



PERCONA  
Performance Consulting Experts

---

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

MySQL Conference & Expo 2010

Yasufumi Kinoshita

Senior Performance Engineer  
Percona Inc.

[MySQLPerformanceBlog.com](http://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 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 SSD)?”*

“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

```
#!/bin/sh
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
rm /tmp/tmp1.txt /tmp/tmp2.txt
```



```
.....
 4 lock on RW-latch at 0x7fb86b2c9138 created in file dict/dict0dict.c line 1356
 6 lock on RW-latch at 0x7fb86b2c4138 created in file dict/dict0dict.c line 1356
12 lock on RW-latch at 0x7fb86b2d9538 created in file dict/dict0dict.c line 1356
20 lock on RW-latch at 0x7fb86b2db138 created in file dict/dict0dict.c line 1356
22 Mutex at 0x7fb86b28f0e0 created file btr/btr0sea.c line 139
30 lock on RW-latch at 0x7fb86b2ba938 created in file dict/dict0dict.c line 1356
36 lock on RW-latch at 0x7fb86b2bad38 created in file dict/dict0dict.c line 1356
71 Mutex at 0x7fb86b28ecb8 created file buf/buf0buf.c line 597
164 lock on RW-latch at 0x7fb86b28f0b8 created in file btr/btr0sea.c line 139
```

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**

```

.....
67 Mutex at 0xd26aa0 created file ibuf/ibuf0ibuf.c line 467
72 lock on RW-latch at 0x7fe6201024f0 created in file dict/dict0dict.c line 1569
118 lock on RW-latch at 0x80ed6c0 created in file btr/btr0sea.c line 170
221 lock on RW-latch at 0x7fe62010e1b0 created in file dict/dict0dict.c line 1569
325 Mutex at 0x80f9878 created file trx/trx0rseg.c line 210
365 lock on RW-latch at 0xd2c900 created in file dict/dict0dict.c line 622
488 Mutex at 0xd2c840 created file buf/buf0buf.c line 955
634 Mutex at 0x80eeb30 created file log/log0log.c line 776
2679 lock on RW-latch at 0x7fe62010d960 created in file dict/dict0dict.c line 1569

```



```

.....
67 'ibuf_mutex'
72 'index->lock'
118 'btr_search_latch'
221 'index->lock'
325 'rseg->mutex'
365 'index->lock'
488 'buf_pool_mutex'
634 'log_sys->mutex'
2679 'index->lock'

```

This contention should cause the others. 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,  
.....
```

Size of insert buffer  
(< 10000 is no problem)

XtraDB has option:

`innodb_ibuf_active_contract = true` (default)

# 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.

```
.....  
-----  
TRANSACTIONS  
-----  
Trx id counter 17F6C0  
Purge done for trx's n:o < 17F6BF undo n:o < 0  
History list length 17  
.....
```

(< 100000 is no problem)

**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

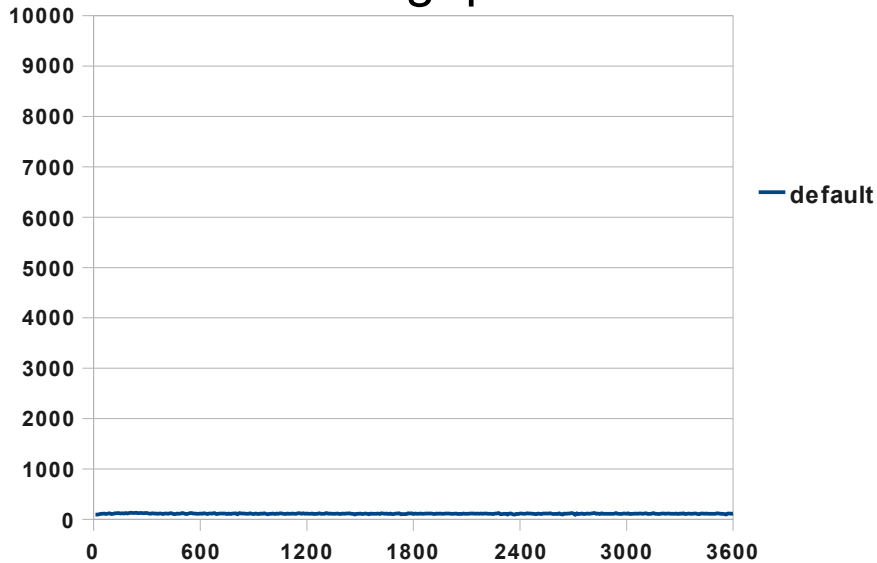
```
innodb_file_per_table = true
innodb_data_file_path = ibdata1:10M:autoextend
innodb_flush_log_at_trx_commit = 1
innodb_flush_method = O_DIRECT
innodb_log_buffer_size = 16M
```

# 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
```

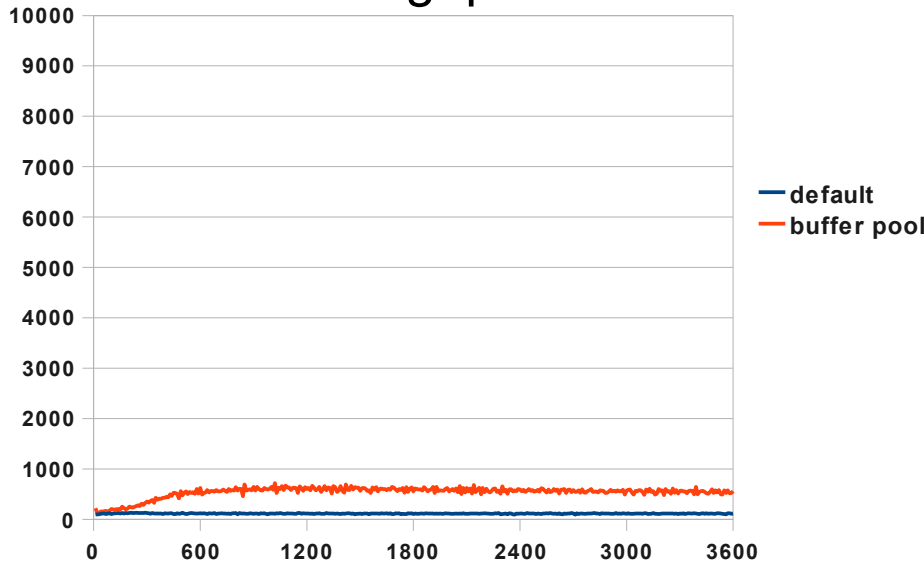
Entirely IO bound...

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

```
.....
-----
BUFFER POOL AND MEMORY
-----
.....
Buffer pool hit rate 872 / 1000
```

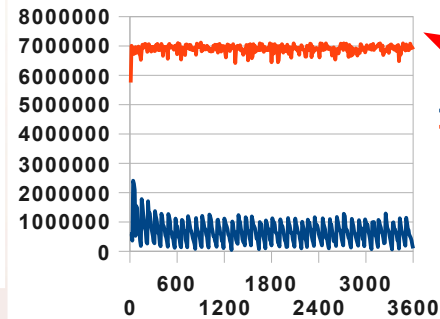
# + Enough Buffer Pool 16G (5.1)

Throughput



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

....Write IO bound



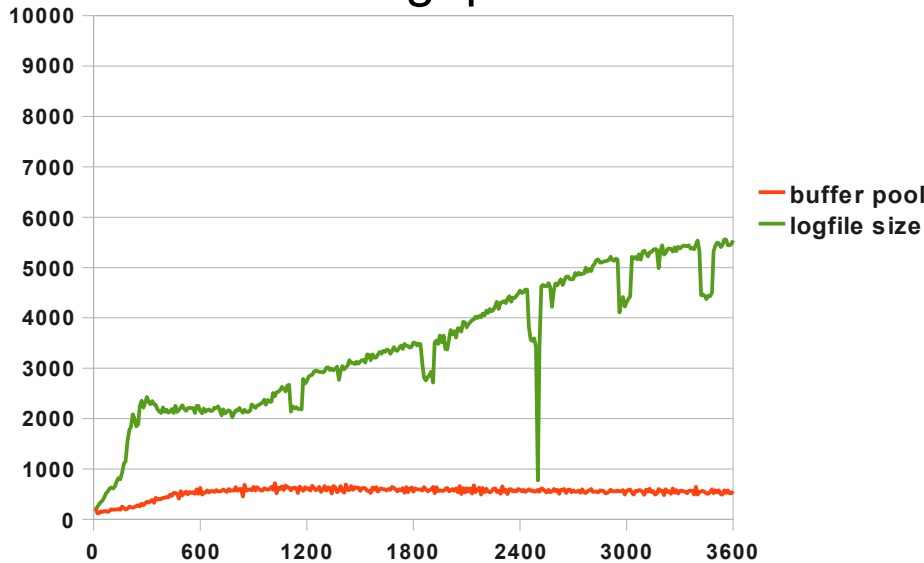
Checkpoint age is controlled to 7MB still

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

```
.....
-----
BUFFER POOL AND MEMORY
-----
.....
Database pages      348170
Modified db pages  15509
.....
Buffer pool hit rate 997 / 1000
```

# + Log files size 2G (5.1)

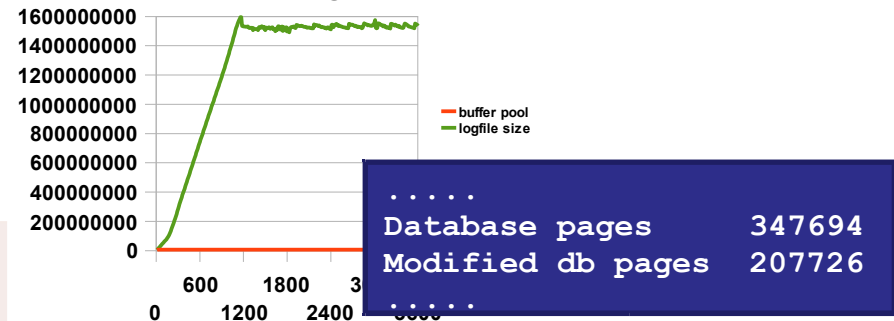
Throughput



```
Average of Pending IO
reads           : 0.4
writes (LRU)    : 0
(flush list)   : 11.425
```

still Write IO bound

Checkpoint Age



*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` x `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.*

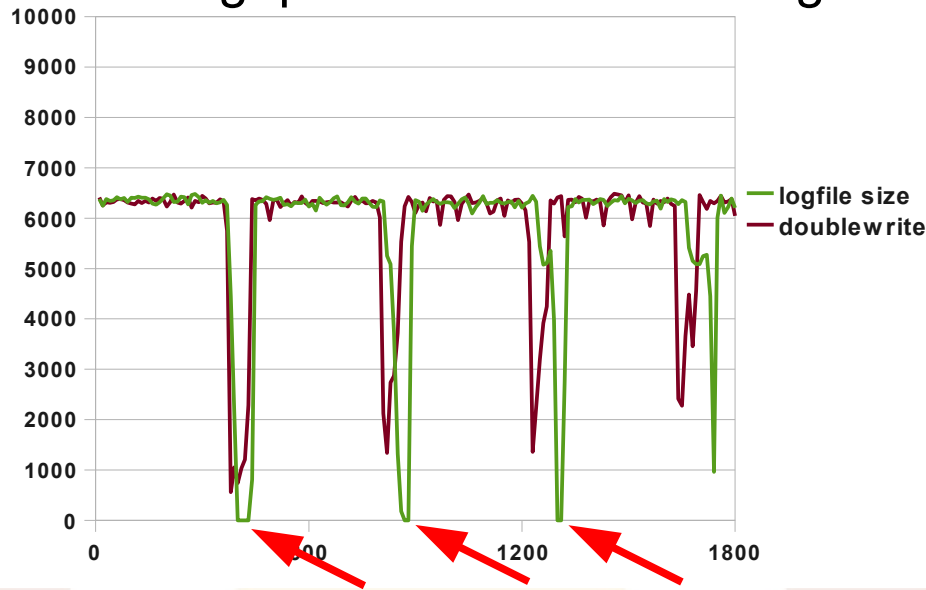
*Whole of transaction log files can be cached by OS.*

*(To avoid read IO caused by writing log)*

*(\* OS cache miss-hit for the transaction log decreases throughput of the system)*

# + (disable doublewrite) (5.1)

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)

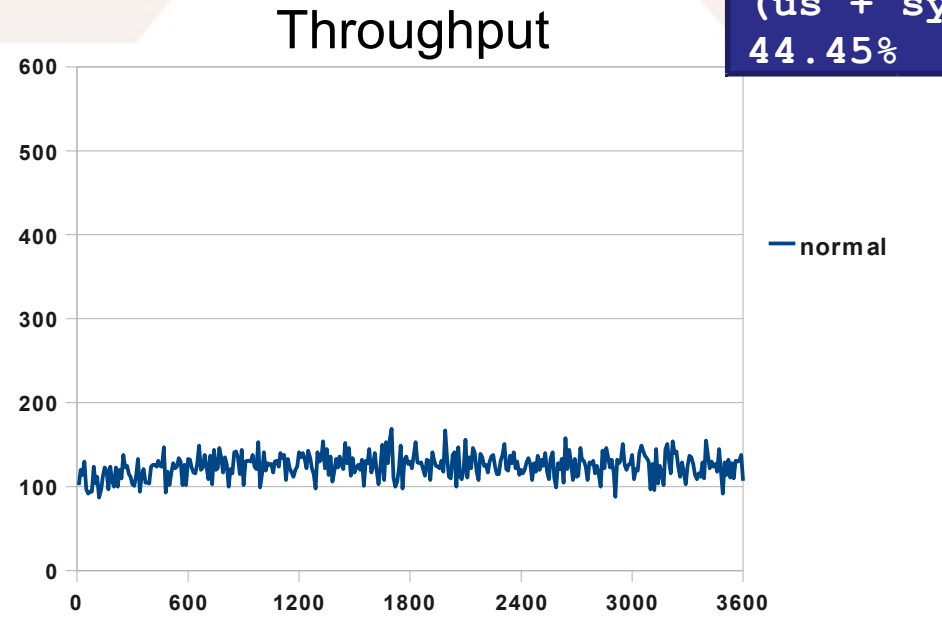
Choose binary and settings for  
**less modifies** and **enough memory** situation

# (2.) Builtin InnoDB 5.1

```
.....  
4 lock (dict/dict0dict.c line 1356)  
218 lock (btr/btr0sea.c line 139)  
251 Mutex (btr/btr0sea.c line 139)
```

Average cpu%  
(us + sy)  
44.45%

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



Hits RW-latch  
implementation problem!

*InnoDB-Plugin* or *XtraDB*  
must be faster

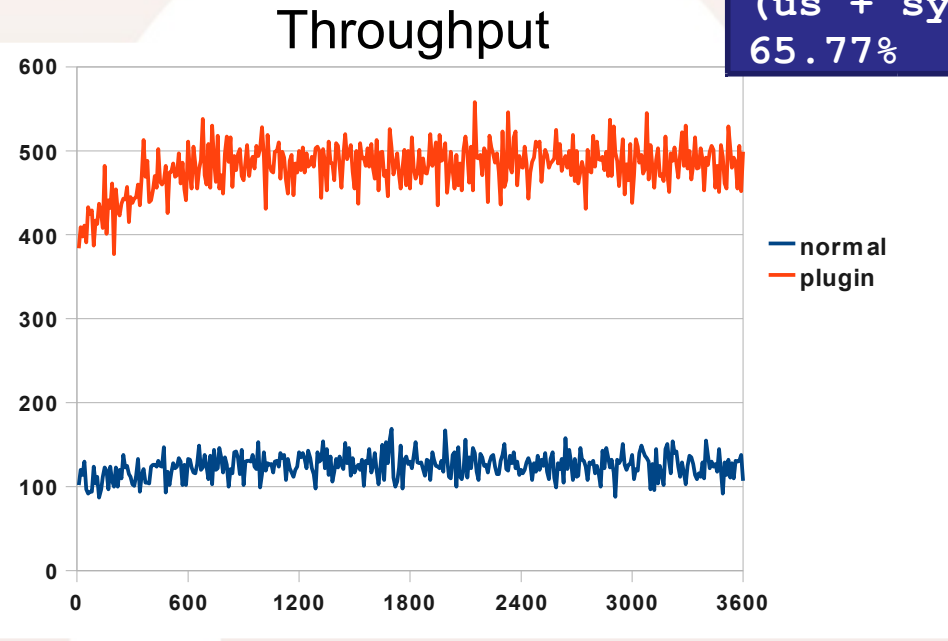
(less modifies and enough memory)

# (2.) InnoDB Plugin

```
.....  
16 Mutex (buf/buf0buf.c line 955)  
16 lock (dict/dict0dict.c line 1569)  
156 lock (btr/btr0sea.c line 170)
```

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

Average cpu%  
(us + sy)  
65.77%



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)

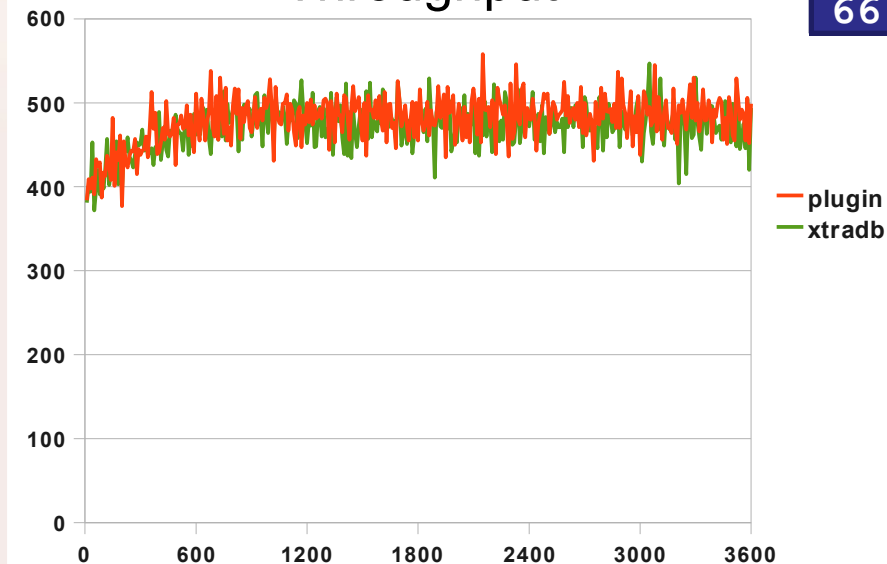
# (2.) XtraDB

```
.....  
3 Mutex '&log_sys->mutex'  
3 Mutex '&flush_list_mutex'  
4 lock 'tpce/trade'  
26 lock 'tpce/trade'  
234 lock '&btr search latch'
```

Average cpu%  
(us + sy)  
66.36%

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

Throughput



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

```

.....
22 Mutex (btr/btr0sea.c line 139)
30 lock (dict/dict0dict.c line 1356)
36 lock (dict/dict0dict.c line 1356)
71 Mutex (buf/buf0buf.c line 597)
164 lock (btr/btr0sea.c line 139)

```

```

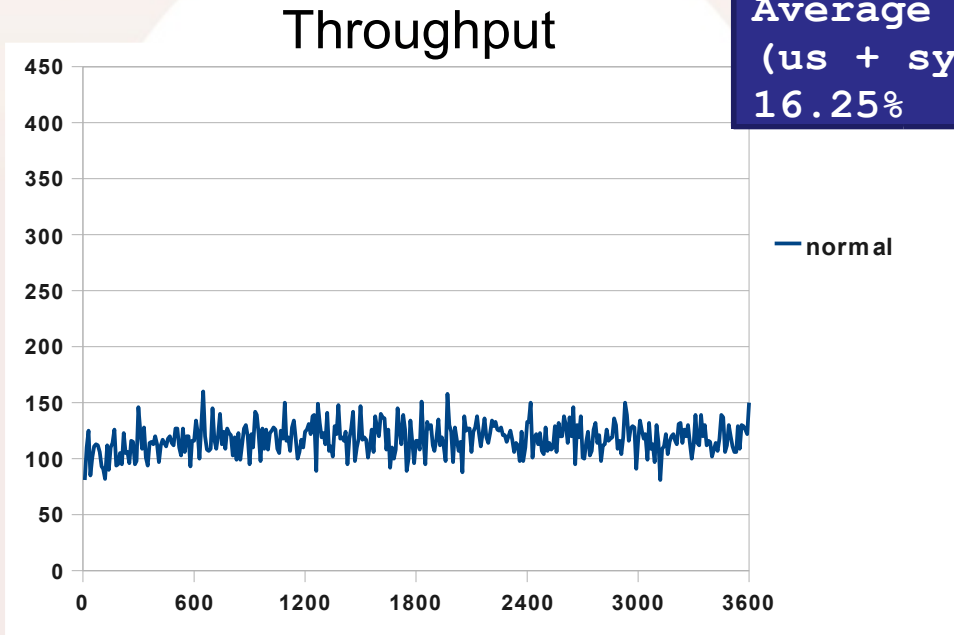
Average of Pending IO
reads           : 22.43
writes (LRU)   : 4.63
(flush list)  : 0

```

```

Average cpu%
(us + sy)
16.25%

```



Read IO bound.  
(next may be lock implement)

*InnoDB-Plugin* or *XtraDB*  
must be faster

(less modifies and read IO intensive)

# (3.) InnoDB Plugin

```

.....
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)

```

```

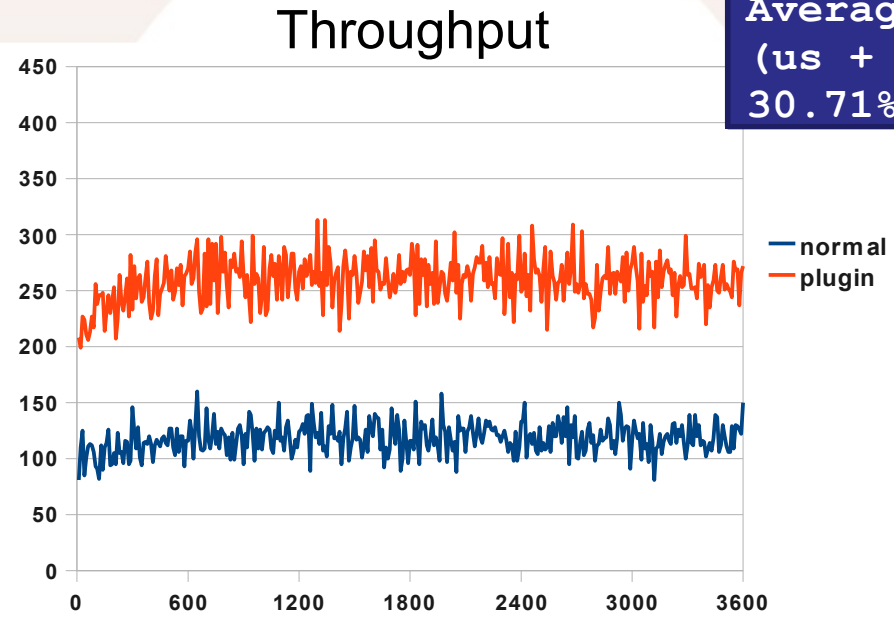
Average of Pending IO
reads           : 21.28
writes (LRU)   : 4.08
(flush list)   : 0

```

```

Average cpu%
(us + sy)
30.71%

```



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!

```

.....
33 lock 'tpce/cash_transaction'
119 lock 'tpce/trade'
149 lock 'tpce/trade'
505 lock 'tpce/trade'
833 lock '&btr_search_latch'

```

```

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

```

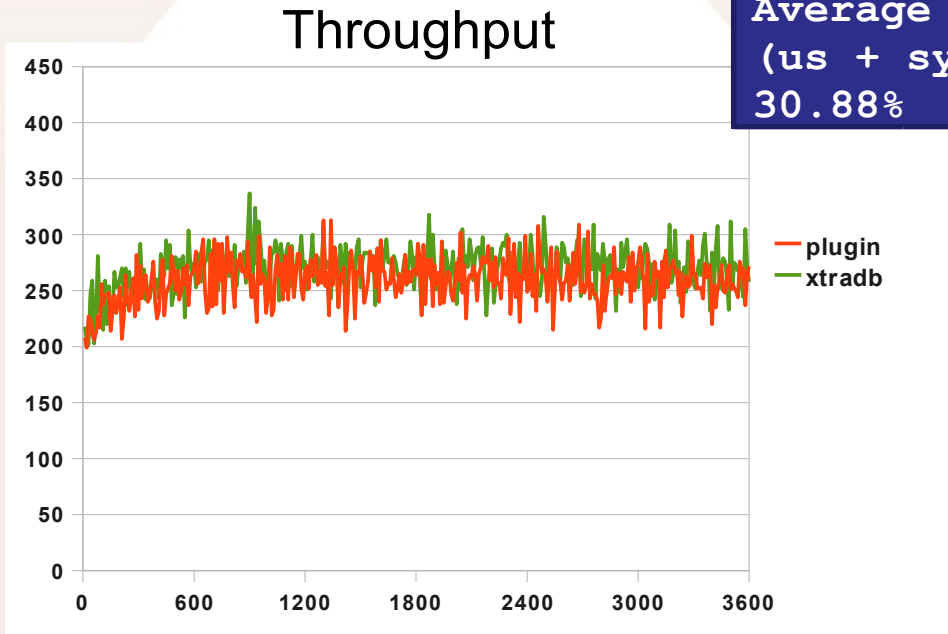
```

Average cpu%
(us + sy)
30.88%

```

Read IO bound.

*InnoDB-Plugin is enough because of IO bound*



(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!!!

```

.....
  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)

```

```

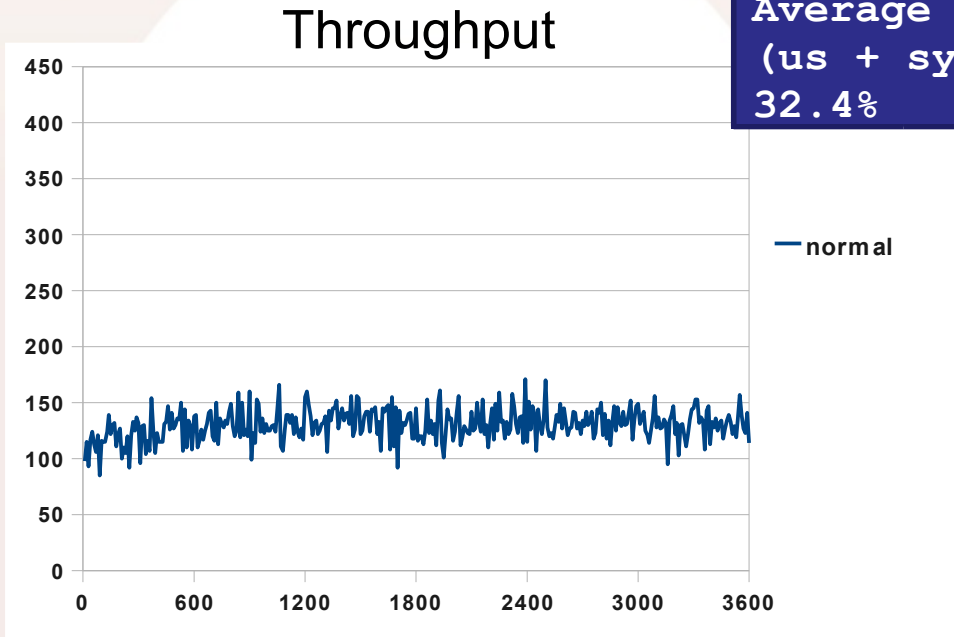
Average of Pending IO
reads           : 3.16
writes (LRU)    : 0.06
(flush list)   : 0

```

```

Average cpu%
(us + sy)
32.4%

```



Hits RW-latch implementation problem!

*InnoDB-Plugin* or *XtraDB* must be faster

(less modifies and read IO intensive [FusionIO])

# (3'.) InnoDB Plugin

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

```

.....
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)

```

```

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

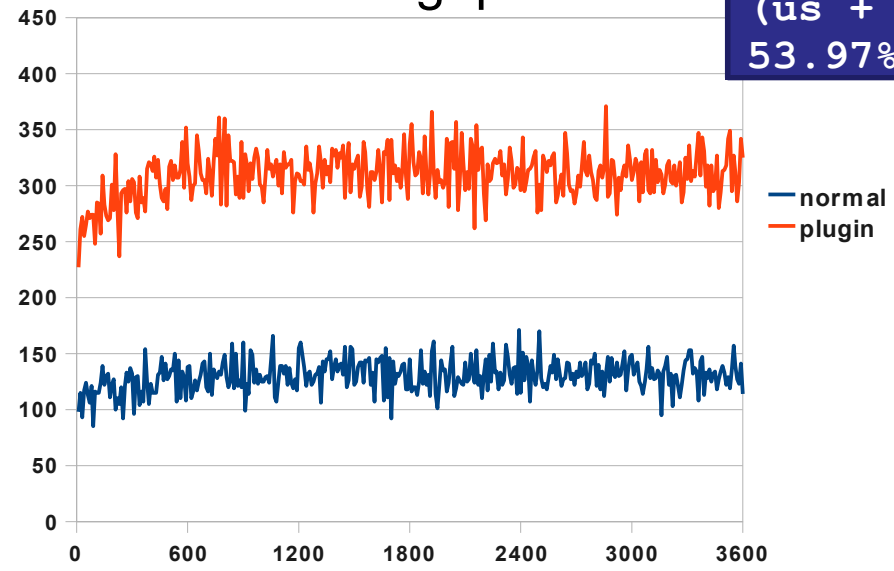
```

```

Average cpu%
(us + sy)
53.97%

```

Throughput



'buf\_pool\_mutex' problem (next must be btr\_search\_latch)

*XtraDB should be faster*

(less modifies and read IO intensive [FusionIO])

# (3'.) XtraDB

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

'buf\_pool\_mutex' contention was fixed!

```

.....
31 Mutex '&kernel mutex'
107 lock '&page hash latch'
118 lock 'tpce/trade'
166 lock 'tpce/trade'
6617 lock '&btr search latch'

```

```

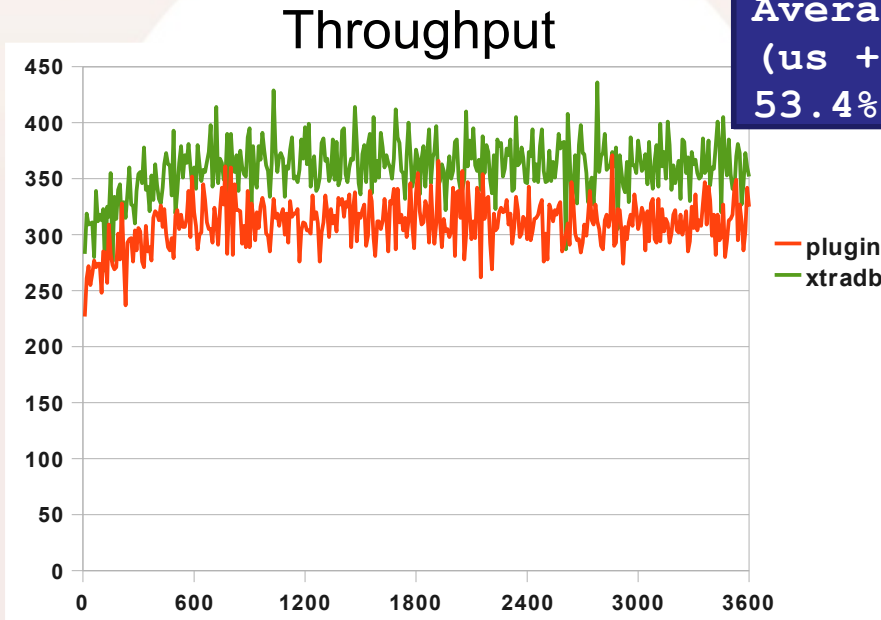
Average of Pending IO
reads           : 3.46
writes (LRU)    : 4.73
(flush list)   : 0.32

```

```

Average cpu%
(us + sy)
53.4%

```



'btr\_search\_latch'  
and 'page\_hash\_latch'

*XtraDB is the best.*  
(but meets the next problem)

(less modifies and read IO intensive [FusionIO])

## 4. Tuning for TPC-C (16G BP)

Choose binary and settings for  
**much modifies** and **write IO intensive** situation

# (4.) Builtin InnoDB 5.1

```

.....
16 Mutex log/log0log.c line 738
34 lock dict/dict0dict.c line 1356
136 lock dict/dict0dict.c line 728
237 Mutex srv/srv0srv.c line 886
274 lock dict/dict0dict.c line 1356

```

```

Average of Pending IO
reads           : 0.13
writes (LRU)    : 0
(flush list)   : 35.25

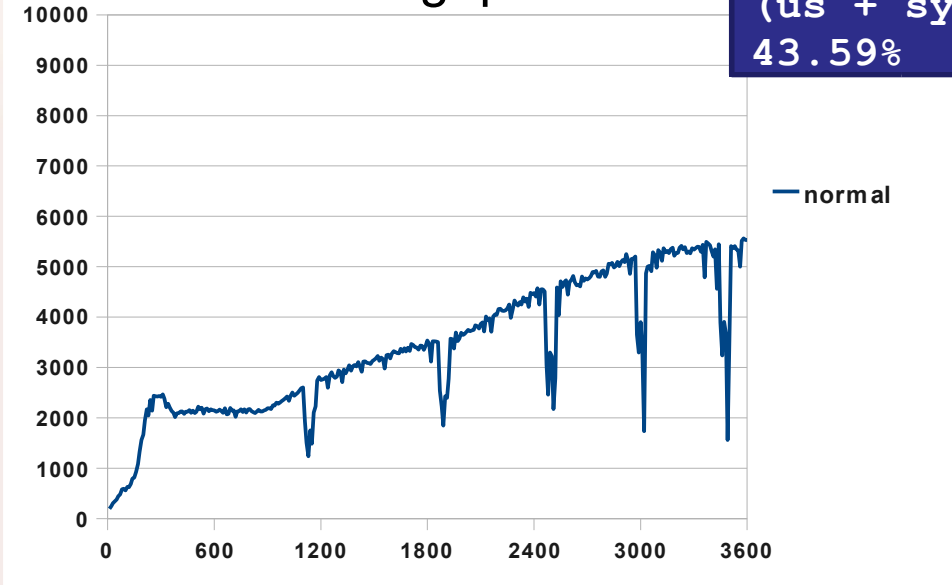
```

```

Average cpu%
(us + sy)
43.59%

```

### Throughput



Write IO bound.

*InnoDB-Plugin* or *XtraDB* must be faster

(much modifies and write IO intensive)

# (4.) InnoDB Plugin

```

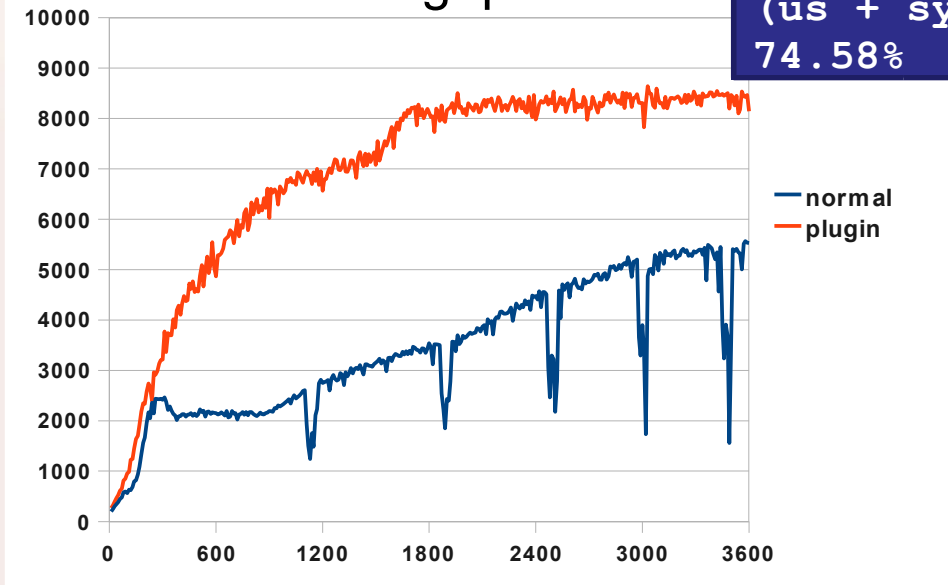
.....
59 Mutex srv/srv0srv.c line 945
59 Mutex trx/trx0rseg.c line 210
69 Mutex log/log0log.c line 776
97 lock dict/dict0dict.c line 622
1247 lock dict/dict0dict.c line 1569

```

Average cpu%  
(us + sy)  
74.58%

Average of Pending IO  
reads : 0.06  
writes (LRU): 0  
(flush list): 28.38

### Throughput



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...*

```

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

```

```

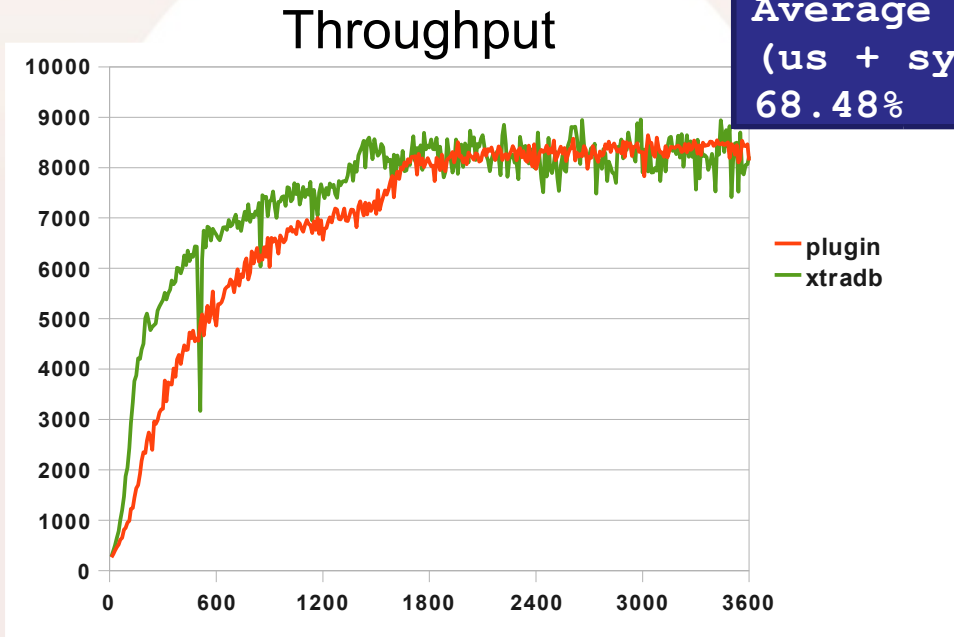
Average of Pending IO
reads           : 0.08
writes (LRU)    : 0
(flush list)   : 27.97

```

```

Average cpu%
(us + sy)
68.48%

```



**!** :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~)

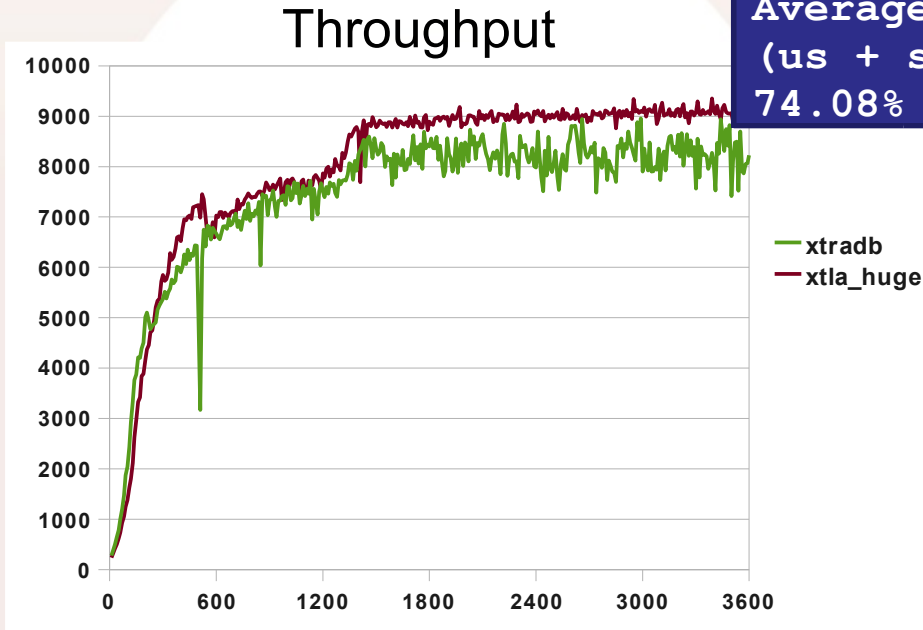
```
innodb_log_file_size = 4G
innodb_log_files_in_group = 2
```

*Write IO seems reduced and seems better "little bit"*

```
.....
44 lock '&new_index->lock'
86 Mutex '&kernel_mutex'
136 lock '&dict_operation_lock'
303 Mutex '&log_sys->mutex'
967 lock '&new_index->lock'
```

```
Average of Pending IO
reads          : 0.06
writes (LRU)   : 0
(flush list)  : 14.63
```

```
Average cpu%
(us + sy)
74.08%
```



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.)

```

.....
23 lock dict/dict0dict.c line 728
29 lock dict/dict0dict.c line 1356
42 Mutex log/log0log.c line 738
107 lock dict/dict0dict.c line 1356
333 Mutex srv/srv0srv.c line 886

```

```

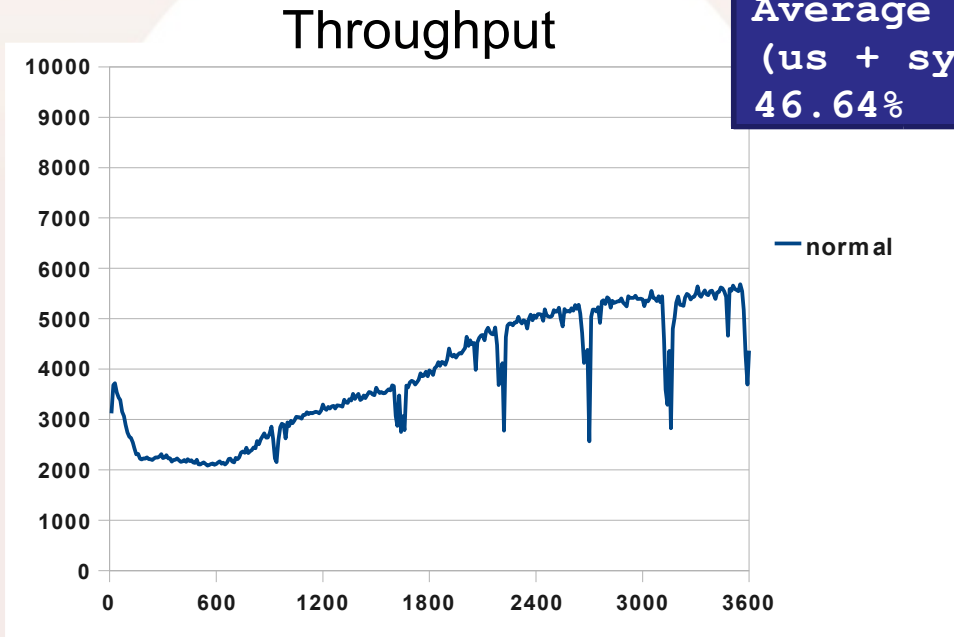
Average of Pending IO
reads           : 0.02
writes (LRU)   : 0
(flush list)   : 24.56

```

```

Average cpu%
(us + sy)
46.64%

```



Write IO bound.

*InnoDB-Plugin* or *XtraDB* must be faster

(much modifies and write IO intensive [FusionIO])

# (4'.) InnoDB Plugin

Almost same to (4.)

```

.....
62 lock dict/dict0dict.c line 622
65 Mutex srv/srv0srv.c line 945
85 Mutex buf/buf0buf.c line 955
93 Mutex log/log0log.c line 776
353 lock dict/dict0dict.c line 1569

```

```

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

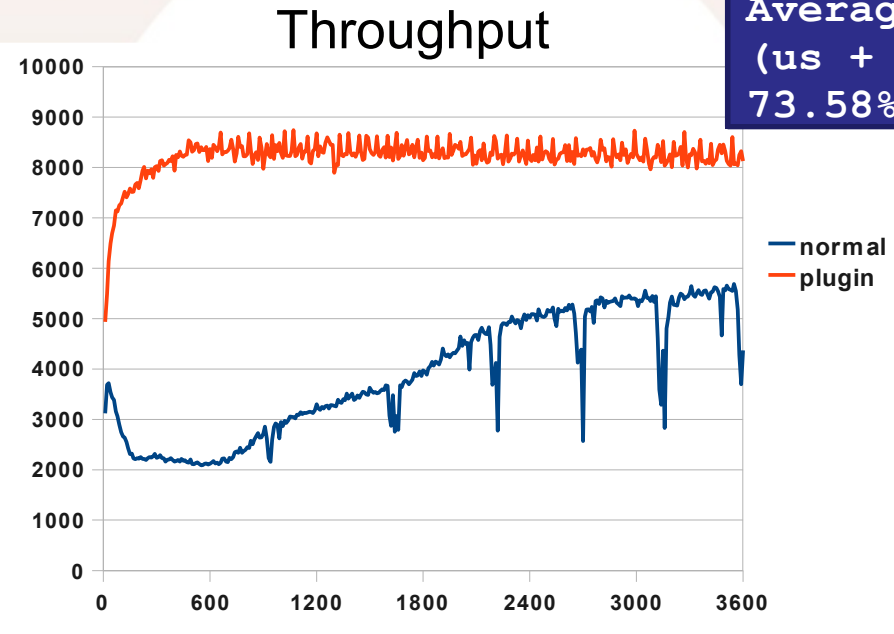
```

```

Average cpu%
(us + sy)
73.58%

```

Write IO bound  
 and 'buf\_pool\_mutex' (weak)  
 (but next is 'log\_sys->mutex')



(much modifies and write IO intensive [FusionIO])

# (4'.) XtraDB

Almost same to (4.)

```

.....
35 Mutex '&rseg->mutex'
43 lock '&dict_operation_lock'
47 Mutex '&kernel_mutex'
324 Mutex '&log_sys->mutex'
372 lock 'tpcc/order_line'

```

```

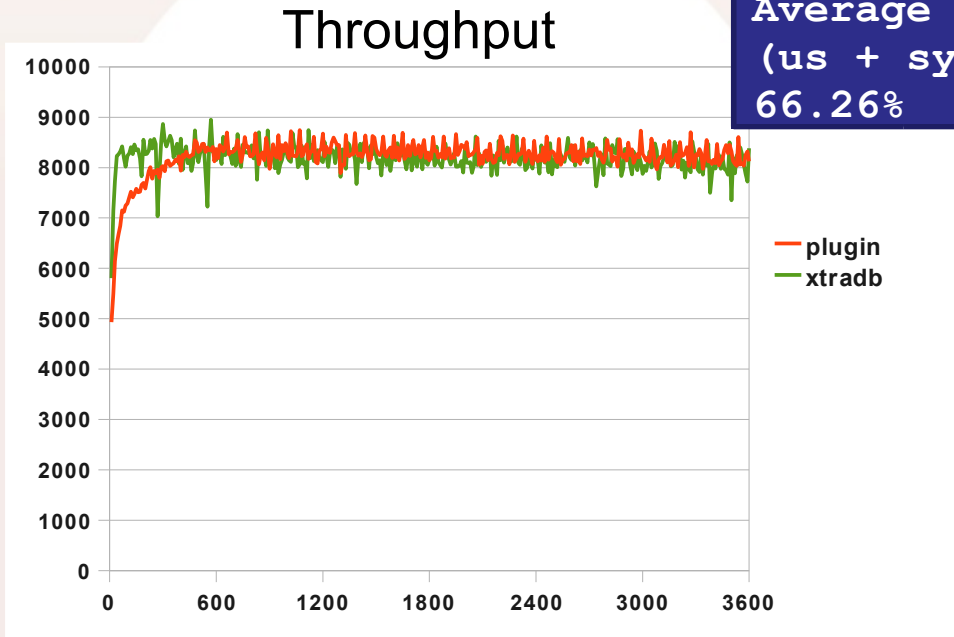
Average of Pending IO
reads           : 0
writes (LRU)   : 0
(flush list)  : 41.63

```

```

Average cpu%
(us + sy)
66.26%

```



(much modifies and write IO intensive [FusionIO])

# (4'.) + huge transaction log (8GB)

*Almost same to (4.)*

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

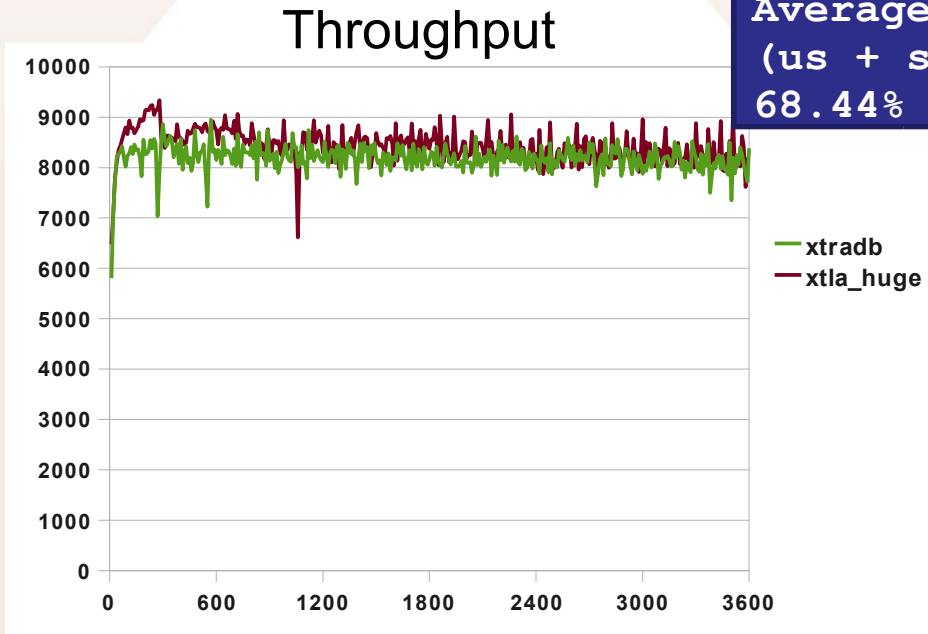
```

.....
37 lock '&dict_operation_lock'
40 Mutex '&rseg->mutex'
46 Mutex '&kernel_mutex'
418 Mutex '&log_sys->mutex'
451 lock '&new_index->lock'

```

Average cpu%  
(us + sy)  
68.44%

Average of Pending IO  
reads : 0.02  
writes (LRU): 0  
(flush list): 16.58



*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.) Builtin InnoDB 5.1

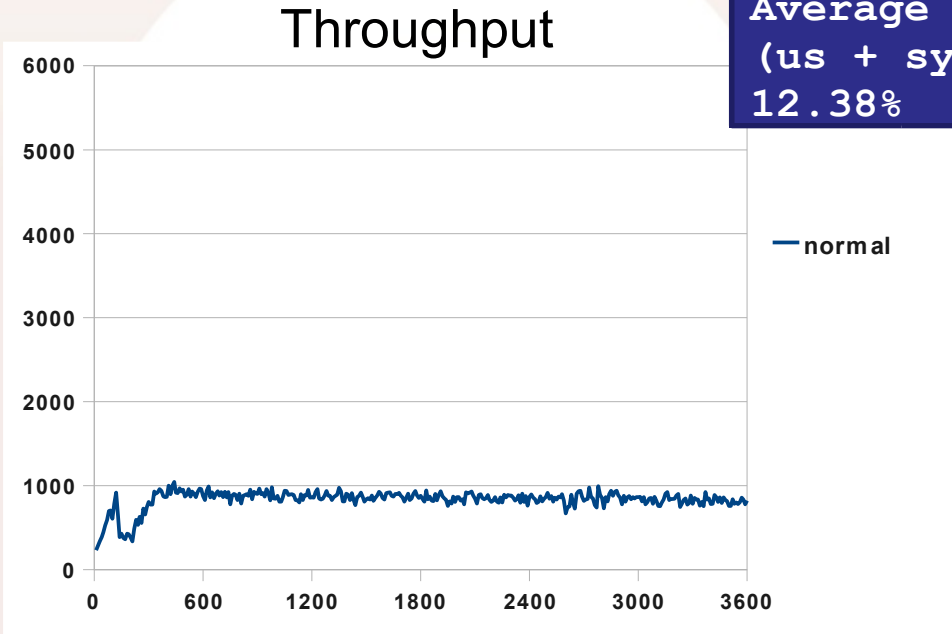
```

.....
17 Mutex srv/srv0srv.c line 886
25 lock dict/dict0dict.c line 1356
50 Mutex buf/buf0buf.c line 597
93 lock dict/dict0dict.c line 728
135 lock dict/dict0dict.c line 1356

```

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

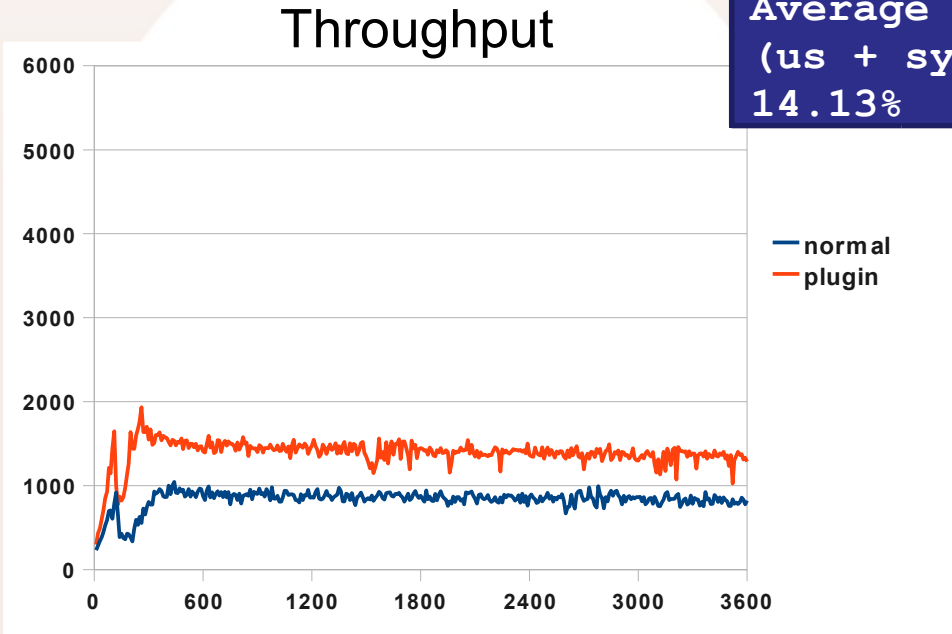
```

. . . . .
77  Mutex buf/buf0buf.c line 955
226 Mutex ibuf/ibuf0ibuf.c line 467
274 lock dict/dict0dict.c line 1569
890 lock dict/dict0dict.c line 622
2118 lock dict/dict0dict.c line 1569

```

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

Average cpu%  
(us + sy)  
14.13%



(much modifies and read IO intensive)

Complete IO bound  
(Write IO is stronger?)  
*XtraDB* may be same  
because of IO bound?

# (5.) XtraDB

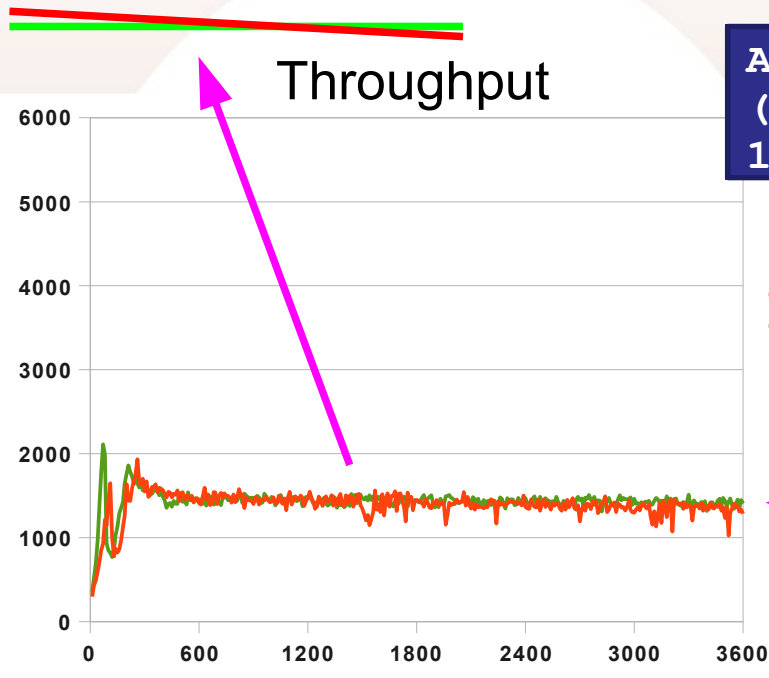
Small difference...

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

```

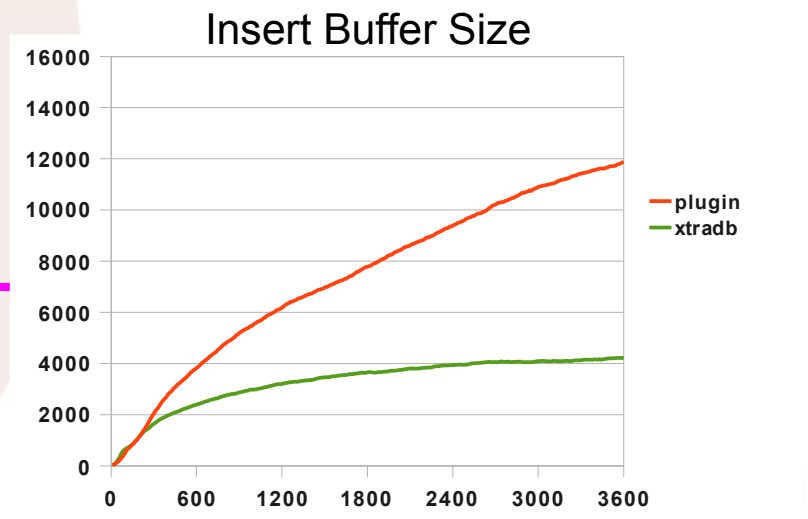
.....
48 Mutex '&log_sys->mutex'
66 lock '&new_index->lock'
186 lock '&new_index->lock'
630 lock '&dict_operation_lock'
1015 lock '&new_index->lock'

```



Average cpu%  
(us + sy)  
17.4%

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



(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!!!

```

.....
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

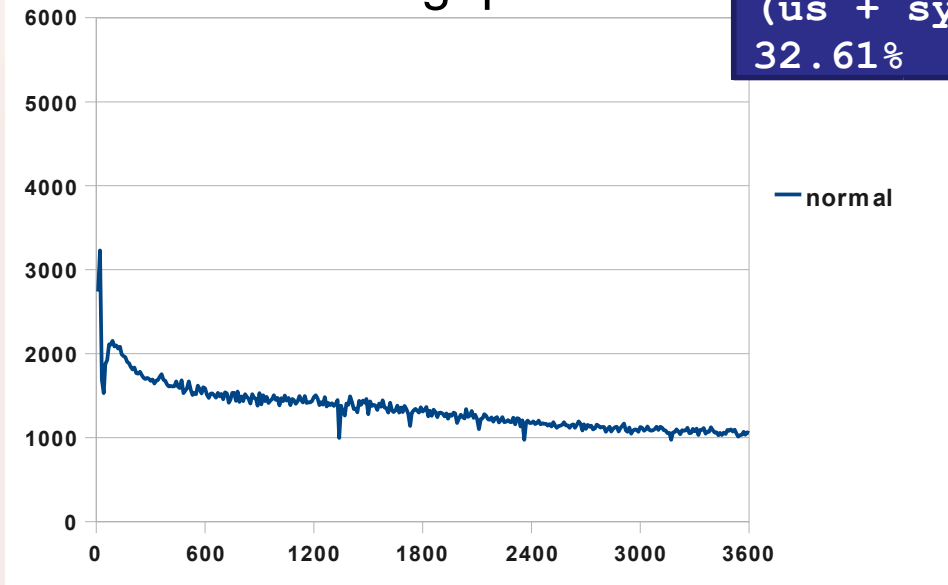
```

```

Average cpu%
(us + sy)
32.61%

```

Throughput



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!!!

```

.....
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

```

```

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

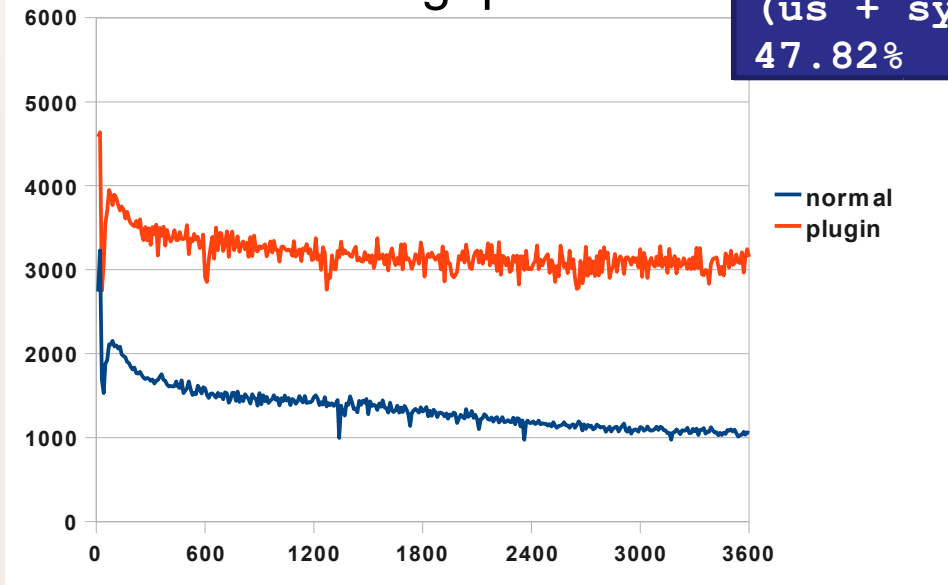
```

```

Average cpu%
(us + sy)
47.82%

```

### Throughput



Write IO bound and 'buf\_pool\_mutex'

(much modifies and read IO intensive [FusionIO])

# (5'.) XtraDB

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

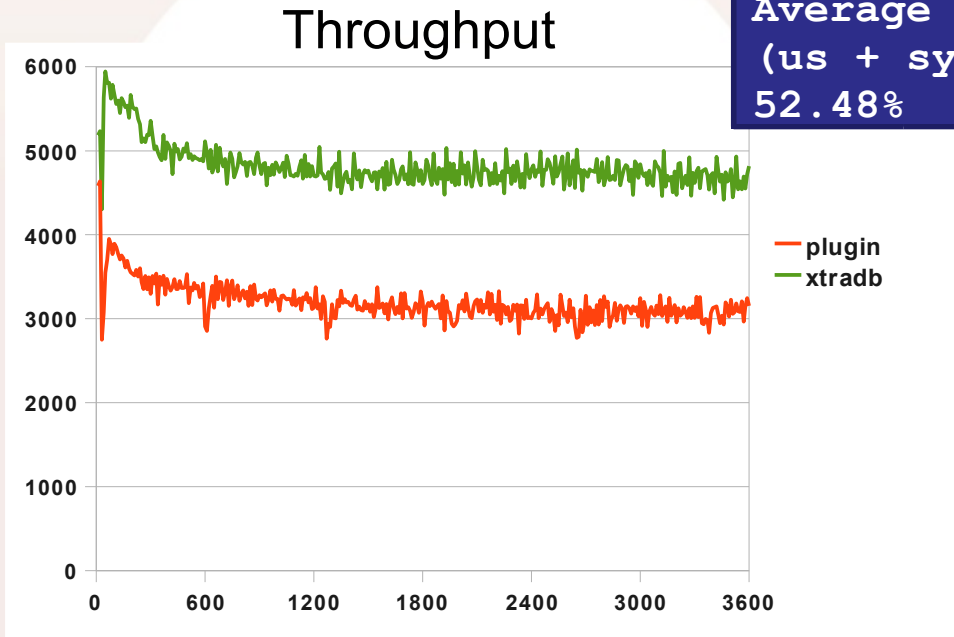
```

.....
186 Mutex '&rseg->mutex'
186 lock '&page_hash_latch'
250 lock 'tpcc/order_line'
564 lock '&dict_operation_lock'
1271 lock 'tpcc/order_line'

```

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*

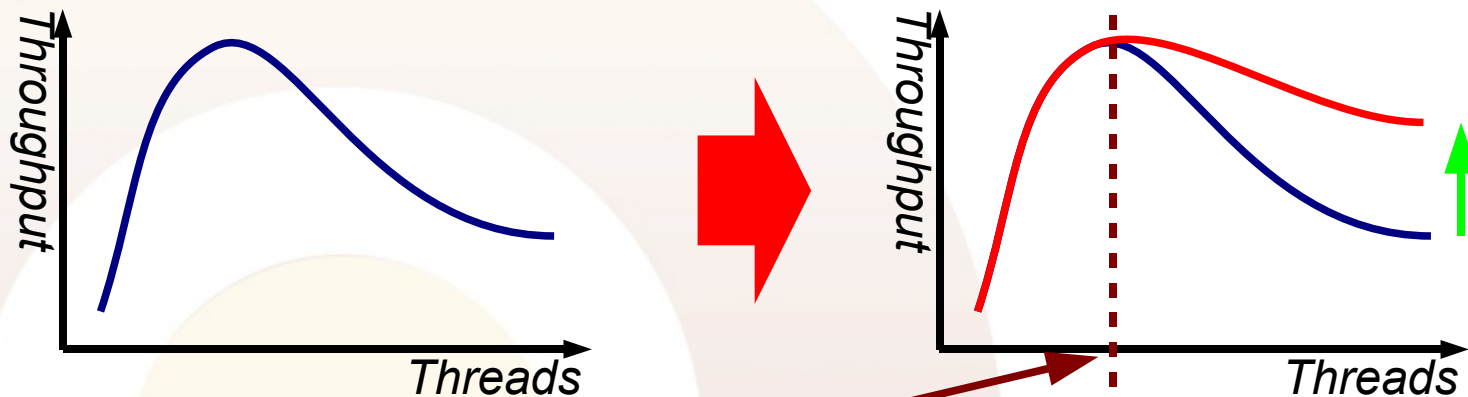
# Other Tips

Tips for the other variables to be tuned  
(no results in this session,  
excuse for no time to prepare....)

# innodb\_thread\_concurrency (Builtin~)

'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]*  
 $\geq$  *[CPU cores the server has]*)

It means “the InnoDB scales enough for you”

*Don't blame InnoDB :-)*

# Other IO tunes for SSD (XtraDB~)

`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

# TODO (XtraDB)

## Mutex/Latch contentions (for fast storage)

- 'btr\_search\_latch'
- 'page\_hash\_latch'

**Fixed  
officially!!!**

## ~~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)

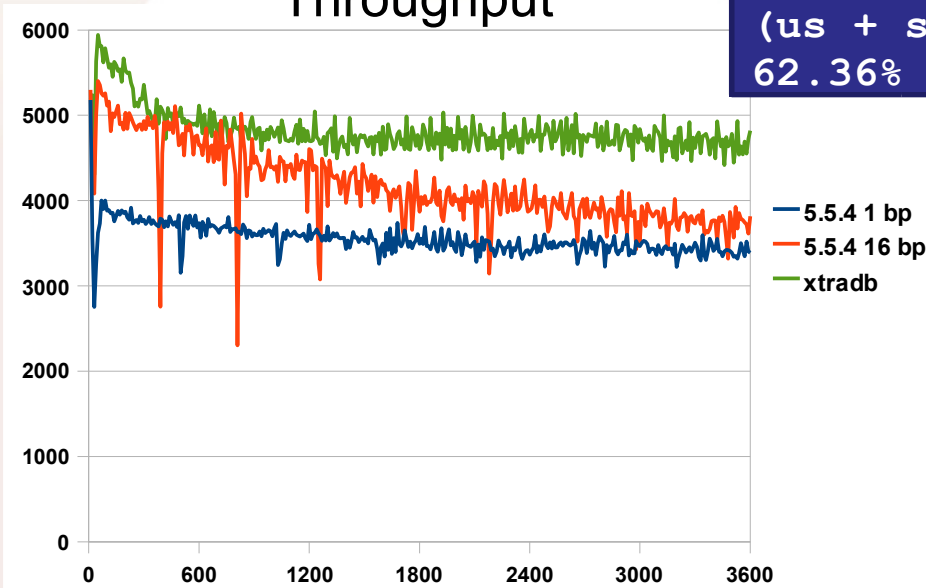
16 seems good for 16 cores

```
innodb_buf_pool_instances = 16
```

*But '16' seems more sensitive to*

*Insert Buffer Size(\*)...*

Throughput



Average cpu%  
(us + sy)  
62.36%

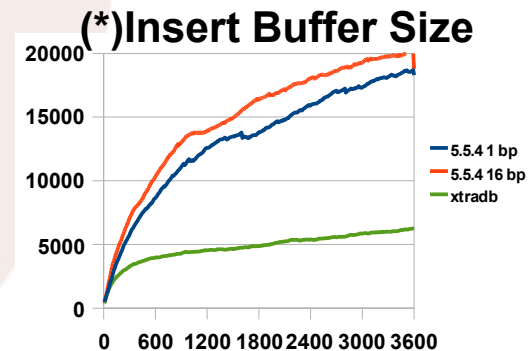
```

... ..
294 Mutex srv/srv0srv.c line 973
488 lock dict/dict0dict.c line 1584
495 Mutex ibuf/ibuf0ibuf.c line 522
651 lock dict/dict0dict.c line 634
1653 lock dict/dict0dict.c line 1584

```

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

Write IO bound and 'kernel\_mutex'  
(and 'ibuf\_mutex' ?)



(much modifies and read IO intensive [FusionIO])