MySQL

ORACLE

MYSQL WORLDWIDE CONFERENCE & EXPO

PERCONA LIVE

MySQL 5.6 : Performance Benchmarks, Tuning and “Best” Practices..

Dimitri KRAVTCHUK
MySQL Performance Architect
http://dimitrik.free.fr/blog / @dimitrik_fr

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.

ORACLE

Are you Dimitri?..

- Yes, it's me :-)
- Hello from Paris! ;-)
- Passionated by Systems and Databases Performance
- Previous 15 years @Sun Benchmark Center
- Started working on MySQL Performance since v3.23
- But during all that time just for fun only ;-)
- Since last years officially @MySQL Performance full time now
- http://dimitrik.free.fr/blog / @dimitrik_fr

ORACLE

Agenda

- Overview of MySQL Benchmarking
- Test workloads and their problems
- Workload oriented tuning
- Performance improvements in MySQL 5.6
- Benchmark results comparing MySQL 5.6, MySQL 5.5, MariaDB 5.5
- Pending issues..
- Q & A

Why benchmarking MySQL ?...

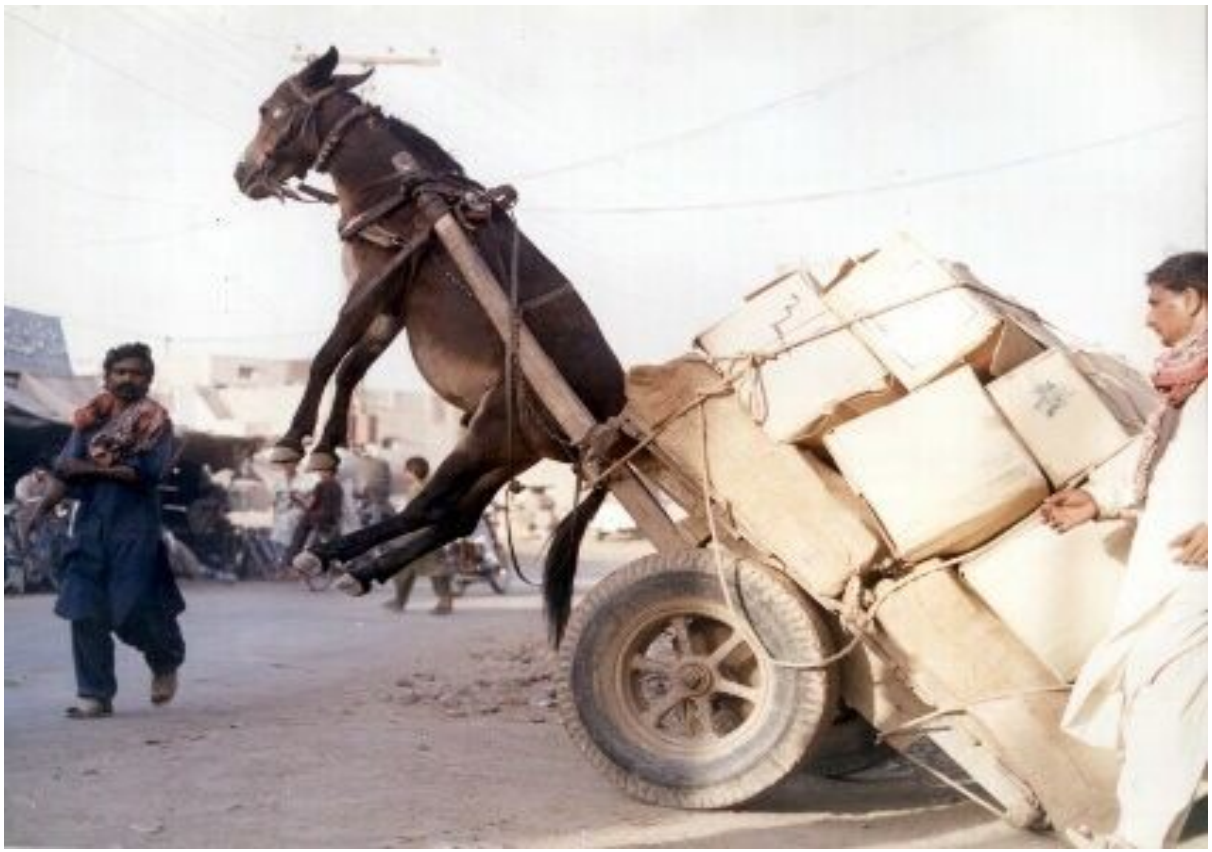
ORACLE

Why benchmarking MySQL?..

- Any solution may look “good enough”...

Why benchmarking MySQL?..

- Until it did not reach its limit..



Why benchmarking MySQL?..

- And even improved solution may not resist to increasing load..

Why benchmarking MySQL?..

- And reach a similar limit..

Why benchmarking MySQL?..

- A good benchmark testing may help you understand ahead the resistance of your solution to incoming potential problems ;-)

Why benchmarking MySQL?..

- But keep it in mind:
  - Even a very powerful solution but leaved in wrong hands may still be easily broken!... :-)

The Main MySQL Performance Tuning #1 Best Practice ?..

ORACLE

The Main MySQL Performance Tuning
#1 Best Practice ?..

USE YOUR BRAIN !!! :-)

ORACLE

Benchmarking: “Good” and “Ugly”..

- Have a clear goal!
  - Otherwise: I've obtained all these results, and now... so what?..
- Want to simulate production workload?..
  - Then just simulate it! (many SW available, not always OSS/free)
  - Hard to simulate? - adapt some generic tests
- Want to know capacity limits of a given platform?
  - Still try to focus on the test which are most significant for you!
- Want just to validate config settings impacts?
  - Focus on tests which are potentially depending on these settings
  - Or any, if the goal to prove there are not depending ;-)
- Well, just keep thinking about what you're doing ;-)

ORACLE

Focus on your test goal from the beginning

- Are you looking to test MySQL or System/ HW Setup/ etc?..
  - Are you sure HW is not a bottleneck on your tests?
  - Does your OS itself is able to scale?..
  - Is your test code reflecting your production workload?..
- Your test workload, is it OK?
  - Are you sure your test code is scaling?
  - Are you sure it's free of deadlocks at least on “application level”?..
  - Are you sure your test tool itself is not limited by HW resources?
  - Are you sure most of CPU time is not spent on your test tool execution, while you're supposed to test your MySQL server?.. ;-)
  - etc...

ORACLE

Check the limits of your platform from the beginning!

- Server:
  - Having faster CPU is still better! 32 cores is good enough ;-)
  - OS is important! - Linux, Solaris, etc.. (and Windows too!)
  - Right malloc() lib!! (Linux: jemalloc, Solaris: libumem)
- Storage:
  - Don't use slow disks! (except if this is a test goal :-))
  - SSD helping random access! (index/data) more and more cheaper
  - FS is important! - ZFS, UFS, QFS, VxFS, EXT3, EXT4, XFS, etc..
  - O_DIRECT or not O_DIRECT, AIO or not AIO, and be aware of bugs! ;-)
  - Do some generic I/O tests first (Sysbench, IObench, iozone, etc.)
- Don't forget network !! :-) (faster is better, 10Gbit is great!)

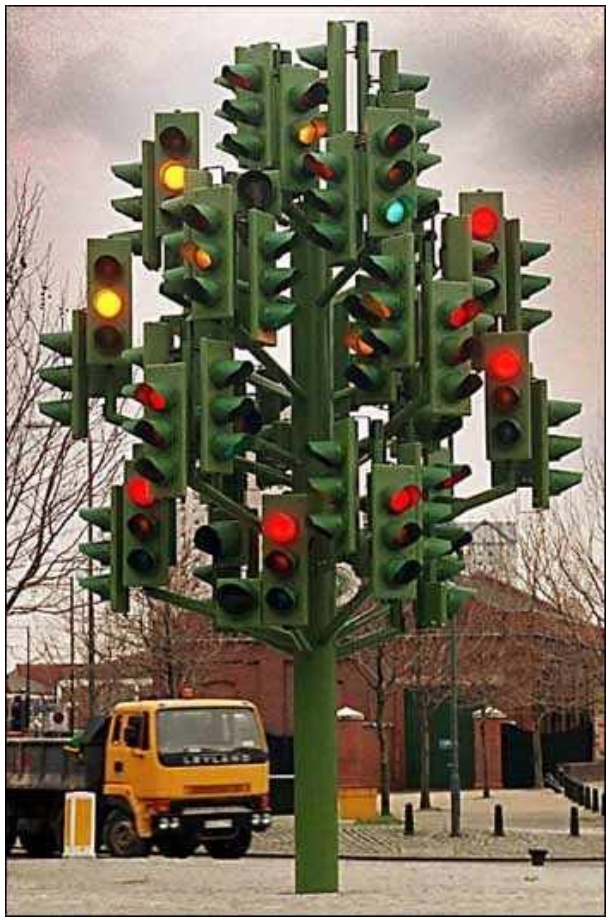
ORACLE

Few bad examples from real life..

- My backup is finished on Linux faster than on Solaris same HW
  - Be sure first there is **really** no more I/O activity once backup is “finished”
- Linux distro: your MySQL code has x4 regression! Fix it!
  - How did you see it? – Our test is taking x4 times more time
  - Which engine? – InnoDB
  - What is innodb_flush_log_at_trx_commit value? – set to 1.. why?
  - And if innodb_flush_log_at_trx_commit=2 ?.. – Oh! You fixed it!! Thx!!
  - Wait, what did you “improve” recently in distro? – FS flushing, why?
  - Just proving that you did not “fsync” on every fsync() before, that's all, but now you get it fixed ;-)

Test Workload

- Before to do something complex...
  - Be sure first you're comfortable with “basic” operations!
  - Single table?
  - Many tables?
  - Short queries?
  - Long queries?
- Remember: any complex load just represents a mix of simple operations..
  - So, start from as simple as possible..
  - And then increase complexity progressively..



Popular “Generic” Test Workloads @MySQL

- Sysbench
  - OLTP, RO/RW, 1-table, since v0.5 N-table, lots load options, deadlocks
- DBT2 / TPCC-like
  - OLTP, RW, very complex, growing db, no options, deadlocks
  - In fact using mostly only 2 tables! (thanks Performance Schema ;-))
- dbSTRESS
  - OLTP, RO/RW, several tables, one most hot, configurable, no deadlocks
- linkbench (Facebook)
  - OLTP, very interesting test workload, yet to analyze ;-)
- DBT3
  - DWH, complex heavy query, loved by Optimizer Team ;-)

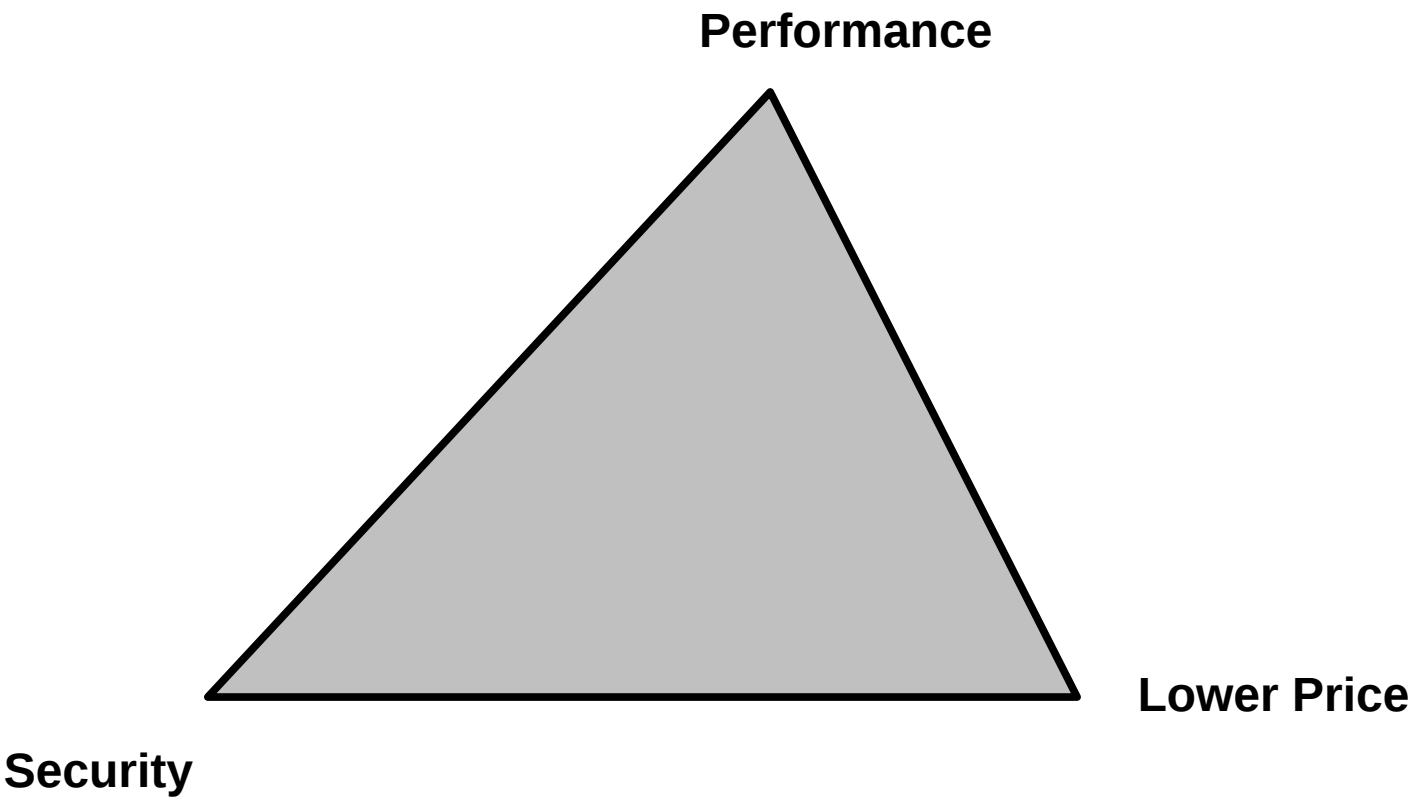
ORACLE

MySQL Storage Engines: focus on InnoDB

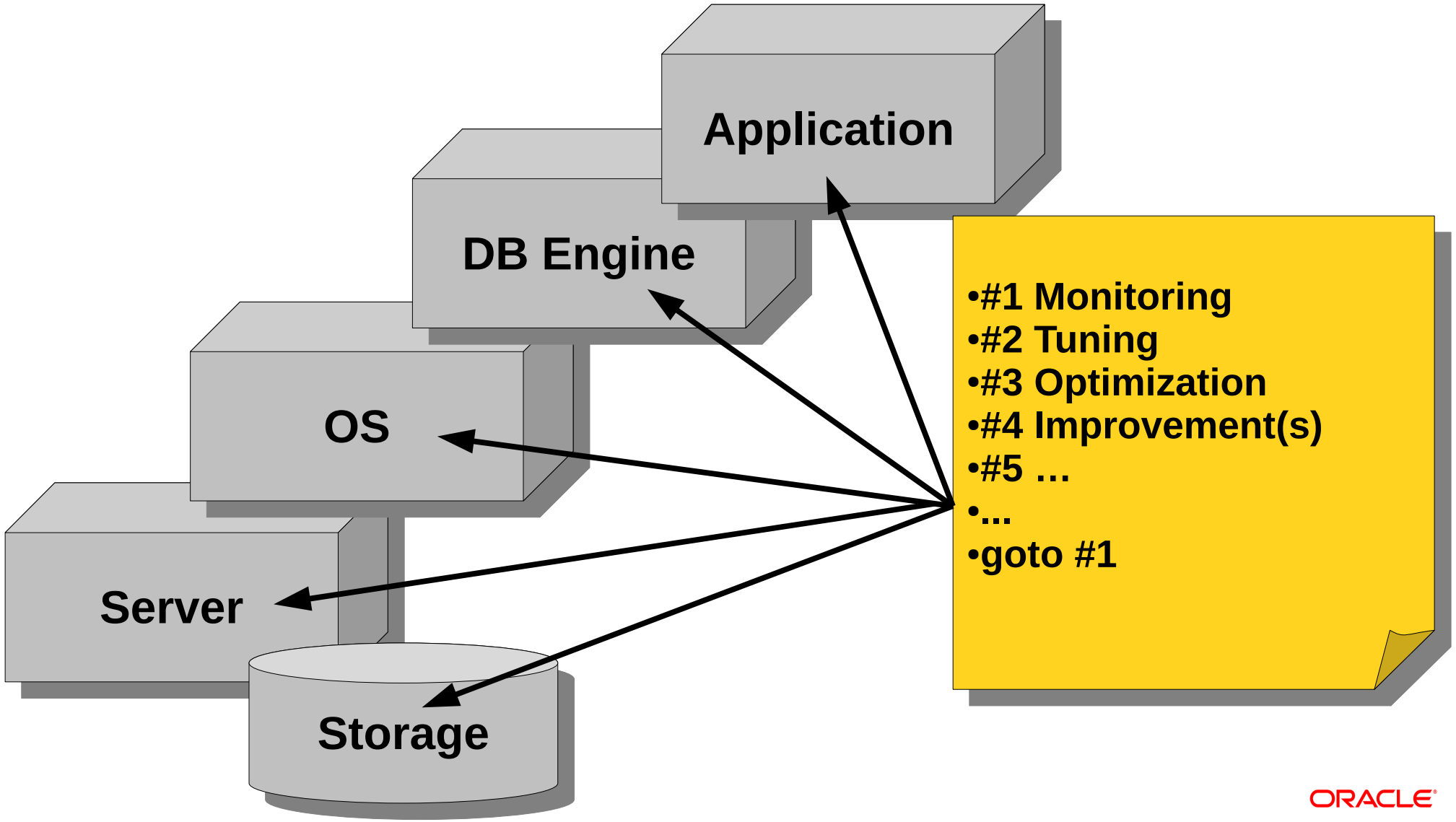
- MyISAM
  - Table level locking for everything, cache(s) for indexes only
  - Easily spending 50% of time on syscalls to read() data..
  - Many global locks, no transactions, no recovery..
- InnoDB
  - Row locking, transactions for everything, Buffer Pool
  - Double write buffer? Checksums?
  - innodb_flush_log_at_trx_commit= 1 / 2 ??
- Binlog / Replication
  - Sync? - binlog group commit is since MySQL 5.6

ORACLE

Seek for your best option..

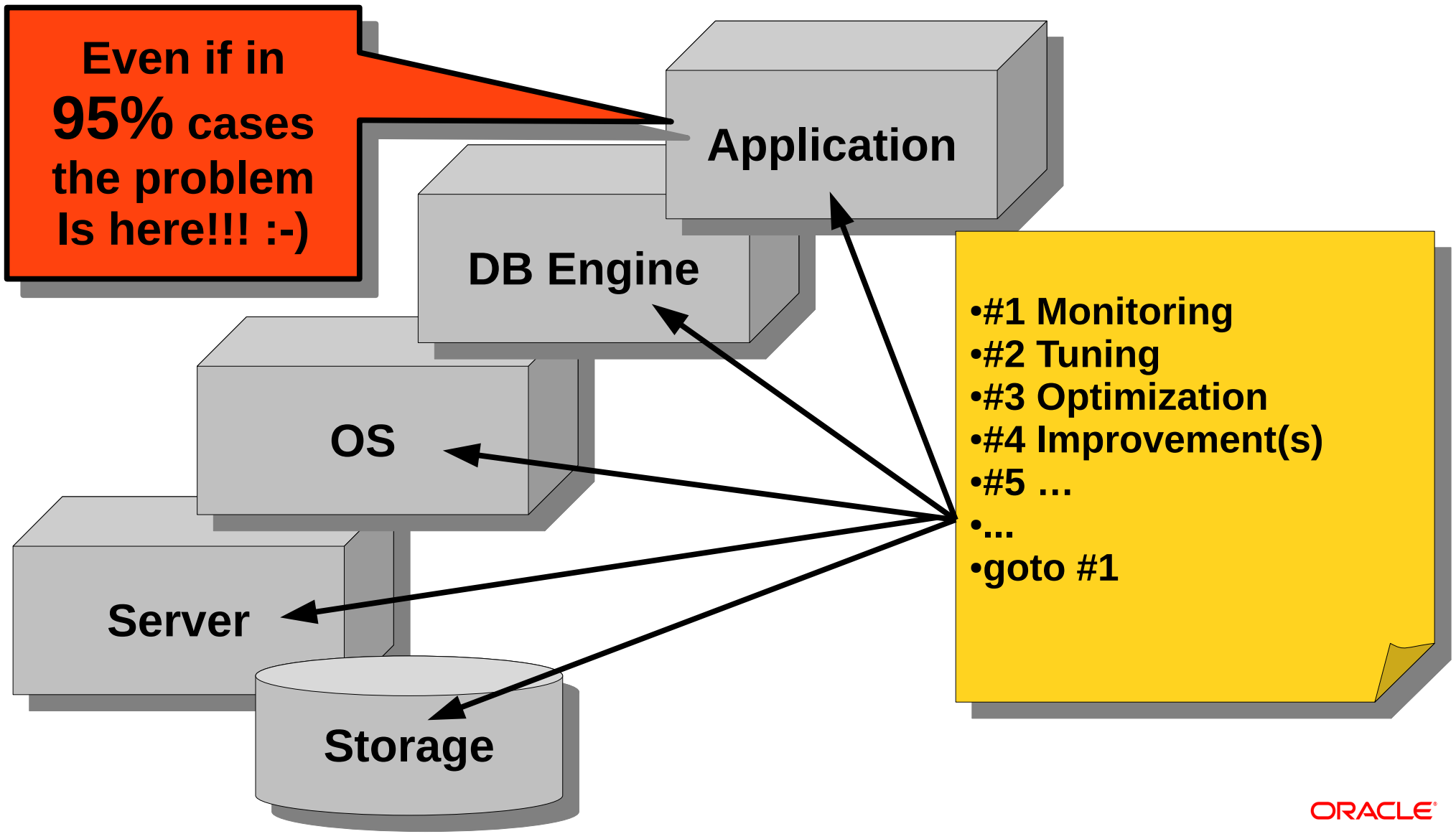


And then Infinitive Loop of Database Tuning begins...

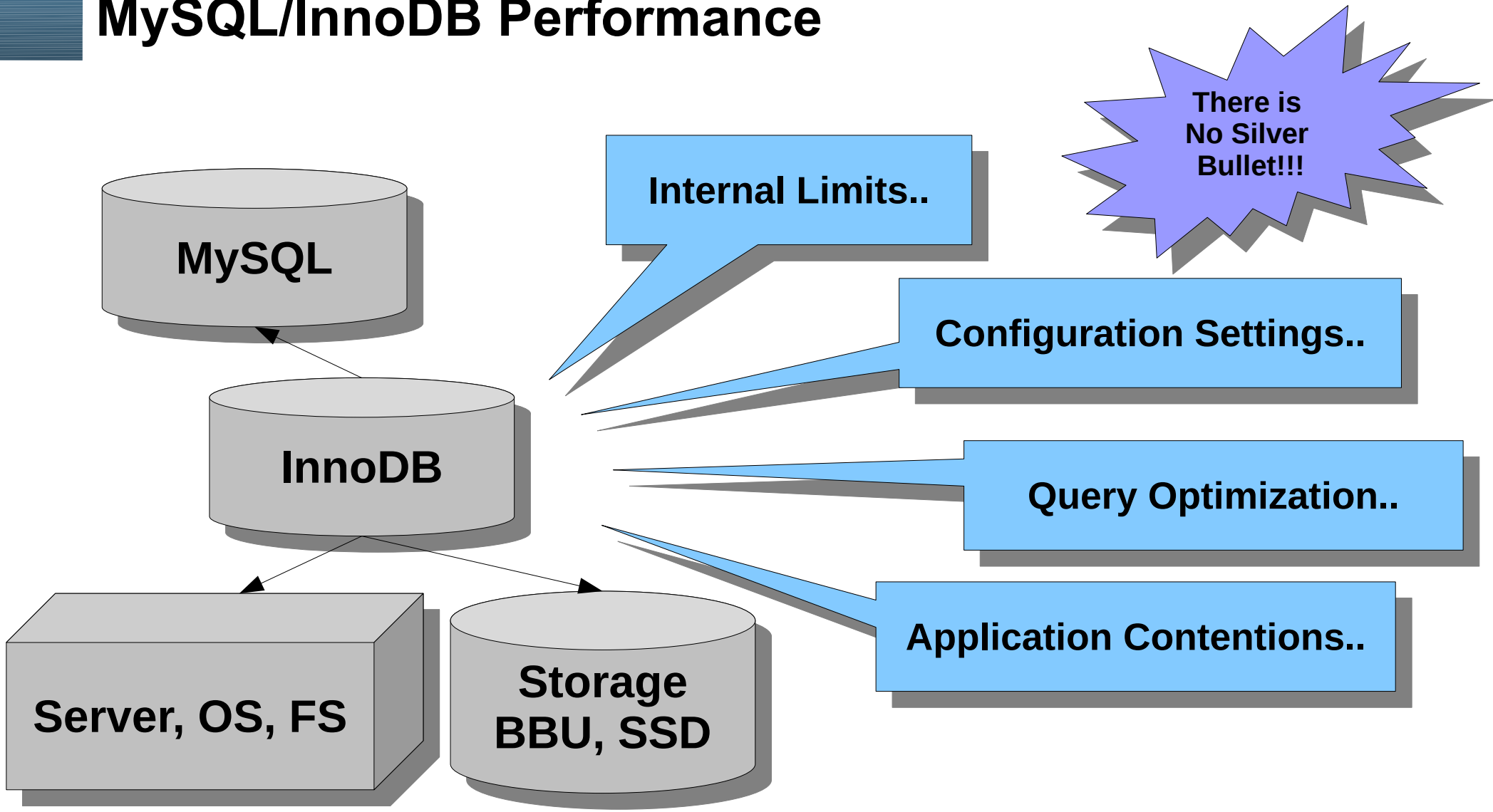


ORACLE

The Infinitive Loop of Database Tuning...



MySQL/InnoDB Performance



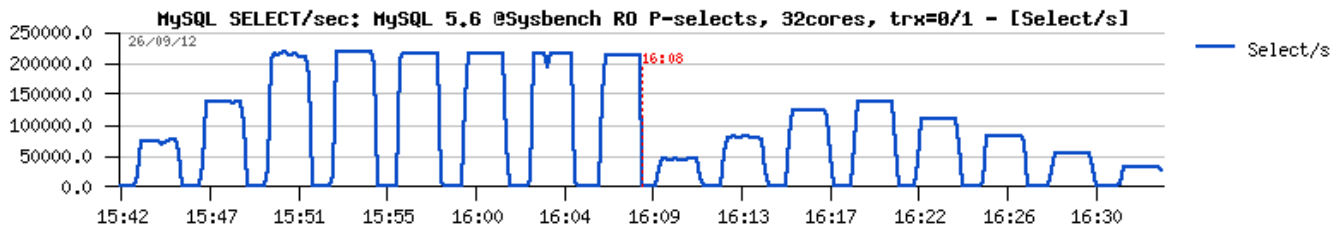
Workload: Read-Only

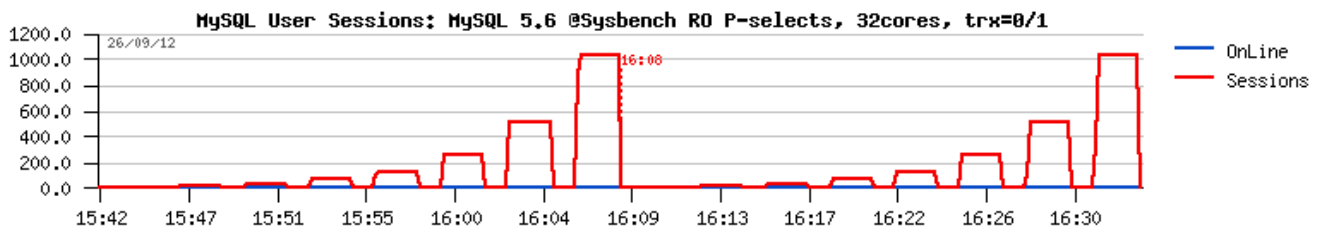
- Bigger Buffer Pool (BP) is better
  - BP < dataset = IO-bound
- TRX list (before kernel_mutex, since 5.6: trx_sys mutex)
- Read view
- Auto-commit or transactions?..
  - Grouping **many** queries within a single transaction may also largely reduce MDL locking, but still keep them **short** ! (check with PFS)
- Prepared statements
  - Observed 10% performance improvement in 5.6 (while Parser time is not more than 3% according to profiler)..
- Read-Only transactions!

ORACLE

InnoDB: Read-Only Transactions in 5.6

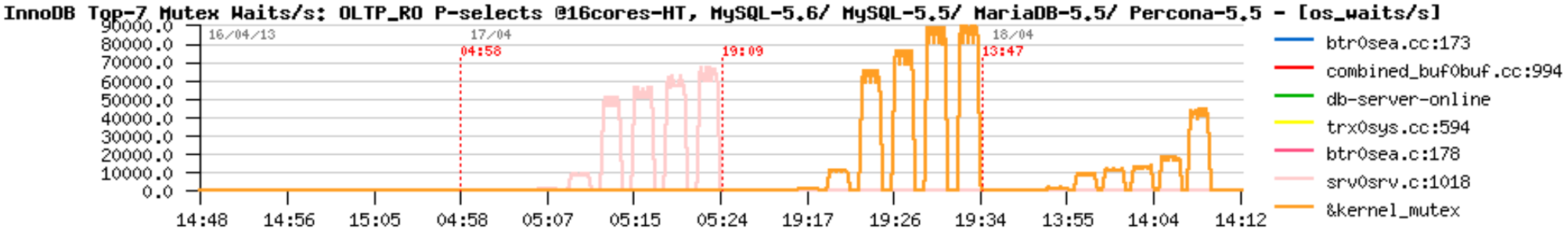
- Sysbench OLTP_RO Point-Selects:
  - Concurrent user sessions: 1, 2, 4 .. 1024
  - Using of transactions in sysbench = 0 / 1

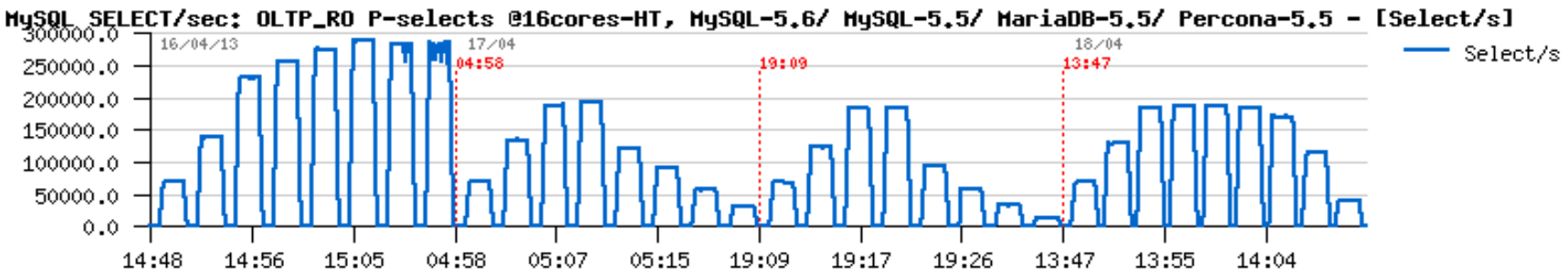


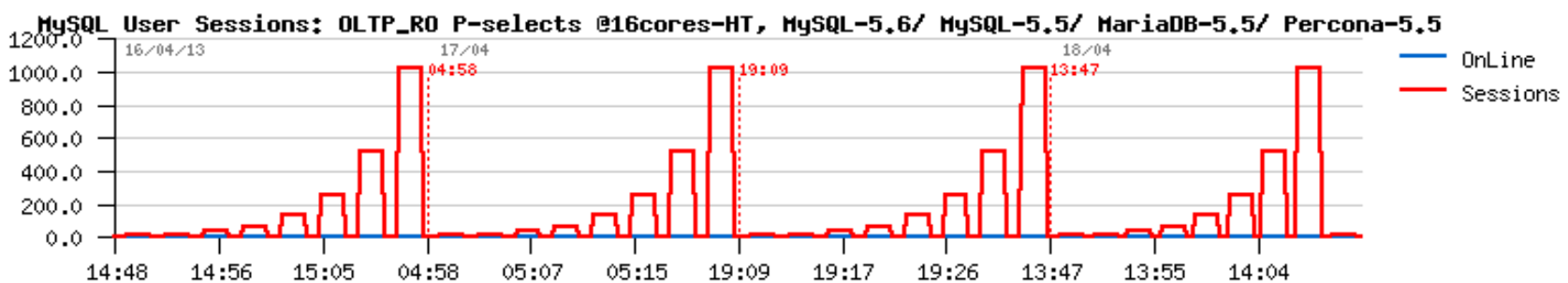


ORACLE

InnoDB: Read-Only Transactions in 5.6



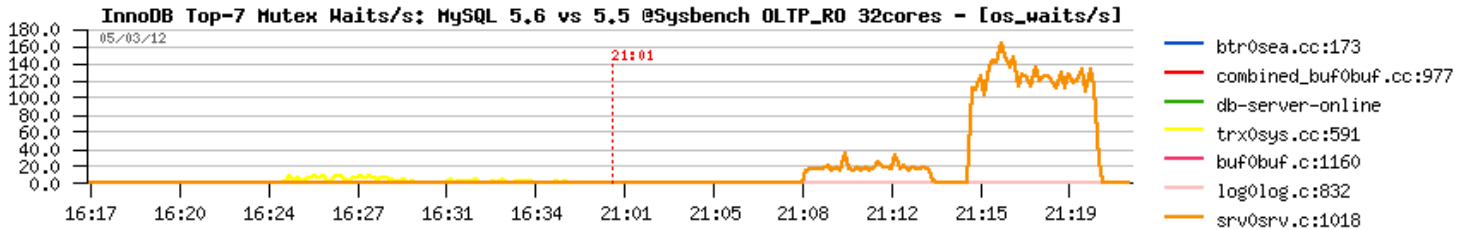


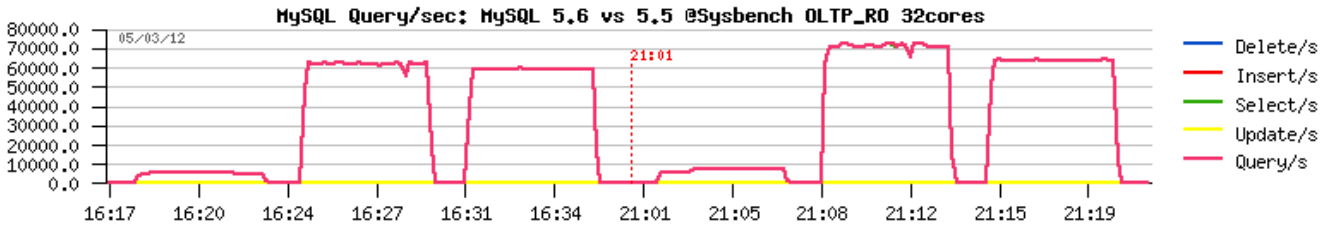


ORACLE

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:

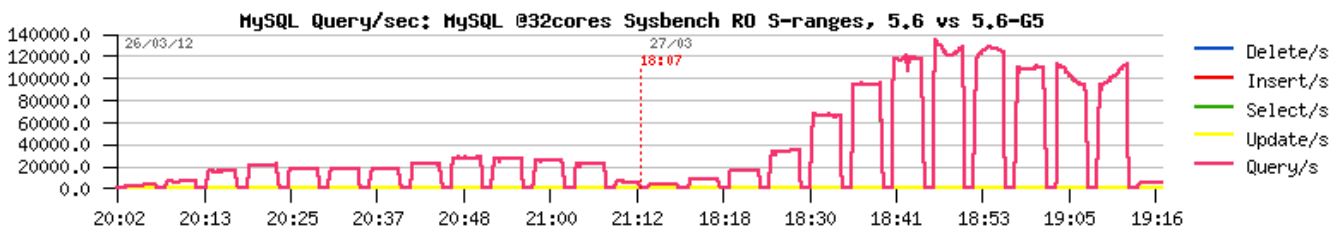


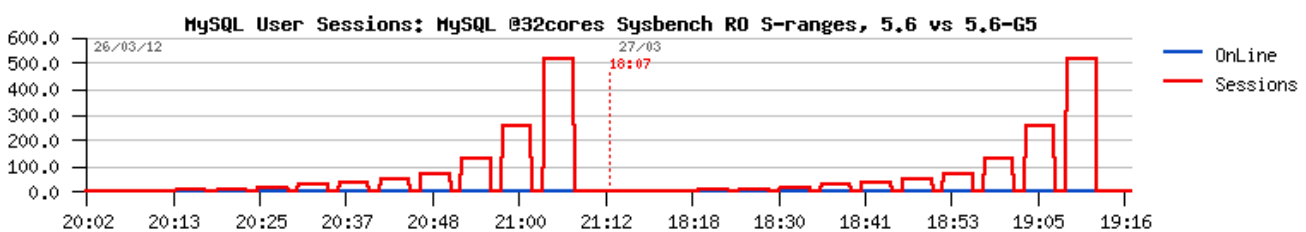


ORACLE

InnoDB : false sharing of cache-line fixed!

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





ORACLE

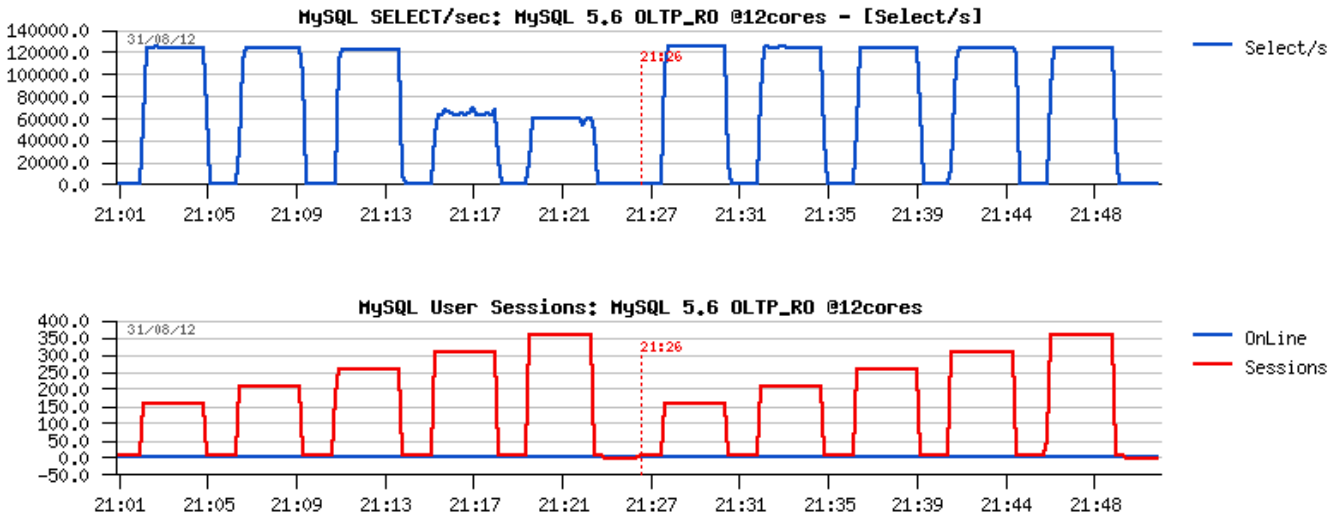
MySQL Internals: “killer” LOCK_open mutex

- 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: several table open cache instances
  - 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

ORACLE

MySQL 5.6 Internals : low table_open_cache

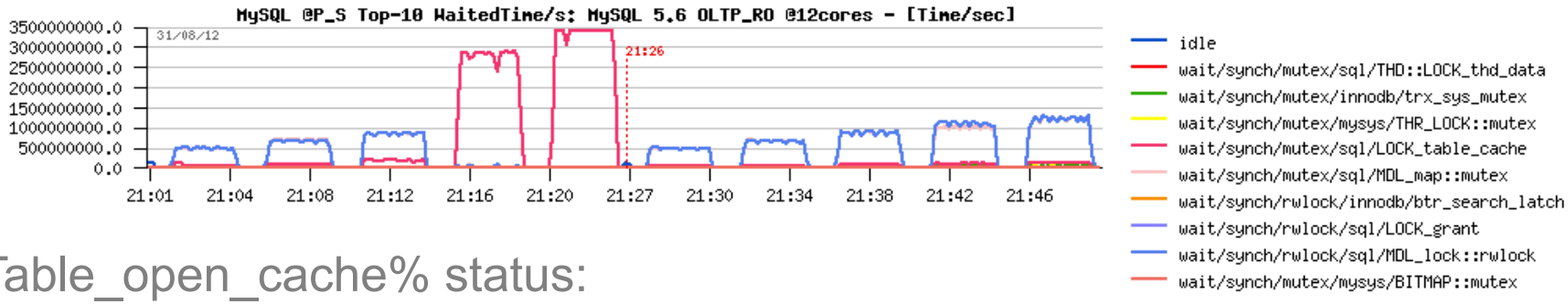
- MySQL 5.6 :
  - Not big enough “table_open_cache” setting



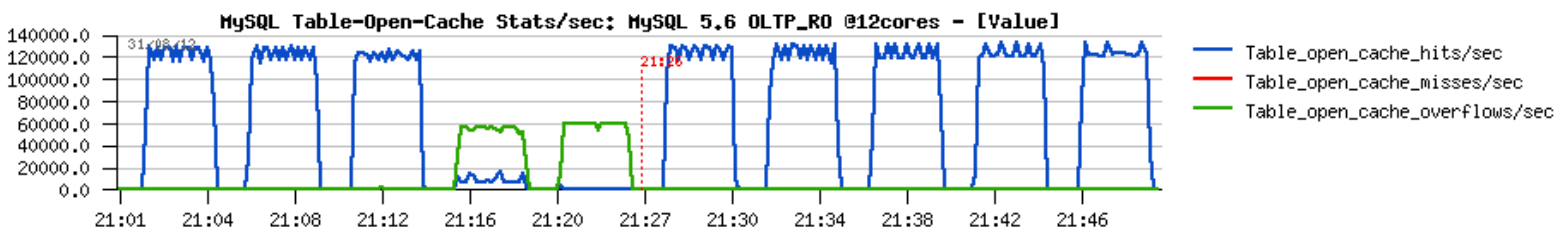
ORACLE

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:



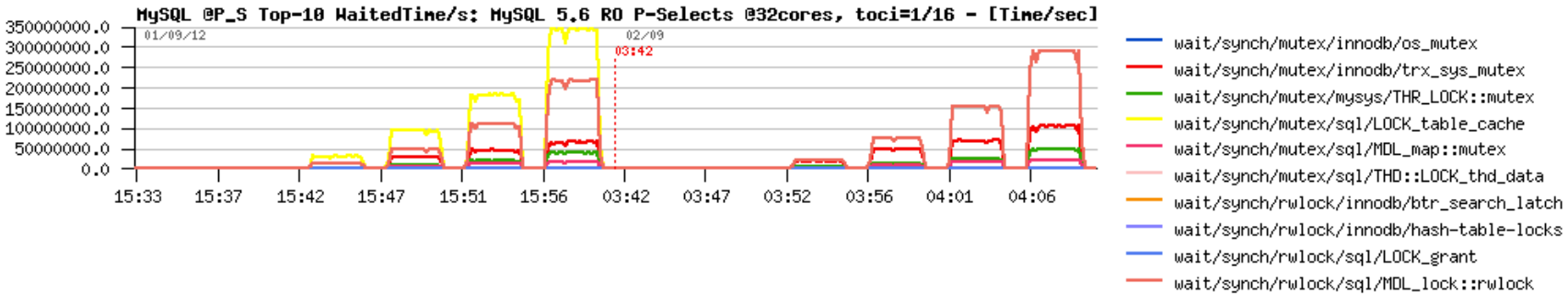
  - Table_open_cache% status:

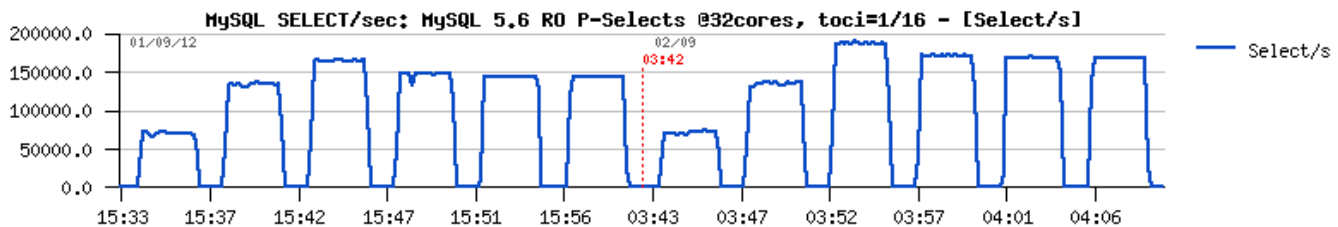


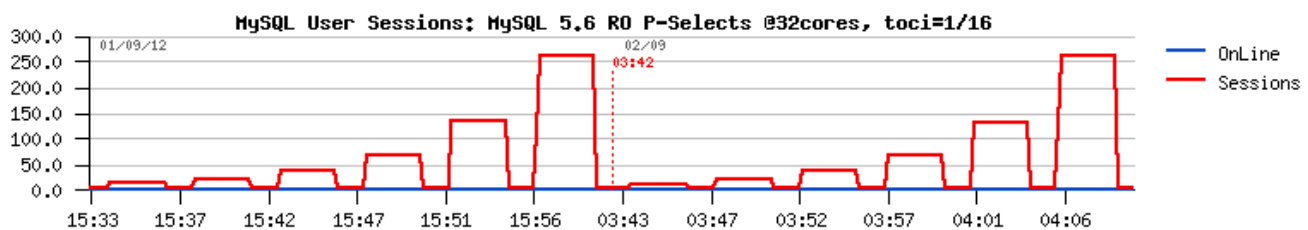
ORACLE

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:







ORACLE

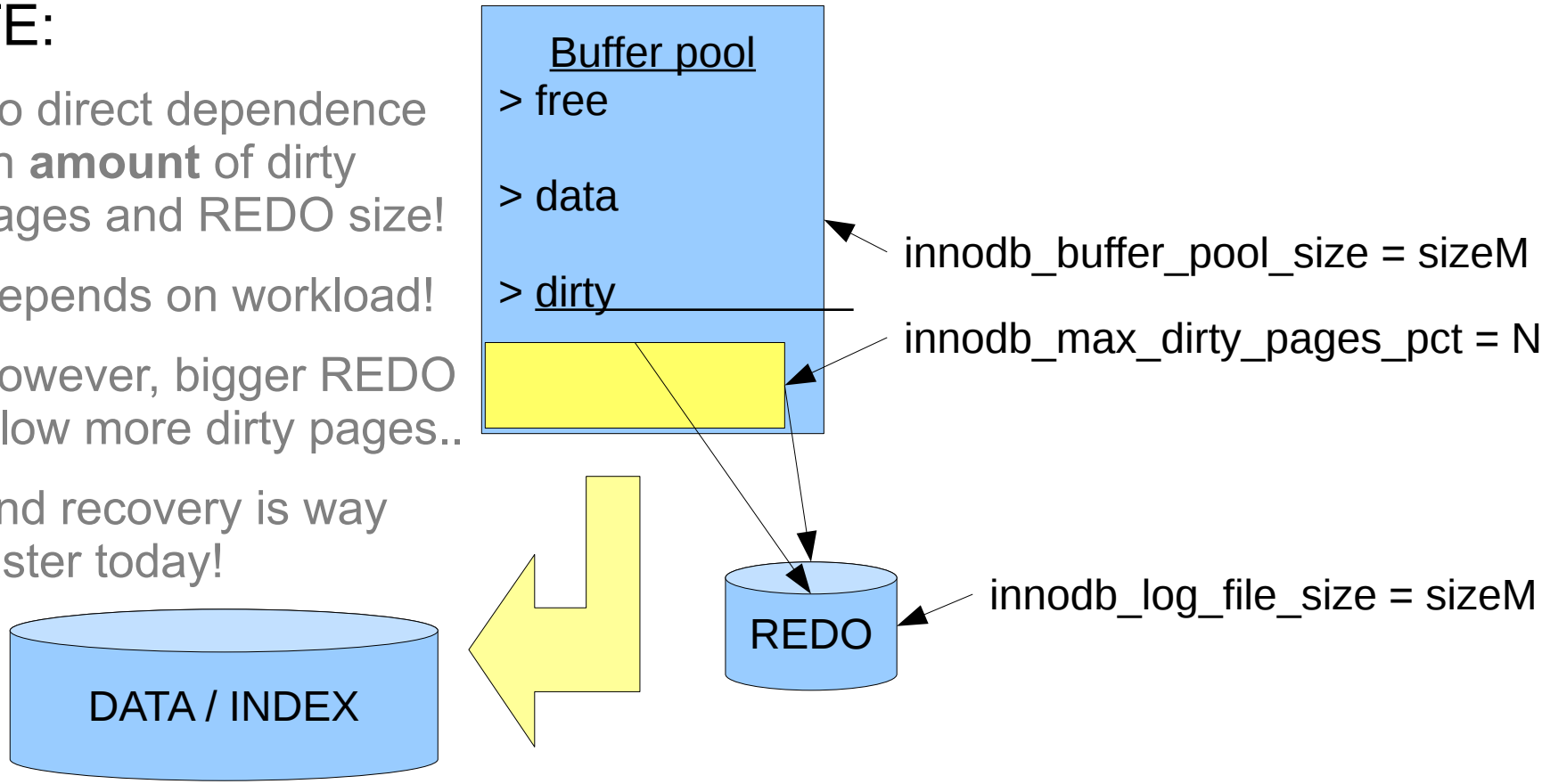
Workload: Read-Write

- RW activity
  - Updates only? Insert? Delete? R/W %ratio?
- Bigger Buffer Pool (BP) is still better
  - BP < dataset = IO-bound Reads(!) or R+W
  - BP > dataset = CPU-bound or IO-bound Writes(!)
- REDO size matters a lot! (up to 2TB in 5.6)
- Adaptive Flushing matters a lot!
- LRU flushing matters a lot too!
- Tip: Neighbor Pages flushing = off

ORACLE

InnoDB: Dirty pages, Flushing

- 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..
  - And recovery is way faster today!



ORACLE

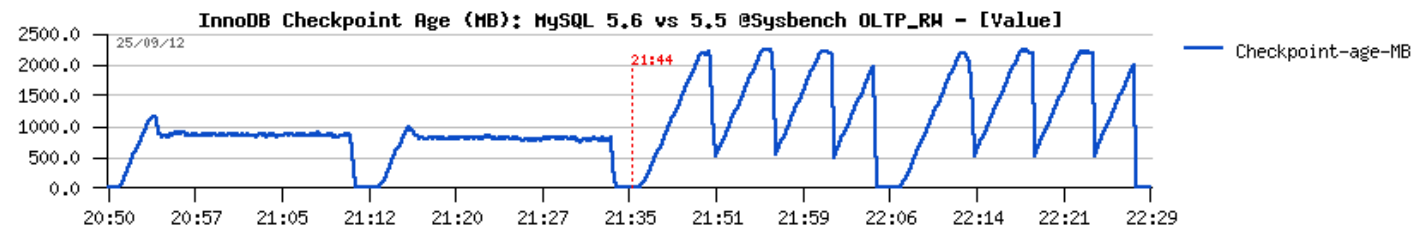
InnoDB: Adaptive Flushing

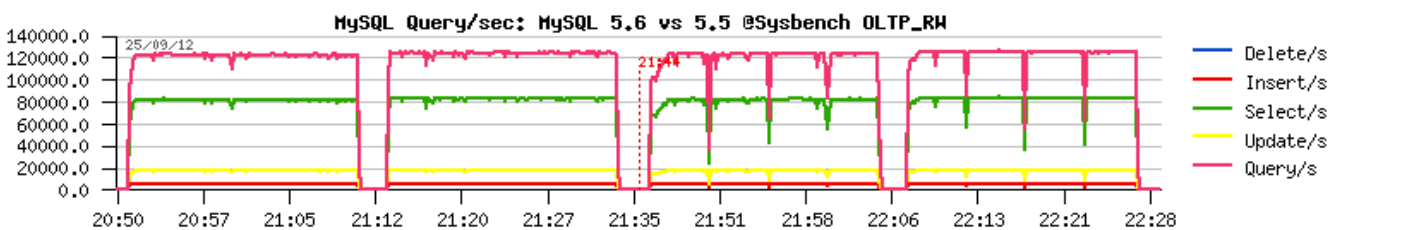
- MySQL 5.5:
  - Estimation based
  - Sometimes works ;-)
- MySQL 5.6 :
  - Based on REDO write rate + I/O capacity Max
  - Involving batch flushing with N pages to flush (progressive, depending on REDO %free) + page age limit (according REDO rate)
- Tuning:
  - innodb_io_capacity
  - innofb_io_capacity_max
  - Both are dynamic!
  - Monitor Checkpoint Age..

ORACLE

Adaptive Flushing: MySQL 5.6 vs 5.5

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

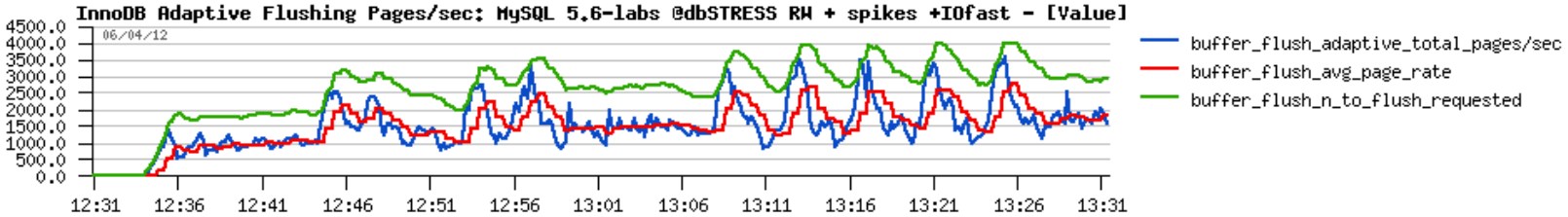


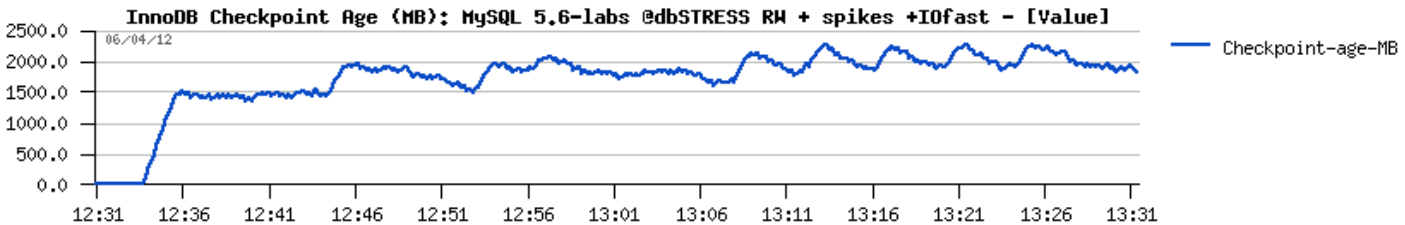


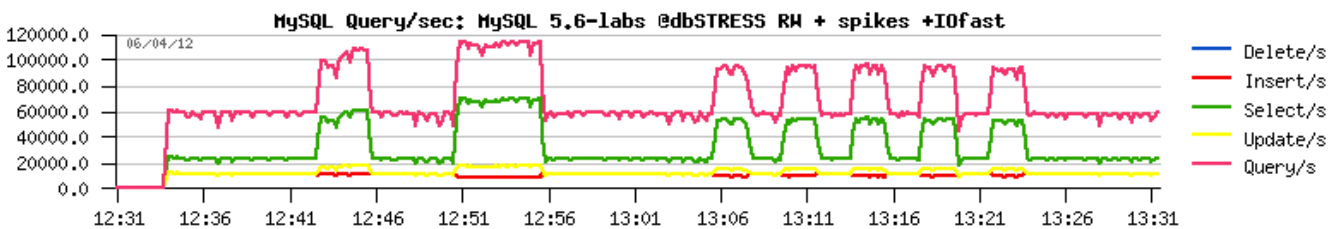
ORACLE

InnoDB : Resisting to activity spikes in 5.6

- dbSTRESS R+W with spikes







ORACLE

User Concurrency scenarios

- Single user?..
  - With a bigger code path today 5.6 simply cannot be faster than 5.5
  - But then, why you're not considering Query Cache? ;-)
- More users?..
  - Up to 8-16 concurrent users all internal contention are not yet hot
  - So, 5.6 will not be better yet..
- More than 16 users?..
  - Then you'll feel a real difference, but if you have at least 16cores ;-)
  - Or if you have really a lot of concurrent users
- But don't forget other 5.6 improvements either!
  - On-line DDL, Binlog group commit, Memcached, etc..

ORACLE

High Concurrency Tuning

- If bottleneck is due a concurrent access on the same data (due application design) – ask dev team to re-design ;-)
- If bottleneck is due MySQL/InnoDB internal contentions, then:
  - If you cannot avoid it, then at least don't let them grow ;-)
  - Try to increase InnoDB spin wait delay (dynamic)
  - Try innodb_thread_concurrency=N (dynamic)
  - CPU taskset / prcset (Linux / Solaris, both dynamic)
  - Thread Pool
  - NOTE: things with contentions may radically change since 5.7, so stay tuned ;-)

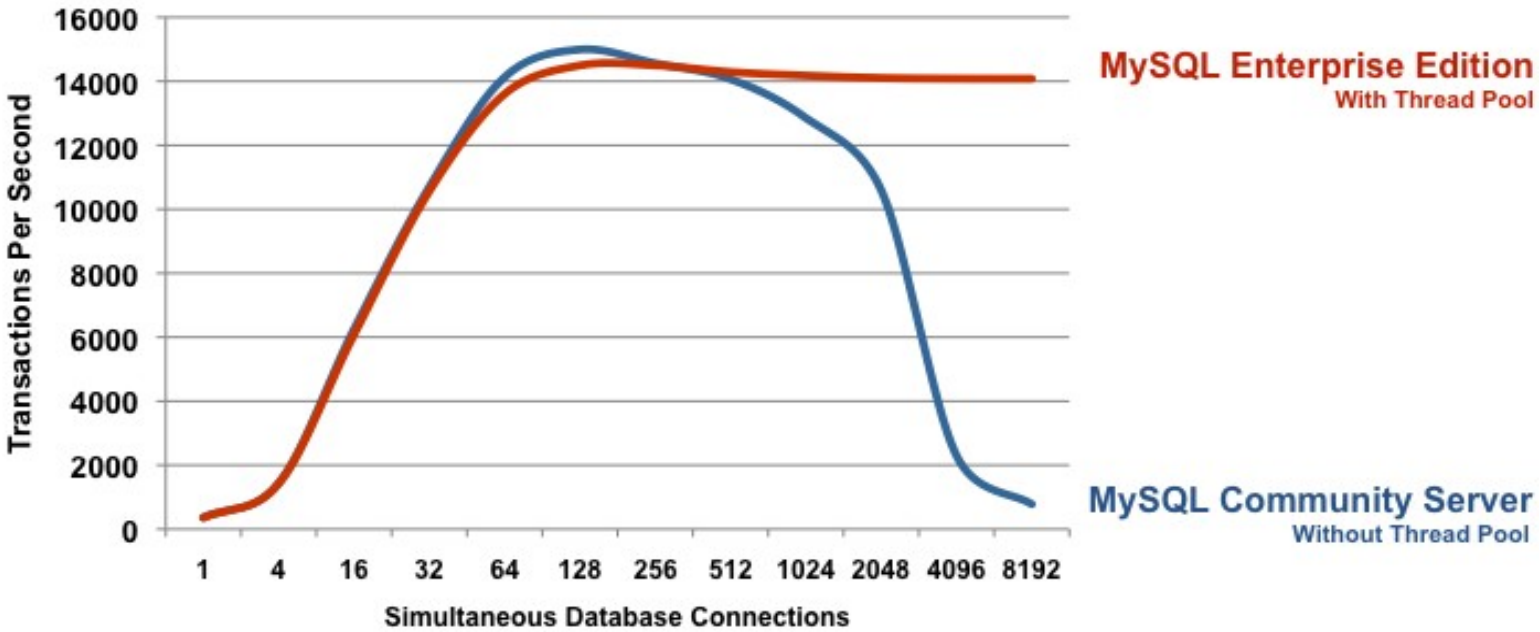
ORACLE

Thread Pool @MySQL

- None of these solutions will help to increase performance!
  - it'll just help to keep the peak level constant (and you yet need to discover on which level of concurrency you're reaching your peak ;-))
- ThreadPool in MySQL 5.5 and 5.6 is aware if I/O are involved!
  - So, better than innodb thread concurrency setting or taskset
  - May still require spin wait delay tuning!
  - The must for high concurrency loads!
  - May still start to show a difference since 32-128 concurrent users! (all depends on workload)..
  - Keep in mind that OS scheduler is not aware how to manage user threads most optimally, but ThreadPool does ;-)

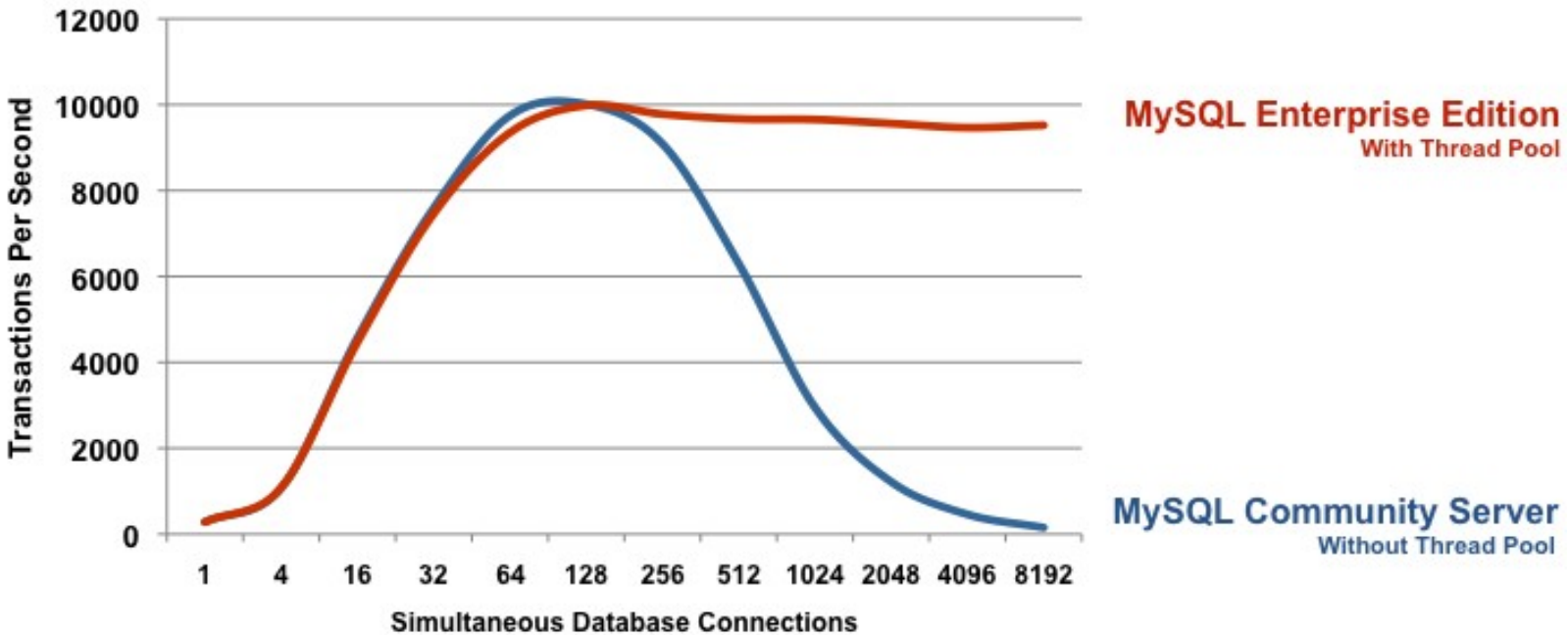
Thread Pool in MySQL 5.6

- OLTP_RO:



Thread Pool in MySQL 5.6

- OLTP_RW:



InnoDB High Concurrency: AHI

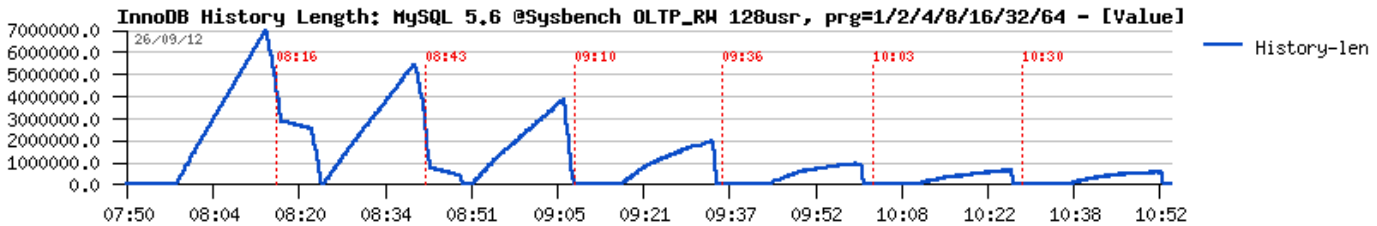
- Adaptive Hash Index (AHI)
  - Helps a lot on Read-Only workloads
  - In fact it helps always until itself become not actively modified
  - AHI contention is seen as its btr_search_latch RW-lock contetnion
  - So, on Read+Write become a huge bottleneck..
  - In many cases on RW the result is better with AHI=off..
  - NOTE: there is still a big mystery around AHI when it's having btr_search_latch contention even when there is no changes at all (pure RO in memory).. - expected to be fixed in 5.7 ;-)

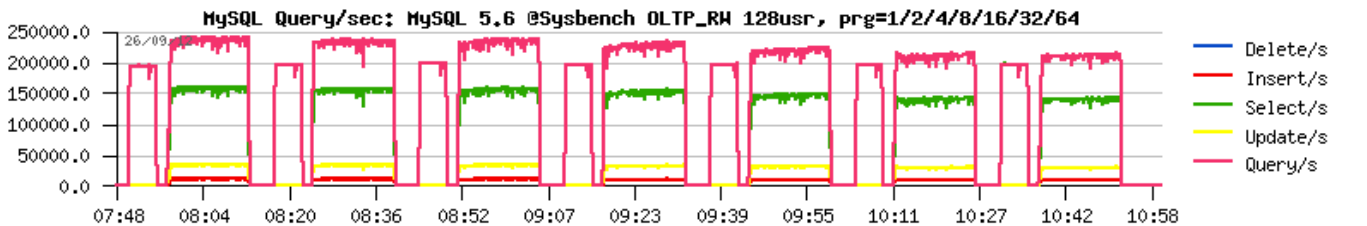
InnoDB Purge

- Purging (similar to Garbage Collecting)
  - Since MySQL 5.5: purge thread
  - Since MySQL 5.6: purge thread(s) (up to 32)
- Having Purge following workload is very important!
  - Ex.: On aggressive RW got 400GB of undo records within few hours(!)
  - Then it took days to reach zero in History Length..
- The main problem is the past – how to dose it?..
  - Since 5.6: with many threads, Purge become auto-stable itself
  - Still missing a dynamic config option to say how many purge threads to run in parallel right now (but it'll be fixed soon ;-))

InnoDB : Purge improvement in 5.6

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

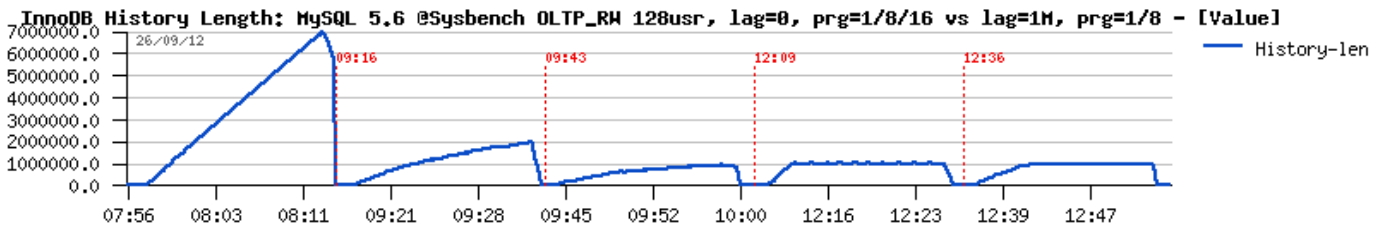


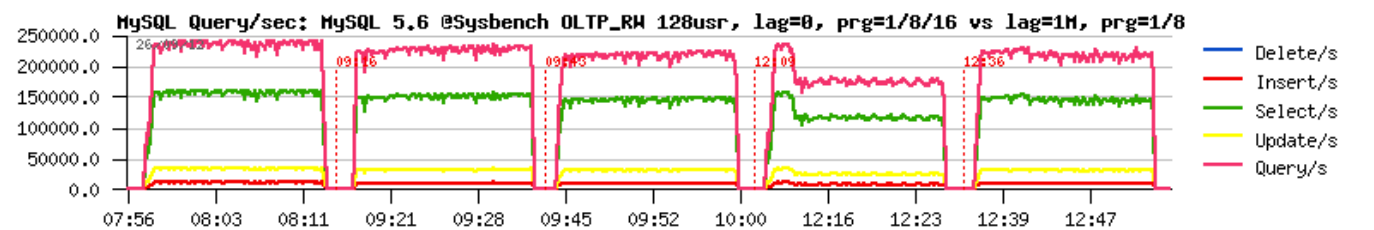


ORACLE

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:



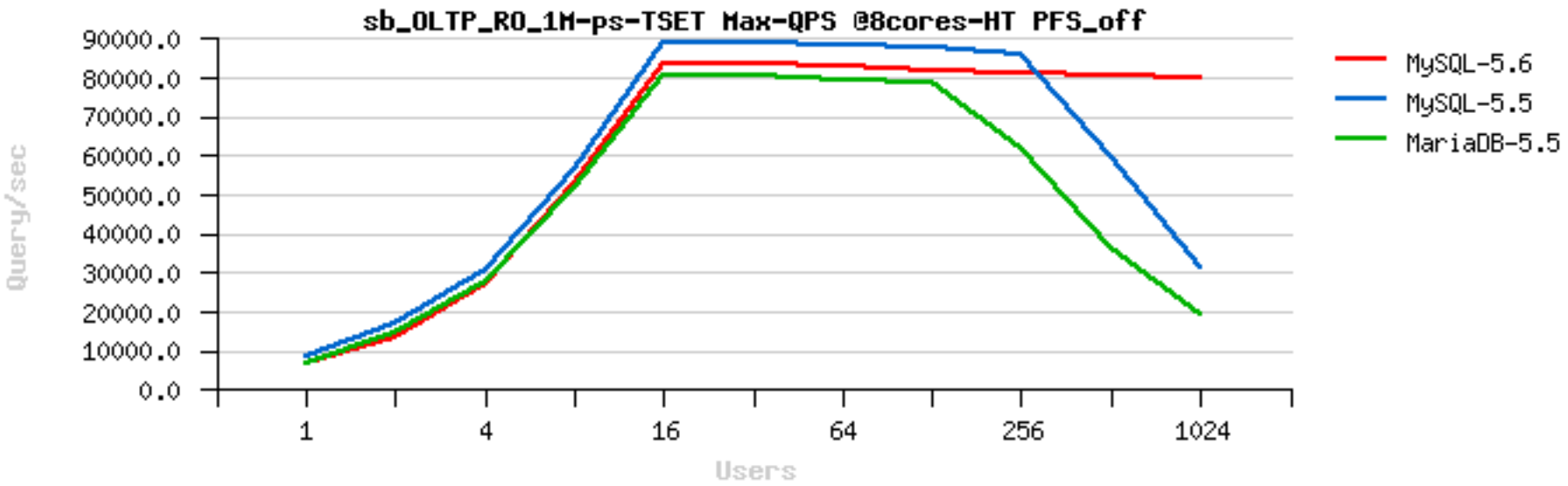


ORACLE

Testing Apples-to-Apples...

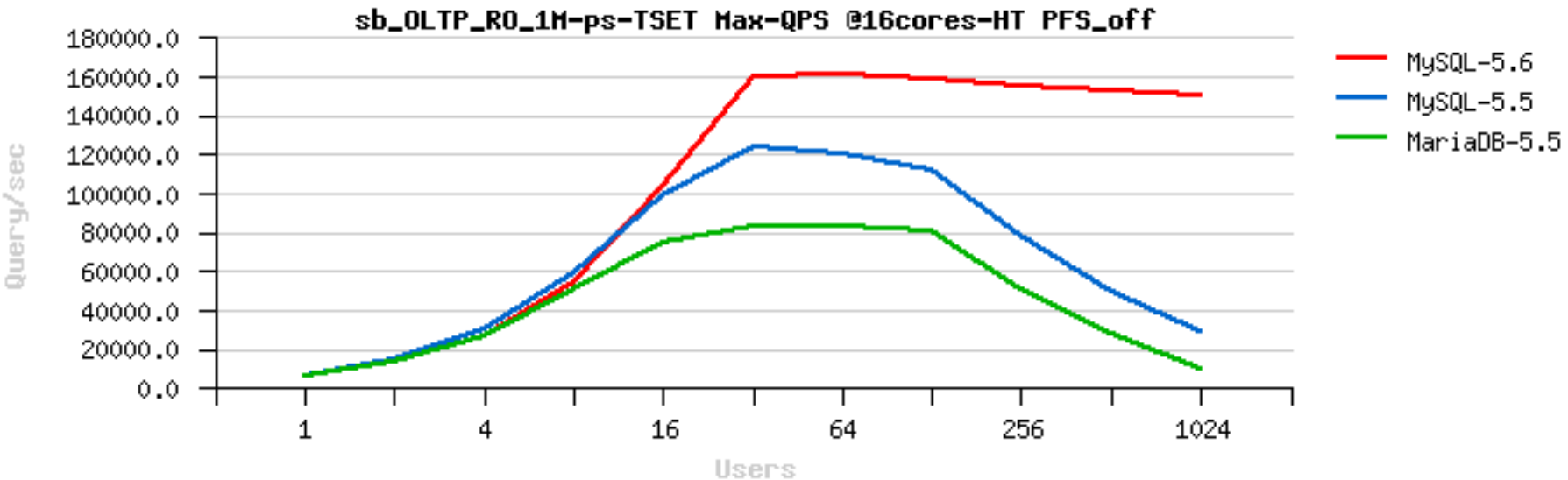
- Comparing MySQL 5.6 vs 5.5 :
  - Don't have G5: dead..
  - Don't have open table cache instances: dead..
  - Don't have improved Adaptive Flushing; dead..
  - Don't have fixed Purge & Lag: danger!..
  - Don't have binlog group commit and use binlog: dead..
  - Etc. etc. etc.
  - NOTE: some “improvement” are also fixes which are making stuff working properly, but coming with additional overhead (like Purge)..
  - NOTE: when comparing 5.6 and 5.5 keep in mind that Performance Schema is enabled by default in 5.6, and not in 5.5, so think to disable it in both (as 5.5 also has a way less PFS instrumentation)..

Sysbench OLTP_RO @8cores-HT



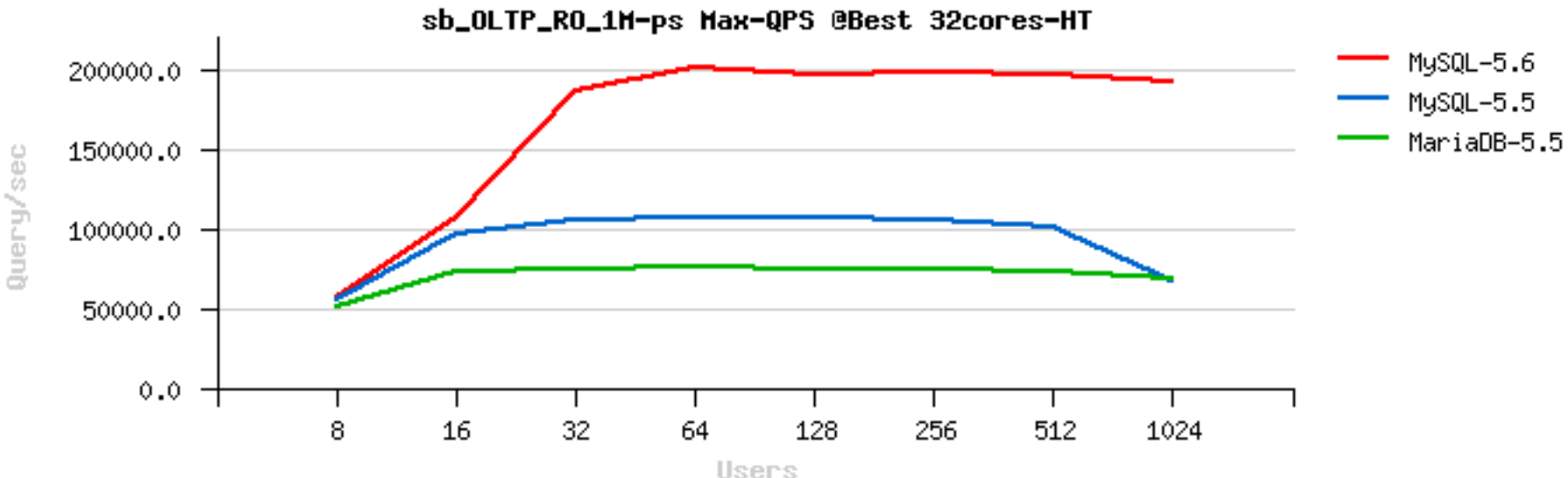
ORACLE

Sysbench OLTP_RO @16cores-HT



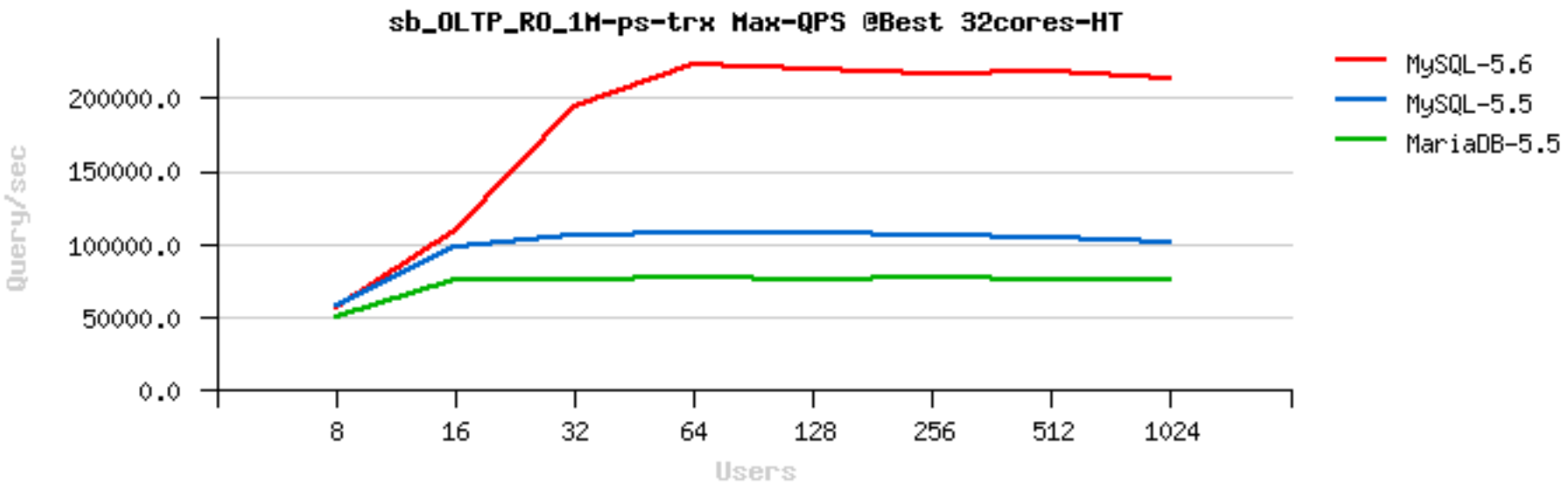
ORACLE

Sysbench OLTP_RO @32cores-HT

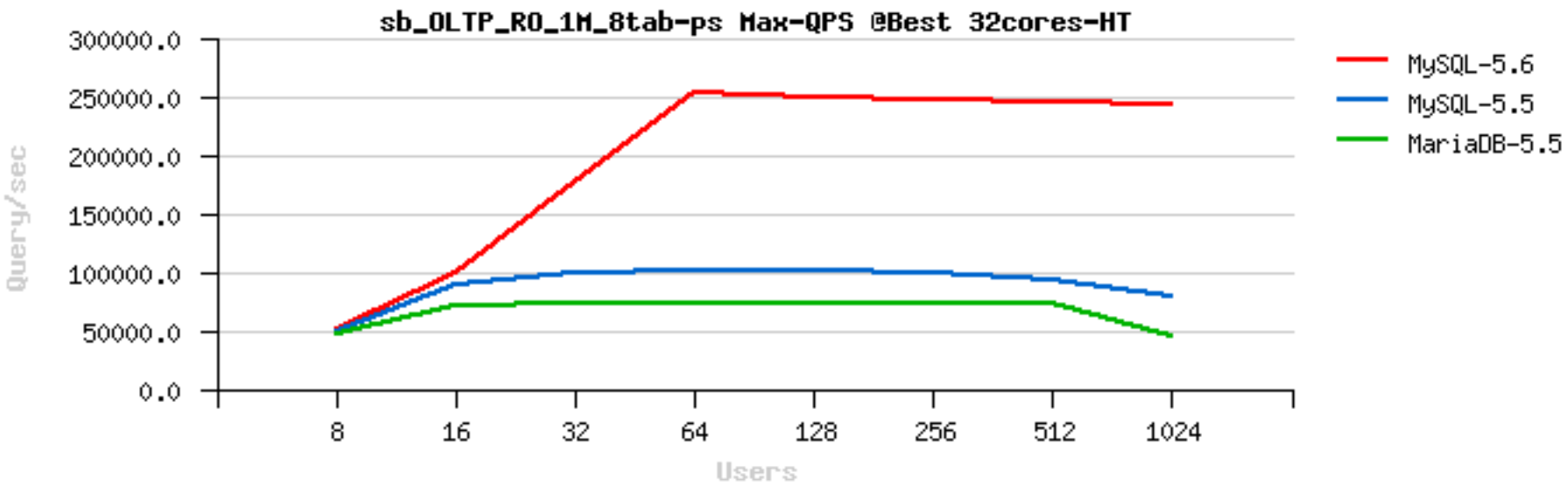


ORACLE

Sysbench OLTP_RO-trx @32cores-HT

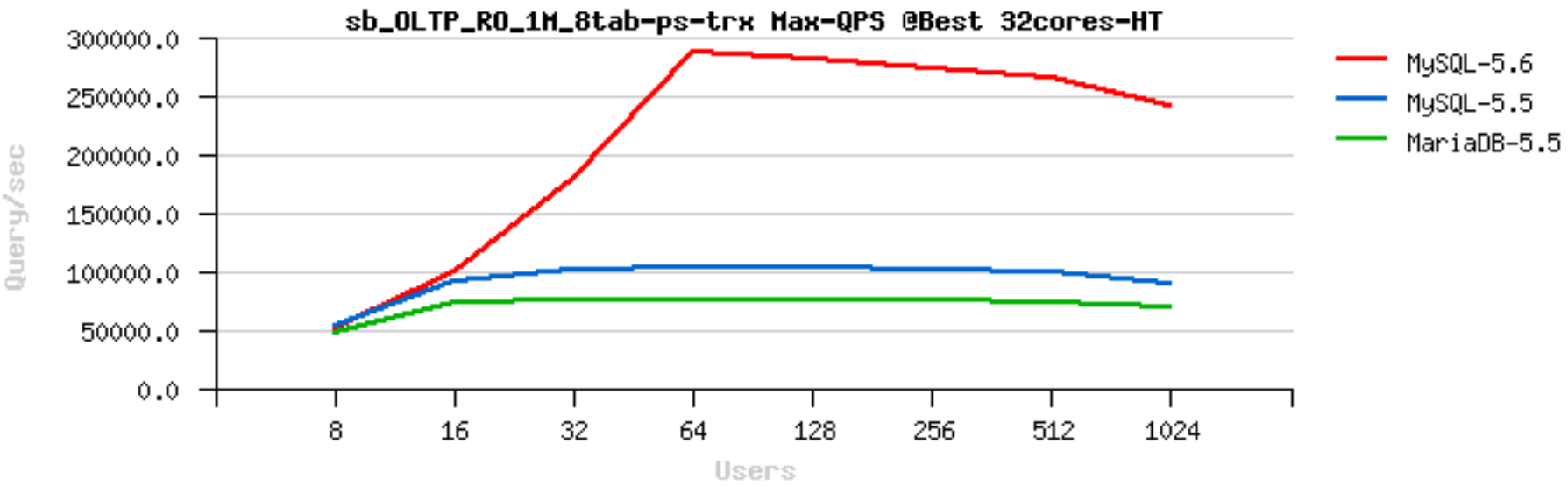


Sysbench OLTP_RO 8-tables @32cores-HT



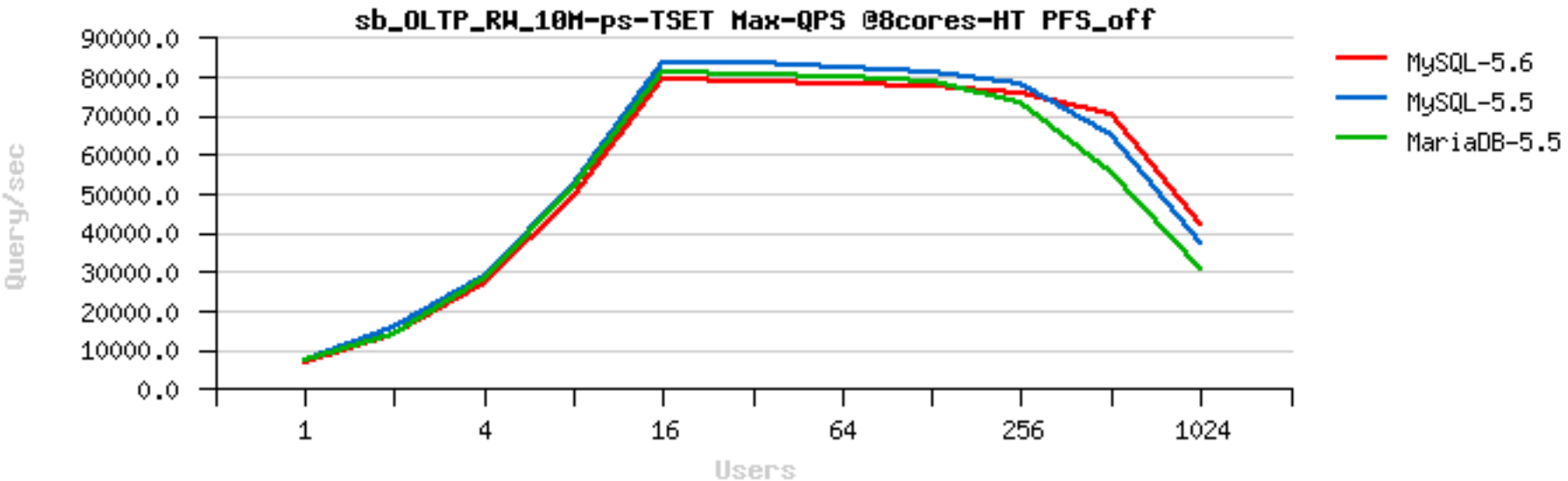
ORACLE

Sysbench OLTP_RO-trx 8-tables @32cores-HT

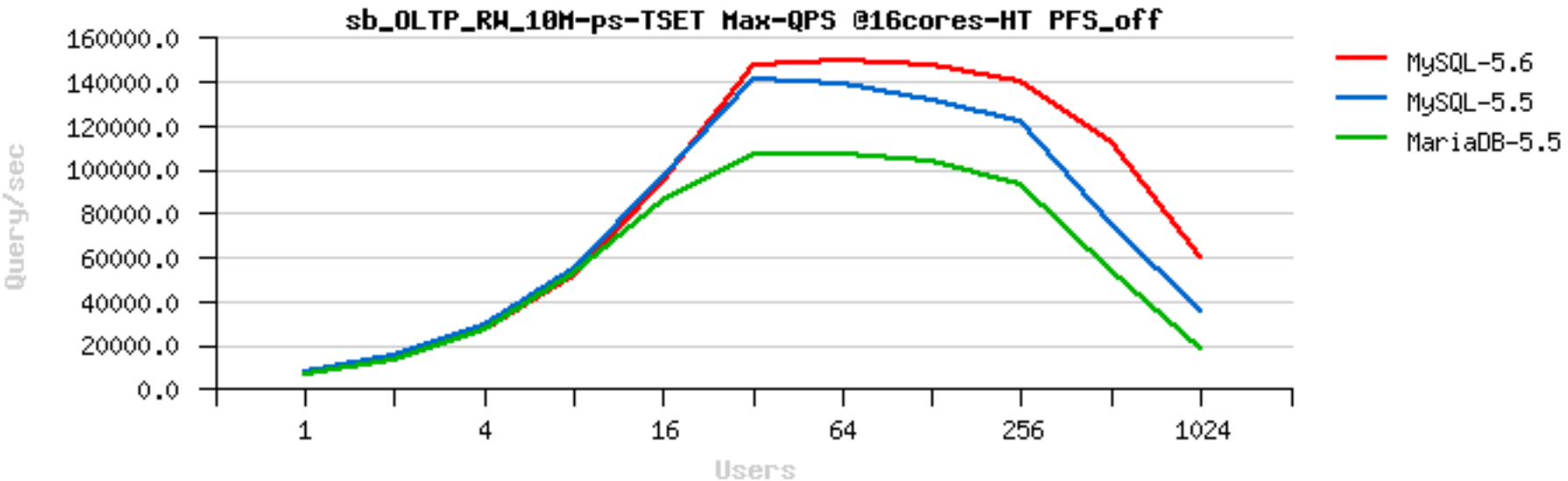


ORACLE

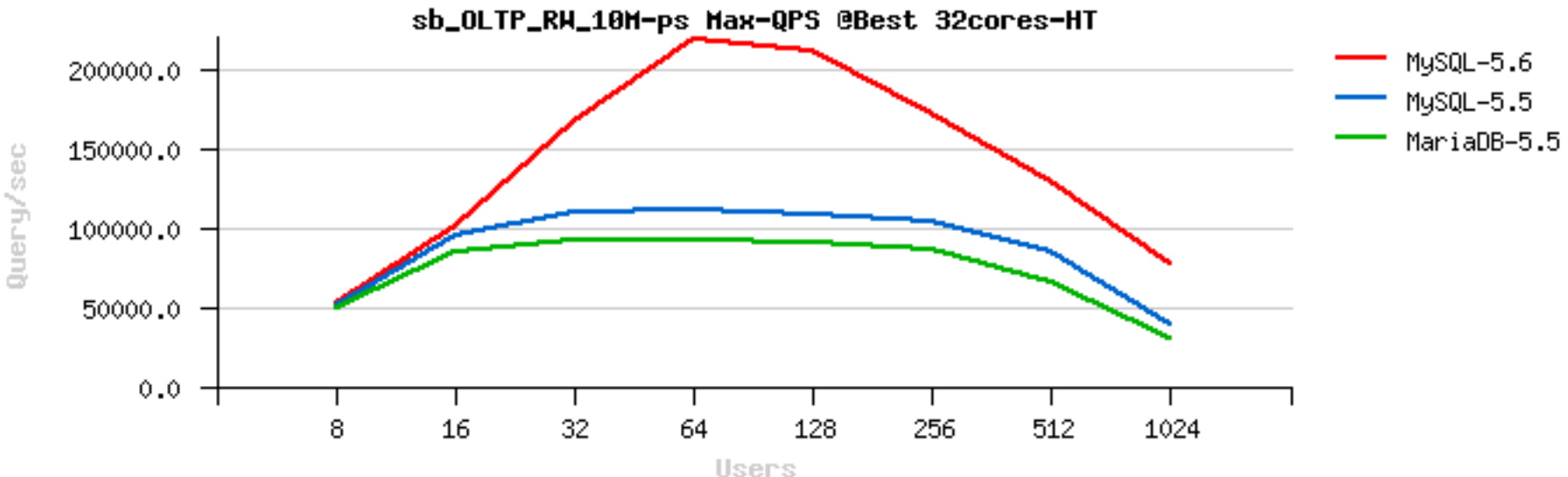
Sysbench OLTP_RW @8cores-HT



Sysbench OLTP_RW @16cores-HT

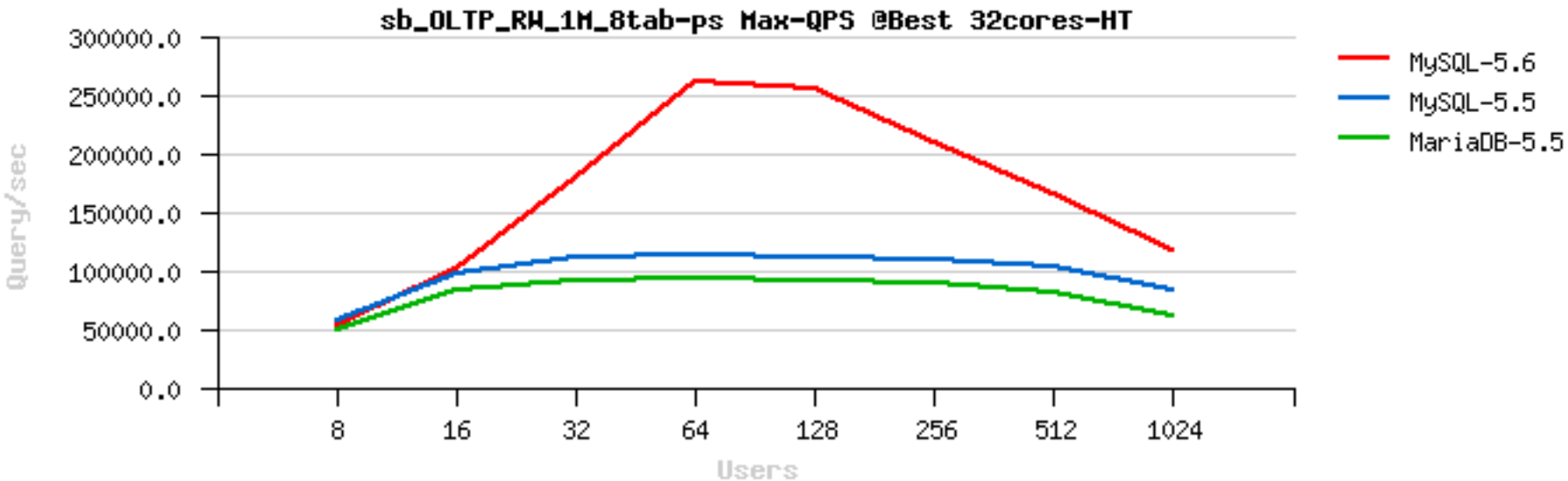


Sysbench OLTP_RW @32cores-HT



ORACLE

Sysbench OLTP_RW 8-tables @32cores-HT



ORACLE

Things are changing constantly, stay tuned ;-)

- 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 over 300.000 QPS(!) + FTS & Memcached

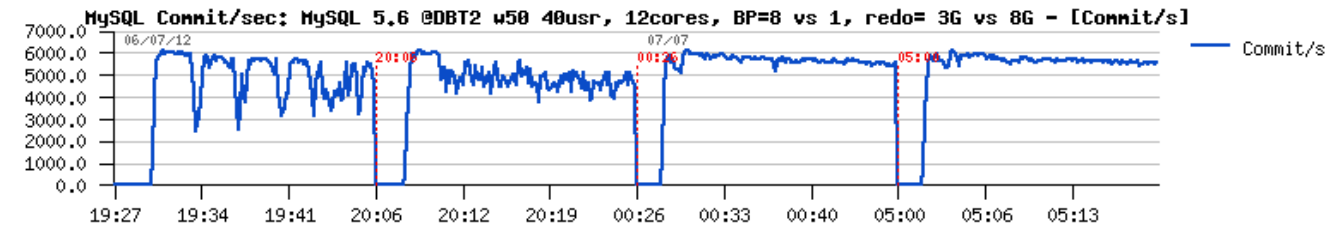
ORACLE

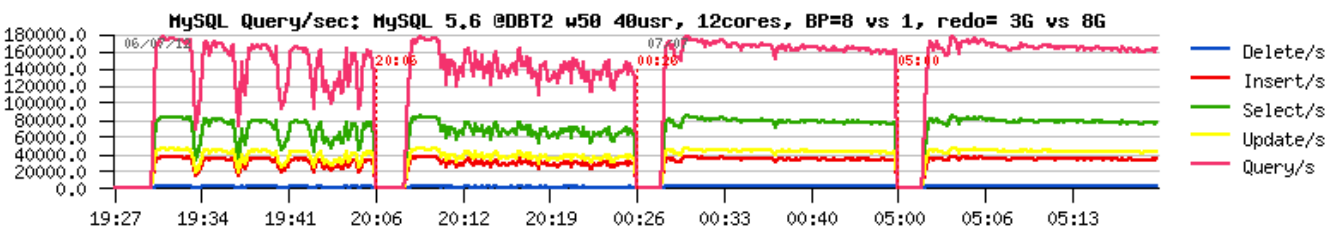
MySQL 5.6: Pending issues

- Index lock..
- Lock_sys contention..
- Trx_sys contention..
- MDL scalability..
- Flushing limits..
- LRU flushing..
- Design bug on block locking.. (was here from the beginning)
- Not able yet to use 100% I/O capacity on a powerful storage..
- “Mysterious” contentions on dbSTRESS..
- etc..

Sequential flushing from BP instances

- Intensive RW workload
  - InnoDB is flushing dirty pages from BP instances one by one..
  - Parallel flushing is needed
  - Workload is solved by using a bigger REDO logs

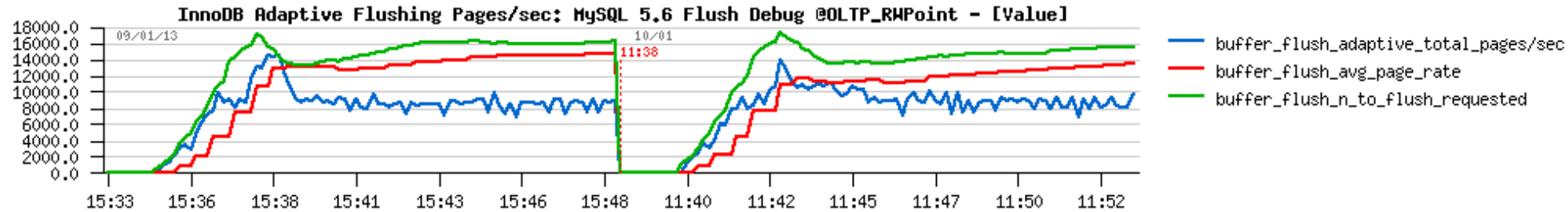


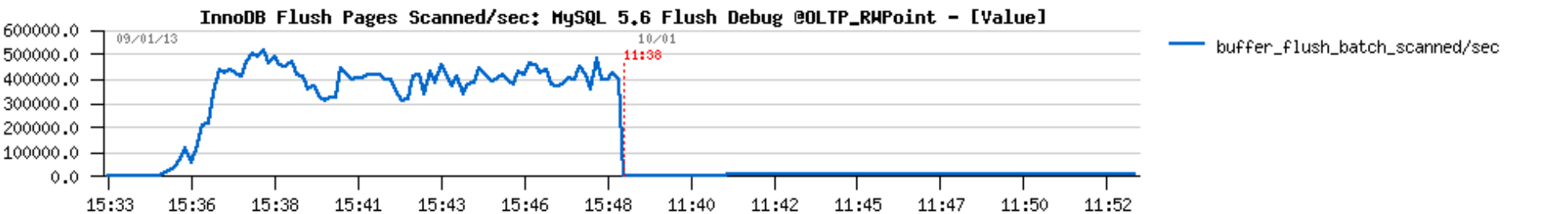


ORACLE

Excessive page scanning during flushing

- Fixed in 5.6.12





ORACLE

dim_STAT (as you'll ask)

- All graphs are built with 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

ORACLE

ORACLE

San Francisco, September 21-23

Oracle.com/mysqlconnect

- Keynotes
- Conferences Sessions
- Birds-of-a-feather sessions
- Tutorials
- Hands-on Labs
- Exhibition Hall
- Demo Pods
- Receptions