Waits monitoring in PostgreSQL

Ildus Kurbangaliev
i.kurbangaliev@gmail.com
Tools

› perf

› pg_stat_statements

› pg_stat_kcache

› SystemTap

› and others...
<table>
<thead>
<tr>
<th>Percentage</th>
<th>Component</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>60.42%</td>
<td>postgres</td>
<td>s_lock</td>
</tr>
<tr>
<td>8.09%</td>
<td>postgres</td>
<td>LWLockAcquire</td>
</tr>
<tr>
<td>7.03%</td>
<td>postgres</td>
<td>LWLockRelease</td>
</tr>
<tr>
<td>5.46%</td>
<td>postgres</td>
<td>PinBuffer</td>
</tr>
<tr>
<td>2.80%</td>
<td>postgres</td>
<td>heap_page_prune_opt</td>
</tr>
<tr>
<td>2.67%</td>
<td>postgres</td>
<td>hash_search_with_hash_value</td>
</tr>
<tr>
<td>2.15%</td>
<td>postgres</td>
<td>heap_hot_search_buffer</td>
</tr>
<tr>
<td>1.25%</td>
<td>postgres</td>
<td>UnpinBuffer</td>
</tr>
<tr>
<td>0.93%</td>
<td>postgres</td>
<td>HeapTupleSatisfiesMVCC</td>
</tr>
<tr>
<td>0.36%</td>
<td>libc-2.12.so</td>
<td>__memcmp_sse4_1</td>
</tr>
<tr>
<td>0.35%</td>
<td>postgres</td>
<td>_bt_next</td>
</tr>
<tr>
<td>0.33%</td>
<td>[kernel]</td>
<td>_spin_lock</td>
</tr>
<tr>
<td>0.29%</td>
<td>postgres</td>
<td>CheckForSerializableConflictOut</td>
</tr>
<tr>
<td>0.29%</td>
<td>postgres</td>
<td>ReadBuffer_common</td>
</tr>
<tr>
<td>0.24%</td>
<td>postgres</td>
<td>hash_any</td>
</tr>
<tr>
<td>0.23%</td>
<td>postgres</td>
<td>HeapTupleIsSurelyDead</td>
</tr>
<tr>
<td>0.23%</td>
<td>postgres</td>
<td>heapgetpage</td>
</tr>
<tr>
<td>0.21%</td>
<td>postgres</td>
<td>get_hash_value</td>
</tr>
</tbody>
</table>

Samples: 3M of event 'cycles', Event count (approx.): 894845079549
Monitored waits

- Locks (heavyweight)
- LWLocks (lightweight locks)
- Latch
- Network
- Storage (IO)
Subtypes of waits

› Locks (heavyweight)
  8 types (9 in 9.5+)
› LWLocks (lightweight locks)
› Latch
› Network
  Reads and writes
› Storage (IO)
  Reads and writes (storage manager, xlog, slru)
LWLocks

Locks (lightweight)

Individual (41)

Groups of locks (buffer manager, processes, predicate and others)

User defined LWLocks (extensions)
Profiling

```
b1=# select * from pg_stat_wait_profile
    where event_name = 'WALWriteLock' limit 1;

- [ RECORD 1 ]-----------------
  pid       | 1804
  class_id  | 1
  class_name | LWLocks
  event_id  | 8
  event_name | WALWriteLock
  wait_time | 8719
  wait_count| 6
```

`wait_time` and `wait_count` fields show time spent on waits.
Profiling

Red - storage waits
```sql
b1=# select * from pg_stat_wait_history limit 1;
-[ RECORD 1 ]-------------------------------
 pid   | 1809
 sample_ts | 2015-10-29 04:58:53.85285-04
 class_id | 3
 class_name | Storage
 event_id | 0
 event_name | SMGR_READ
 wait_time | 10299
 p1     | 1663
 p2     | 16384
 p3     | 12214
 p4     | 0
 p5     | 1
```
Tracing

It has a big **overhead** and can be used only within separate sessions

**terminal 1:** $ psql b1

**terminal 2:**
$ ps ax | grep postgres
<...>
11085 ? Ss 0:00 postgres: postgres b1 [local] idle

$ psql b1 -c "select pg_start_trace(11085, '/tmp/f.trace')"

**terminal 1:**
b1=# CREATE TABLE t1 AS SELECT i, i*10 AS i1 FROM generate_series(1,10) i;
SELECT 10
terminal 2:
$ tail -f /tmp/f.trace
stop 2015-07-10 10:03:35.603458-04 Network
start 2015-07-10 10:03:35.603464-04 Network READ 0 0 0 0 0
stop 2015-07-10 10:03:44.099587-04 Network
start 2015-07-10 10:03:44.100401-04 Storage READ 1663 16384 1259 2 0
stop 2015-07-10 10:03:44.100424-04 Storage
start 2015-07-10 10:03:44.102549-04 Network WRITE 0 0 0 0 0
stop 2015-07-10 10:03:44.102573-04 Network
start 2015-07-10 10:03:44.102582-04 Network READ 0 0 0 0 0
stop 2015-07-10 10:05:33.029975-04 Network
start 2015-07-10 10:05:33.030205-04 Storage READ 1663 16384 2691 0 28
stop 2015-07-10 10:05:33.030233-04 Storage
start 2015-07-10 10:05:33.030246-04 Storage READ 1663 16384 1255 0 50
stop 2015-07-10 10:05:33.03026-04 Storage
Overhead

Server configuration:

- Intel(R) Xeon(R) CPU X5675@3.07GHz, 24 cores
- RAM 24 GB
- pgbench -S 500 ~ 1.6 Gb

Reduce impact of disk I/O:

- fsync off
- tmpfs
$ pgbench -S b1 -c 96 -j 4 -T 300
starting vacuum...end.
transaction type: SELECT only
scaling factor: 500
query mode: simple
number of clients: 96
number of threads: 4
duration: 300 s
number of transactions actually processed: 39349816
latency average: 0.732 ms
tps = 131130.859559 (including connections establishing)
tps = 131153.752204 (excluding connections establishing)
$ pgbench -S b1 -c 96 -j 4 -T 300
starting vacuum...end.
transaction type: SELECT only
scaling factor: 500
query mode: simple
number of clients: 96
number of threads: 4
duration: 300 s
number of transactions actually processed: 39172607
latency average: 0.735 ms

\[
\text{tps} = 130574.626755 \quad \text{(including connections establishing)}
\]
\[
\text{tps} = 130600.767440 \quad \text{(excluding connections establishing)}
\]
Thank you

Ildus Kurbangaliev
i.kurbangaliev@gmail.com