Tips & Tricks on Tuning MySQL Performance

Shlomi Noach

Interbit T & C

Java Tech Day
Israel, June 2009

Performance tuning

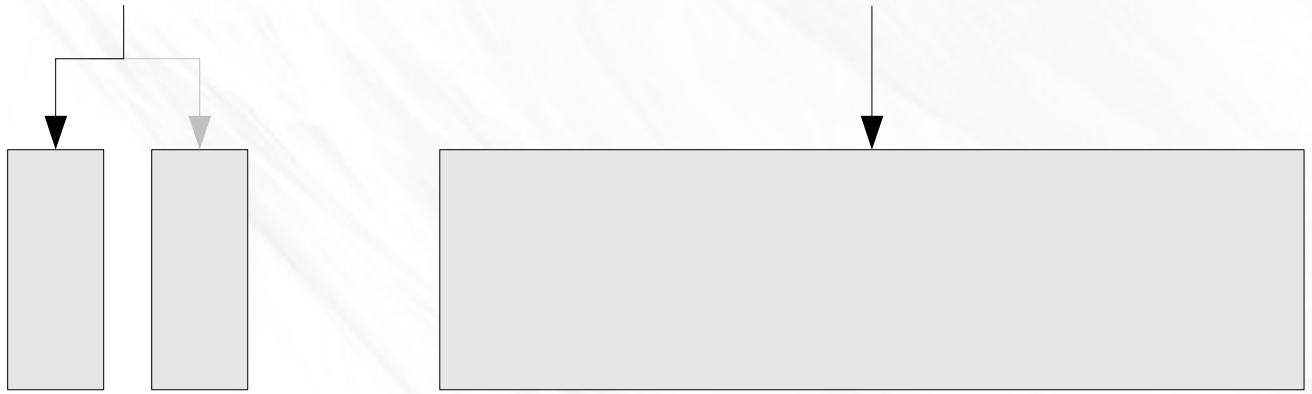
- One of the last things to tune.
- Data types, Schema structure, Queries: usually have more impact.
- However...

Sometimes you're on the edge

- You may be suffering high spikes of usage
- You may be scaling faster than you thought
- You need to minimize downtime

InnoDB transaction logs

- Transactions are written both to buffer pool and to transaction logs.
- Buffer pool is not flushed on transaction commit.



Transaction logs **Buffer pool**

InnoDB transaction logs

- The smaller the transaction log, the more frequently page flush must occur on the buffer pool.
- The smaller the transaction log, the more random I/O operations occur.
- The larger the transaction log, the more data it contains, alleviating the need to flush the pool.
- The larger the transaction log, the more serial I/O operations occur.

Large transaction logs have downsides

- The larger the transaction log, the longer it will take for crash recovery.
- Very large transaction logs (hundreds of MB up to few GB) can make for *many hours* of recovery time.
- A trade-off must take place.

Calculating desired transaction log size

- Check for log sequence number in SHOW ENGINE INNODB STATUS.

```
mysql> pager grep sequence -
mysql> SHOW ENGINE INNODB STATUS\G
SELECT SLEEP(60); SHOW ENGINE
INNODB STATUS\G
Log sequence number 184 4145541287
...
Log sequence number 184 4150607879
```

How much transactional MB do we write per hour?

- (4150607829-4145541287)*60/1024/1024 = 290 MB/Hr
- If we want to be able to recover within one hour, the combined transaction log size must not exceed 290MB

Need to restart?

- Restarting an InnoDB MySQL server (change of parameters, upgrade, machine reboot), may take a while.
- First attempt to minimize shutdown time is to set **innodb_fast_shutdown**.
- But this will be costly on next startup.

Gradual shutdown

set global innodb_max_dirty_pages_pct=0;

- Will cause InnoDB to keep less dirty pages in memory. InnoDB will flush pages frequently.
- Reaching 0 is not possible on loaded server.
- Server is still up and responsive during this time.
- Upon shutdown, there's less flushing to do, hence shutdown is faster.

Helping out some more

SHOW OPEN TABLES;

- Manually execute **FLUSH TABLE T** for each **T** in list of open tables.
- Will lock each table on turn.

Query cache

- Allows for caching query results.
- A query's result can be stored in the query cache, to be directly served to a later identical query, without reading actual table data.

Query cache

- Allows for caching query results.
- A query's result can be stored in the query cache, to be directly served to a later identical query, without reading actual table data.

How good is my query cache?

```
mysql> SHOW GLOBAL STATUS LIKE 'qcache%';
+-------------------------+----------+
| Variable_name           | Value    |
+-------------------------+----------+
| Qcache_free_blocks      | 2        |
| Qcache_free_memory      | 66066864 |
| Qcache_hits             | 2951917  |
| Qcache_inserts          | 1657123  |
| Qcache_lowmem_prunes    | 0        |
| Qcache_not_cached       | 130096   |
| Qcache_queries_in_cache | 16       |
| Qcache_total_blocks     | 47       |
+-------------------------+----------+
```

But comparing consequent status:

- Doesn't look too good now...

```
+---------------------------+----------+
| Variable_name             | Value    |
+---------------------------+----------+
| Qcache_hits               |       10 |
| Qcache_inserts            |      257 |
+---------------------------+----------+
```

Query cache

- Incurs management overhead.
- Has a single locking mutex.
- Busy multicore system may suffer from using query cache.
- Consider using Memcached instead.

Memcached

- An application level caching server.
- Has client API for all major programming languages.
- Has client API for MySQL: memcached functions for MySQL.
- Can be utilized by stored procedures or triggers for cache invalidation, for example.

```
SELECT memc_set('mykey', 'The answer is 42');
SELECT memc_delete('mykey')INTO @discard;
```

Table cache

- Caches table file descriptors.
- Usually should be very large (e.g. in the thousands)
- When schema contains huge amount of tables (e.g. ~ millions), table cache becomes bottleneck.

Table cache

- Like query cache, has global, single mutex.
- When flushing old tables from cache, LRU algorithm requires mutex lock.
- Time to hold lock is in proportion to cache size.
- When DB has too amny tables, consider *reducing* table cache size.
- Flushing will still occur, but with smaller lock contention periods.

Thank you!

Visit http://openark.org