
MySQL™ Connect

MySQL

ORACLE

Demystified MySQL/InnoDB Performance Tuning

Dimitri KRAVTCHUK
MySQL Performance Architect

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract.
It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Agenda

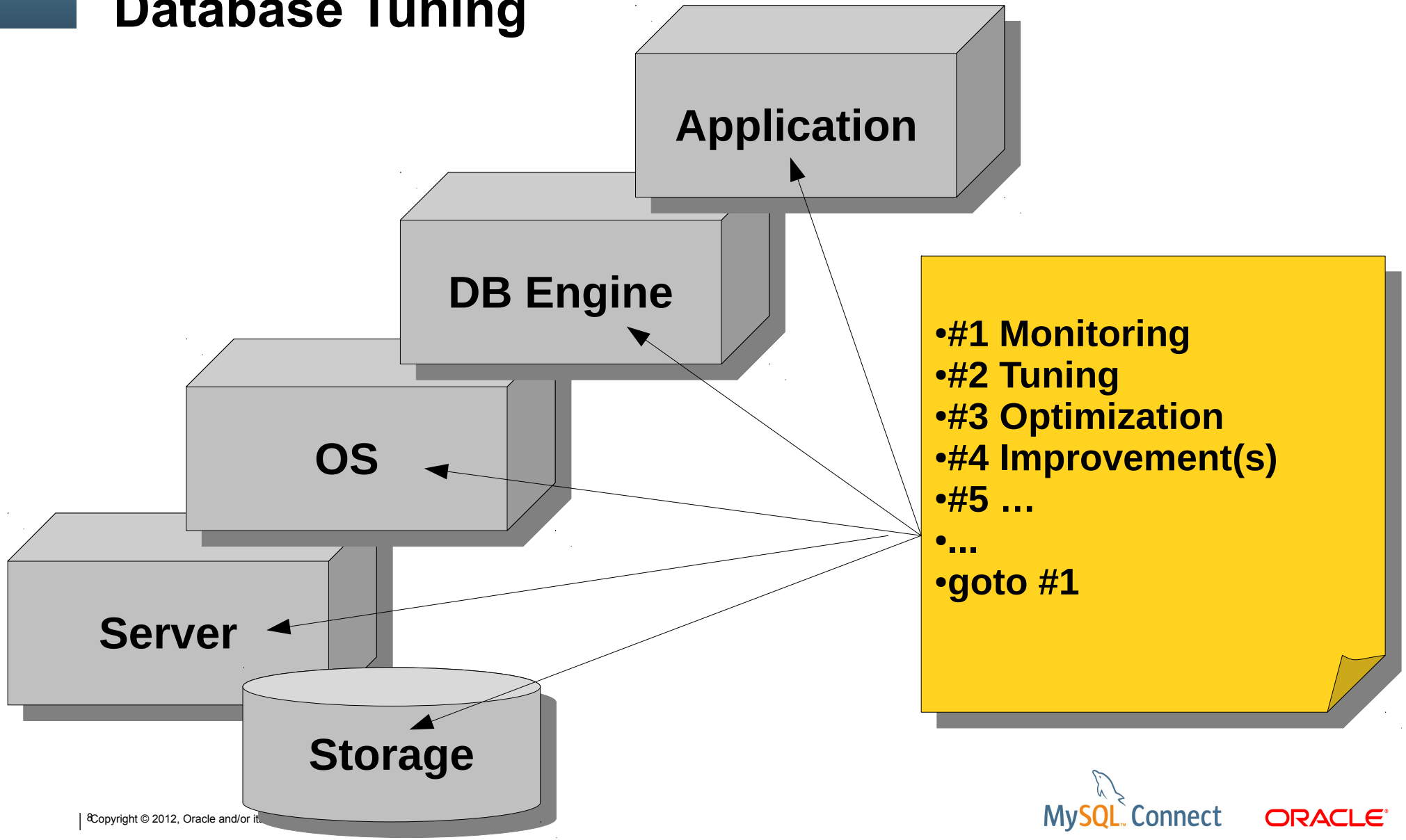
- MySQL Performance overview
- General tuning overview
- MySQL bottlenecks and solutions
- InnoDB bottlenecks and solutions
- Performance improvements made in MySQL 5.5 and 5.6
- Pending issues..
- Q & A

Preface: Too Close Look...

Preface: Less Close Look, Zoom--

Preface: Overall view, Zoom= Zoom/10

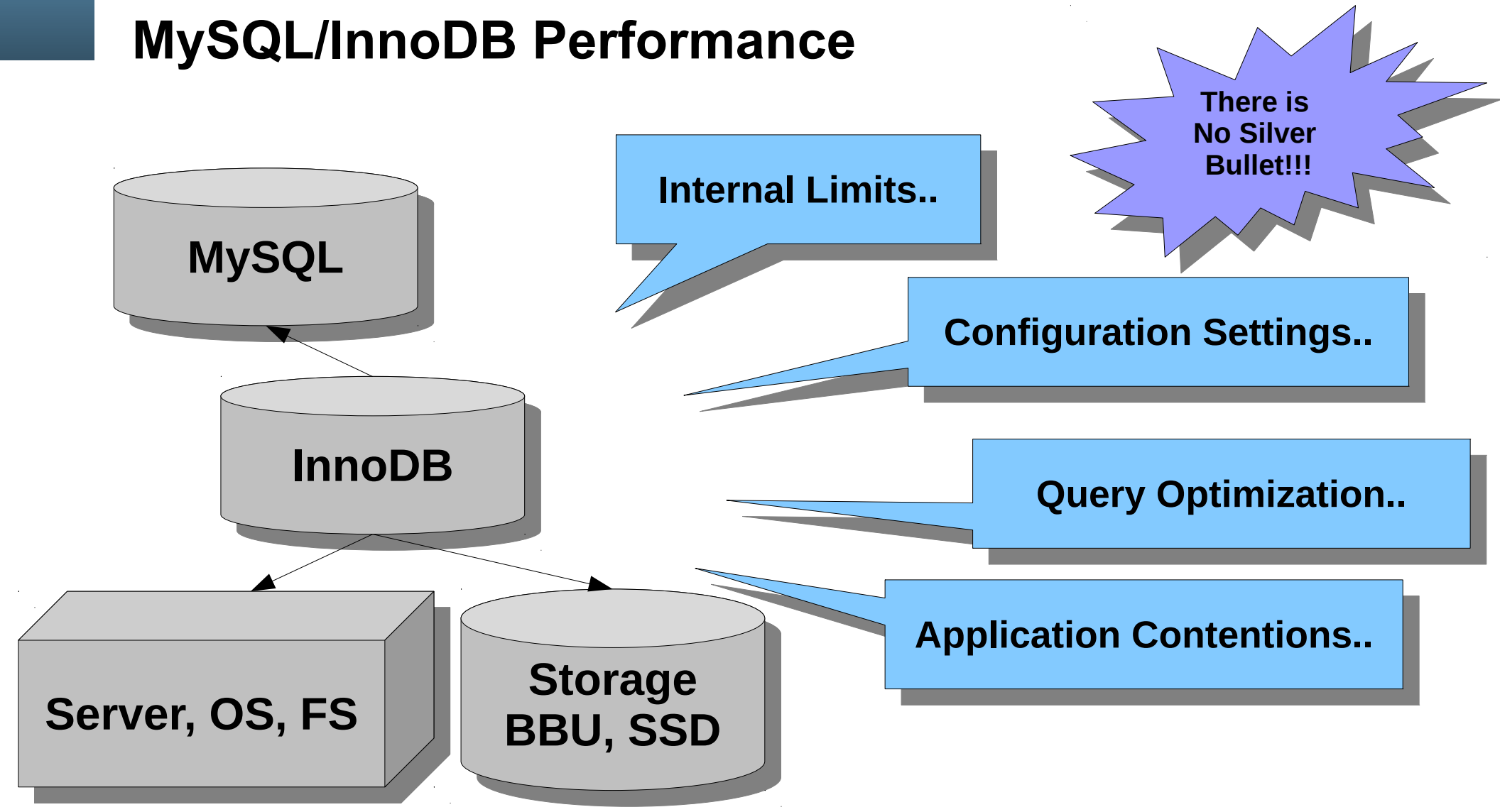
Database Tuning



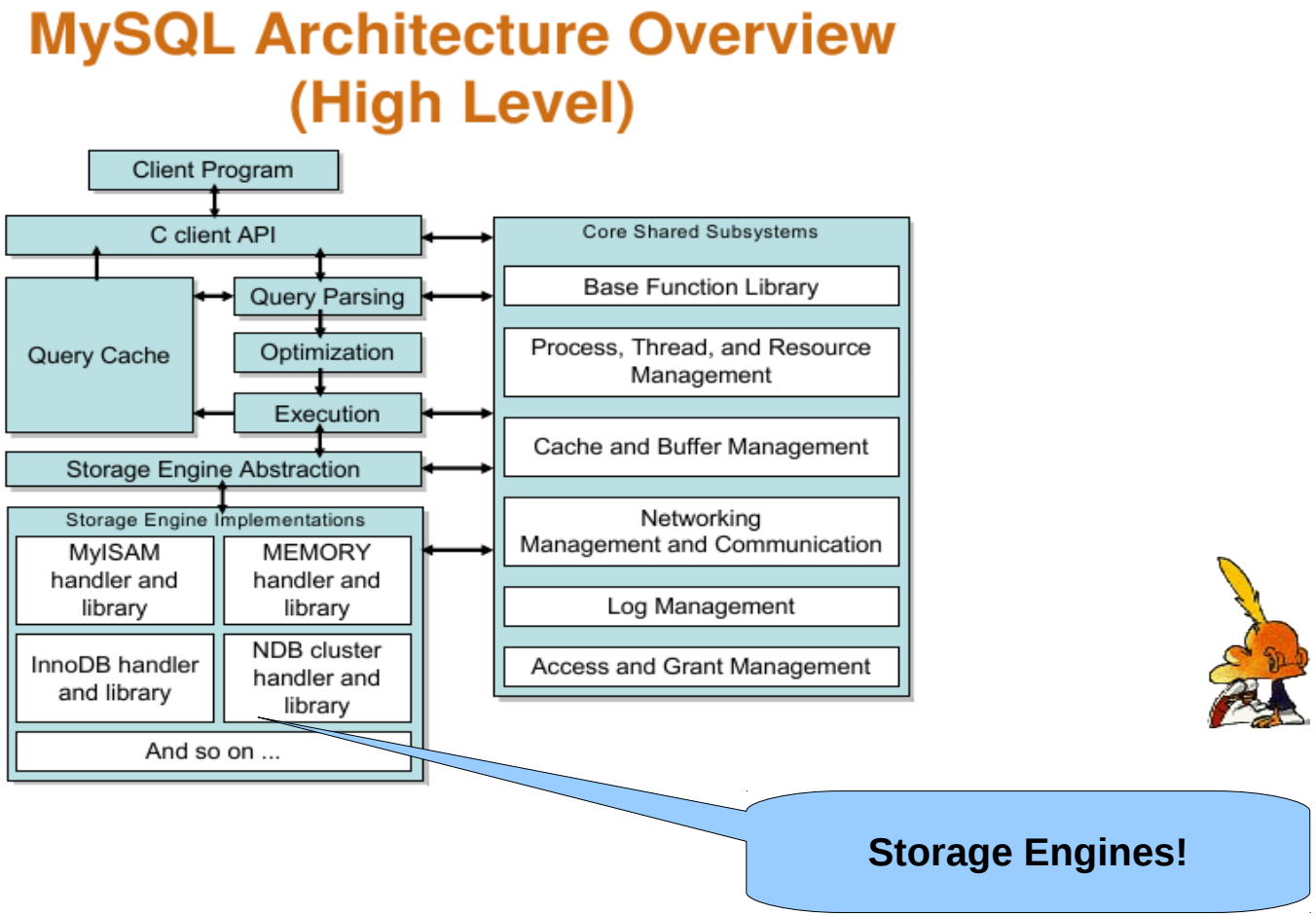
Start points for an optimal MySQL Server

- Think “Performance” from the beginning :-)
- Server:
  - Faster CPU is better! 32 cores is good enough ;-)
  - OS is important! - Linux, Solaris, etc.. (and Windows too!)
- Storage:
  - Don't use slow disks :-))
  - Having battery protected write cache helps REDO writes!
  - Having SSD drives helping random access! (index/data)
  - FS is important! - ZFS, UFS, QFS, VxFS, EXT3, EXT4, XFS, etc..
  - O_DIRECT or not O_DIRECT ;-)
- Network:
  - Don't forget!! :-) - 10Gbit is great! (faster is better)

MySQL/InnoDB Performance



MySQL Design



MySQL Design

- Multi-Threaded database
  - Fast context switch!
  - Simplified data access!
  - Concurrent access?..
  - Scalability?..

- Storage Engines
  - Initially: MyISAM only
  - Then, with InnoDB: started to match expectations of a “true RDBMS” ;-)
  - Many other engines (MEMORY, CSV, NDB, PBXT, etc.)
  - CREATE TABLE ... ENGINE=<NAME_OF_ENGINE>
  - ALTER TABLE ... ENGINE=<NAME_OF_ENGINE>
  - Did you choose a right Engine?..

MyISAM Engine (since 1994)

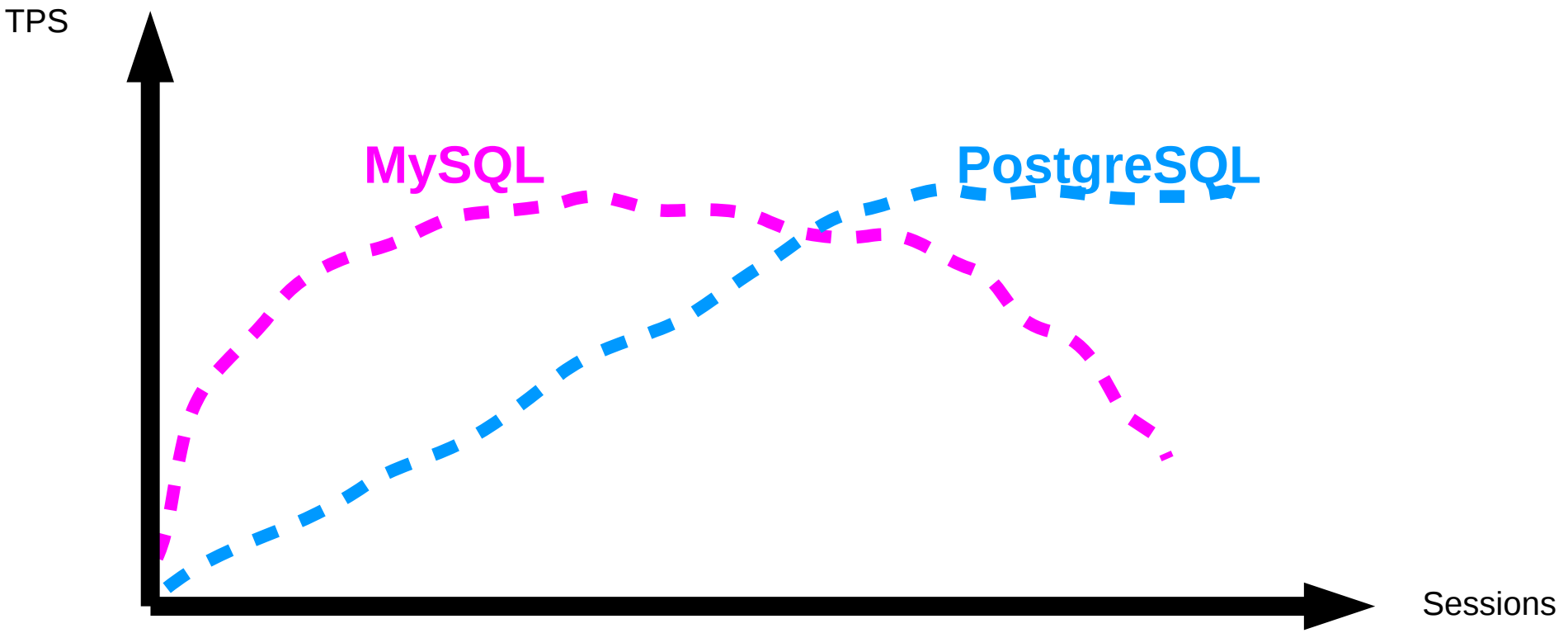
- Non-transactional! / No fast recovery! :-)
- Cache
  - Index only
  - Data => FS cache
  - mysql> flush tables;
- Single Writer @Table
  - Main bottleneck! => single writer
  - Solutions: delayed inserts, low priority
- Query plan
  - Index forcing may be necessary (hint)
- Extremely simple and lightweight

Why MySQL + MyISAM was successful?..

- Full Text search queries out-of-the-box!
- SELECT count(*) ... :-))
- Extremely SIMPLE!
  - my.conf => configuration parameters
  - mysql.server start / stop
  - Database => directory
  - Table => directory/Table.MYD, Table.MYI, Table.frm
  - $ cp Base1/Table.* /other/mysql/Base2

- Data binary compatibility! (ex: reports via NFS)
- Replication ready!
- Very FAST! (until some limit :-))
- RW workload is killing.. (but on 2CPU servers it was ok ;-))

RW benchmark: MyISAM vs PostgreSQL (in 2000)



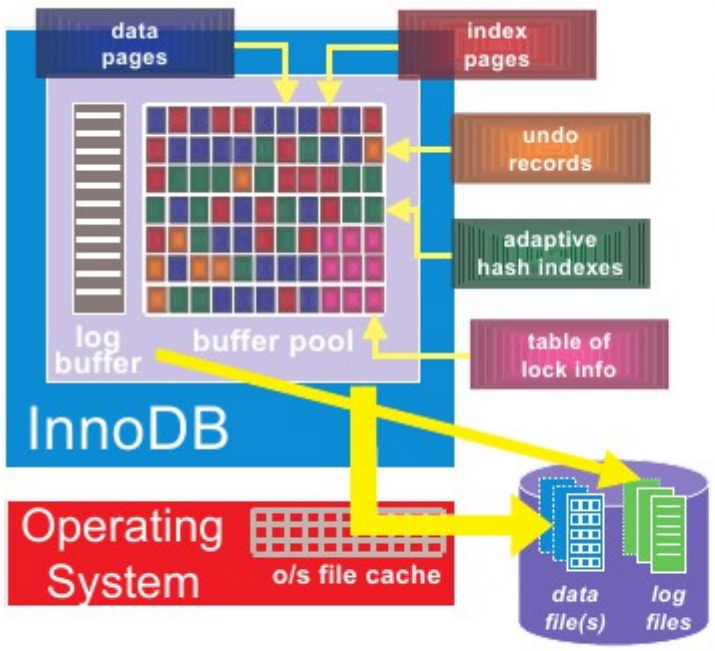
InnoDB Changing the game (since 2001)

- Row-level locking
- Index-only reads
- True transactions / UNDO
- Auto recovery
- Double write / Checksums
- Tablespaces or File-per-Table option
- Buffer pool
- Multi-threaded
- Currently the fastest transactional disk-based MySQL Storage Engine!

InnoDB Design

- All Data going via Buffer Pool
- Fuzzy Checkpoint
- Group Commit
- Log flush policy
- Threads:
  - User sessions
  - Master
  - Read Ahead
  - Page Writer
  - Log Writer

- Performance



- When data & index pages are read, they are cached for re-use, and are replaced when least-recently-used
- InnoDB on-the-fly creates adaptive hash indexes depending on query pattern (more on this later)
- On updates, row-level locking information is maintained in an efficient bit-map
- Undo info, used to reverse a transaction's changes, is cached in memory, later written to sys tblspace
- Compact representation of changes is kept in log buffer. On commit, log records are written to log file (but no buffer pool pages need to be written)
- Eventually, data, index and undo pages are flushed to data files

INNOBASE

MySQL Performance (traditionally, in the past)

- Choose the right Engine for each of your table/database
  - Read-Only / Text search => MyISAM
  - Read+Write / Transactions => InnoDB
  - Short/Small Transactions + DB fits in RAM => NDB

- Tune / Optimize your queries
- Once scalability limit is reached => go for Distributed:
  - Sharding
  - Master / Slave(s) => role-based workload
  - Any other similar :-)

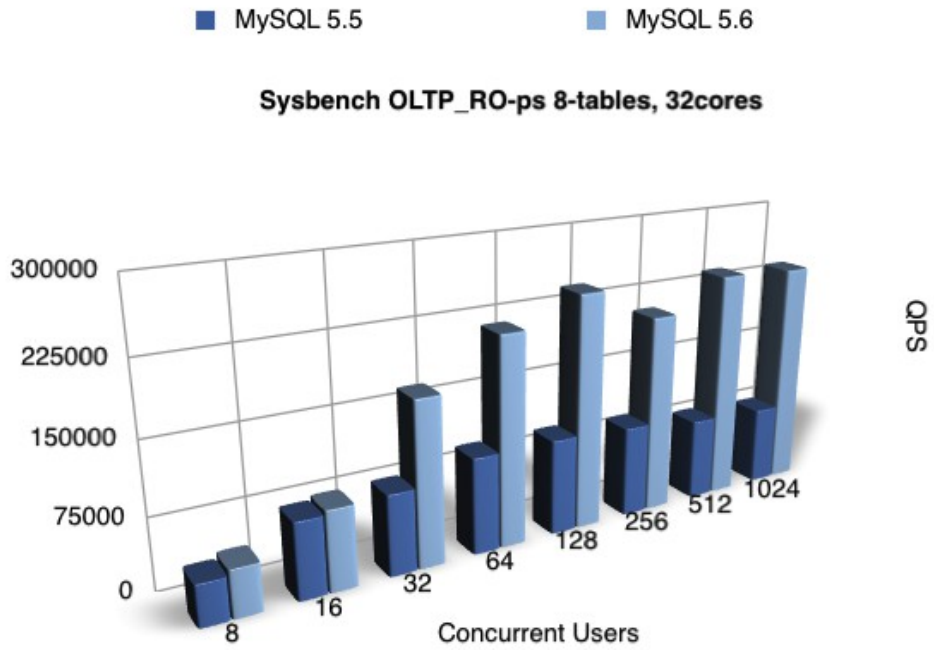
- Big Users: Google, Facebook, Amazon, (and even USA elections in 2008 ;-))..
- Scalability = Main Performance Problem!...

Things are changing constantly... :-)

- MySQL/InnoDB Scalability:
  - 2007 : up to 2CPU...
  - 2008 : up to 4CPU cores
  - 2009 : up to 16CPU cores (+Sun)
  - 2010 : up to 32CPU cores (+Oracle)
  - 2012 : up to 48CPU cores..
  - 2014 : …?? ;-)
  - NOTE: on the same HW performance is better from version to version!
- InnoDB today:
  - At least x4-8 times better performance than 2-3 years ago ;-)
  - Capable of ~~100.000~~ over 300.000 QPS(!)
  - Full Text Search (FTS) and Memcached
- Stay tuned ;-)

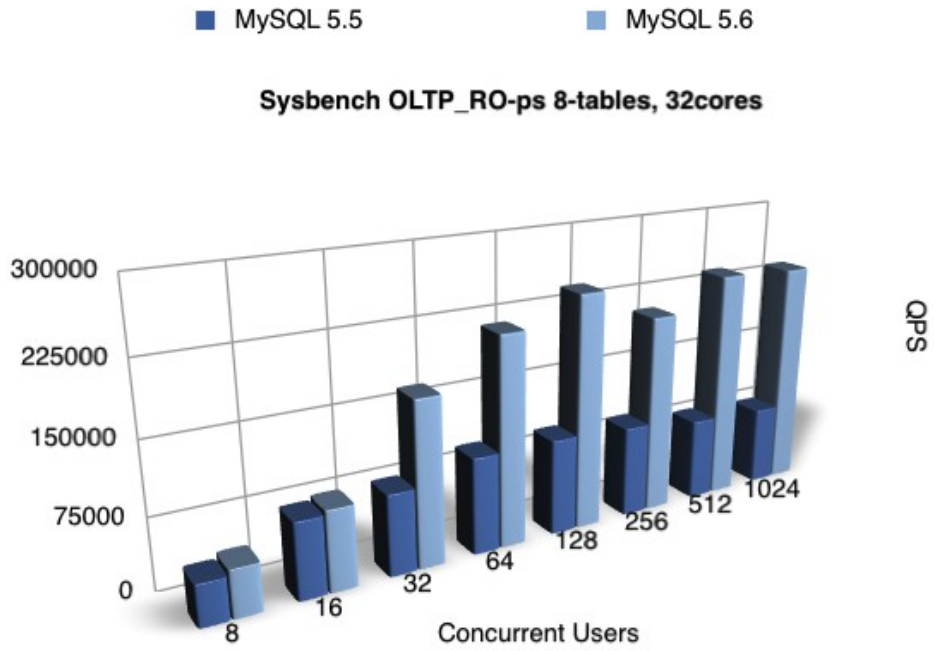
Sysbench OLTP_RO: MySQL 5.5 vs 5.6

- Test details:
  - All results are obtained on the same 32core Linux Server
  - Results are presented in QPS
  - Only Read + Write operations reported by Sysbench are counted
  - Each QPS number is representing the best possible performance level obtained on each MySQL version with the most optimal tuning applied..



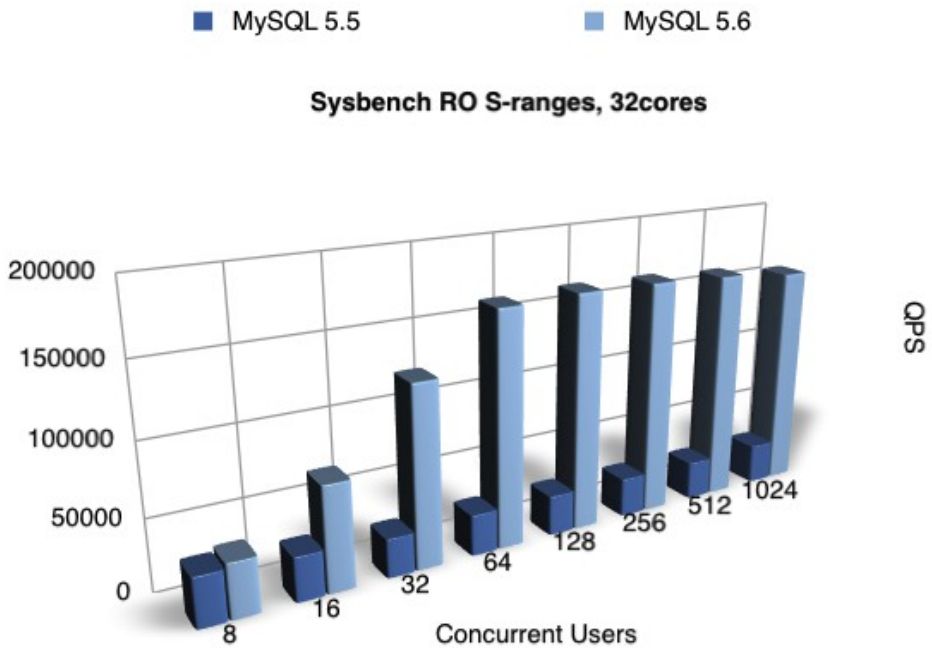
Sysbench OLTP_RO 8-tables: MySQL 5.5 vs 5.6

- Test details:
  - All results are obtained on the same 32core Linux Server
  - Results are presented in QPS
  - Only Read + Write operations reported by Sysbench are counted
  - Each QPS number is representing the best possible performance level obtained on each MySQL version with the most optimal tuning applied..



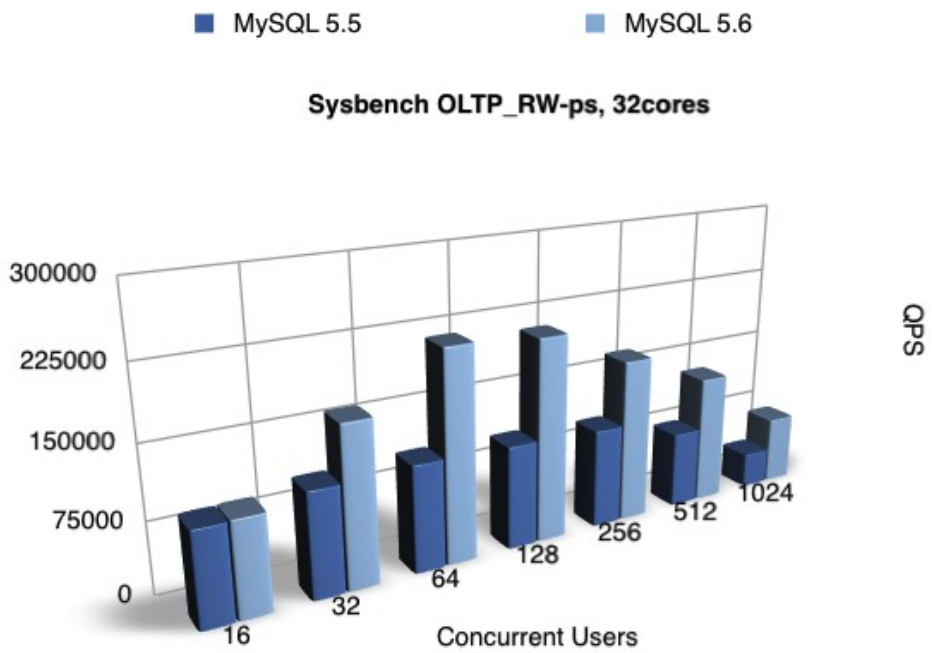
Sysbench RO Simple-Ranges: MySQL 5.5 vs 5.6

- Test details:
  - All results are obtained on the same 32core Linux Server
  - Results are presented in QPS
  - Only Read + Write operations reported by Sysbench are counted
  - Each QPS number is representing the best possible performance level obtained on each MySQL version with the most optimal tuning applied..



Sysbench OLTP_RW: MySQL 5.5 vs 5.6

- Test details:
  - All results are obtained on the same 32core Linux Server
  - Results are presented in QPS
  - Only Read + Write operations reported by Sysbench are counted
  - Each QPS number is representing the best possible performance level obtained on each MySQL version with the most optimal tuning applied..



But it was not easy... Except if.. :-)

- Let's split the problems...
- MySQL + System
- MySQL Internals
- InnoDB Internals
- InnoDB and I/O

MySQL & OS : memory allocation

- Be Aware: “Default” malloc() LIB is usually not MT-concurrency oriented !!
- Concurrent memory allocation may freeze threads
- Important part of MySQL is written on C++
  - Creation of any variable calls malloc() automatically in C++ !

- Use MT-oriented solutions:
  - HOARD (was the first and most fast before)
  - Solaris: libmtmalloc, libumem
  - Linux: libtcmalloc, jemalloc
  - Force with LD_PRELOAD:

```
$ LD_PRELOAD=/usr/lib64/libjemalloc.so.1
$ export LD_PRELOAD
```

NOTE: it's a choice according your workload!

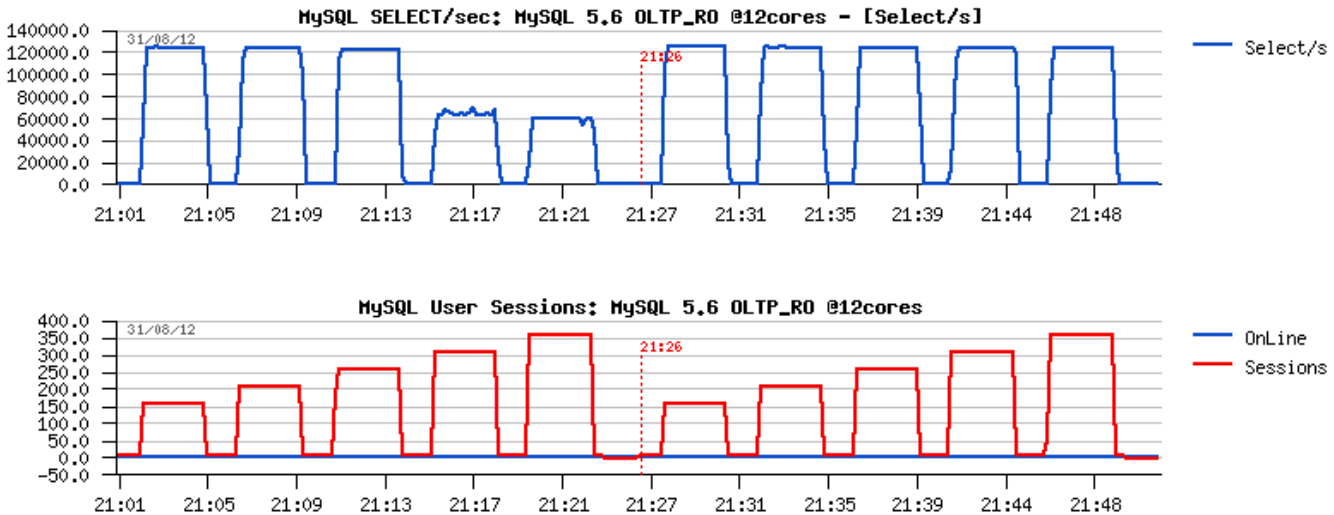
MySQL Internals : LOCK_open contention

- MySQL 5.5 and before:
  - Keep “table_open_cache” setting big enough!
  - Monitor global status for '%opened%'
  - Once this contention become the most hot – well, time to upgrade to 5.6 ;-))

- Since MySQL 5.6:
  - Fixed, part of MDL now :-)
  - But it doesn't mean “table_open_cache” can be low ;-)
  - Monitor PFS Waits!
  - Monitor “table_open_cache%” status variables!
  - Keep “table_open_cache_instances” at least bigger than 1
  - But MDL lock contention become the most hot if other contentions are gone..
  - MDL mutex lock contention => fix in progress (already a working prototype)
  - MDL rw-lock contention => more complex, planned for MySQL 5.7

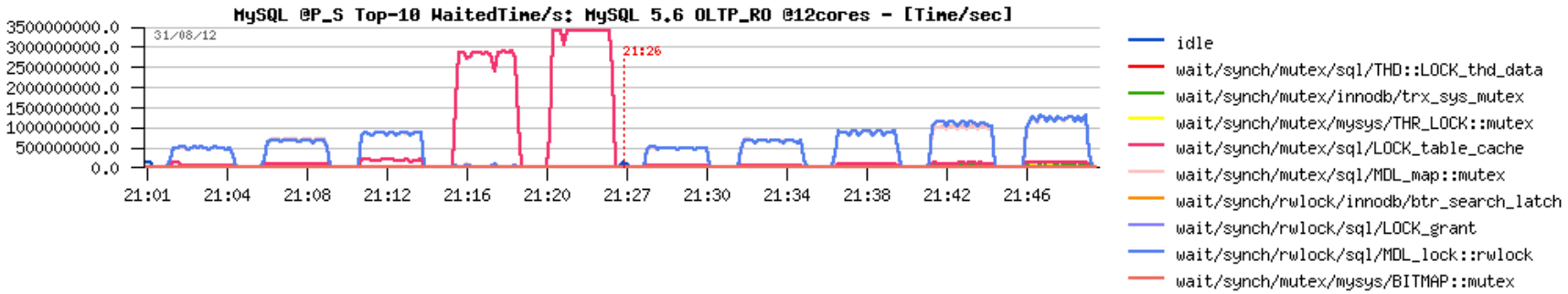
MySQL 5.6 Internals : low table_open_cache

- MySQL 5.6 :
  - Not big enough “table_open_cache” setting

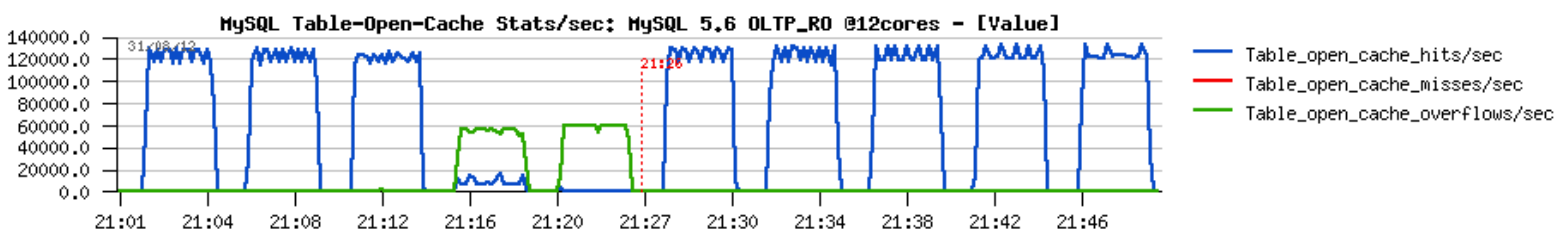


MySQL 5.6 Internals : low table_open_cache (2)

- MySQL 5.6 :
  - Not big enough “table_open_cache” setting
  - PFS Waits monitoring: LOCK_table_cache become the most hot:

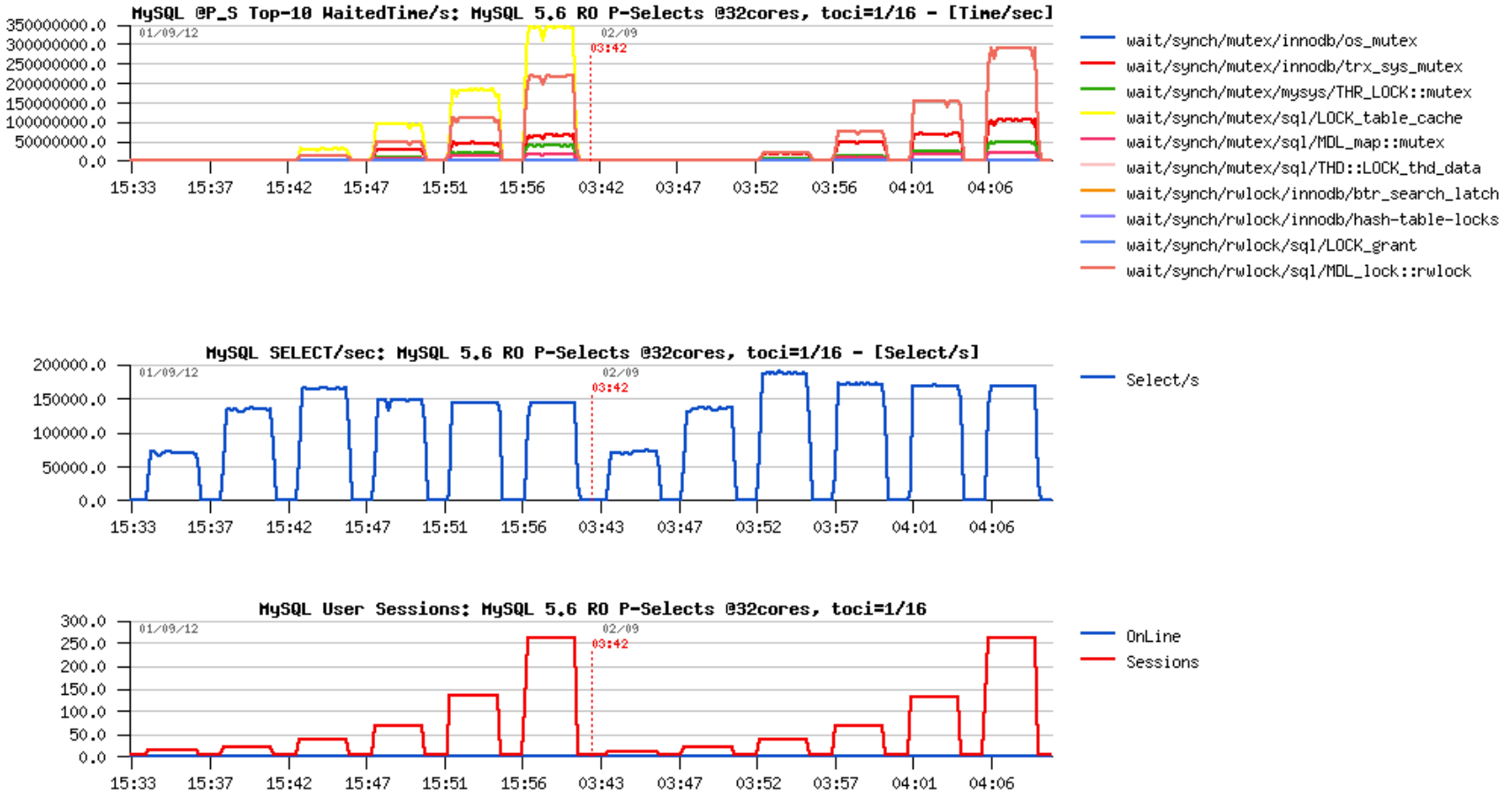


  - Table_open_cache% status:



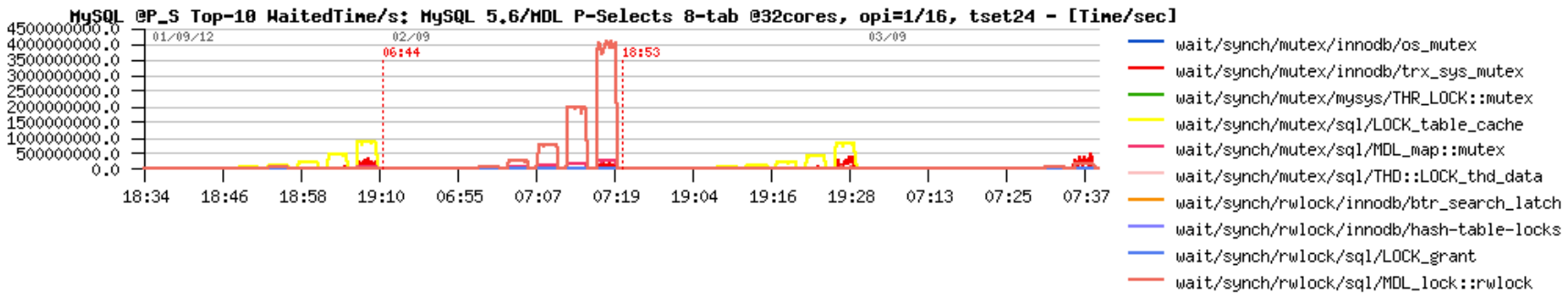
MySQL 5.6 Internals : table_open_cache_instances

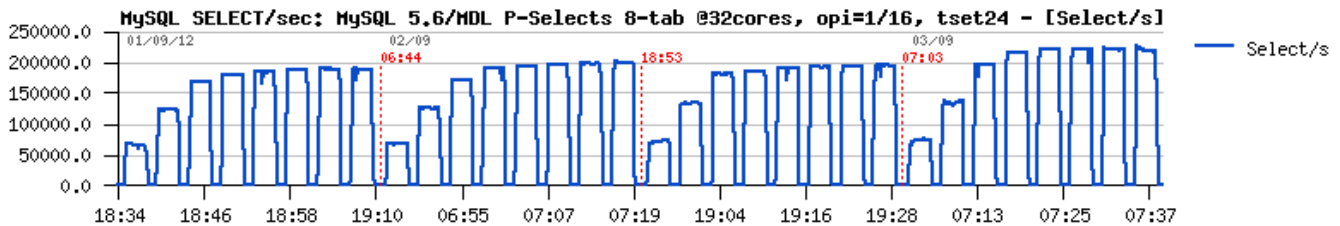
- MySQL 5.6 :
  - When LOCK_table_cache wait is on top, the gain is usually well visible:

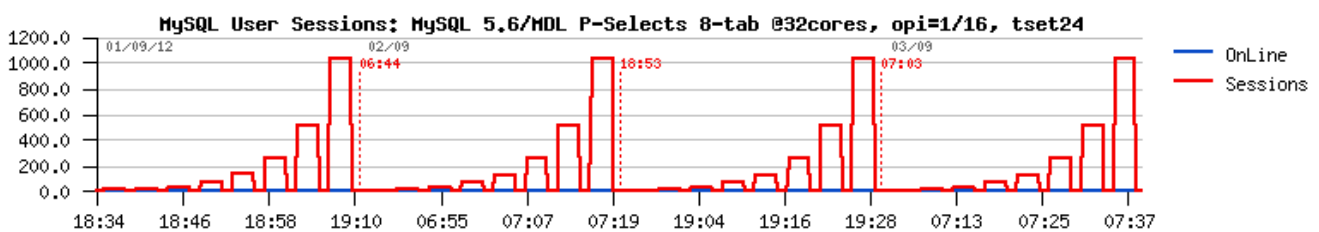


MySQL 5.6 Internals : MDL contention

- MDL lock involved on <u>every</u> table for every query, fix in progress..
  - Test: 5.6 opi=1, 5.6 opi=16, 5.6-mdl-fix opi=1, 5.6-mdl-fix opi=16

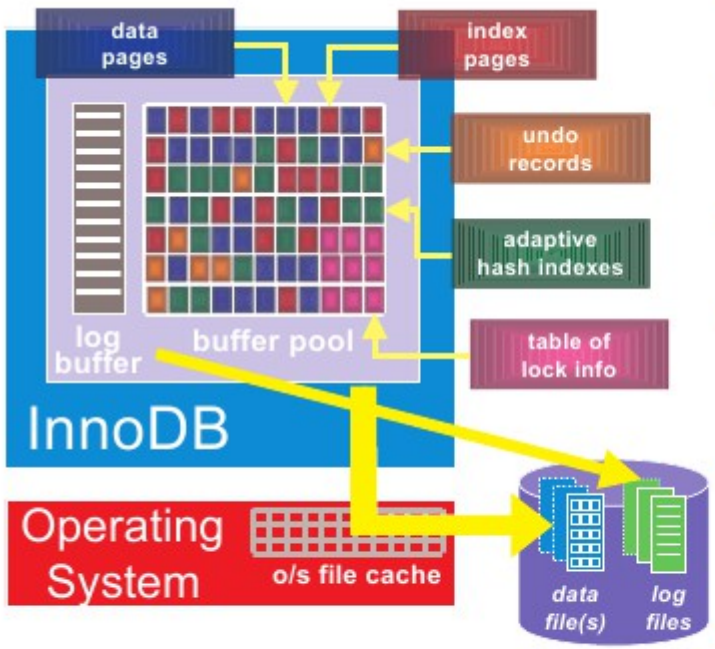






InnoDB : sources of performance issues

- Buffer Pool
- Internal locks
- Concurrency
- IO capacity
- Purge
- Flushing
- etc...



- When data & index pages are read, they are cached for re-use, and are replaced when least-recently-used
- InnoDB on-the-fly creates adaptive hash indexes depending on query pattern (more on this later)
- On updates, row-level locking information is maintained in an efficient bit-map
- Undo info, used to reverse a transaction's changes, is cached in memory, later written to sys tblspace
- Compact representation of changes is kept in log buffer. On commit, log records are written to log file (but no buffer pool pages need to be written)
- Eventually, data, index and undo pages are flushed to data files

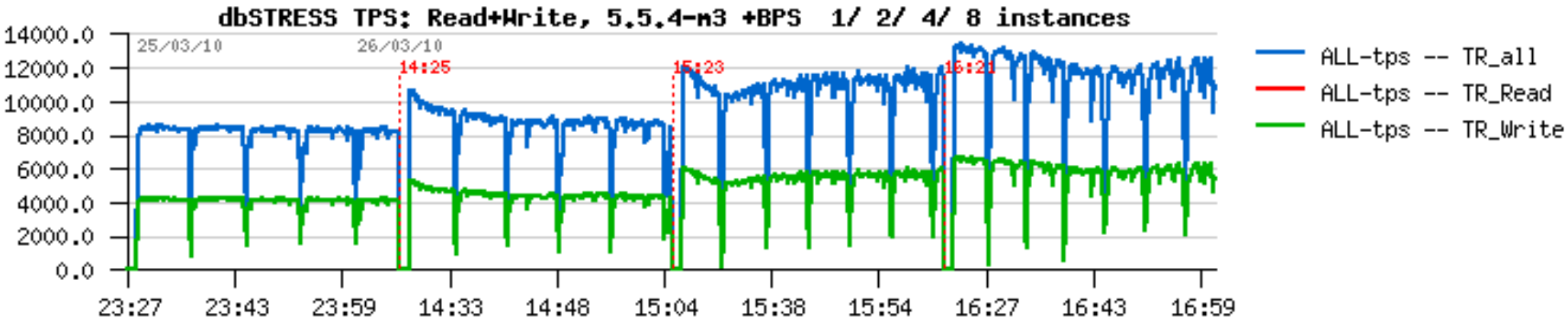
INNOBASE

InnoDB : Buffer Pool instances

- Before 5.5.4:
  - even all your data pages are seating in the Buffer Pool there was still a contention on the Buffer Pool mutex.. - so, needs a split!
  - was the main showstopper for all the following improvement in 5.5 !
- Split via hash table:
  - ok, but may require a huge hash table, hit hash table code scalability limits..
- Multiple buffer instances :
  - scales better, open door for new features!
- NOTE :
  - in 5.5 contention still depends on the data/page distribution within a Buffer Pool (as in Oracle RDBMS too)
  - In 5.6 hashing is added to the pages, so the impact is less dramatic + more stable performance

InnoDB : Buffer Pool instances in MySQL 5.5

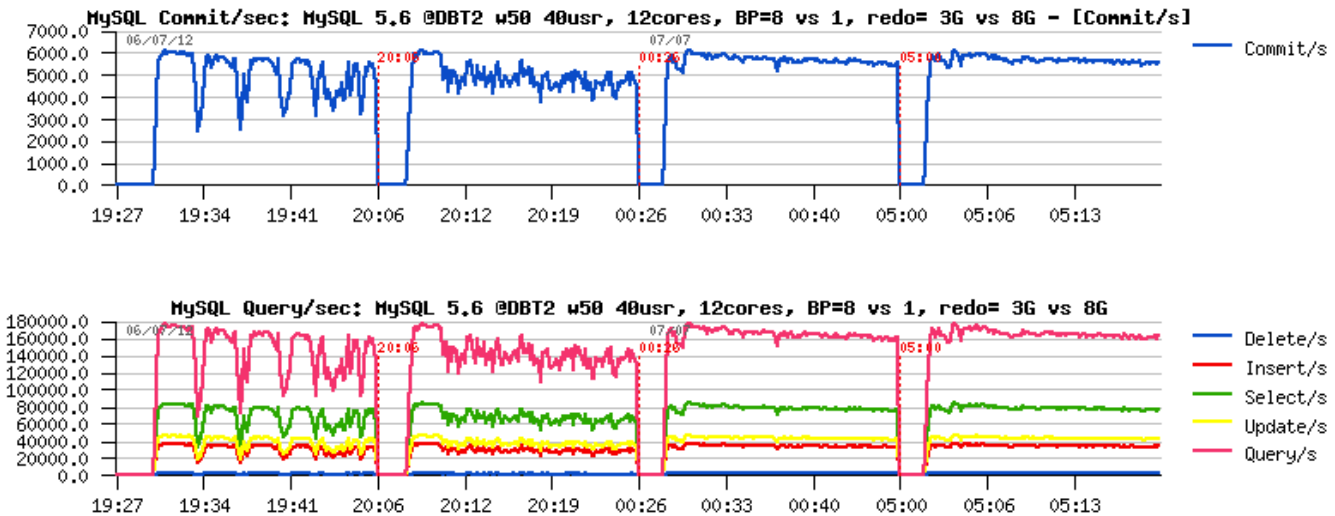
- RW Workload
  - innodb_buffer_pool_instances= 1/ 2/ 4/ 8



InnoDB : Buffer Pool instances in MySQL 5.6

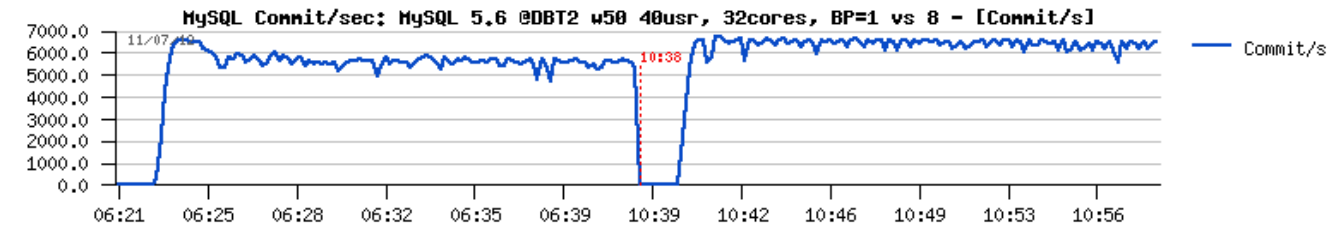
- DBT2 (heavy RW) Workload
  - innodb_buffer_pool_instances= 1/ 8
  - Problem: sync flushing...
  - Solutions: bigger REDO, faster storage, tuned Adaptive Flushing

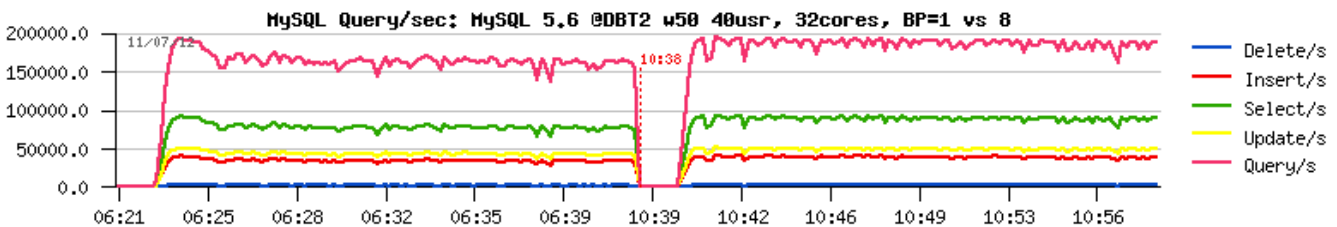
- Bigger REDO:



InnoDB : Buffer Pool instances in MySQL 5.6 (2)

- DBT2 (heavy RW) Workload
  - innodb_buffer_pool_instances= 1/ 8
  - Same REDO, just faster storage:



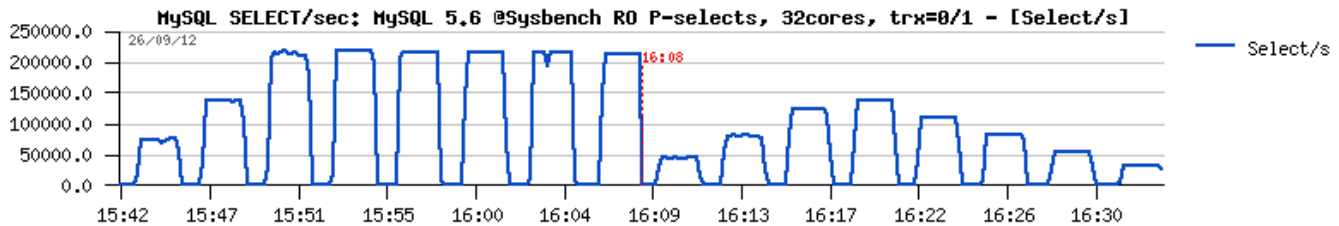


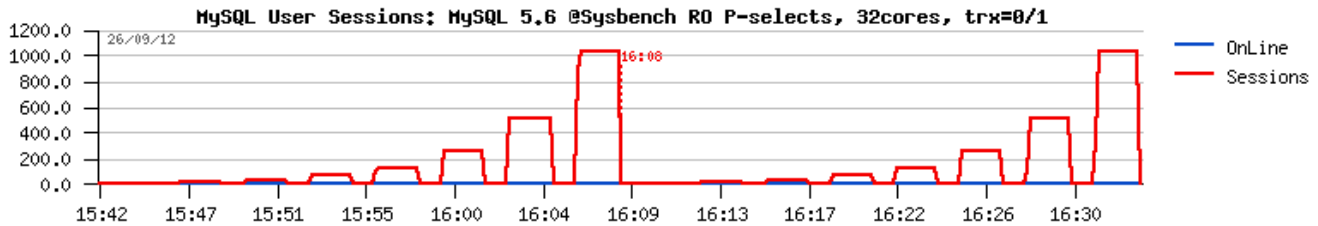
InnoDB : Internal locks

- MySQL 5.5 :
  - kernel_mutex: the most hot, blocking factor in most of workloads (RO & RW)
  - btr_search_latch RW-lock: used by AHI, can be avoided by switching AHI off
  - Index lock: blocking writes and purge(!), contention can be reduced by using partitions (but not always applicable)..
- MySQL 5.6 :
  - split of kernel_mutex !!
  - other mutex contentions became more hot, finally reducing overall performance..
  - trx_sys mutex became a showstopper for any further improvement..
  - trx_sys is mainly used by every transaction, even RO workload.. hm..
- RO transactions(!) :
  - explicit START TRANSACTION read only
  - or having AUTOCOMMIT=1 and not involving any start transaction / begin/commit

InnoDB : Read-Only Transactions in MySQL 5.6

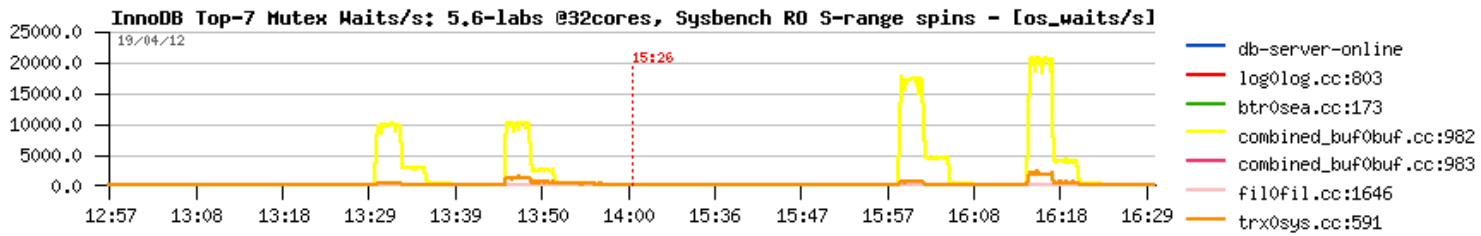
- Sysbench POINT-SELECTs RO Workload
  - Concurrent user sessions: 1, 2, 4 .. 1024
  - Using of transactions in sysbench = 0 / 1

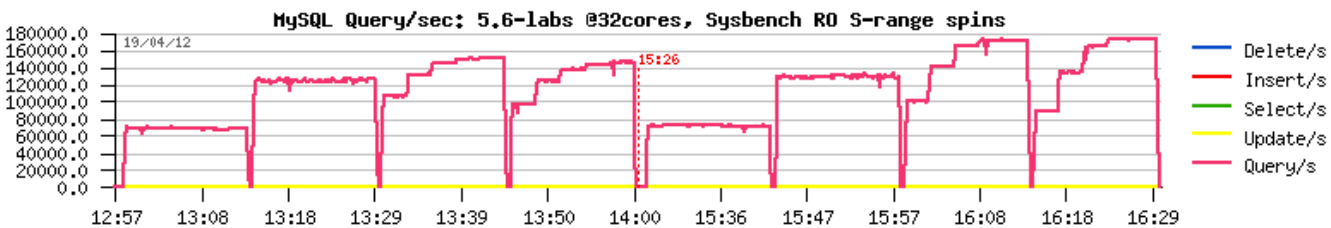




InnoDB : trx_sys mutex contention with more CPUs

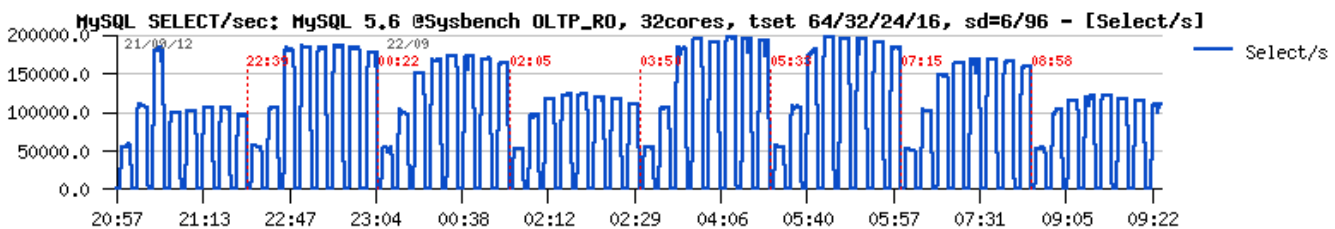
- Sysbench RO Workloads
  - With more CPU cores contention become more hot
  - Bind of mysqld to less cores helps, but the goal is to get benefit from more cores ;-)
  - Using innodb_thread_concurrency is not helping here anymore..
  - So, **innodb_spin_wait_delay** is entering in the game:

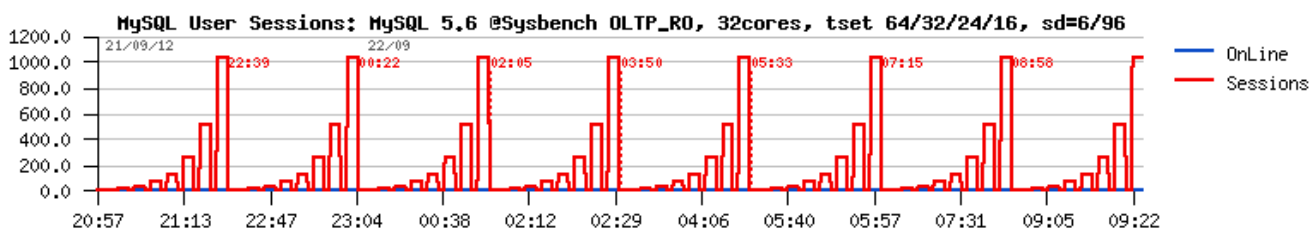




InnoDB : adapting innodb_spin_wait_delay

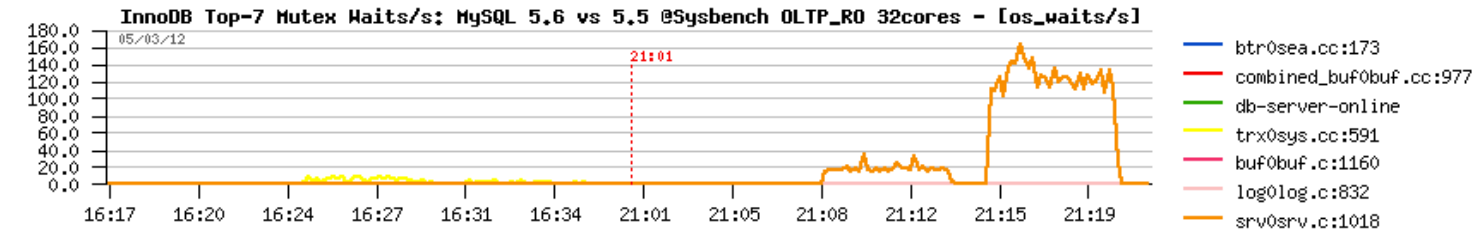
- RO or RW Workloads
  - NOTE: bigger innodb_spin_wait_delay is simply increasing the max random delay that user thread may sleep within a spin loop in wait for lock..
  - Ideally should be auto-adaptive.. while the same tuning works for 5.5 as well ;-)
  - General rule: default is 6, may need an increase with more cores
  - Test: 32-bi-threadi/ 32/ 24/ 16cores, spin delay = 6 / 96 :

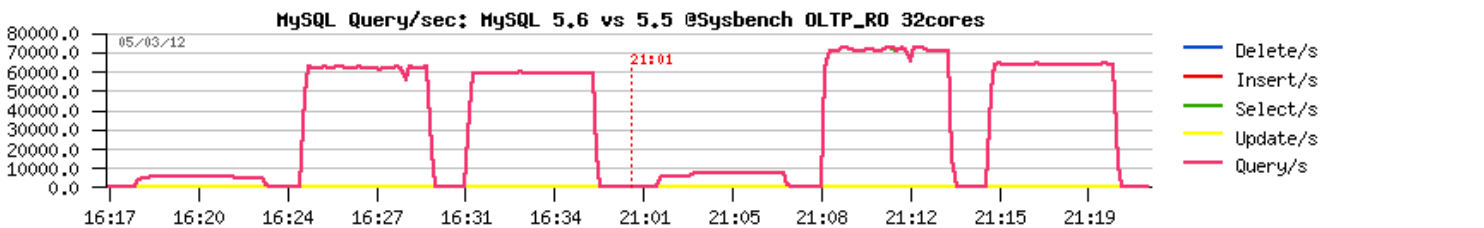




InnoDB : false sharing of cache-line = true killer

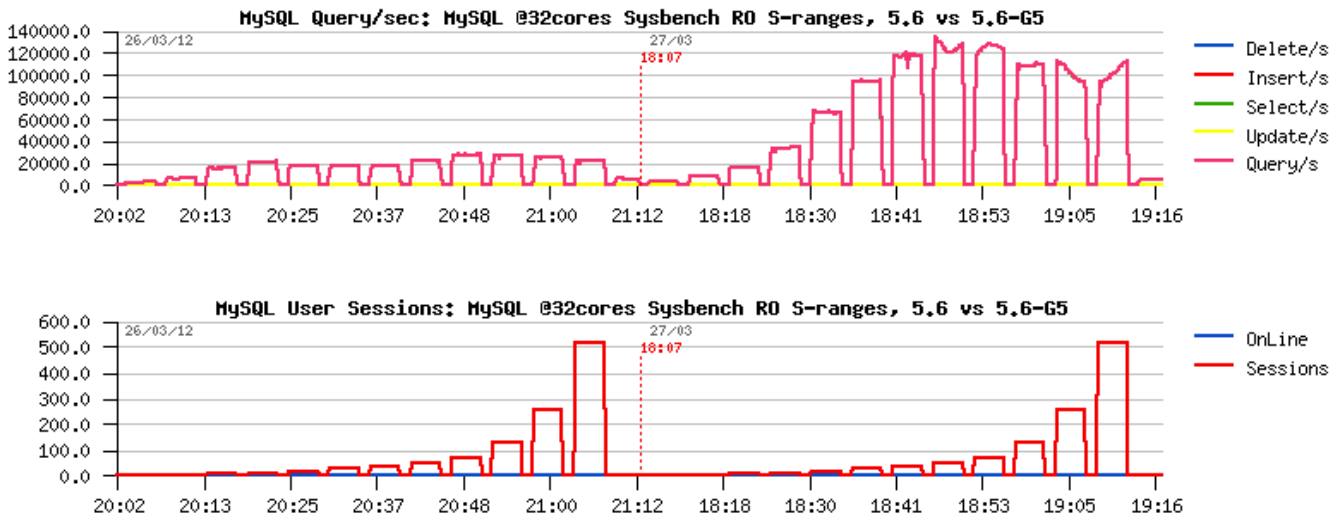
- RO or RW Workloads
  - Same symptoms in 5.5 & 5.6 : no QPS improvement between 16 and 32 user sessions:





InnoDB : false sharing of cache-line fixed!

- RO or RW Workloads
  - “G5” patch! :-)
  - x2 times better on Sysbench OLTP_RO, and x6 times on SIMPLE-Ranges!
  - NOTE: unfortunately the fix is not applicable on 5.5..



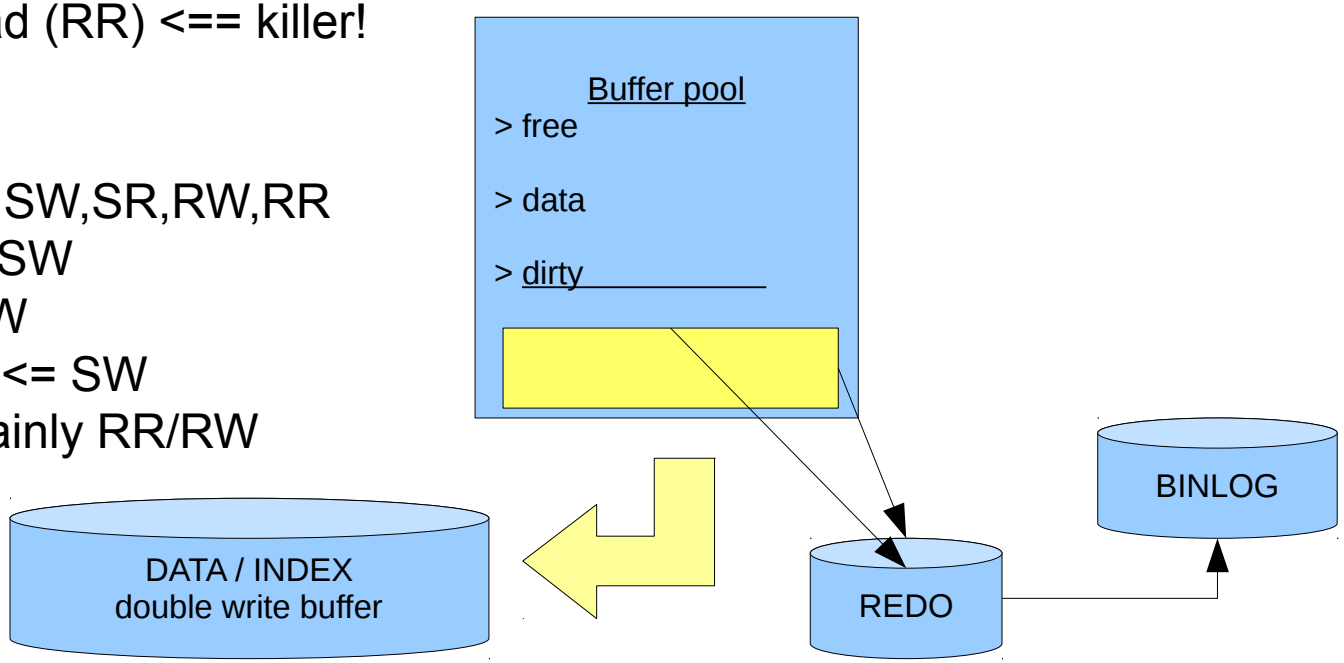
InnoDB : I/O level design

- Initially:
  - limited to 100 writes/sec...
  - 1 Read-Ahead thread
  - 1 Page-Write thread
  - 1 RSB
- Since MySQL 5.5 :
  - I/O capacity option!
  - Multiple Read helper and Write helper threads, + AIO support on Linux
  - x10-x100 faster recovery
  - Adaptive Flushing
  - Purge Thread
  - 128 RSB (+configurable!)
- Since MySQL 5.6 :
  - Several Purge threads, Page cleaner thread, improved Adaptive Flushing, over 4GB REDO logs, more...

InnoDB : I/O operations nature

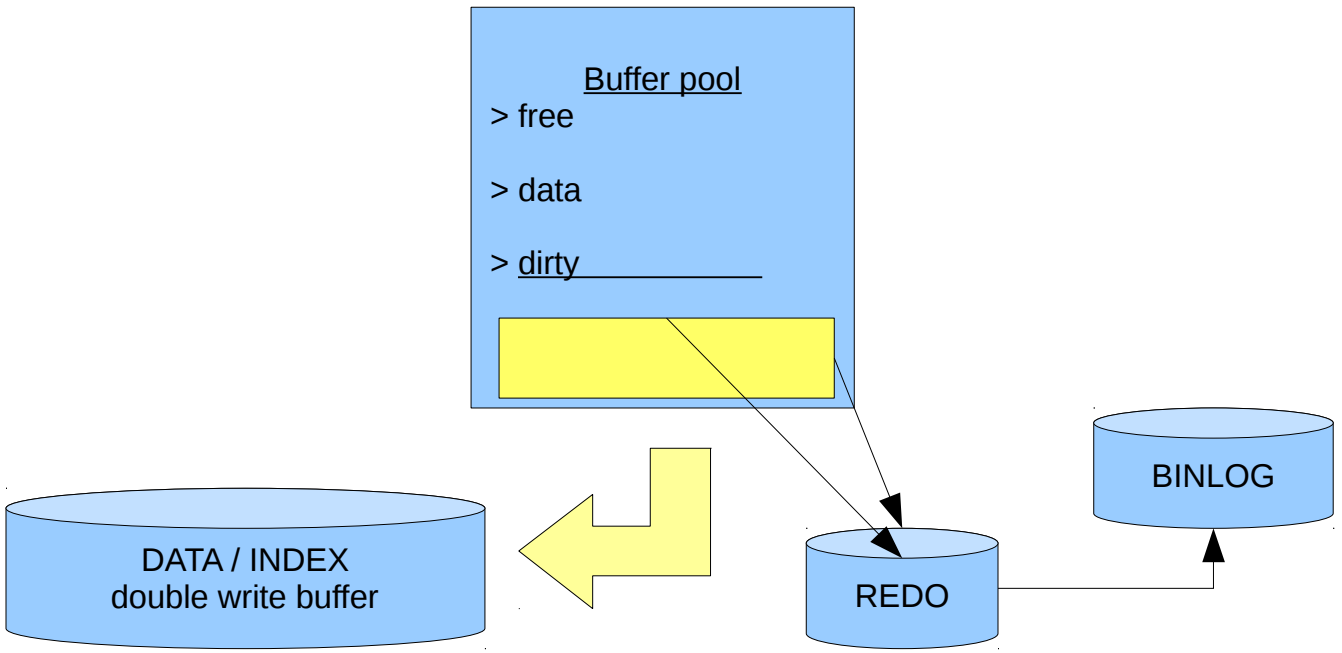
- Keep in mind the nature of I/O operation!
  - Sequential Write (SW)
  - Sequential Read (SR)
  - Random Write (RW)
  - Random Read (RR) <== killer!

- InnoDB
  - Data files <= SW,SR,RW,RR
  - Redo log <= SW
  - Bin log <= SW
  - Double write <= SW
  - UNDO <= mainly RR/RW



InnoDB : mix of I/O operations impact

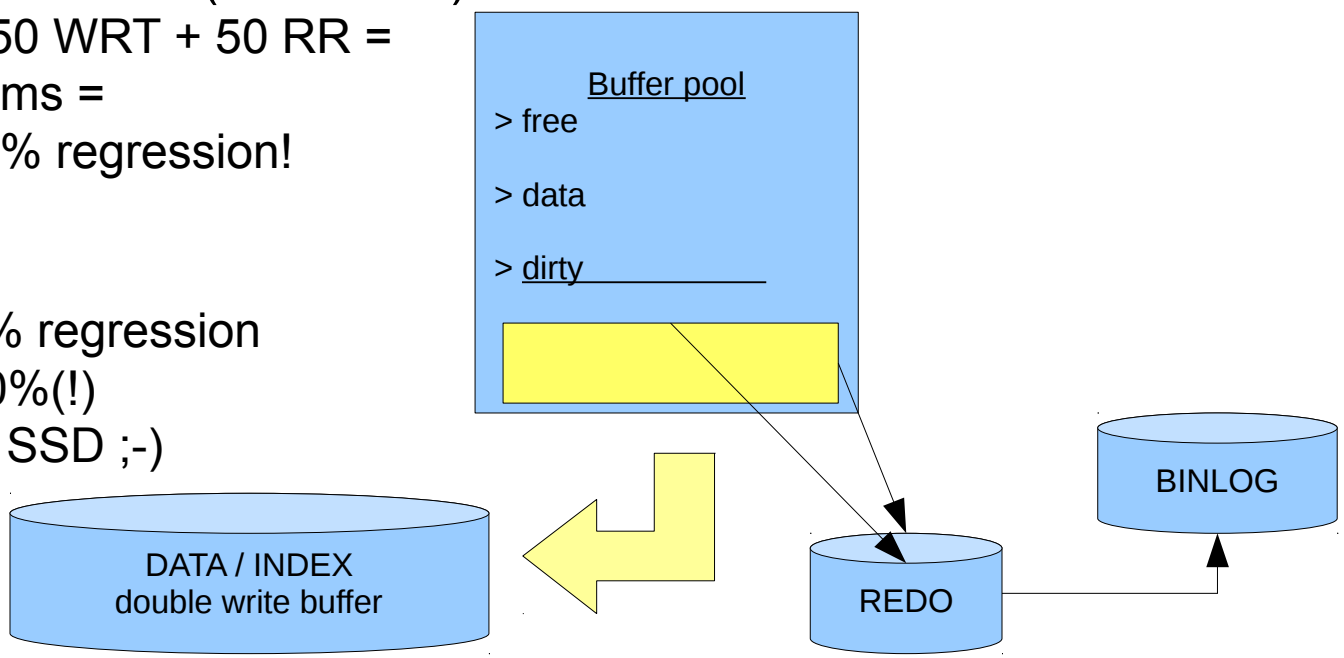
- For ex: you're doing 1000 writes/sec
- Now, 5% of your I/O operations become random read.. <= any % regression?



InnoDB : mix of I/O operations impact on HDD

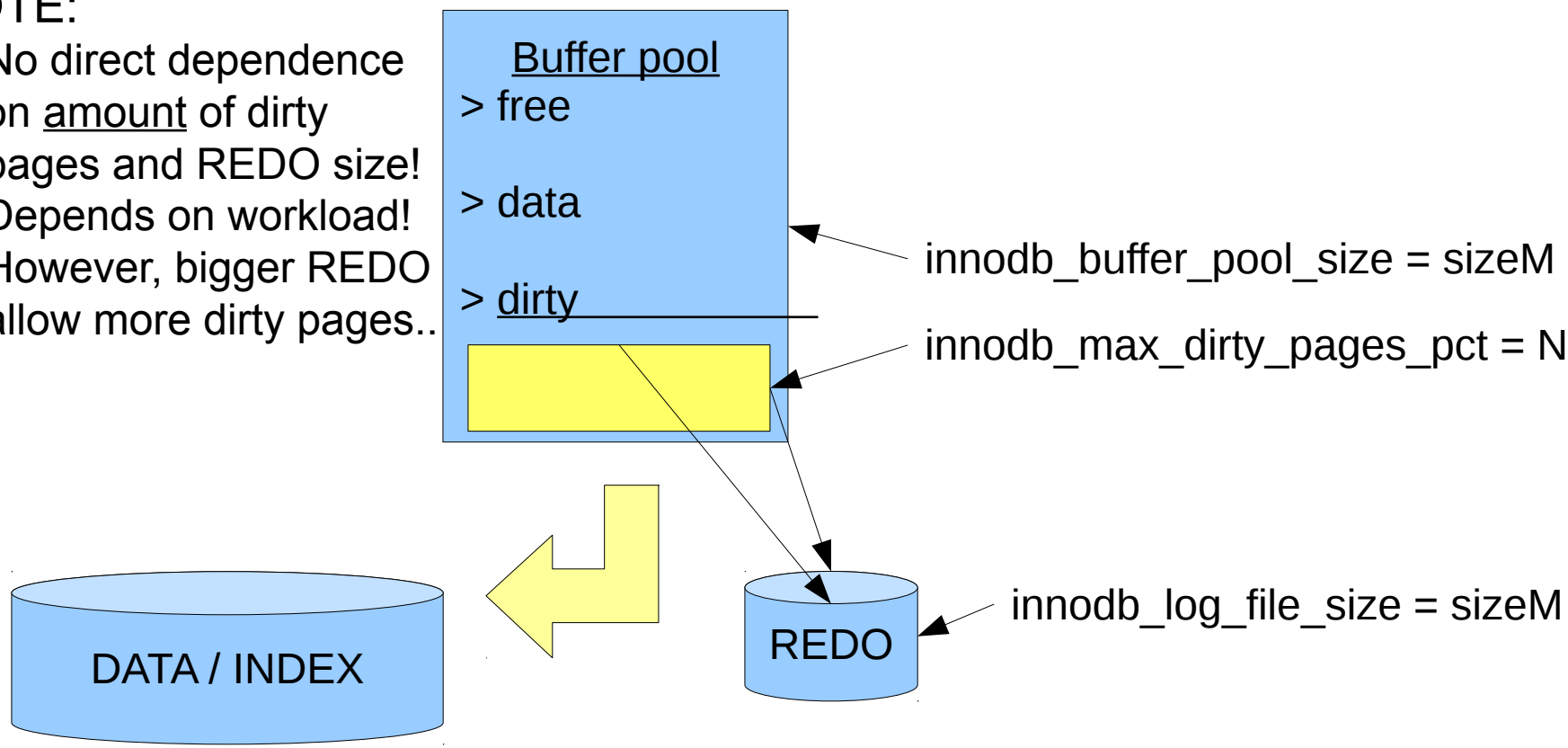
- For ex: you're doing 1000 writes/sec
- Now, 5% of your I/O operations become random read.. <= any % regression?
- So, before 1000 I/O = 1000ms
- Since 5% become RR (5ms each) :
  - 1000 I/O = 950 WRT + 50 RR =
  - 950ms + 250ms =
  - 1200ms = 20% regression!

- So far:
  - 5% RR = 20% regression
  - 10% RR = 40%(!)
  - so, love your SSD ;-)



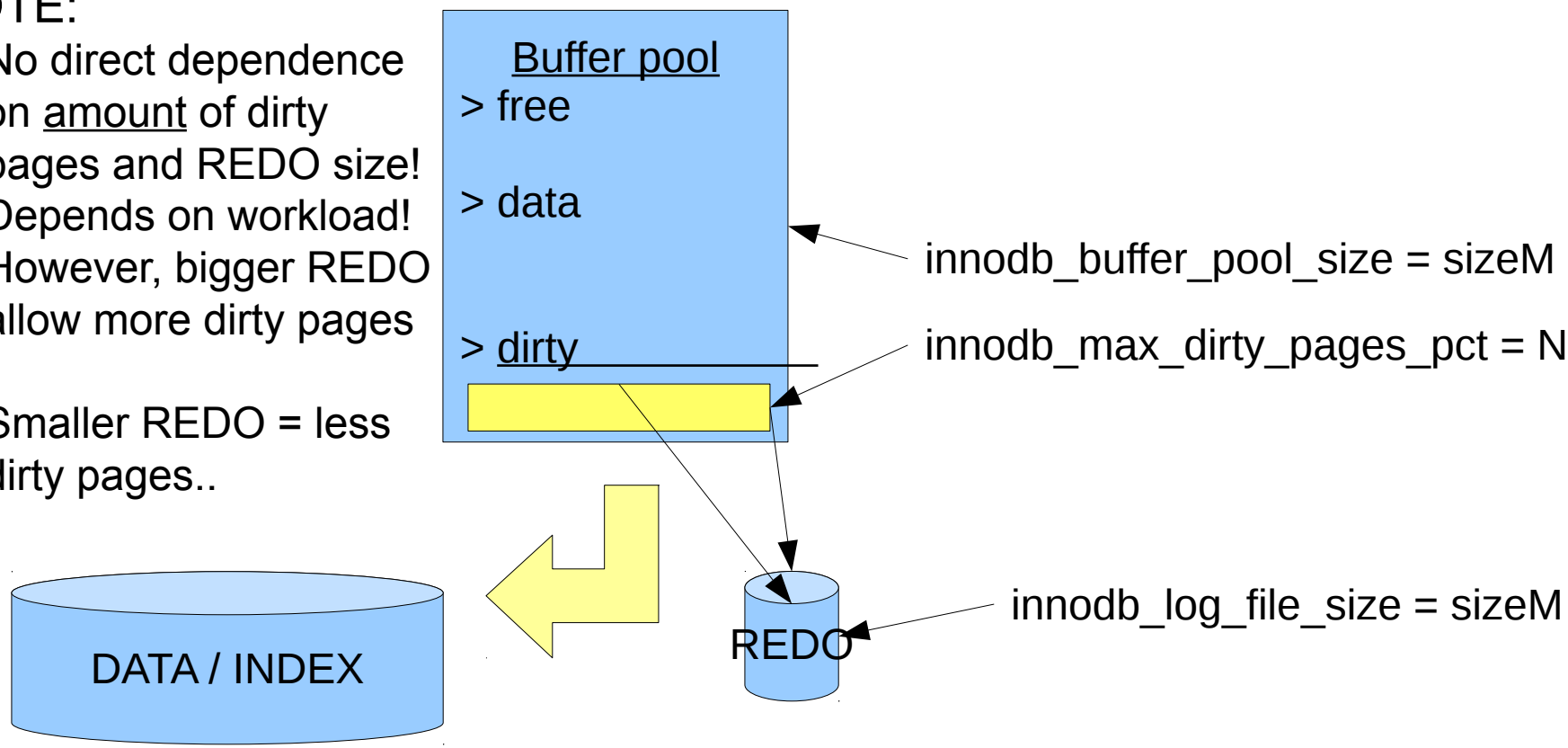
InnoDB : Dirty pages

- Direct dependence on REDO log size
- NOTE:
  - No direct dependence on amount of dirty pages and REDO size!
  - Depends on workload!
  - However, bigger REDO allow more dirty pages..



InnoDB : Dirty pages

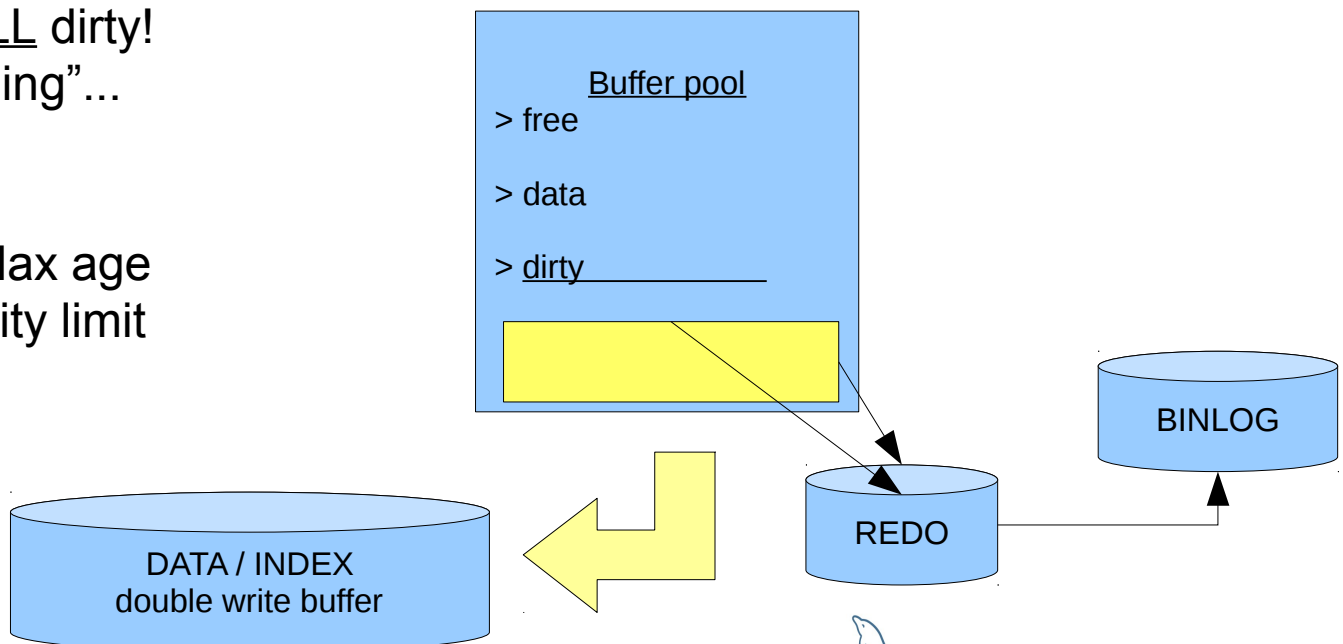
- Direct dependence on REDO log size
- NOTE:
  - No direct dependence on amount of dirty pages and REDO size!
  - Depends on workload!
  - However, bigger REDO allow more dirty pages
  - Smaller REDO = less dirty pages..



InnoDB : REDO log constraints

- REDO log constraints: (Always monitor Checkpoint Age!!!)
  - Cyclic, need free space
  - Checkpoint age: diff between the current LSN in REDO and the oldest dirty page LSN
  - Checkpoint age cannot out-pass the max checkpoint age (redo log size)
  - If Checkpoint age >= 7/8 of Max age => Flush ALL dirty!
    => AKA “furious flushing”...

- Adaptive Flushing:
  - Keep REDO under Max age
  - Respecting IO capacity limit



InnoDB : Adaptive Flushing

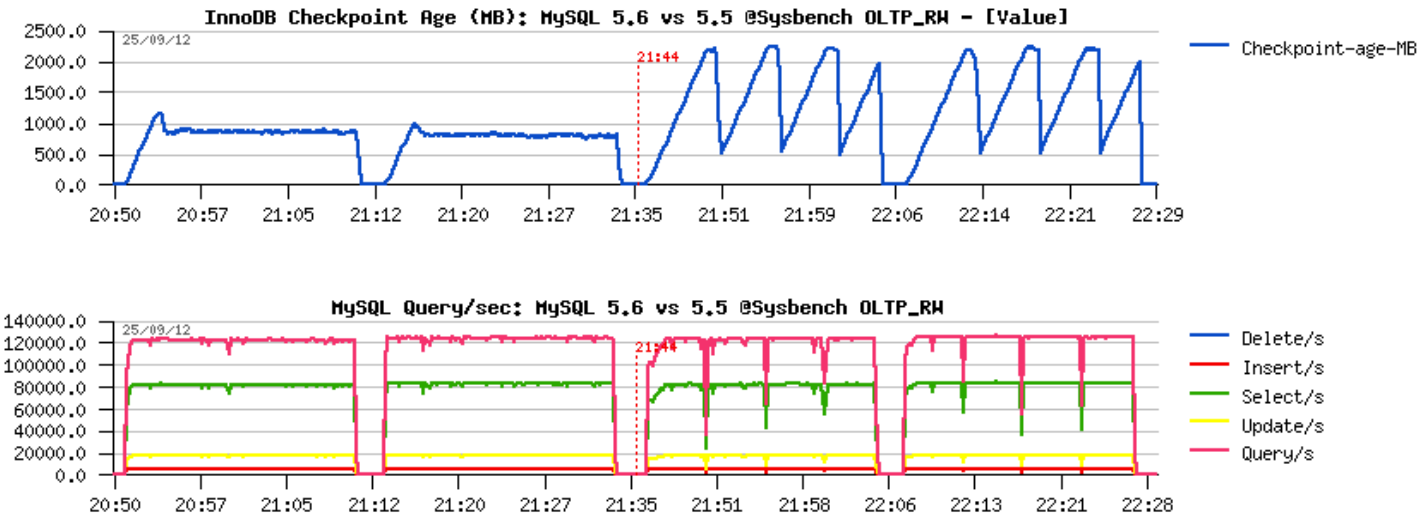
- MySQL 5.5:
  - Estimation based
  - Sometimes works ;-)

- MySQL 5.6 :
  - Based on REDO write rate + I/O capacity Max
  - Involving batch flushing with N pages to flush (progressive, depending on REDO %free) + page age limit (according REDO rate)

  - Tuning:
    - innodb_io_capacity
    - innofb_io_capacity_max
    - Both are dynamic!
    - Monitor Checkpoint Age..

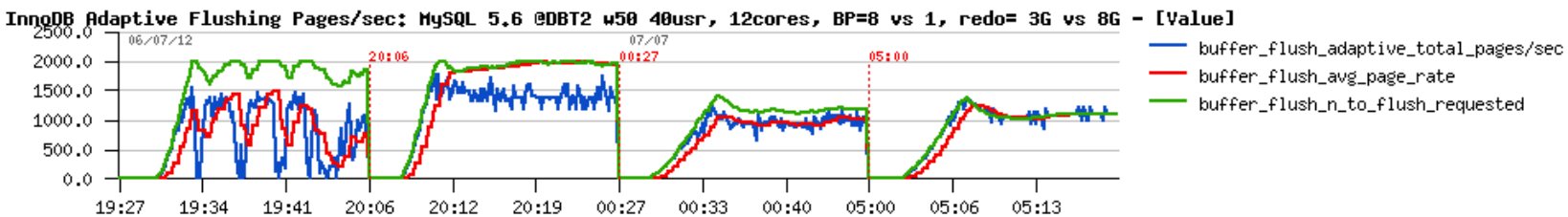
InnoDB : Adaptive Flushing in 5.5 vs 5.6

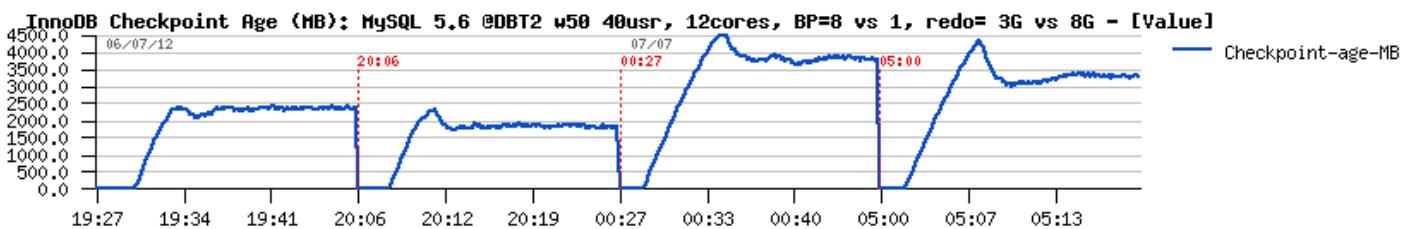
- OLTP_RW Workload:
  - Same IO capacity
  - Different logic..



InnoDB : tuning Adaptive Flushing in 5.6

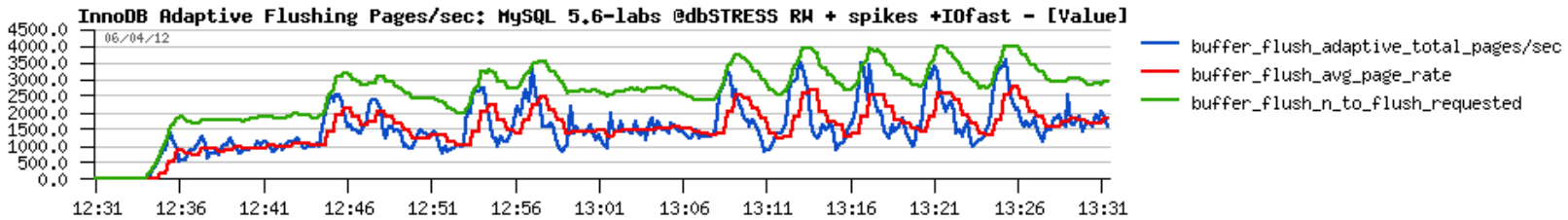
- Monitoring:
  - Flush adaptive pages/sec
  - Pages requested to flush
  - Flush AVG page rate

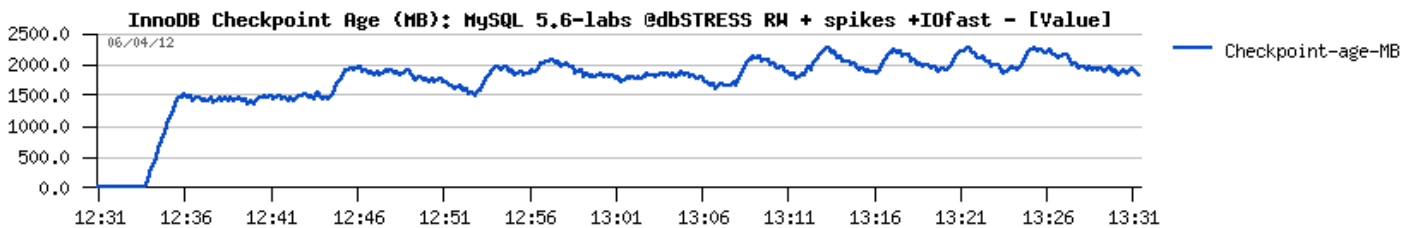


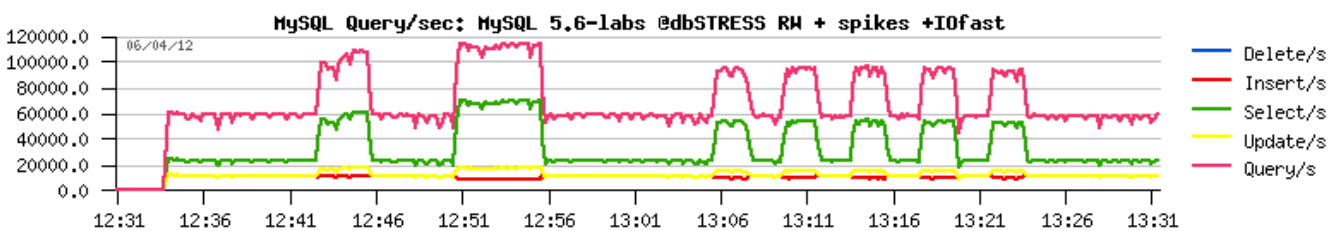


InnoDB : Resisting to activity spikes in 5.6

- dbSTRESS RW Workload + RW spikes:





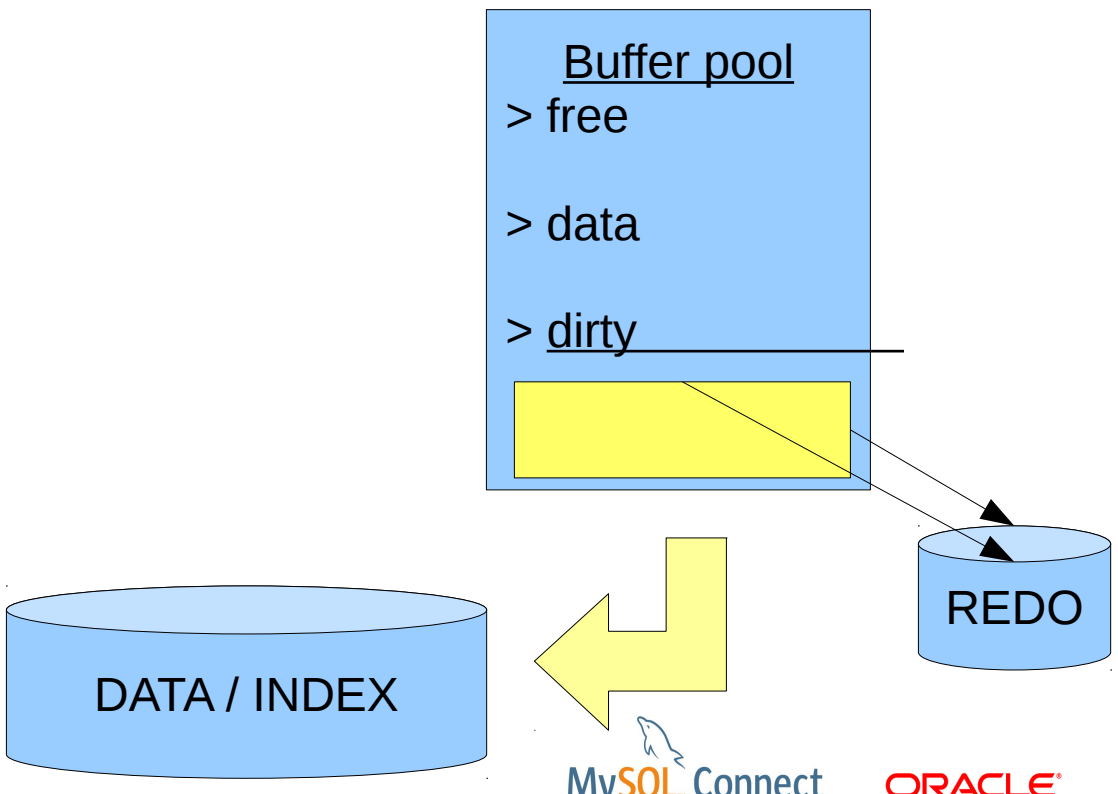


InnoDB : Why Purge Thread?..

- Reading source code:

Master Thread

```
loop: //Main loop
...
if( dirty pct > limit)
  flush_batch( 100% IO);
...
do {
  pages= trx_purge();
  if( 1sec passed )  flush_log();
} while (pages);
...
goto loop;
```



InnoDB : Purge Thread is the must!

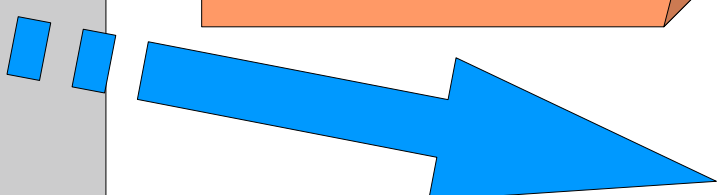
- Improved code:

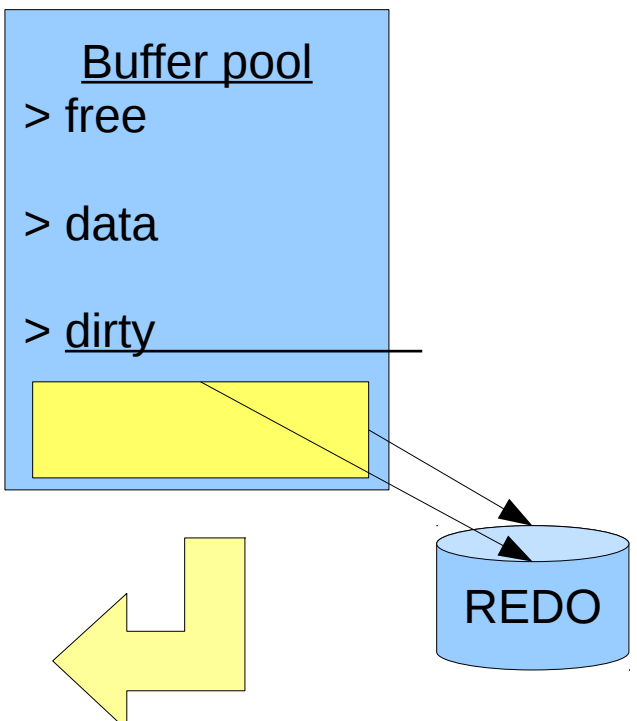
Master Thread

```
loop: //Main loop
...
sleep( 1 );
...
if( dirty pct > limit)
  flush_batch( 100% IO);
...
 flush_log();
...
goto loop;
```

Purge Thread

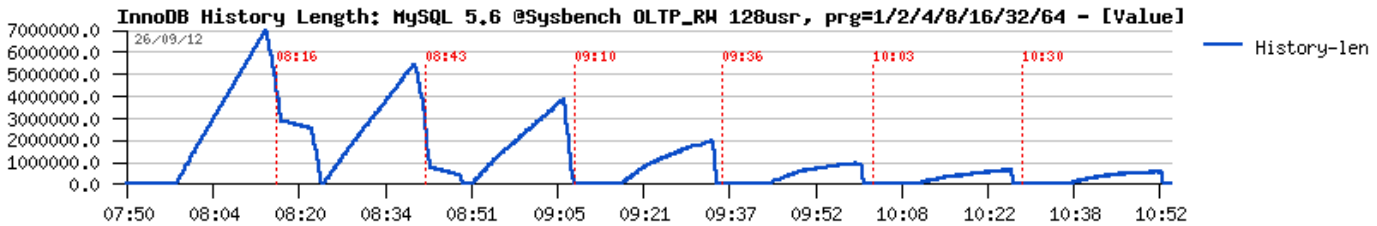
```
loop:
sleep( ... );
do {  pages=
     trx_purge();
} while (pages);
goto loop;
```

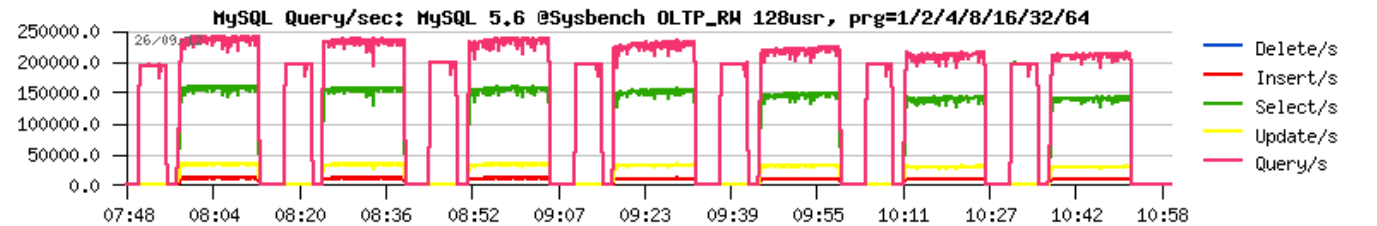




InnoDB : Purge improvement in 5.6

- Several Purge Threads :
  - NOTE: activation is auto-magical (I'm serious ;-))

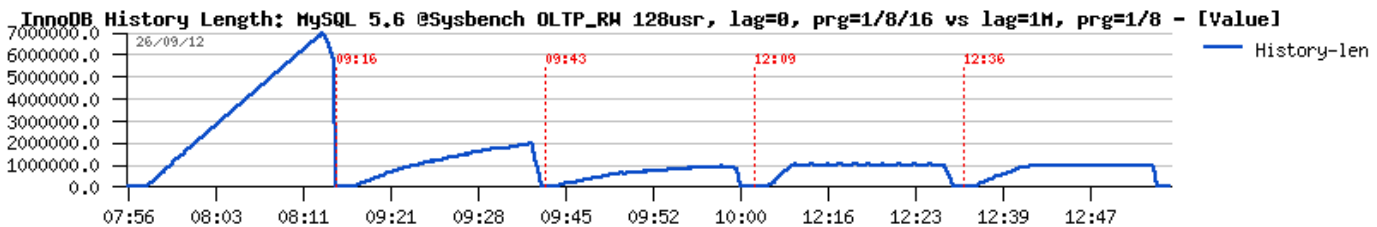


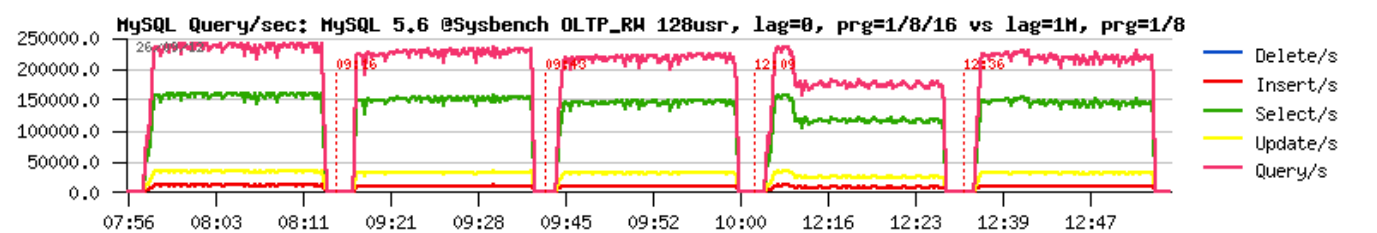


InnoDB : Purge improvement in 5.6

- Fixed max purge lag code!
  - innodb_max_purge_lag
  - innodb_max_purge_lag_delay <= configurable!

- Setting innodb_max_purge_lag=1M:





InnoDB : Other performance issues

- RW limitations:
  - Index lock..
  - Flushing limits..
  - Do we really using 100% of available I/O capacity?..
  - Work continues ;-)

- dbSTRESS workload :
  - Some mysterious contentions..
  - More in depth analyzing required..

MySQL Monitoring

- SQL> status;
- SQL> show global status;
- SQL> show processlist;
- SQL> show engine innodb status \G
- SQL> show engine innodb mutex ;
- INFORMATION_SCHEMA
- PERFORMANCE_SCHEMA !! <= let you surprise by “ps_helper” !!
- InnoDB METRICS TABLE !!
- etc..

MySQL Monitoring Tools

- MySQL Enterprise Monitoring!
- Cacti, etc.
- dim_STAT (http://dimitrik.free.fr)
  - All System load stats (CPU, I/O, Network, RAM, Processes,...)
  - Manly for Solaris & Linux, but any other UNIX too :-)
  - Add-Ons for Oracle, MySQL, PostgreSQL, Java, etc.
  - MySQL Add-Ons:
    - mysqlSTAT : all available data from “show status”
    - mysqlLOAD : compact data, multi-host monitoring oriented
    - mysqlWAITS : top wait events from Performance SCHEMA
    - InnodbSTAT : most important data from “show innodb status”
    - innodbMUTEX : monitoring InnoDB mutex waits
    - innodbMETRICS : all counters from the METRICS table
  - And any other you want to add! :-)

THANK YOU !!!

- All details about presented materials you may find on:
  - http://dimitrik.free.fr - dim_STAT, Benchmark Reports
  - http://dimitrik.free.fr/blog - Articles about MySQL Performance