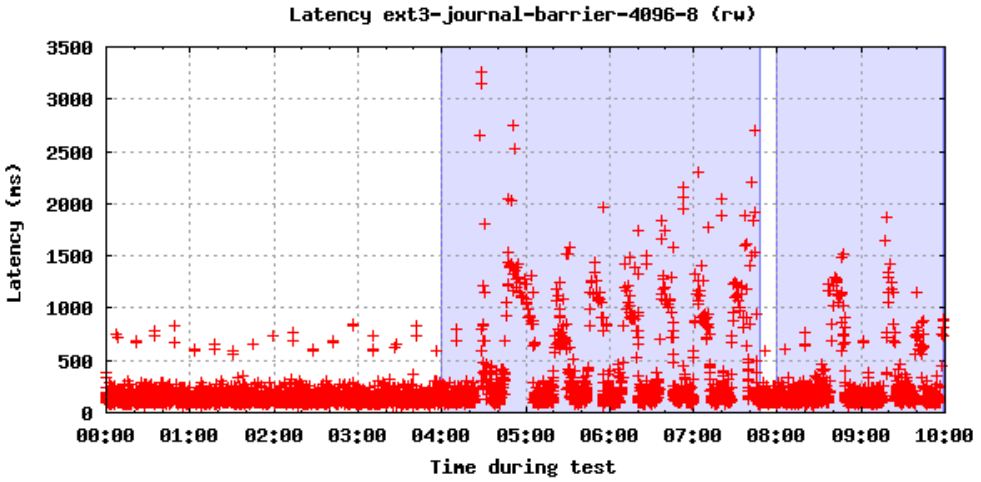

TPS ext3-journal-barrier-4096-8 (rw)
TPS
100
80
60
40
20
0
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
Time during test

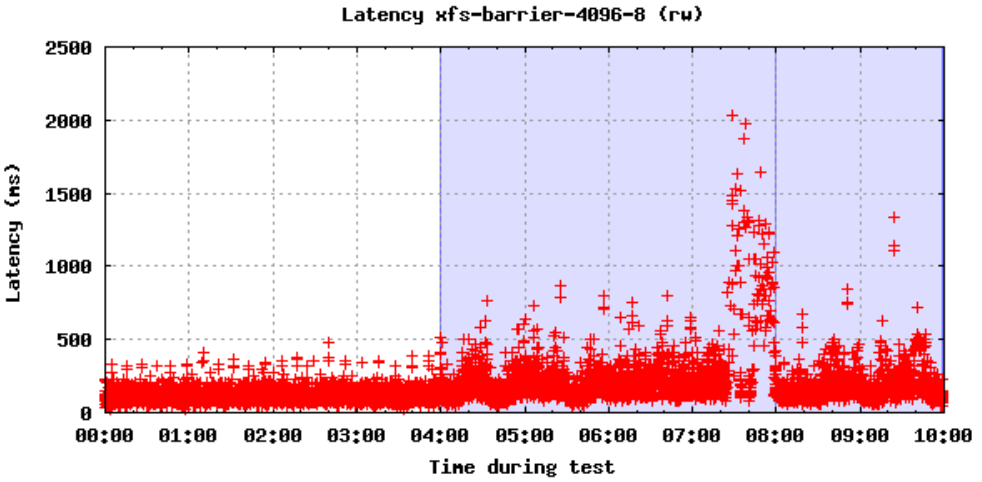

TPS ext4-journal-barrier-4096-8 (rw)
TPS
100
80
60
40
20
0
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
Time during test

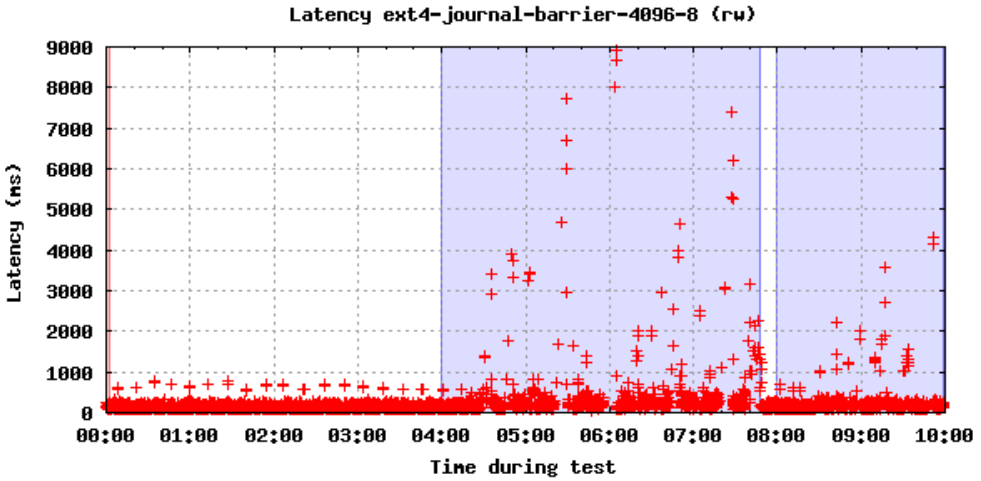

TPS xfs-barrier-4096-8 (rw)
TPS
Time during test
0
20
40
60
80
100
120
140
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00

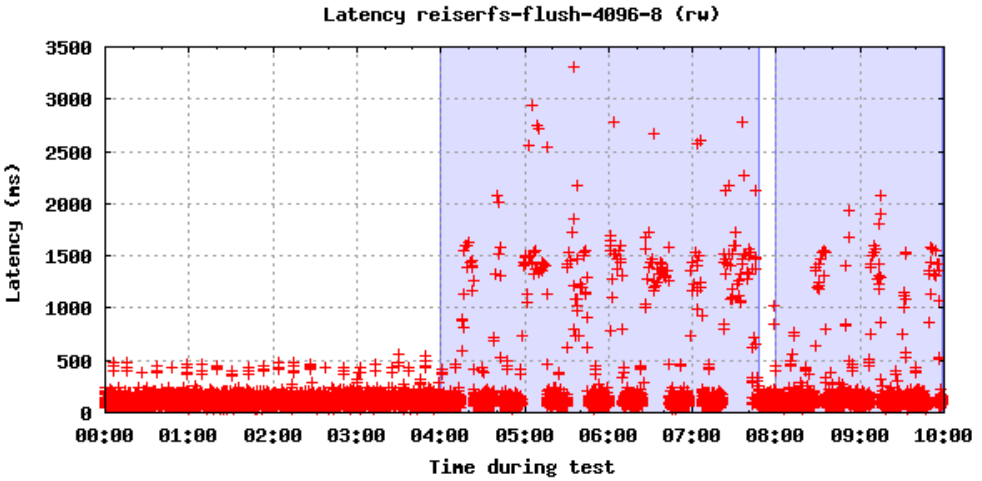

TPS reiserfs-flush-4096-8 (rw)
TPS
Time during test
0
20
40
60
80
100
120
140
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00

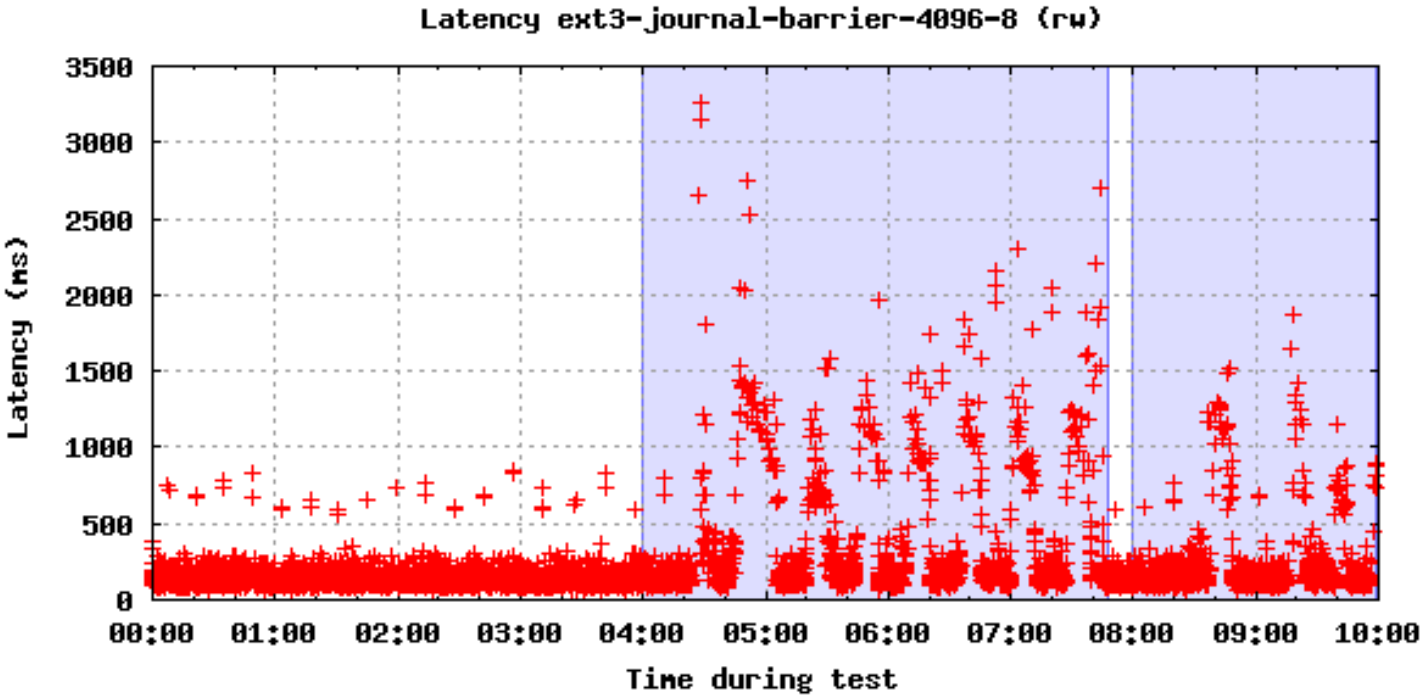
dopad checkpointu (latence)

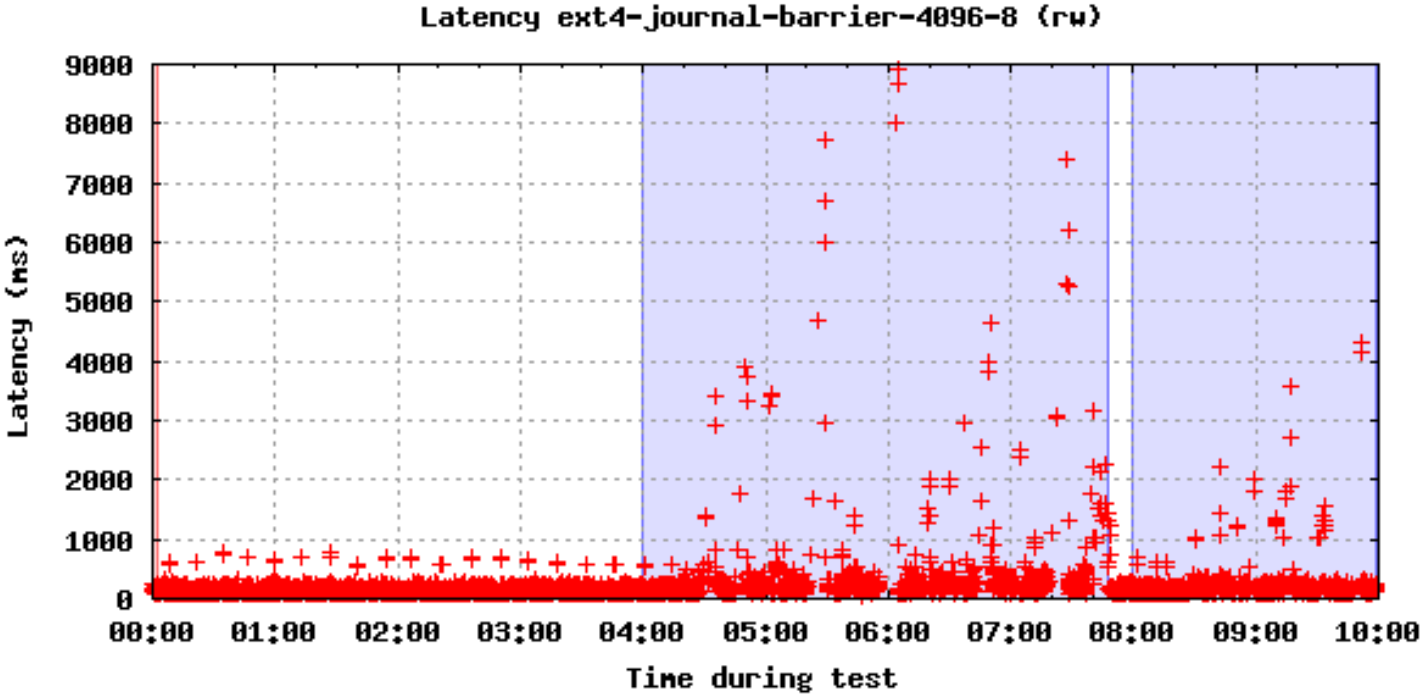


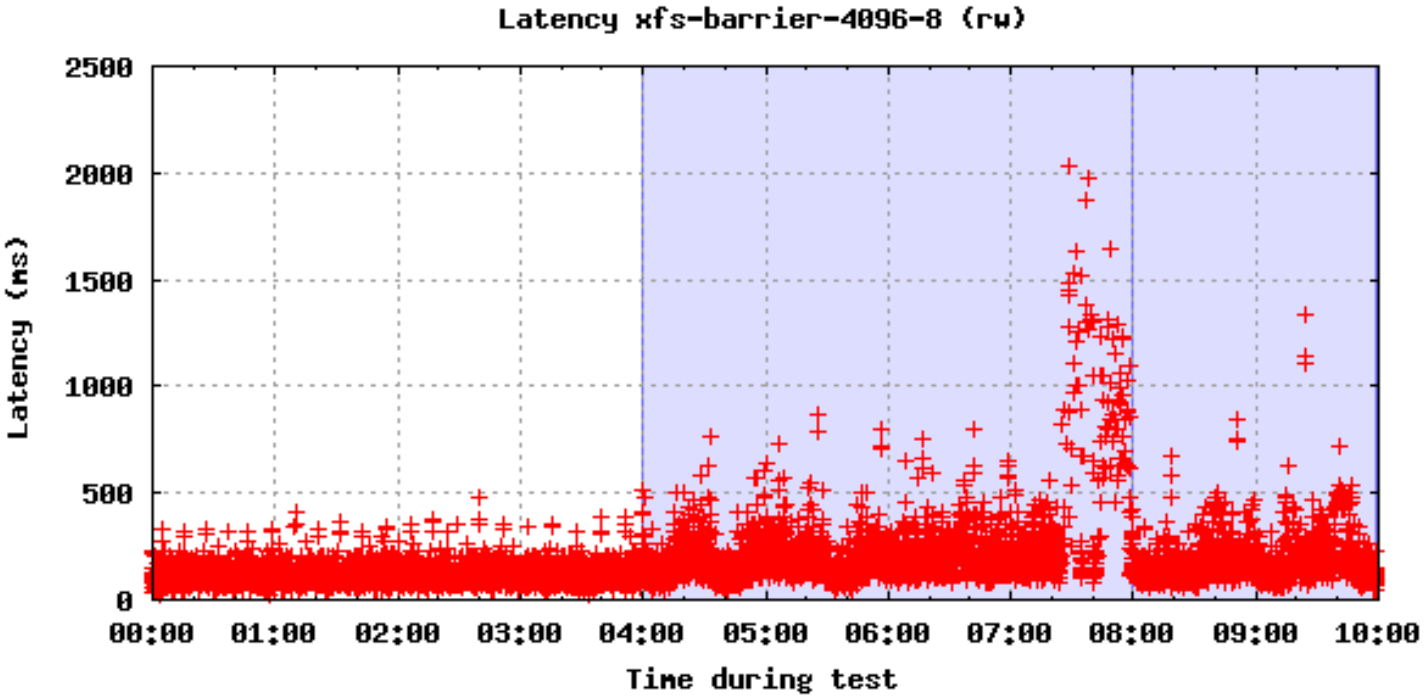


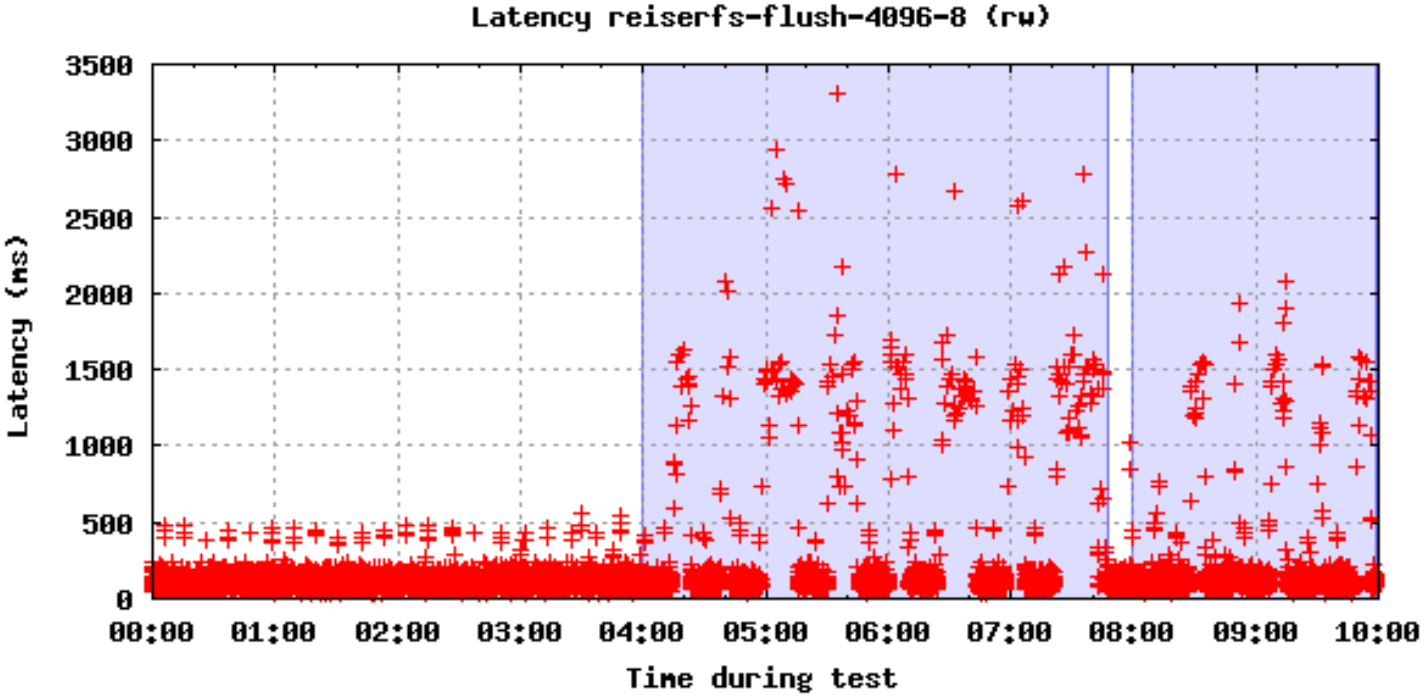





Latency ext3-journal-barrier-4096-8 (rw)
Latency (ms)
3500
3000
2500
2000
1500
1000
500
0
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
Time during test


Latency ext4-journal-barrier-4096-8 (rw)
Latency (ms)
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
Time during test


Latency xfs-barrier-4096-8 (rw)
Latency (ms)
2500
2000
1500
1000
500
0
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
Time during test


Latency reiserfs-flush-4096-8 (rw)
Latency (ms)
3500
3000
2500
2000
1500
1000
500
0
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
Time during test

stupňování problémů s page cache

1. zapisujete velký objem dat (např. noční load / batch)

2. dosáhnete dirty_background_ratio
   - pdflush začne zapisovat na disk
   - DB zapisuje do page cache
   - disk nestíhá čistit cache

3. dosáhnete dirty_ratio
   - pdflush zapisuje na disk
   - DB musí zapisovat přímo na disk (bez cache)

4. peklo

Ladění checkpointů

page cache, databáze, aplikace ...

Hardware people always seem convinced that any problem can be fixed by more complex software (software people, in contrast, know that problems can always be solved by more hardware).

[TheRaven64 @ slashdot.org]

Ladění …

- latence vs. propustnost
  - latence – reakce na jednotlivé požadavky
  - propustnost – celkový počet požadavků
  - většinou nemůžete mít oboje :-(

- cíle snahy
  - nenutit DB k checkpointu příliš často (kumulace změn)
  - když už je nutný checkpoint, tak postupný (spread)
  - omezení množství zápisů (vhodnou úpravou algoritmu, MVCC)
  - eliminace „šoku“ page cache (náhle zápis velkého objemu)

informace o checkpointech

log_checkpoints = on

- první co byste měli udělat
- základní informace (interval, kolik bufferů, trvání sync)
- umožňuje korelovat data oproti jiným datům (běžel checkpoint?)

pg_stat_bgwriter

- agregovaná data (kolik checkpointů jakého typu, kolik bufferů)
- ideální pro sběr snapshotů (kolik snapshotů za hodinu apod.)
- informace i o dalších zdrojích zápisů (bgwriter, backendy)

informace o checkpointech

log_checkpoints

```
2011-10-10 14:20:50 LOG: checkpoint starting: time
2011-10-10 14:21:45 LOG: checkpoint complete: wrote 7231
     buffers (1.1%); 0 transaction log file(s) added, 0 removed,
     2 recycled; write=29.911 s, sync=24.899 s, total=55.037 s
```

pg_stat_bgwriter

```
checkpoints_timed | checkpoints_req | buffers_checkpoint | buffers_clean
            3.236 |          83.044 |      1.376.460.896 |    59.124.159

 maxwritten_clean | buffers_backend | buffers_alloc
          304.410 |     285.595.787 | 6.643.047.623
```

operační systém

- souborový systém s rozumným fsync chováním
  - měl by umět fsync na úrovni souborů
  - xfs, ext4, …
- snížení „dirty“ limitů (menší objemy, častěji)
  - dirty_background_ratio = 2 (viděl jsem i 0)
  - dirty_ratio = 5
  - dirty_expire_centisecs = 500 (5 vteřin)
  - případně „_bytes“ varianty (po upgrade RAM stále platí)

konfigurace PostgreSQL

přímo konfigurace checkpointů

- checkpoint_segments
- checkpoint_completion_target
- checkpoint_timeout

nepřímý dopad

- wal_level
- shared_buffers
- background writer

segments / timeout

checkpoint_segments = 3

- výchozí hodnota je velmi nízká (jen 48MB dat)
- zvýšení omezí frekvenci „xlog“ checkpointů (není místo ve WAL)

checkpoint_timeout = 5 min

- zvýšení maximální doby mezi checkpointy (maximum 1h)

důsledky zvýšení

- možnost opakované modifikace bloku mezi checkpointy (snížení I/O)
- prodloužení recovery v případě pádu (aplikace WAL segmentů od posledního checkpointu)

completion_target

checkpoint_completion_target

- před verzí 8.3 se při checkpointu jelo „na plný plyn“ (hodně I/O)
- nově se zápisy rozloží na dobu dalšího checkpointu (timed i xlog)

příklad

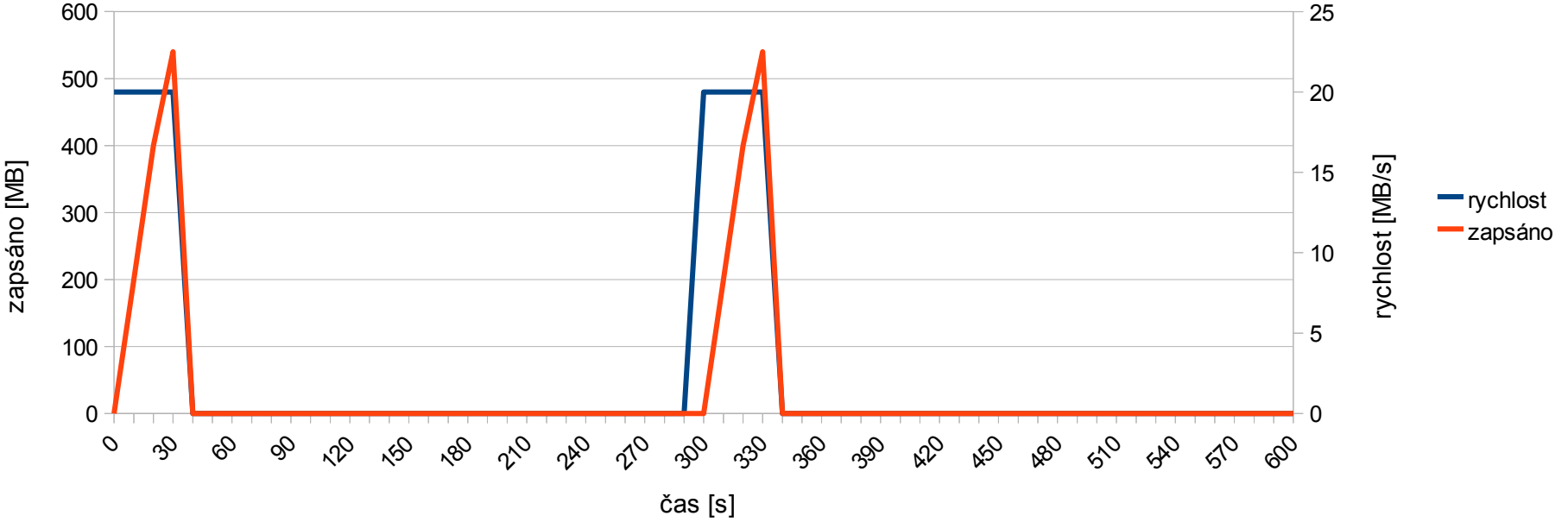
- vypršel timeout (5 minut), 540 MB dat k zápisu (v shared buffers)
- completion_target * timeout = 0.9 * 300s = 270s (konec zápisu)
- 540 MB / 270 s = 2 MB/s (průměrná rychlost)

důsledky

- zvýšení počtu segmentů (cca na dvojnásobek)

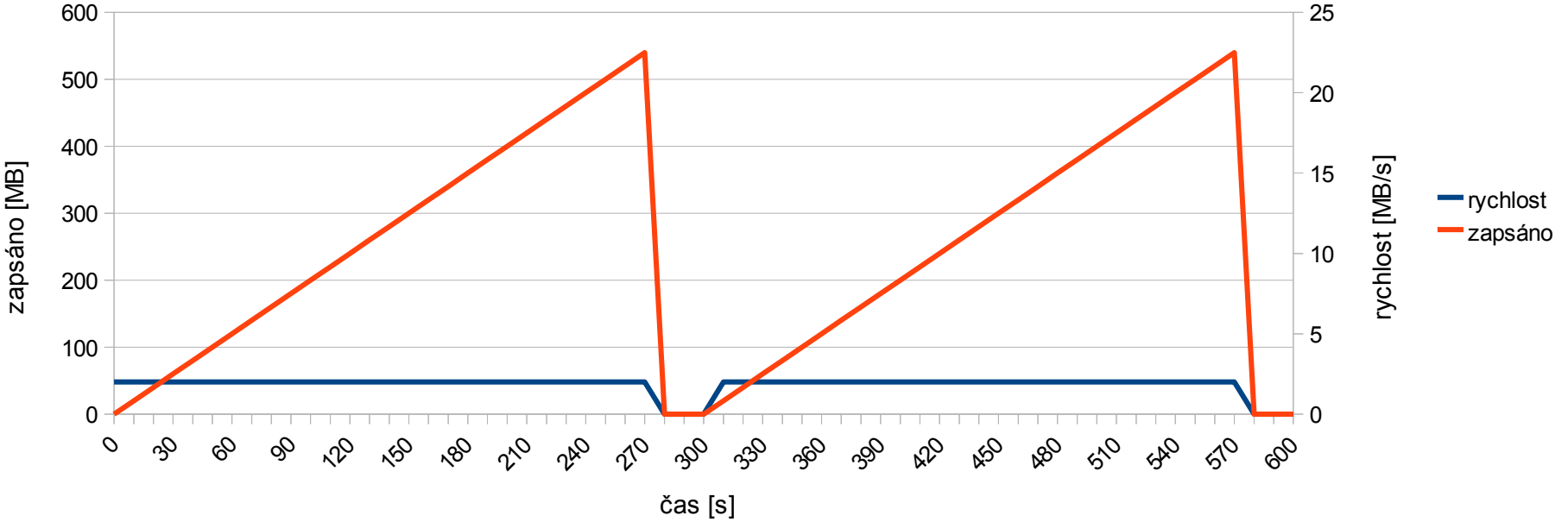
Old-style checkpointy

540 MB k zápisu, 300s timeout, nárazově



Spread checkpoints

540MB k zápisu, 300s timeout => 2 MB/s průměrná rychlost



wal_level

- ne vždy je třeba „wal_level = archive“
  - nemáte standby
  - nechcete PITR (Point-In-Time-Recovery)
- „wal_level = minimal“ omezí objem WAL (u některých operací)
  - CREATE TABLE AS
  - CREATE INDEX
  - CLUSTER
  - COPY (do nové tabulky – i partition je tabulka)
- může velmi výrazně zlepšit load dat apod.
  - ostatní transakce novou relaci nevidí
  - data jdou přímo na disk (ne do buffers)

shared_buffers

- občas se doporučuje zmenšit shared buffers (méně dat k zápisu)
  - spíše zastaralá (pre-8.3) alternativa k spread checkpointům
  - kombinace s agresivní konfigurací bgwriteru
  - http://www.westnet.com/~gsmith/content/postgresql/chkp-bgw-83.htm
- při čtení to většinou nevadí (díky page cache)
- při zápisu může mít negativní důsledky
  - backendy jsou nuceny zapisovat data samy (tj. iowait)
  - projevuje se jako růst pg_stat_bgwriter.buffers_backend

shared_buffers

moje doporučení (>= 8.3)

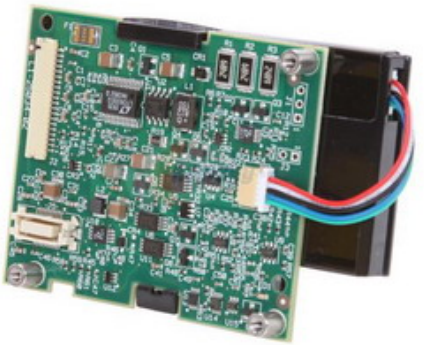
1. nastavit konzervativní hodnotu
2. postupně zvyšovat a sledovat výkon
   - cache „hit ratio“ (bohužel bez page cache)
   - výkon aplikace (ideální metrika)
3. zastavit jakmile výkon přestane růst
4. případně agresivnější bgwriter (snížení hodnot backend_clean)
5. v každém kroku jen jedna změna

minimální velikost shared buffers, maximální efektivita

Ladění … hardware

řadič s write cache

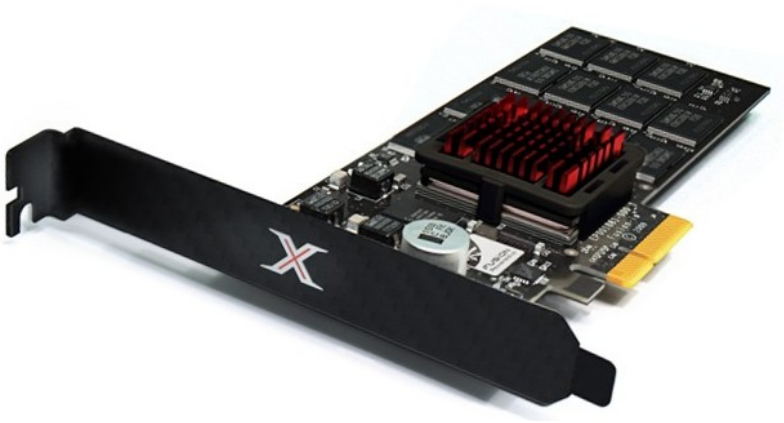
- nejlépe s baterkou (ale dle libosti)
- cache má omezenou velikost, nezvládne všechno
- spread checkpointy výrazně zlepšují (menší objemy, průběžně)



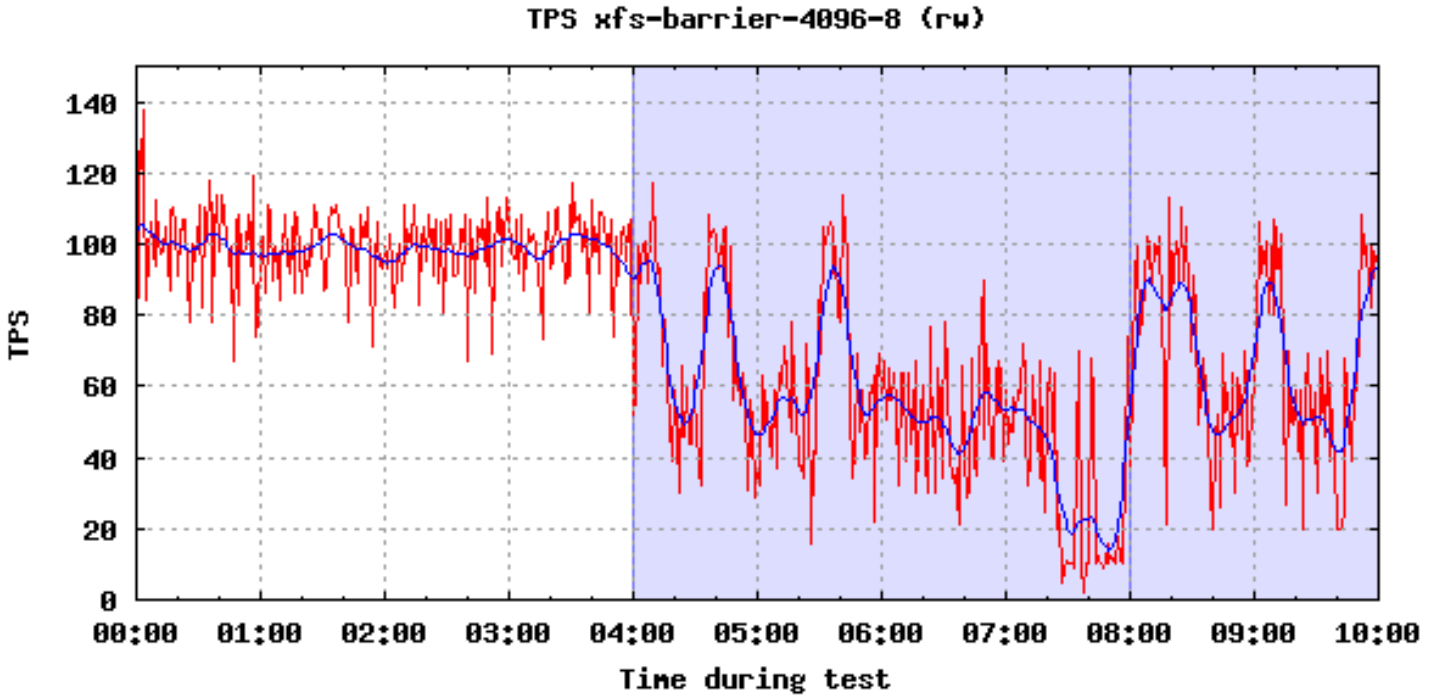
Ladění … hardware

SSD

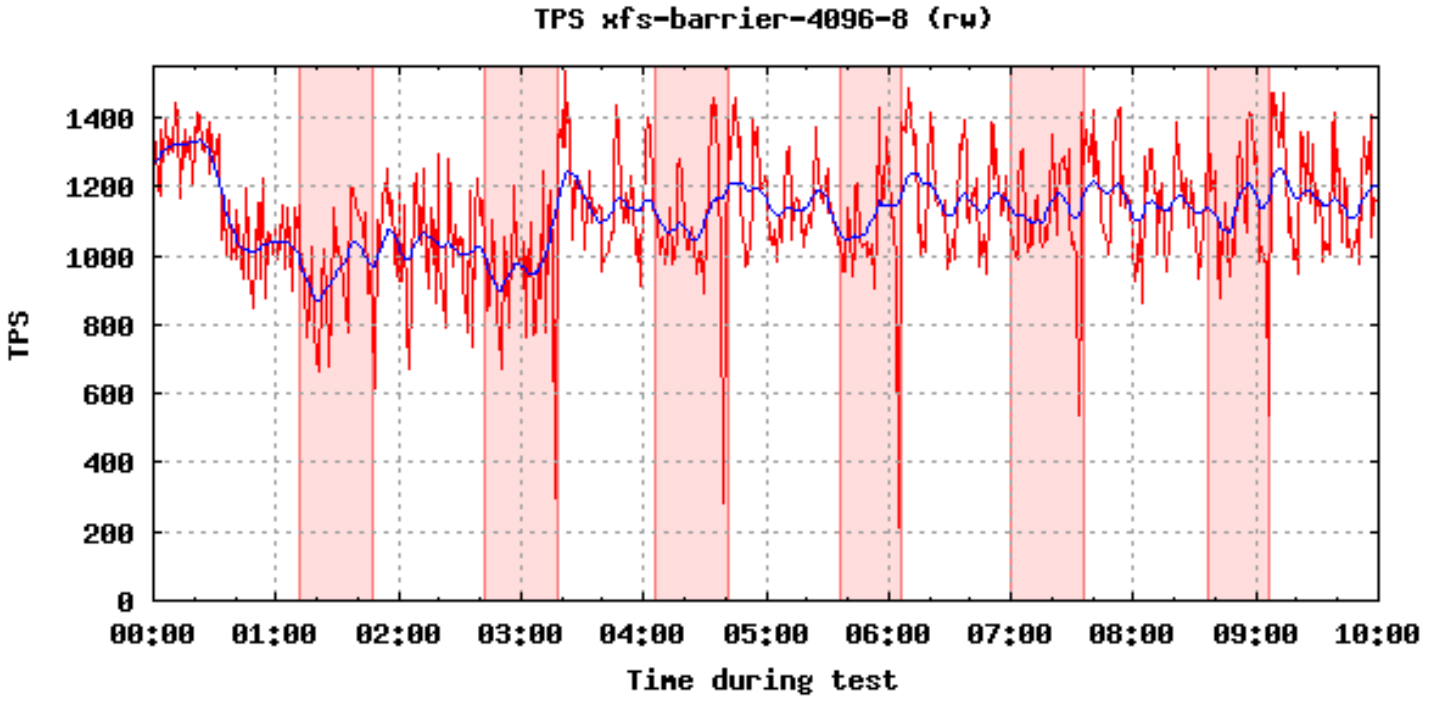
- víceméně řeší problém s náhodným přístupem (čtení vs. zápisy)
- omezený počet zápisů (problémy s checkpointy => write-heavy db)
- dražší (?)
- spolehlivost (?)



7.2k SATA



Intel 320 SSD



Q&A