The Write Stuff

Greg Smith

2ndQuadrant US

05/20/2011

About this presentation

- The master source for these slides is
  http://projects.2ndquadrant.com

2ndQuadrant Professional PostgreSQL

The buffer cache

- shared_buffers sets size
- 256MB - 8GB is typical
- Traditional tuning suggests around 25% of total RAM

Checkpoints

- All dirty data in buffer cache must be flushed to disk eventually
- WAL segments are 16MB
- Checkpoint requested after every checkpoint_segments worth of writes
- Timed checkpoint every checkpoint_timeout (5 minute default)
- Traditional tuning sets checkpoint_segments 16-256

Checkpoint spikes

- Before 8.3, all dirty data written in one burst
- 8.3 added Spread Checkpoints
- Defaults aim to finish 50% of the way through next checkpoint
- fsync flush to disk happens at end of checkpoint
- Optimal behavior: OS already wrote data out before fsync call
- Attempts to spread the sync out didn't work usefully
- Spikes still happen

2ndQuadrant
Professional PostgreSQL

Linux filesystem trivia

- Checkpoint rewrite tests all on Linux
- Default and only stable Linux filesystem then was ext3
- ext3 handles fsync by writing all cached data to disk
- Spread sync can't help if every fsync writes all data out
- WAL writes do fsync too
- One reason why separating WAL and database disks helps so much
- XFS and ext4 allow granular sync
- Recent Linux kernels (around 2.6.32) make ext3 much better too

Linux write caching

- dirty_ratio and dirty_background_ratio control % of RAM to allow dirty
- More aggressive writing happens when thresholds crossed
- Writes can become blocked
- Ideally, dirty RAM fits in battery backed cache size
- Kernel before 2.6.22: 10%/40% of RAM are thresholds
- Kernel 2.6.22 and later: 5%/10% are defaults
- Kernel 2.6.29 and later: dirty_bytes and dirty_background_bytes allow setting exact amount of RAM to allow dirty

Write caching, 16GB Server

Dirty	Wrback	Written	Dirty%
1134660	12	0	7.5
1213692	0	0	8.0
1293152	12	0	8.5
1372200	0	36	9.1
1451268	0	20	9.6
1530332	12	411196	10.1
1153944	107000	343440	7.6
881480	109120	293936	5.8
719060	10460	40	4.7

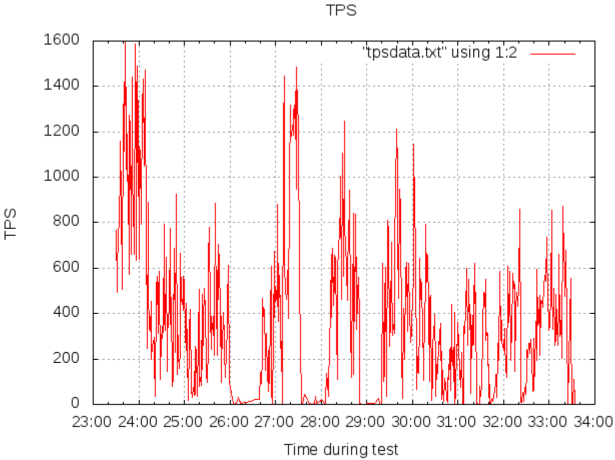
2ndQuadrant
Professional PostgreSQL

Having a bad day on purpose with ext3

- log_checkpoints shows sync time
- 8GB of RAM in server
- 5% dirty=400MB
- 10% dirty=800MB
- 256MB of battery-backed cache
- Standard pgbench test dirties data very fast

2ndQuadrant
Professional PostgreSQL

pgbench write stalls



2ndQuadrant
Professional PostgreSQL

Checkpoint sync times, ext3, new 9.1 logging

```
LOG: checkpoint starting: xlog
DEBUG: checkpoint sync:
number=1 file=base/16385/16480 time=10422.859 msec
number=2 file=base/16385/16475_vm time=2896.614 msec
number=3 file=base/16385/16475.1 time=57.836 msec
number=4 file=base/16385/16466 time=20.080 msec ...
number=8 file=base/16385/16475 time=35.164 msec
LOG: checkpoint complete: wrote 2143 buffers (52.3%);
0 transaction log file(s) added, 0 removed, 3 recycled;
write=1.213 s, sync=13.589 s, total=24.744 s;
sync files=8, longest=10.422s, average=1.698s
```

2ndQuadrant
Professional PostgreSQL

A really bad day on a popular web site

- XFS
- Lots of RAM
- shared_buffers=512MB, typically under 200MB dirty at checkpoint time
- Often gigabytes of write cache dirty with random writes
- Still well under 10%, Linux is unfortunately not too concerned
- sync time = 50 minutes?!
- Not even 1MB/second into a medium sized disk array

2ndQuadrant
Professional PostgreSQL

Another bad day, on a heavily queried internal system

- LOG: checkpoint complete: wrote 33282 buffers (3.2%);
- 0 transaction log file(s) added, 60 removed, 129 recycled;
- write=228.848 s, sync=4628.879 s, total=4858.859 s
- (That's 80 minutes for 264MB of writes!)

2ndQuadrant
Professional PostgreSQL

Writes in PostgreSQL

- Checkpoint write: most efficient
- Background writer write: still good
- Backend write, fsync aborbed by background writer: fine if OS caches
- Backend write, BGW queue filled, backend does fsync itself: bad

2ndQuadrant
Professional PostgreSQL

Backend sync counts

```
$ psql -x -c "select * from pg_stat_bgwriter"
checkpoints_timed | 0
checkpoints_req | 4
buffers_checkpoint | 6
buffers_clean | 0
maxwritten_clean | 0
buffers_backend | 654685
buffers_backend_sync | 84
buffers_alloc | 1225
```

The root problem

- Background writer stop working normally while running sync
- Never pauses to fully consume the fsync queues backends fill
- Once filled, all backend writes do their own fsync
- Serious competition for the checkpoint writes

2ndQuadrant Professional PostgreSQL

Possible solutions

- Introduce a pause to spread out writes after each file sync
- During the pause time, continue running regular background writer work
- Improve general fsync queue management
- Upgrade Linux kernel, reduce write cache to small number of bytes

2ndQuadrant
Professional PostgreSQL

Spread sync: pause after each sync, cleanup fsync queue

- Helped keep fsync contention under control
- Deployed into production
- Works, but improvement hard to replicate on testbed

Use a tiny Linux write cache

- Drop dirty_bytes and dirty_background_bytes to 128MB/64MB
- ext3: 10-15% drop in transaction rate, but latency drops to under 1/4 of standard config
- XFS: Performance generally worse
- Problem: VACUUM time is 48% to 71% longer!
- Ring buffer in VACUUM needs a large OS write cache to run efficiently

2ndQuadrant
Professional PostgreSQL

Compact fsync queue: comitted for 9.1

- Many fsync requests in the queue were repeated requests for the same file
- Client backends who find the queue full compact it themselves, by removing duplicates
- No longer need the background writer to catch this worst-case scenario
- Works perfectly in synthetic benchmarks
- Zero buffers_backend_sync, 10% gain in write performance
- Gains from other approaches marginal after this change

Planning impact

- Be careful using large settings for shared_buffers with heavy writes
- Monitor size of OS cache dirty data to measure problems here
- grep "Dirty:" /proc/meminfo
- ext3 can be increasingly bad as total system memory continues to increase
- Revival of XFS popularity for over 16TB filesystems makes it more viable now
- Need to use nobarrier option when you have a battery-backed cache
- Status of ext4 still not explored well
- Logging sync timing and compact fsync queue are both easy to backport changes

2ndQuadrant
Professional PostgreSQL