How to Fulfill the Potential of InnoDB's Performance and Scalability

MySQL Conference & Expo 2010

Yasufumi Kinoshita

Senior Performance Engineer
Percona Inc.
MySQLPerformanceBlog.com
About me...

http://MysqlPerformanceBlog.com

Yasufumi Kinoshita
Senior Performance Engineer, Percona Inc
Primary Developer of XtraDB(*), XtraBackup and percona-patch

* XtraDB
Enhanced InnoDB based on InnoDB-Plugin
Open for third-party patches
Number of specific tuning options are added
http://www.percona.com/docs/wiki/percona-xtradb:start
https://launchpad.net/percona-xtradb
What is this session about?

To know what should be fixed next for scalability, needed to know how it has been fixed correctly.

Tuning procedure based on “SHOW INNODB STATUS”
  - IO bound scalability
  - CPU bound scalability

Tuning examples using benchmarks
  - When should we upgrade to InnoDB Plugin or XtraDB
  - Using fast storage like SSD

Additional TIPS about specific options

XtraDB's TODO for more scalability
Tuning Procedure

Tuning until you are satisfied with performance. So, if you are satisfied with performance, you can stop tuning.
1. Check IO bound or not

Check the pending IO in “SHOW INNODB STATUS”

```
......
----------------------
BUFFER POOL AND MEMORY
----------------------
......
Pending reads 28
Pending writes: LRU 0, flush list 0, single page 0
......
```

Sampling several times and average the each values

If (>10), it must be IO bound for InnoDB.

“Pending reads” : Read IO bound
“Pending writes” (LRU or flush list) : Write IO bound
2. Check IO bound scalability

Check the IO throughput by output of “vmstat” etc...

“Are you satisfied with the throughput as your storage (e.g. RAID or SDD)?”

- **Yes**
  - Tune your SQLs :-) (not described in this session)

- **No**
  - Use InnoDB Plugin or XtraDB and tune
    - `innodb_read_io_threads`
    - `innodb_write_io_threads`

  (* RAID and SSD can accept parallel IO requests*)
3. Check CPU bound scalability

Check the CPU activity by output of “vmstat” etc...

“Are you satisfied with the throughput as your number of CPU cores?”

{ “Yes”
  Tune your SQLs :-) (not described in this session)

  “No”
  Investigate the contention point in InnoDB
  (after next page... )
4. Check “true” contention point

Check the contention in “SHOW INNODB STATUS”

......
SEMAPHORES
......
OS WAIT ARRAY INFO: reservation count 28702892, signal count 18960799
--Thread 140426528233808 has waited at btr/btr0sea.c line 774 for 0.00 seconds
S-lock on RW-latch at 0x7fb86b28f0b8 created in file btr/btr0sea.c line 139
a writer (thread id 140426530642256) has reserved it in mode exclusive
number of readers 0, waiters flag 1
Last time read locked in file btr/btr0sea.c line 774
Last time write locked in file btr/btr0sea.c line 1024
--Thread 140426532649296 has waited at btr/btr0cur.c line 443 for 0.00 seconds
S-lock on RW-latch at 0x7fb7a62f71d0 created in file buf/buf0buf.c line 547
a writer (thread id 140426532649296) has reserved it in mode exclusive
number of readers 0, waiters flag 1
Last time read locked in file btr/btr0sea.c line 794
Last time write locked in file buf/buf0buf.c line 1797
......

Sampling several times and aggregate the entries
4. Check “true” contention point
(ex.) aggregate the entries by shell script

```bash
#!/bin/sh

# Mutex contention
cat $1.innodb | grep "Mutex at " | cut -d"," -f1 | sort | uniq -c > /tmp/tmp1.txt

cat $1.innodb | grep "lock on " | cut -d"-" -f2- | sort | uniq -c > /tmp/tmp2.txt

cat /tmp/tmp1.txt /tmp/tmp2.txt | sort -n > $1.contention

cat $1.innodb | cut -d"," -f1 | sort | uniq -c > /tmp/tmp1.txt

cat $1.innodb | cut -d"-" -f2- | sort | uniq -c > /tmp/tmp2.txt

cat /tmp/tmp1.txt /tmp/tmp2.txt | sort -n > $1.contention

rm /tmp/tmp1.txt /tmp/tmp2.txt
```

Pickup the name of the Mutex/RW-latch from source code...

(* current XtraDB print their names directly, so you don't need to lookup source code)
4. Check “true” contention point

(1) If you use built-in InnoDB of MySQL 5.1 or older...

Any RW-latch contention
(especially for btr_search_latch)
should be bottle-neck.

“Top entry is RW-latch”

Upgrade to InnoDB-Plugin or XtraDB

(* InnoDB-Plugin or XtraDB have more native RW-latch implementation)
4. Check “true” contention point

(2) The top entry may not be “true” contention point

Solve based on latch-order priority

“latter is more prior”

The contentions should be affected by the latter-ordered mutex/latch
4. Check “true” contention point

<Priority of typical Mutex/Latch contentions to be solved>

1. buf_pool_mutex
   * Use XtraDB

2. btr_search_latch
   * Use InnoDB-Plugin or XtraDB

3. log_sys->mutex
   * XtraDB may solve a little

4. kernel_mutex
   * No way for now

5. rseg->mutex
   * Use XtraDB and option innodb_extra_rsegments

(* The order can be looked at include/sync0sync.h)
4. Check “true” contention point

(ex.) One case of InnoDB-Plugin CPU scale bound

- Mutex at 0xd26aa0 created file ibuf/ibuf0ibuf.c line 467
- Lock on RW-latch at 0x7fe6201024f0 created in file dict/dict0dict.c line 1569
- Lock on RW-latch at 0x80ed6c0 created in file btr/btr0sea.c line 170
- Lock on RW-latch at 0x7fe62010e1b0 created in file dict/dict0dict.c line 1569
- Mutex at 0x80f9878 created file trx/trx0rseg.c line 210
- Lock on RW-latch at 0xd2c900 created in file dict/dict0dict.c line 622
- Mutex at 0xd2c840 created file buf/buf0buf.c line 955
- Mutex at 0x80eeb30 created file log/log0log.c line 776
- Lock on RW-latch at 0x7fe62010d960 created in file dict/dict0dict.c line 1569

This contention should cause the anothers. XtraDB will solve this scale problem.

Its results is shown later....
5. Checks to stabilize throughput

Check the followings to avoid decreasing throughput or periodical drop

5.1. Too many too old modified blocks

5.2. Too large insert buffer

5.3. Too large history list (rseq)
5.1. Too many too old modified blocks

Increasing “checkpoint age” without “contiguous proper flush of modified blocks” may cause sudden stormy flush periodically.

---

LOG
---

Log sequence number 34136918674
Log flushed up to 34136917188
Last checkpoint at 32580790171

Checkpoint age 1556128503

----------------------

BUFFER POOL AND MEMORY
----------------------

32.90 reads/s, 11.40 creates/s, 2189.18 writes/s

[checkpoint age] = [Log sequence number] - [Last checkpoint at]

XtraDB prints directly

Amount of flushing
5.1. Too many too old modified blocks

InnoDB-Plugin and XtraDB have the each strategies for “contiguous proper flush of modified blocks” in default. The both are worth to try.

**InnoDB-Plugin:**

- `innodb_adaptive_flushing` = true (default)

**XtraDB:**

- `innodb_adaptive_flushing` = false (default)
- `innodb_adaptive_checkpoint` = estimate (default)
5.2. Too large insert buffer

The insert buffer is good architecture. (insert to 2ndary indexes as background async tasks)
The problem is when “growing too large”
It should be processed more actively in such case. (stopping ibuf simply may decrease performance.)

```
......
-------------------------------------
INSERT BUFFER AND ADAPTIVE HASH INDEX
-------------------------------------
Ibuf: size 1704, free list len 1209, seg size 2914,
......
```

XtraDB has option:

```
innodb_ibuf_active_contract = true (default)
```

Size of insert buffer (< 10000 is no problem)
5.3. Too large history list (rseg)

Any data modifications of InnoDB must keep previous value for the older transactions viewing. It is stored to the rollback segment (rseg). Enough old entries can be removed (purge). Too large size of rseg is also bad for performance.

XtraDB has option for more active purging:

```
innodb_use_purge_thread = 1 (default)
```
Tuning Examples

Tuning benchmark on 16 core server
(32GB RAM: RAID storage)
based on the procedure
Benchmark settings

- Using workload is original TPC-C or TPC-E like
- Initial dataset is chosen around 10GB
- Options for not InnoDB is already set properly (table_cache etc...)
- InnoDB tuning is started from default settings
- Some InnoDB variables are fixed beforehand

```
innodb_file_per_table = true
innodb_data_file_path = ibdata1:10M:autoextend
innodb_flush_log_at_trx_commit = 1
innodb_flush_method = O_DIRECT
innodb_log_buffer_size = 16M
```
1. Basic Tuning

Tuning builtin-InnoDB of MySQL 5.1
TPC-C based workload
Almost Default InnoDB (5.1)

Throughput

Average of Pending IO
reads         : 13.225
writes   (LRU): 10.7917
    (flush list): 0

Entirely IO bound...

Yes! Buffer Pool (capacity for blocks) is too short!

Buffer pool hit rate 872 / 1000
+ Enough Buffer Pool 16G (5.1)

Average of Pending IO
reads : 0.4
writes (LRU): 0
(flush list): 54.975

....Write IO bound

Checkpoint age is controlled to 7MB still

Transaction log size (capacity for modifies) is too short!

Database pages 348170
Modified db pages 15509

Buffer pool hit rate 997 / 1000
Both of “Buffer Pool size” and “Log files size” is important to control capacity of data
TIPS: Buffer Pool and Logfiles

(1) `buf_pool_size` + `log_file_size` × `log_files_in_group` < “allowed memory for mysqld”

(2) Use “innodb_flush_method = O_DIRECT”

The optimal usage of physical memory is...

Datafile pages are cached only in buffer pool.
Whole of transaction log files can be cached by OS.
(To avoid read IO caused by writing log)

(* OS cache miss-hit for the transaction log decreases throughput of the system)
Throughput from cached enough

In the worst case, `doublewrite` line goes to 0 in a while....

In the end, the followings variables are added for base settings of the following tests

- `innodb_buffer_pool_size = 16G`
- `innodb_log_file_size = 1024M`
- `innodb_log_files_in_group = 2`
- `innodb_doublewrite = false`
2. Tuning for TPC-E (16G BP)

Choose binary and settings for 
less modifies and enough memory situation
(2.) Builtin InnoDB 5.1

Throughput

- Hits RW-latch implementation problem!
- *InnoDB-Plugin* or *XtraDB* must be faster

Average cpu% (us + sy): 44.45%

Average of Pending IO:
- reads: 0.025
- writes (LRU): 0
- (flush list): 0

Remarks:
- 4 lock (dictdict0dict0dict.c line 1356)
- 218 lock (btr/btr0sea.c line 139)
- 251 Mutex (btr/btr0sea.c line 139)

Note: less modifies and enough memory
(2.) InnoDB Plugin

Even if 'buf_pool_mutex' is fixed, 'btr_search_latch' may not be fixed enough...

**Average of Pending IO**
- reads : 0.017
- writes (LRU): 0
  - (flush list): 0.392

**Throughput**
- Average cpu% (us + sy): 65.77%
- Normal
- Plugin

Even if 'buf_pool_mutex' is fixed, 'btr_search_latch' may not be fixed enough...

**XtraDB may not be faster**

(less modifies and enough memory)
(2.) XtraDB

Throughput

Average cpu\% (us + sy) 66.36\%

Average of Pending IO reads : 0.03
writes (LRU) : 0
(flush list) : 0

As same as estimated, still 'btr_search_latch' contention.

InnoDB-Plugin is enough
(for less modifies and enough memory)

(less modifies and enough memory)
3. Tuning for TPC-E (3G BP)

Choose binary and settings for less modifies and read IO intensive situation
(3.) Builtin InnoDB 5.1

Average cpu% (us + sy) 16.25%

Average of Pending IO reads : 22.43
writes (LRU) : 4.63
(flush list) : 0

Read IO bound.
(next may be lock implement)

InnoDB-Plugin or XtraDB must be faster

(less modifies and read IO intensive)
(3.) InnoDB Plugin

Throughput

- Average cpu% (us + sy): 30.71%
- Throughput: Average of Pending IO
  - reads: 21.28
  - writes (LRU): 4.08
  - (flush list): 0

54 lock (dict/dict0dict.c line 1569)
153 lock (dict/dict0dict.c line 1569)
165 Mutex (buf/buf0buf.c line 955)
481 lock (dict/dict0dict.c line 1569)
738 lock (btr/btr0sea.c line 170)

Read IO bound.
(next may be buf_pool_mutex)

XtraDB may be same because of IO bound

(less modifies and read IO intensive)
(3.) XtraDB

'buf_pool_mutex' contention was fixed!

Throughput

Average cpu% (us + sy)
30.88%

Average of Pending IO
reads : 22.36
writes (LRU): 1.32
   (flush list): 0

Read IO bound.

InnoDB-Plugin is enough because of IO bound

(plugin and xtradb)

(less modifies and read IO intensive)
3'. Tuning for TPC-E (3G BP) 

(using FusionIO 320GB)

Choose binary and settings for less modifies and read IO intensive situation (on very fast storage)
(3'.) Builtin InnoDB 5.1

Read IO is much faster than ordinarily RAID!!!

Hits RW-latch implementation problem!

InnoDB-Plugin or XtraDB must be faster

Average cpu%
(us + sy) 32.4%

Average of Pending IO
reads : 3.16
writes (LRU): 0.06
(flush list): 0

Throughput

3 Mutex (srv/srv0srv.c line 886)
16 lock (dict/dict0dict.c line 1356)
120 Mutex (btr/btr0sea.c line 139)
180 Mutex (buf/buf0buf.c line 597)
726 lock (btr/btr0sea.c line 139)


(less modifies and read IO intensive [FusionIO])
(3'.) InnoDB Plugin

Read IO is much faster than ordinarily RAID!!!

Average of Pending IO
reads : 2.03
writes (LRU): 4.29
(flush list): 0

buf_pool_mutex' problem
(next must be btr_search_latch)

XtraDB should be faster

Throughput

Average cpu%
(us + sy)
53.97%

15 Mutex (fil/fil0fil.c line 1513)
40 lock (dict/dict0dict.c line 1569)
280 lock (dict/dict0dict.c line 1569)
819 Mutex (buf/buf0buf.c line 955)
6409 lock (btr/btr0sea.c line 170)

Read IO is much faster than ordinarily RAID!!!

(less modifies and read IO intensive [FusionIO])
(3') XtraDB

Read IO is much faster than ordinarily RAID!!!

'buf_pool_mutex'
contention was fixed!

Throughput

Average of Pending IO
reads : 3.46
writes (LRU): 4.73
(flush list): 0.32

'btr_search_latch'
and 'page_hash_latch'

XtraDB is the best.

(but meets the next problem)

Average cpu%
(us + sy)
53.4%
4. Tuning for TPC-C (16G BP)

Choose binary and settings for much modifies and write IO intensive situation
(4.) Built-in InnoDB 5.1

Write IO bound.

InnoDB-Plugin or XtraDB must be faster

(much modifies and write IO intensive)
(4.) InnoDB Plugin

Throughput

Average cpu% (us + sy) 74.58%

Write IO bound and 'log_sys->mutex'

Simply commit may wait log synchronization....

(much modifies and write IO intensive)
(4.) XtraDB

Almost same situation, in the end...

Throughput

Average cpu% (us + sy) 68.48%

Average of Pending IO
reads : 0.08
writes (LRU): 0
(flush list): 27.97

33 Mutex '&kernel_mutex'
60 Mutex '&rseg->mutex'
131 lock '&dict_operation_lock'
258 Mutex '&log_sys->mutex'
1041 lock 'tpcc/order_line'

⚠️: It may be wrong
Putting transaction log in the other storage may help... (not tested yet)

(much modifies and write IO intensive)
(4.) + huge transaction log (8GB)

4GB x 2 logfiles (XtraDB-10~)

innodb_log_file_size = 4G
ingodb_log_files_in_group = 2

Write IO seems reduced and seems better “little bit”

Average cpu%
(us + sy)
74.08%

Throughput

It may depend on log serialization

It may be logical limit scale for this workload, in the end...

(much modifies and write IO intensive)
4'. Tuning for TPC-C (16G BP) 
*(using FusionIO 320GB)*

Choose binary and settings for much modifies and write IO intensive situation (on very fast (write IO?) storage)
(4'.) Builtin InnoDB 5.1

Almost same to (4.)

Throughput

Average cpu% (us + sy) 46.64%

Average of Pending IO reads : 0.02
writes (LRU) : 0
(flush list) : 24.56

Write IO bound.

InnoDB-Plugin or XtraDB must be faster

(much modifies and write IO intensive [FusionIO])
(4').) InnoDB Plugin

Almost same to (4.)

Throughput

Average cpu% (us + sy): 73.58%

Write IO bound and 'buf_pool_mutex' (weak)
(but next is 'log_sys->mutex')

(46) InnoDB Plugin

Throughput

Average of Pending IO
reads: 0.01
writes (LRU): 0
(flush list): 21.84

Write IO bound and 'buf_pool_mutex' (weak)
(but next is 'log_sys->mutex')

(much modifies and write IO intensive [FusionIO])
Almost same to (4.)

Throughput

Average cpu% (us + sy) 66.26%

Average of Pending IO
reads : 0
writes (LRU): 0
flush list: 41.63

(much modifies and write IO intensive [FusionIO])
Almost same to (4.)

In the end, Write IO is not so much faster than ordinarily RAID..?

SSD doesn't solve write IO bound of RAID and

It may be logical limit scale for this workload, in the end...

(much modifies and write IO intensive [FusionIO])
5. Tuning for TPC-C (3G BP)

Choose binary and settings for much modifies and read IO intensive situation
(5.) Built-in InnoDB 5.1

Throughput

Average cpu% (us + sy) 12.38%

Average of Pending IO
reads : 20.27
writes (LRU): 27.82
(flush list): 8.2

Complete IO bound

InnoDB-Plugin or XtraDB must be faster

(much modifies and read IO intensive)
(5.) InnoDB Plugin

Throughput

Average cpu% (us + sy) 14.13%

Average of Pending IO reads: 8.83
writes (LRU): 27.95
(flush list): 41.78

Complete IO bound (Write IO is stronger?)

XtraDB may be same because of IO bound?

(much modifies and read IO intensive)
(5.) XtraDB

Small difference...

XtraDB line doesn't decline by Insert Buffer Size growing.

Average of Pending IO
reads : 22.43
writes (LRU): 72.73
(flush list): 8.1

Throughput

Average cpu% (us + sy) 17.4%

Insert Buffer Size

(much modifies and read IO intensive)
5'. Tuning for TPC-C (3G BP) *(using FusionIO 320GB)*

Choose binary and settings for much modifies and read IO intensive situation (on very fast storage)
(5'.) Builtin InnoDB 5.1

Read IO is **much faster** than ordinarily RAID!!!

Throughput

```
.....
31 lock dict/dict0dict.c line 1356
53 lock dict/dict0dict.c line 728
91 lock dict/dict0dict.c line 1356
184 Mutex srv/srv0srv.c line 886
187 Mutex buf/buf0buf.c line 597 :
```

**Average of Pending IO**
- reads : **2.14**
- writes (LRU): **10.2**
- (flush list): **2.13**

Write IO bound and 'buf_pool_mutex' (?)

*Plugin* should be faster.

*XtraDB* is more faster (?)

(much modifies and read IO intensive [FusionIO])
(5'.) InnoDB Plugin

Read IO is much faster than ordinarily RAID!!

Throughput

Average cpu% (us + sy)
47.82%

Average of Pending IO
reads : 1.02
writes (LRU): 18.17
(flush list): 25.42

Write IO bound and 'buf_pool_mutex'

(5'.) InnoDB Plugin

Read IO is much faster than ordinarily RAID!!

Throughput

Average cpu% (us + sy)
47.82%

Average of Pending IO
reads : 1.02
writes (LRU): 18.17
(flush list): 25.42

Write IO bound and 'buf_pool_mutex'

(much modifies and read IO intensive [FusionIO])

....

325 Mutex trx/trx0rseg.c line 210
365 lock dict/dict0dict.c line 622
488 Mutex buf/buf0buf.c line 955
634 Mutex log/log0log.c line 776
2679 lock dict/dict0dict.c line 1569
(5'.) XtraDB

Read IO is much faster than ordinarily RAID!!!

Read IO is much faster than ordinarily RAID!!!

Throughput

Average cpu% (us + sy) 52.48%

Average of Pending IO reads : 9.65
writes (LRU): 58.41
(flush list): 22.83

Write IO bound and 'page_hash_latch' (?)
(next may be 'rseg->mutex')

(much modifies and read IO intensive [FusionIO])
When are Plugin or XtraDB needed?

InnoDB-Plugin:
- IO bound for RAID or SSD
- 'btr_search_latch' contention

XtraDB:
- IO bound for RAID or SSD
- 'btr_search_latch' contention
- IO intensive workload for fast storage
  ("Hot" data is larger than buffer pool)
  -> 'buf_pool_mutex' contention is solved
Other Tips

Tips for the other variables to be tuned
(no results in this session,
excuse for no time to prepare....)
'innodb_thread_concurrency' is not negative variable! It is **good** to retain throughput for many threads.

But, you should tune it by yourself, if you want “the best”.

* if ([the best innodb_thread_concurrency for you] ≥ [CPU cores the server has])

It means “the InnoDB scales enough for you”

*Don't blame InnoDB :-)*
Other IO tunes for SSD

**innodb_flush_neighbors** [true (default)]:
- SSD has no advantage for neighbor access
- “false” may be good for some cases of SDD

**innodb_read_ahead** [“linear” (default)]:
- SSD has no advantage for sequential reading
- “none” may be good for some case of SDD
Experimental tunes for SSD (XtraDB~)

`innodb_fast_checksum [false (default)]:`
- 4-bytes word based calculation for page instead of 1-byte based, if true is set.
- It may speed up the each IO for datafile.

`innodb_page_size [16K (default)]:`
- Smaller size may limit chunk of flushing.
- 4K or 8K are alternatives.
TODO for XtraDB

For more performance, scalability, usefulness
Mutex/Latch contentions (for fast storage)
- 'btr_search_latch'
- 'page_hash_latch'

Recovery speed ('scanning log' phase)
- Several GB checkpoint age recovery is very slow and it seems to be CPU bound of 1 CPU
- ('applying log' phase was already fixed (XtraDB))

Range optimizer (access tree really)
- Not-unique search accesses index for estimation
- Change to statistic based optimization
Questions?

Thank you for coming!

Let's happy tuning
with deeper understanding
and more accurate analysis
to find next step of InnoDB engine!

We need customers
for next research and implementation (in TODO page).
Please contact us, if you can pay for them.
(5'.) MySQL 5.5.4 (appendix)

16 seems good for 16 cores

```
inoddbuf_pool_instances = 16
```

**But '16' seems more sensitive to Insert Buffer Size(***)...**

**Throughput**

- **Average cpu %**
  - (us + sy)
  - 62.36%

**Average of Pending IO**
- **reads**: 2.65
- **writes (LRU)**: 45.36
- **(flush list)**: 1.32

**Write IO bound and 'kernel_mutex' (and 'ibuf_mutex' ?)**

*(*)**Insert Buffer Size**

(***) MySQL 5.5.4 (appendix)

(much modifies and read IO intensive [FusionIO])