MySQL 5.7 Performance: Scalability & Benchmarks

Dimitri KRAVTCHUK
MySQL Performance Architect @Oracle

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.

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 2011 “officially” @MySQL Performance full time now
- http://dimitrik.free.fr/blog / @dimitrik_fr

Agenda

- Overview of MySQL Performance
- Performance improvements in MySQL 5.7 & Benchmark results
- Pending issues..
- Q & A

ORACLE

Why MySQL Performance ?...

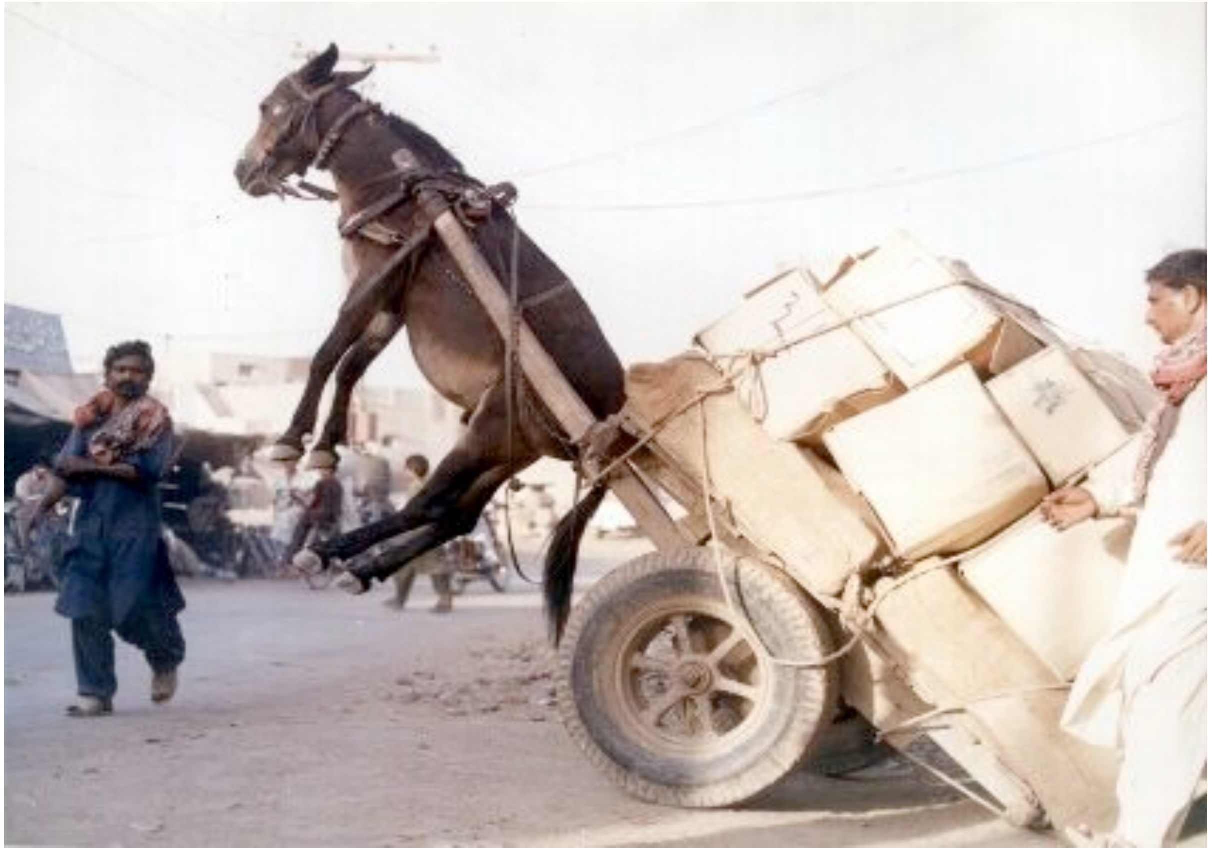
Why benchmarking MySQL?..

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

ORACLE

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 to understand ahead the resistance of your solution to incoming potential problems ;-)

ORACLE

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 is... ???..

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

USE YOUR BRAIN !!!... ;-)

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

USE YOUR BRAIN !!!... ;-)

THE MAIN SLIDE! ;-))

Think “Database Performance” from the beginning!

malloc() !!!
I/O & FS !!

- 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 validation goal :-))
  - Flash helps when access is random! (reads are the most costly)
  - 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!)

Test Workload

- Before to jump into something complex...
  - Be sure first you're comfortable with “basic” operations!
  - Single table? Many tables?
  - Short queries? Long queries?
- Remember: any complex load in fact is just a mix of simple operations..
  - So, try to split problems..
  - Start from as simple as possible..
  - And then increase complexity progressively..
- NB : **any** test case is important !!!
  - Consider the case rather reject it with “I’m sure you’re doing something wrong..” ;-))

ORACLE

“Generic” Test Workloads @MySQL

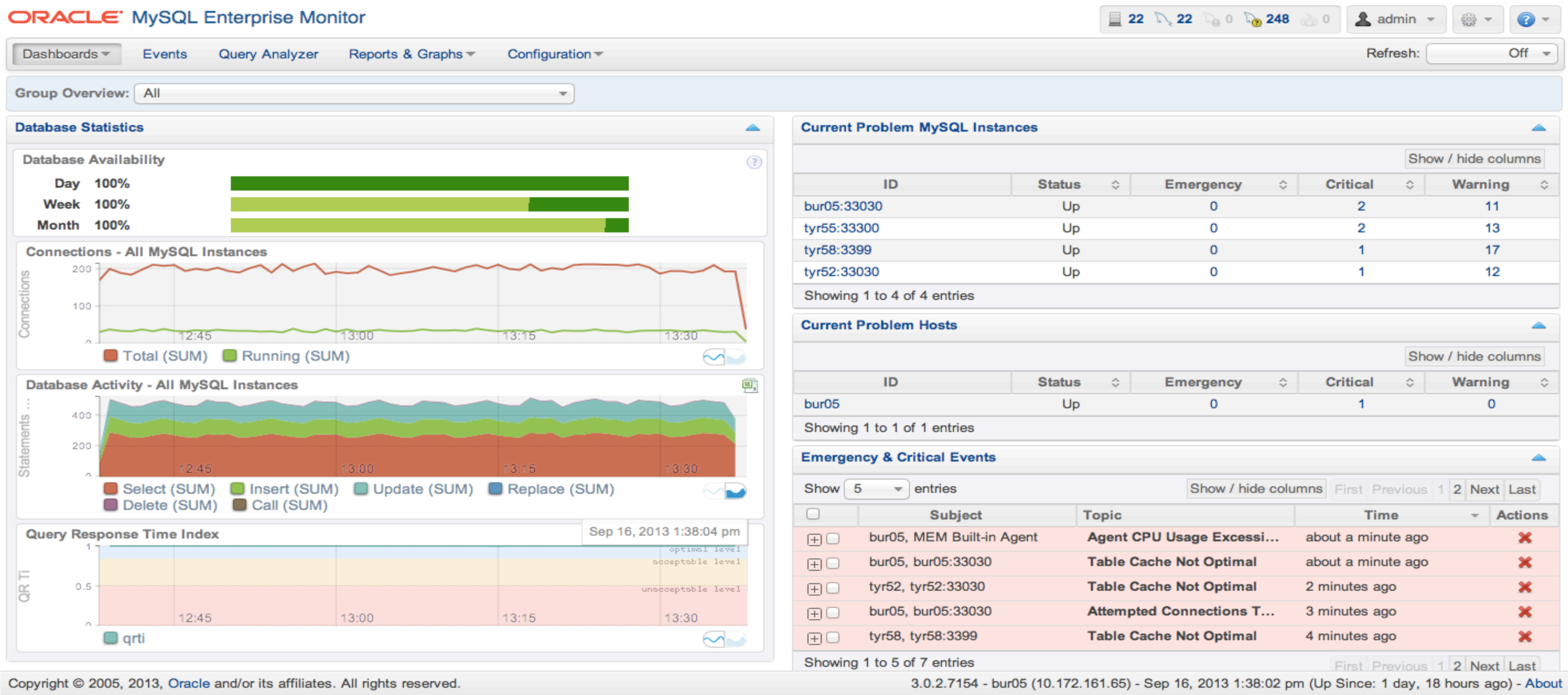
- Sysbench
  - OLTP, RO/RW, 1-table, since v0.5 N-table(s), lots load options, deadlocks
- DBT2 / TPCC-like
  - OLTP, RW, very complex, growing db, no options, deadlocks
  - In fact using mostly only 2 tables! (thanks Performance Schema ;-))
- dbSTRESS
  - OLTP, RO/RW, several tables, one most hot, configurable,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

Monitoring is THE MUST !

even don’t start to test anything without monitoring.. ;-)

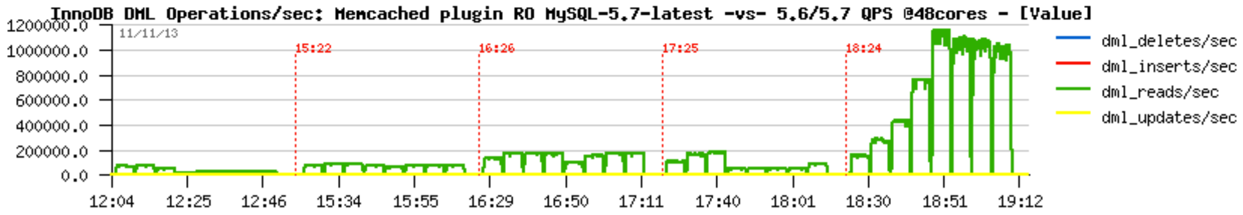
MySQL Enterprise Monitor

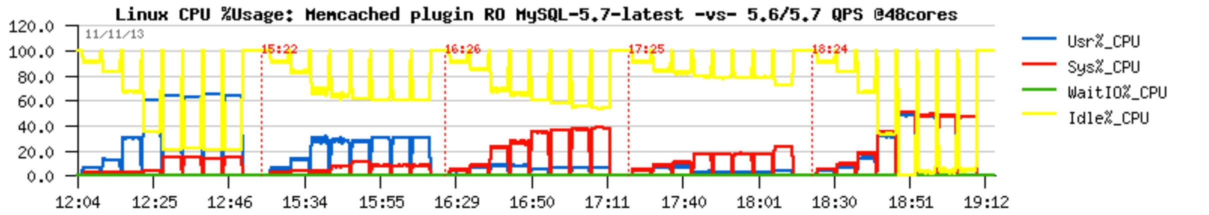
- Fantastic tool!
  - Did you already try it?.. Did you see it live?..



Other Monitoring Tools

- Cacti, Zabbix, Nagios, Etc.....
- dim_STAT
  - well, I'm using this one, sorry ;-)
  - all graphs within presentation were made with it
  - details are in the end of presentation..



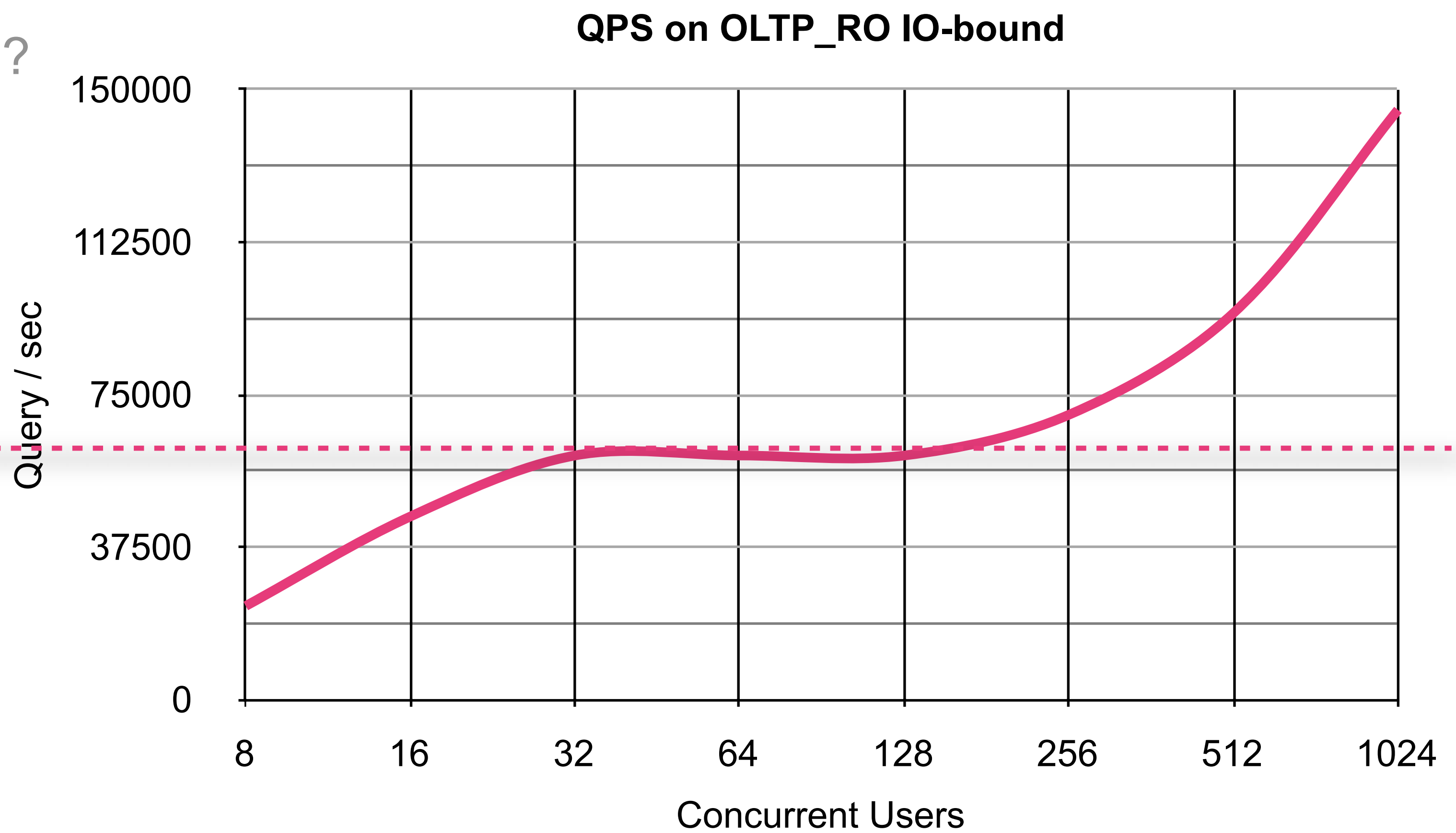


Be sure you can trust your Benchmark results ;-)

- Know your HW platform **limits**
- Understand what your **Workload** is doing
- Keep in mind MySQL Server **internals**
- There is **NO** “Silver Bullet” !!!
  - Think about the #1 MySQL Performance Best Practice ;-))

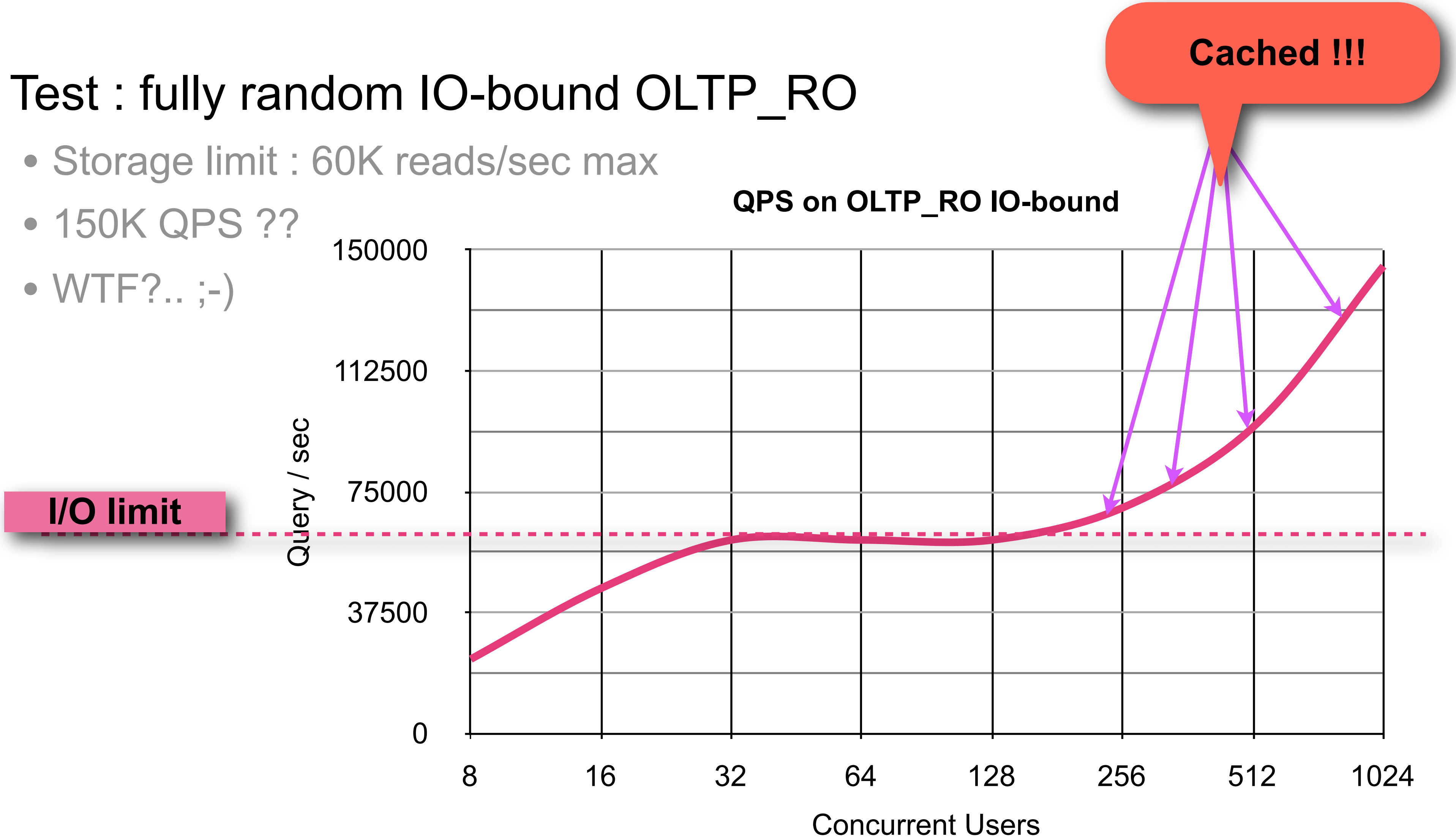
Let’s analyze the following benchmark result..

- Test : fully random IO-bound OLTP_RO
  - Storage limit : 60K reads/sec max
  - 150K QPS ??
  - WTF?.. ;-)



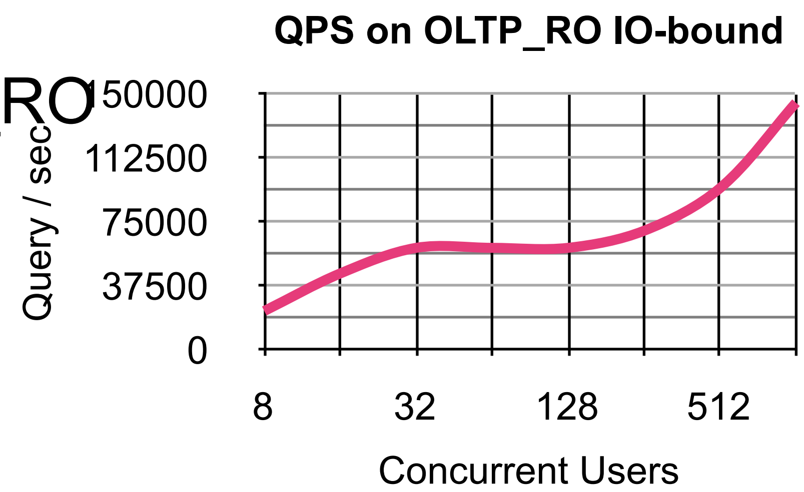
Let’s analyze the following benchmark result..

- Test : fully random IO-bound OLTP_RO
  - Storage limit : 60K reads/sec max
  - 150K QPS ??
  - WTF?.. ;-)



Let's analyze the following benchmark result..

- Test : fully random IO-bound OLTP_RO
  - Storage limit : 60K reads/sec max
  - 150K QPS ??
  - WTF?.. ;-)
- The issue:
  - the random ID for a row acces is not that random as expected..
  - and with a higher workload the probability to get the same "random" row ID on the same time and by different threads only increasing..
  - workaround : for some of the tests started to use as many Sysbench processes as user threads (1 connection = 1 sysbench process)..



ORACLE

Analyzing Workloads: RO -vs- RW

- Read-Only (RO) :
  - Nothing more simple when comparing DB Engines, HW configs, etc..
  - RO In-Memory : data set fit in memory / BP / cache
  - RO IO-bound : data set out-passing a given memory / BP / cache
- Read+Write (RW) :
  - I/O is **ALWAYS** present ! - storage performance matters a lot !
  - may be considered as always IO-bound ;-)
  - RW In-Memory : same as RO, data set fit in memory, but :
    - small data set => small writes
    - big dataset => big writes ;-)
  - RW IO-bound : data set out-passing a memory
    - means there will be (a lot of?) reads !
    - don't forget that I/O random reads = I/O killer !

ORACLE

Read-Only Scalability @MySQL / InnoDB

- Depends on a workload..
  - sometimes the limit is only within your memcpy() rate ;-)
- But really started to scale only since MySQL 5.7
  - due improved TRX list management, MDL, THR_lock, etc..
  - scaling up to 48 CPU cores for sure, reported on more cores too..
  - Note : code path is growing with new features! (small HW may regress)
- IO-bound :
  - could be limited by storage (if you’re not using a fast flash)
  - or by internal contentions (InnoDB file_sys mutex)
- Limitations
  - there are still some limitations “by design” (block lock, file_sys, etc..)
  - all in TODO to be fixed, but some are needing a deep redesign

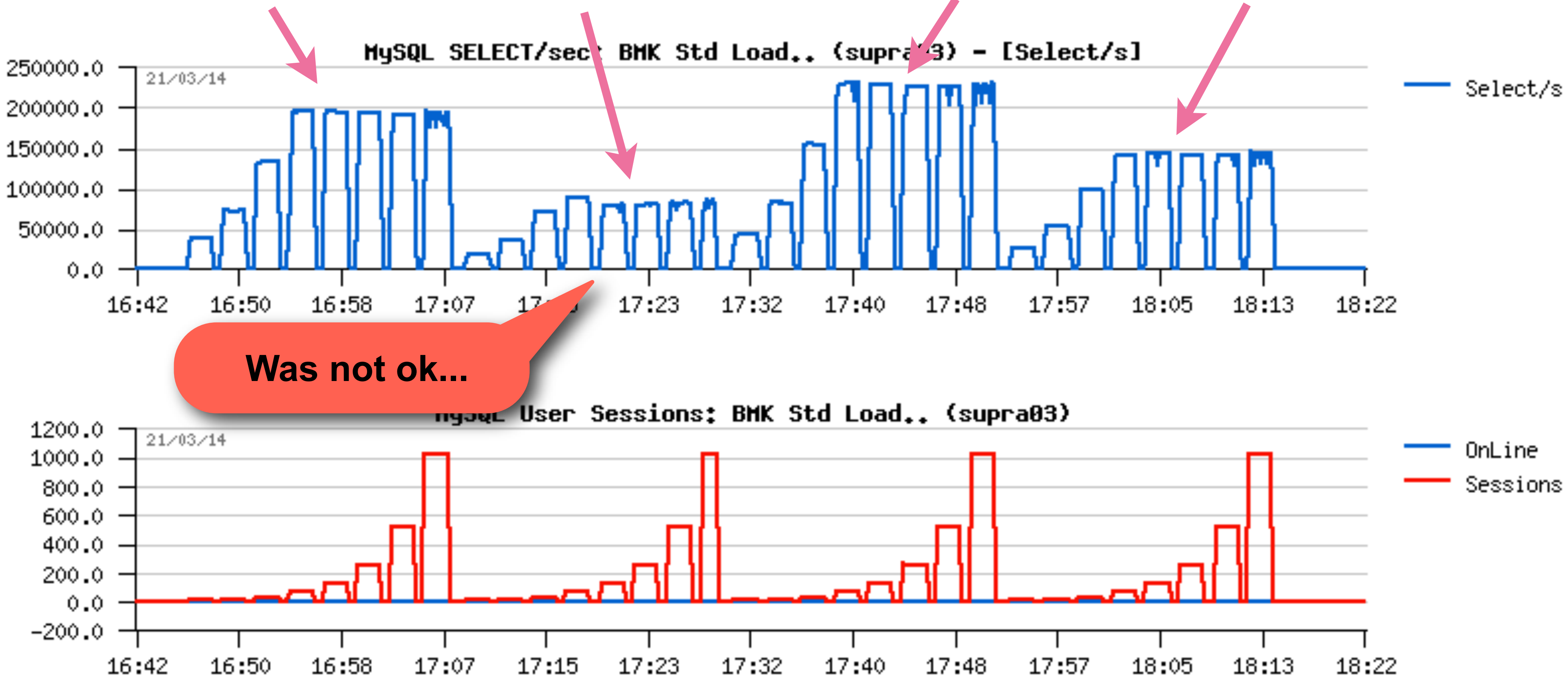
RO related starter configuration settings

- my.conf :

```
join_buffer_size=32K
sort_buffer_size=32K

table_open_cache = 8000
table_open_cache_instances = 16
query_cache_type = 0

innodb_buffer_pool_size= 64000M (2/3 RAM ?)
innodb_buffer_pool_instances=32
innodb_thread_concurrency = 0 / 32 / 64
innodb_spin_wait_delay= 6 / 48 / 96

innodb_stats_persistent = 1
innodb_adaptive_hash_index= 0 / 1
innodb_monitor_enable = '%'
```

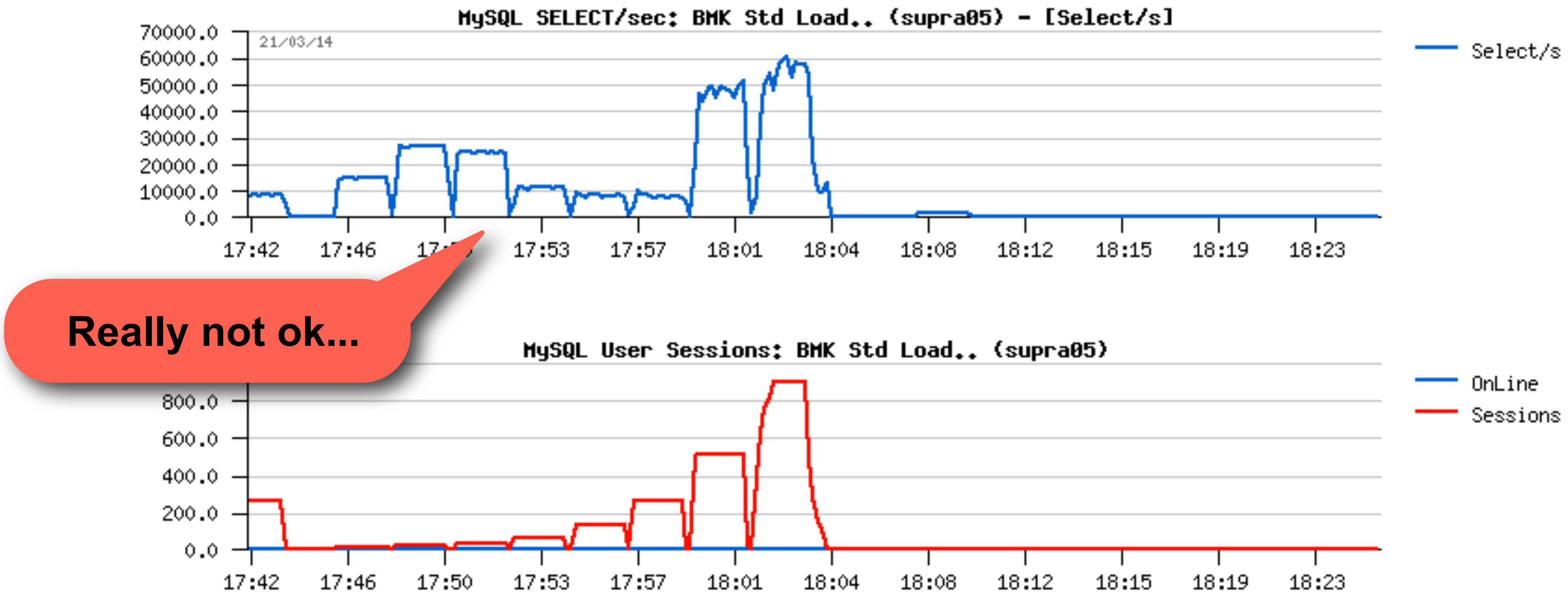
Sysbench OLTP_RO Workloads @MySQL 5.7

- Simple ranges, Distinct ranges, SUM ranges, Ordered ranges



Story #1 : mysterious kernel contention

- Sysbench RO Distinct Selects
  - 40cores-HT server



ORACLE

Story #1 : mysterious kernel contention (2)

- Sysbench RO Distinct Selects
  - 40cores-HT server
- Profiler:

```
80.52%  [kernel]      [k] _spin_lock
 7.36%  [kernel]      [k] native_write_msr_safe
 2.08%  [kernel]      [k] smp_invalidate_interrupt
 0.82%  [kernel]      [k] find_next_bit
 0.76%  [kernel]      [k] flush_tlb_others_ipi
 0.69%  [kernel]      [k] __bitmap_empty
 0.53%  [kernel]      [k] native_flush_tlb
...
```

Really not ok...

Story #1 : mysterious kernel contention (3)

- Sysbench RO Distinct Selects
  - 40cores-HT server
- Profiler:

```
 80.52%  [kernel]                [k] _spin_lock
  7.36%  [kernel]                [k] native_write_msr_safe
  2.08%  [kernel]                [k] smp_invalidate_interru
  0.82%  [kernel]                [k] find_next_bit
  0.76%  [kernel]
  0.69%  [kernel]
  0.53%  [kernel]
...
```

```
 73.13%   mysqld  [kernel.kallsyms]                    [k] _spin_lock
          |
          --- _spin_lock
             |
             |--99.96%-- flush_tlb_others_ipi
             |          native_flush_tlb_others
             |          flush_tlb_mm
             |          zap_page_range
             |          sys_madvise
             |          system_call_fastpath
             |          madvise
             |          |
             |          |--2.73%-- 0x7f03db1e1818
...
```

But who is the killer ?...

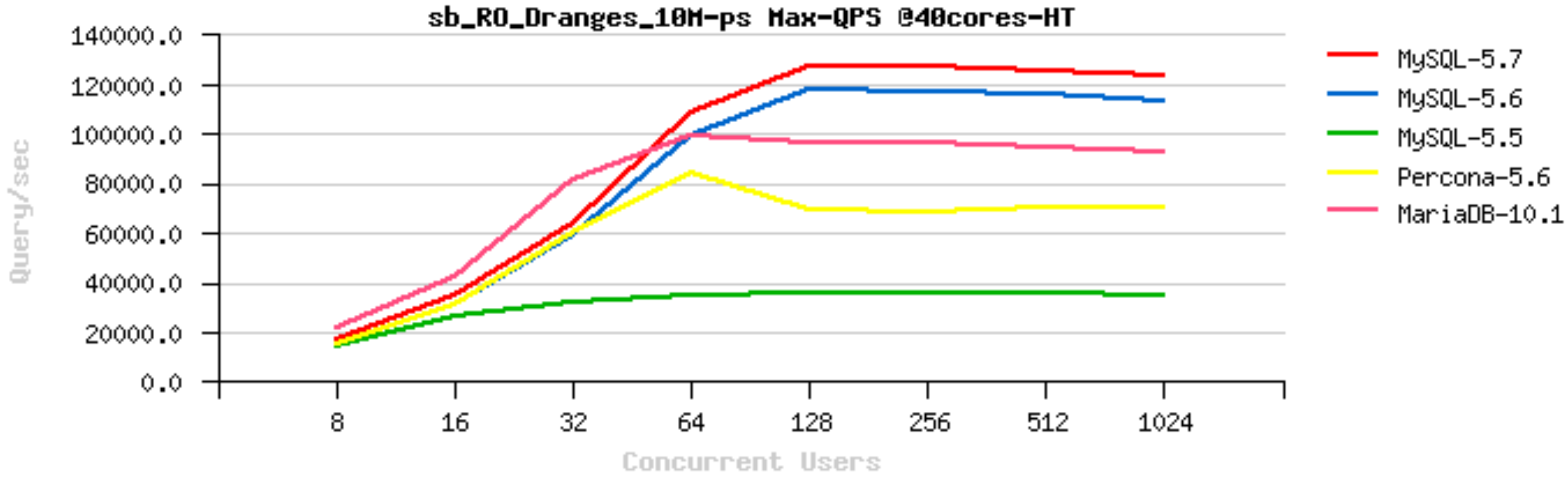
Story #1 : mysterious kernel contention (4)

- Sysbench RO Distinct Selects
  - 40cores-HT server
- And the killer is... - **jemalloc** !!! ;-)
  - Distinct Selects workload is extremely hot on malloc (HEAP)
    - in fact any SELECT involving HEAP temp tables will be in the same case..
    - ex: small results via group by, order by, etc..
  - jemalloc has a smart memory free stuff...
  - trigger OS via madvise()..
  - disabling this jemalloc feature resolving the problem ;-)

```
LD_PRELOAD=/apps/lib/libjemalloc.so  ; export LD_PRELOAD
MALLOC_CONF=lg_dirty_mult:-1         ; export MALLOC_CONF
```

OLTP_RO Distinct Selects with “fixed” jemalloc

- Max QPS @40cores-HT :

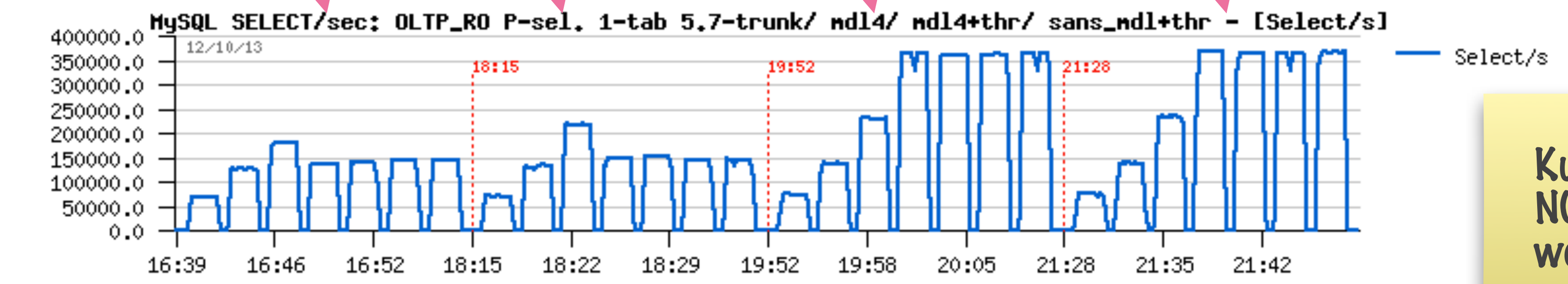


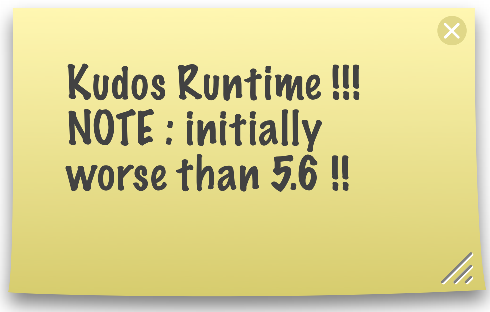
Story #2 : contentions around a hot table

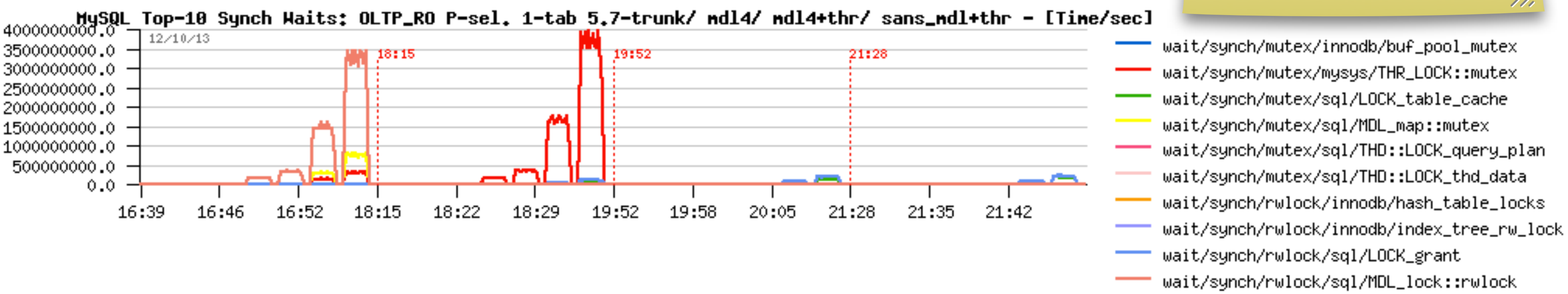
- Once again a game of contentions :
  - improved TRX list ==> more hot MDL..
  - more hot MDL ==> regression on all single-table workloads..
  - improved MDL ==> more hot THR_lock..
  - more hot THR_lock ==> regression on all single-table workloads..
  - improved THR_lock ==> “next-level” locks become visible now!
  - expectation for today : once the next level lock are fixed, there should be no one new unexpected contention for a while ;-)

Story #2 : contentions around a hot table (2)

- Making-off @OLTP_RO Point-Selects, single-table :
  - original | MDL-fix | MDL&THR_lock fix | original w/out MDL&THR_lock

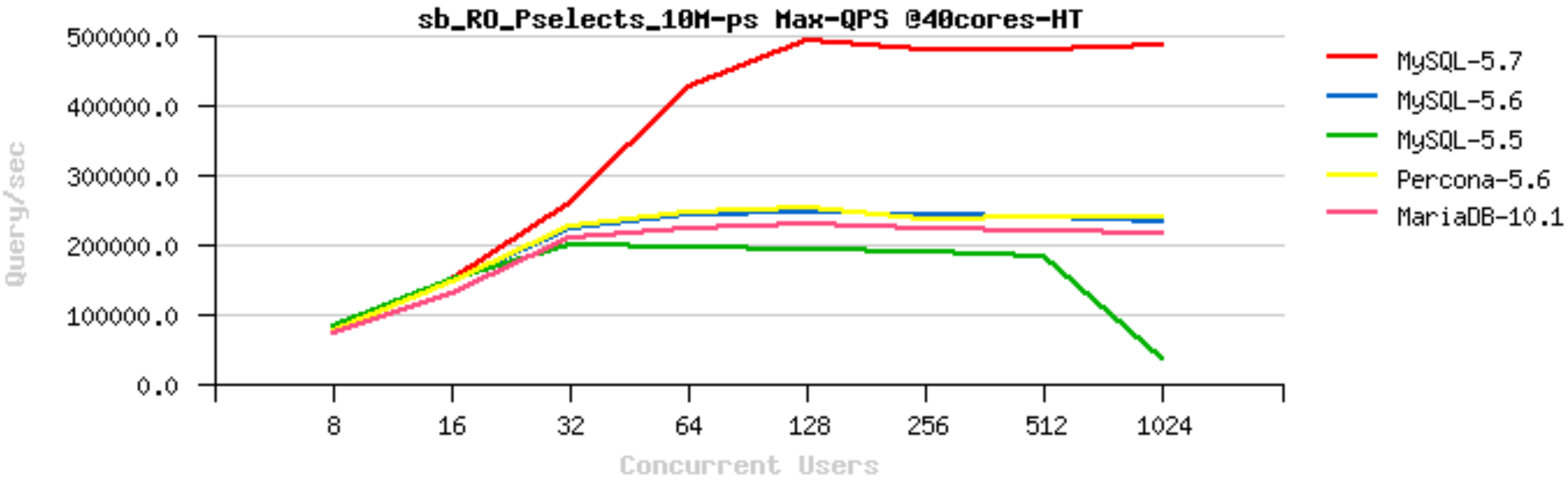






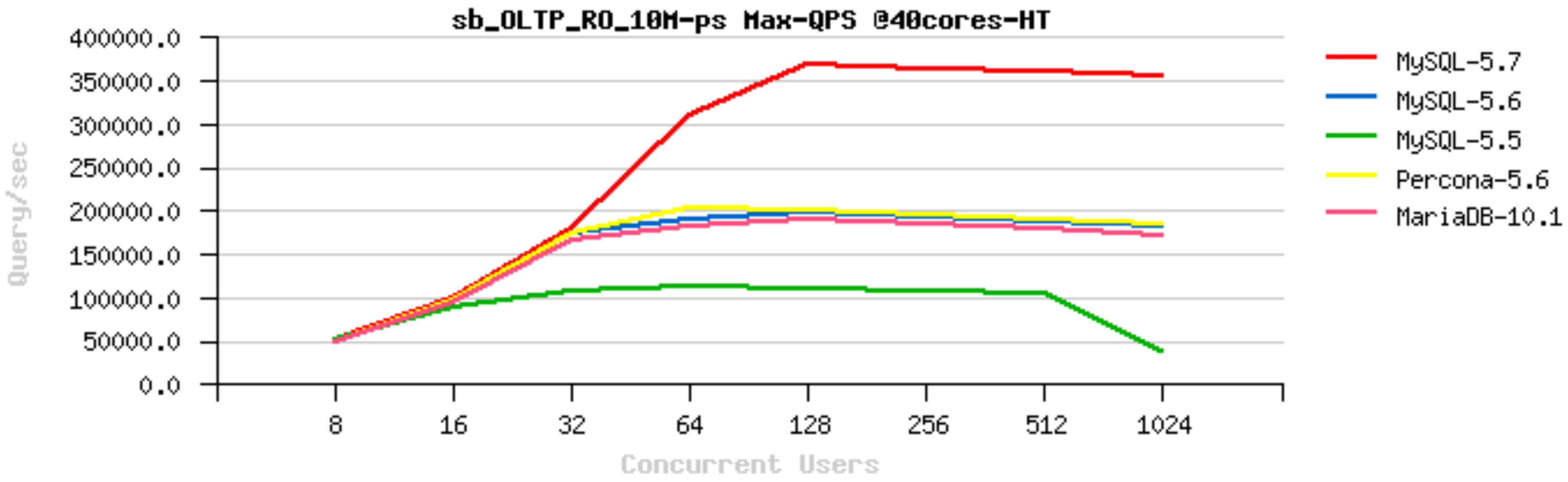
Sysbench OLTP_RO Point-Selects single-table

- Max QPS @40cores-HT :



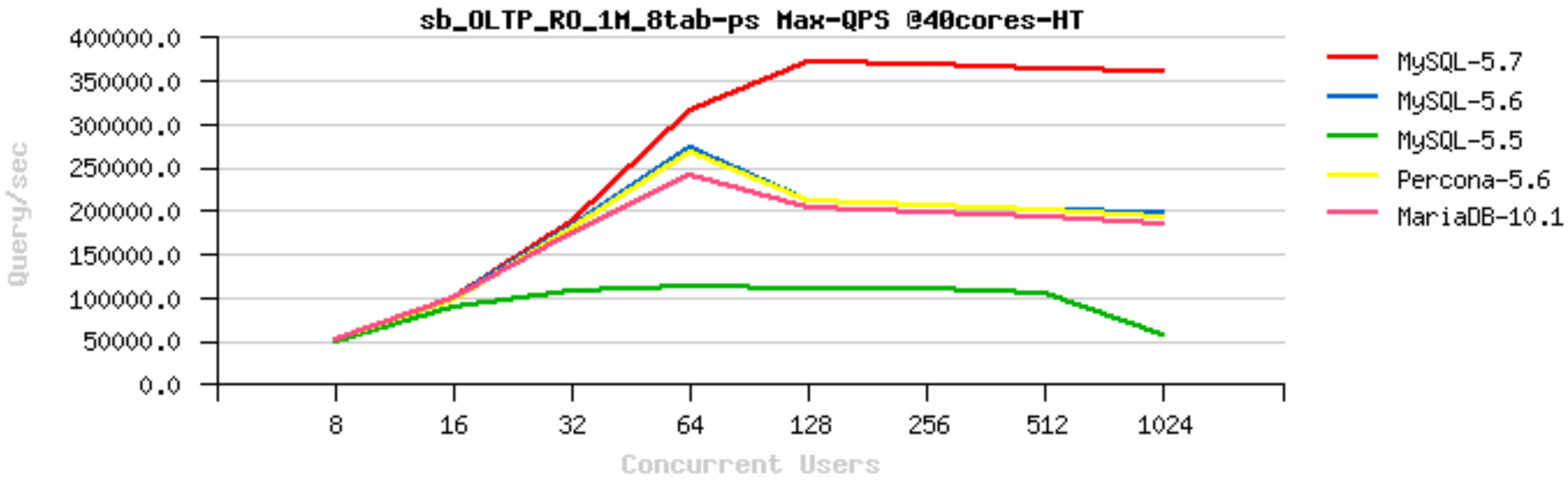
Sysbench OLTP_RO Single-table

- Max QPS @40cores-HT :



RO In-Memory @MySQL 5.7

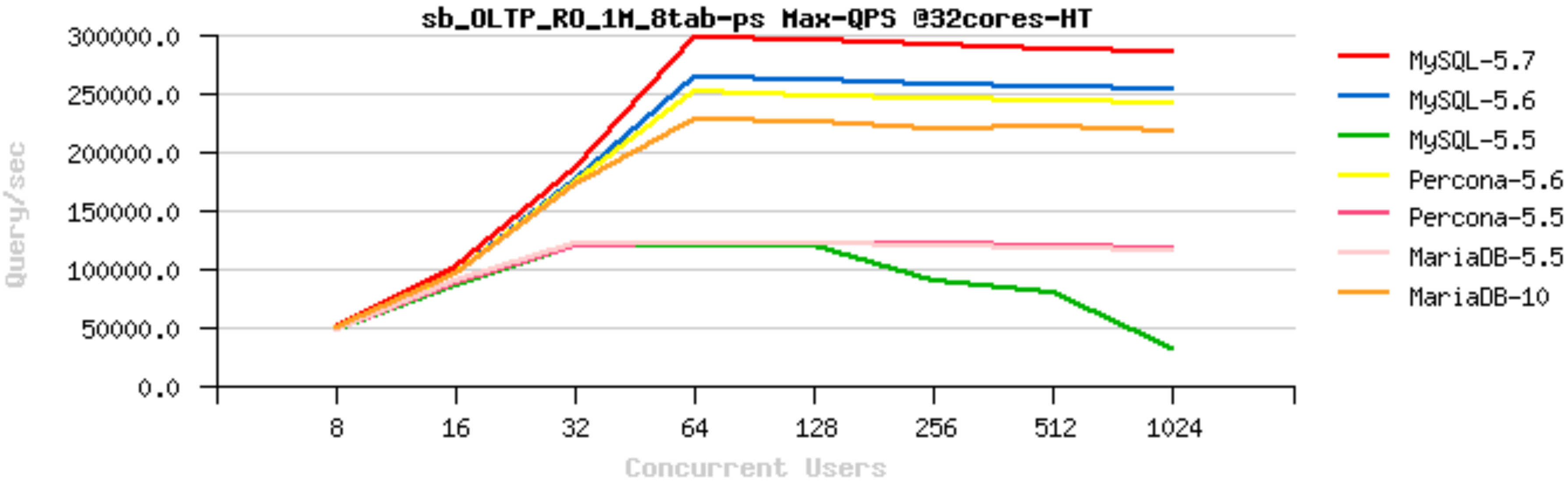
• Sysbench OLTP_RO 8-tables, 40cores-HT :



ORACLE

RO In-Memory @MySQL 5.7

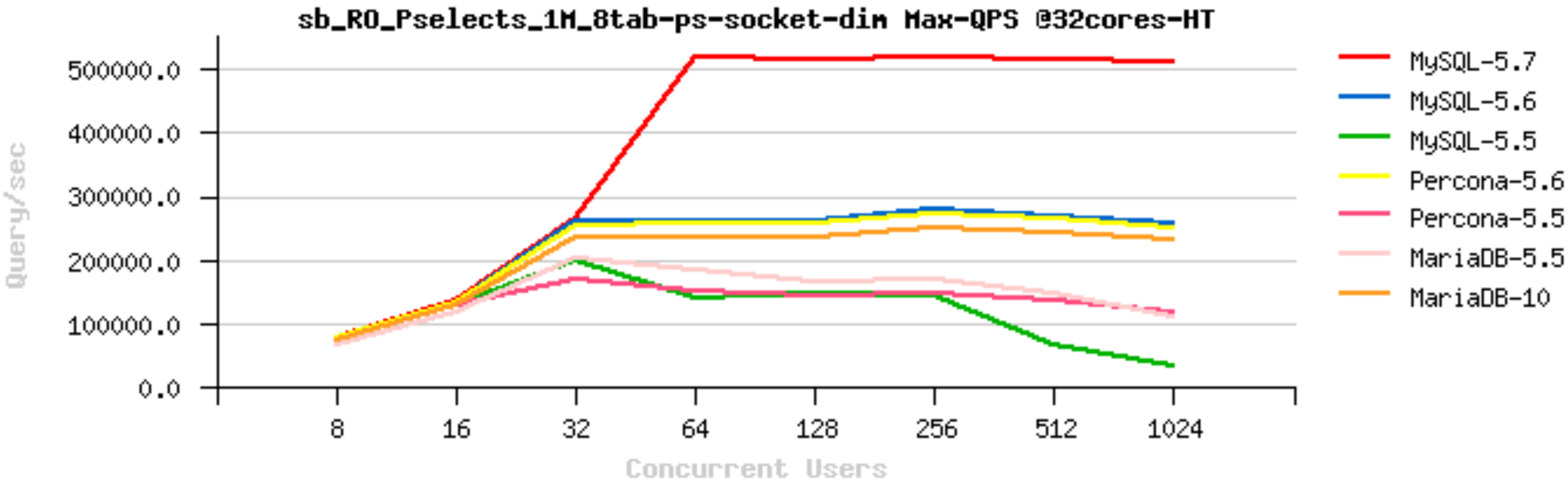
- Sysbench OLTP_RO 8-tables, 32cores-HT :



ORACLE

RO In-Memory @MySQL 5.7

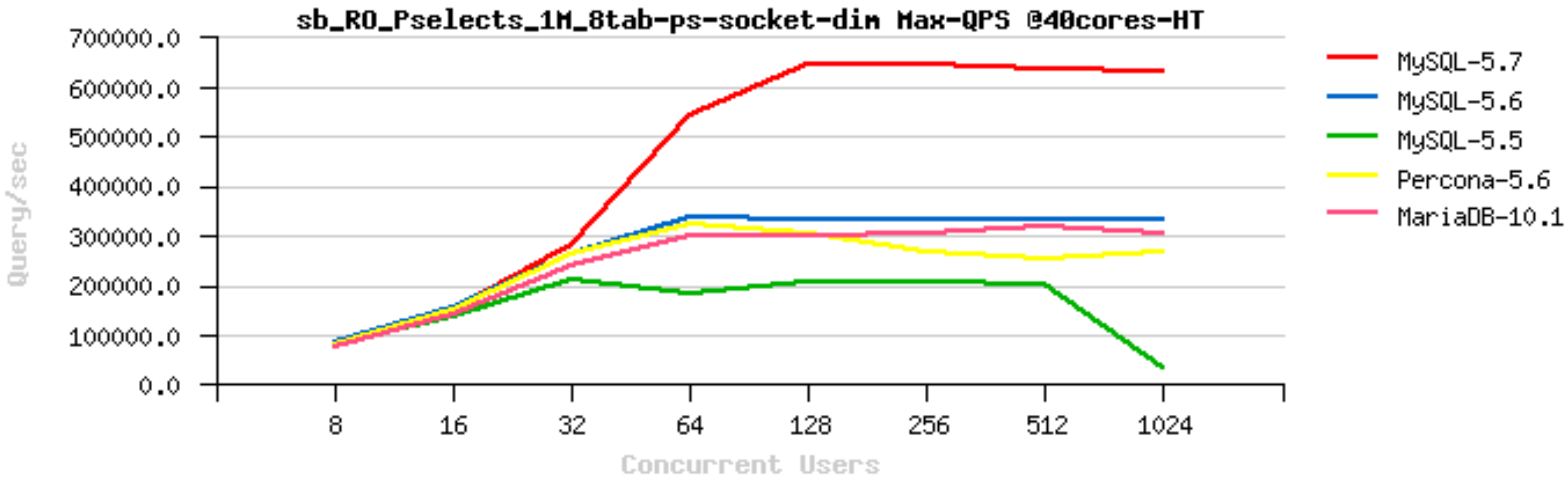
- **500K QPS** Sysbench Point-Selects 8-tab, 32cores-HT :



ORACLE

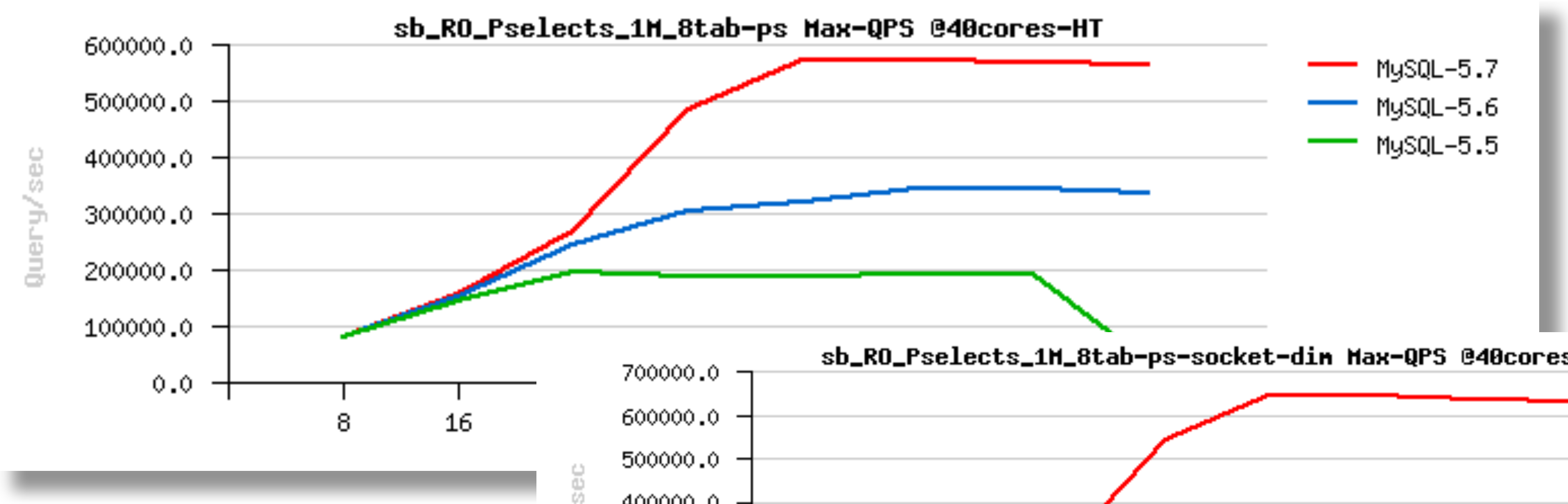
RO In-Memory @MySQL 5.7

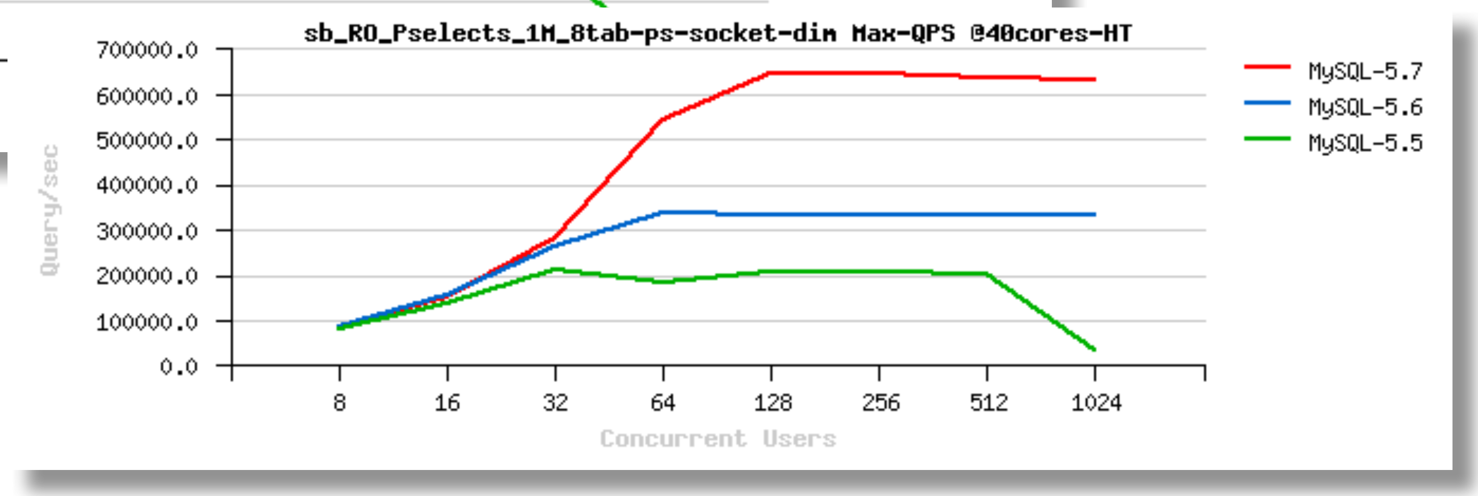
- **645K QPS** Sysbench Point-Selects 8-tab, 40cores-HT :



Few words about RO scalability

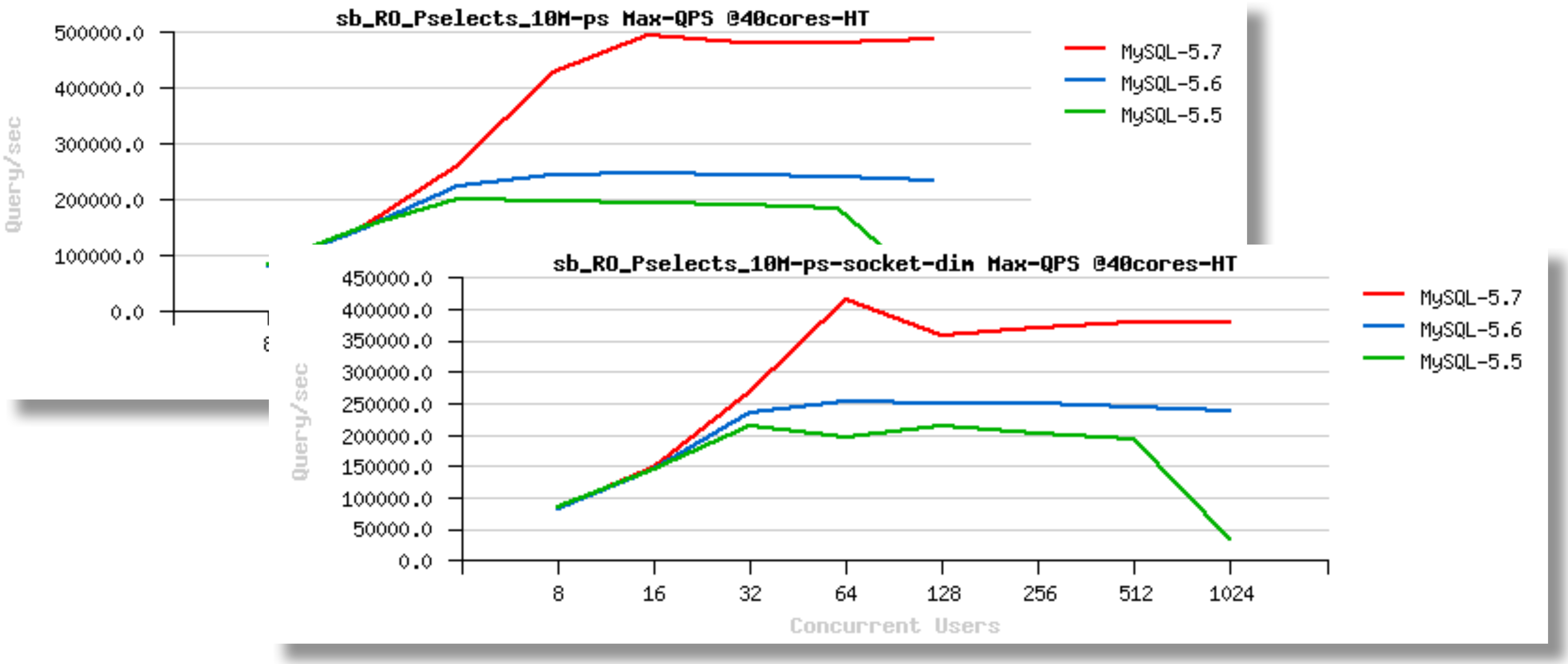
- OLTP_RO Point-selects **8**-tables, the same 40cores host
  - IP socket & sysbench 0.4.13 -vs- UNIX socket & sysbench 0.4.8 :





Few words about RO scalability (bis)

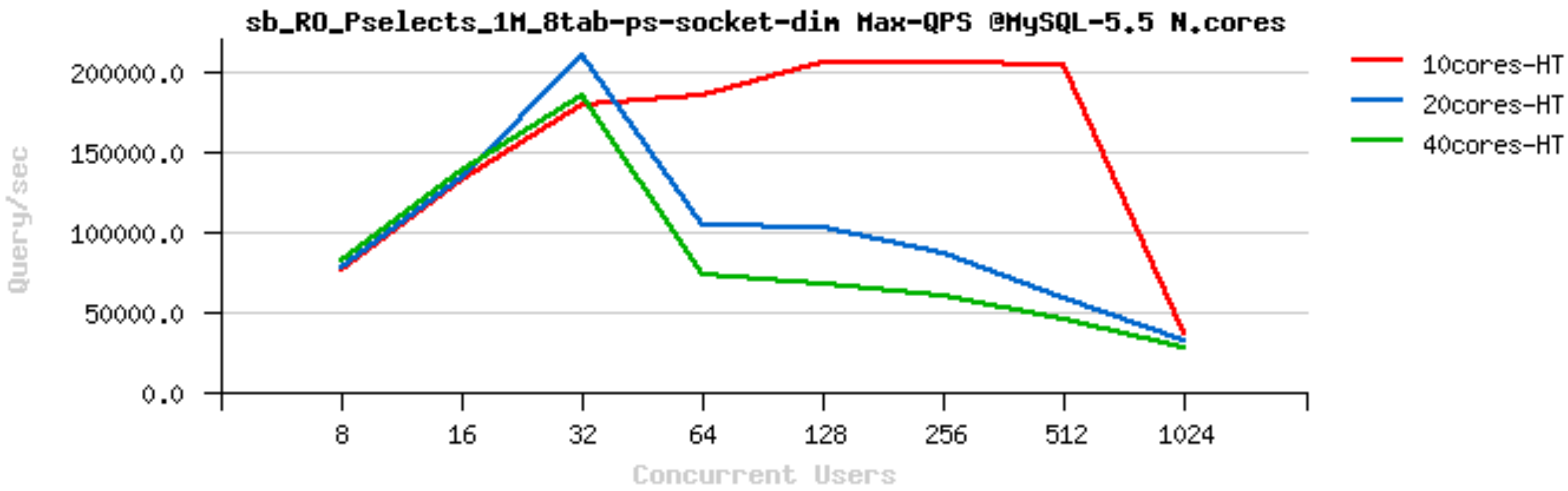
- OLTP_RO Point-selects **1**-table, the same 40cores host
  - IP socket & sysbench 0.4.13 -vs- UNIX socket & sysbench 0.4.8 :



ORACLE

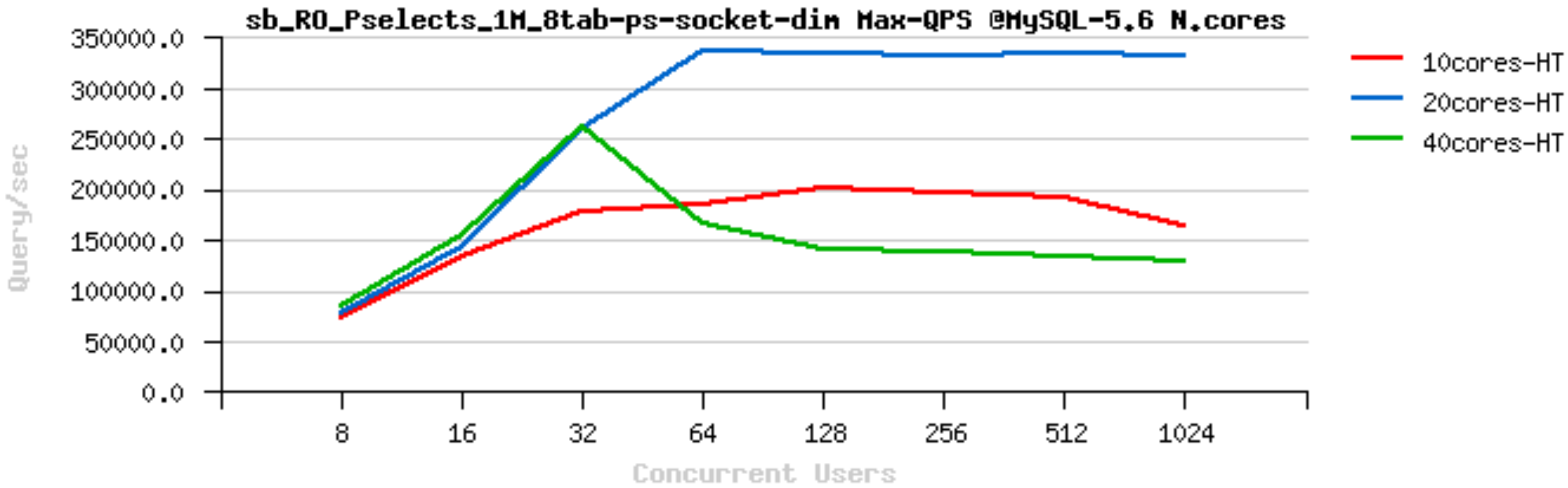
Few words about RO scalability (2)

- OLTP_RO Point-selects 8-tables
  - MySQL 5.5 : Max QPS is @10cores...



Few words about RO scalability (3)

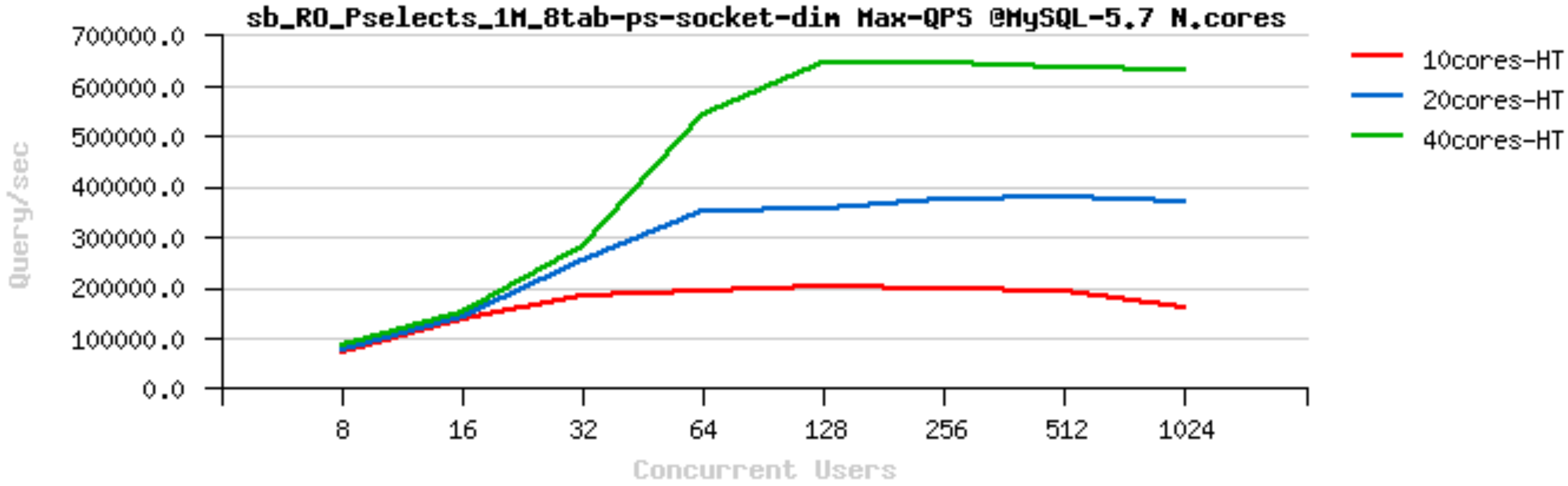
- OLTP_RO Point-selects 8-tables
  - MySQL 5.6 : Max QPS is @20cores..



ORACLE

Few words about RO scalability (4)

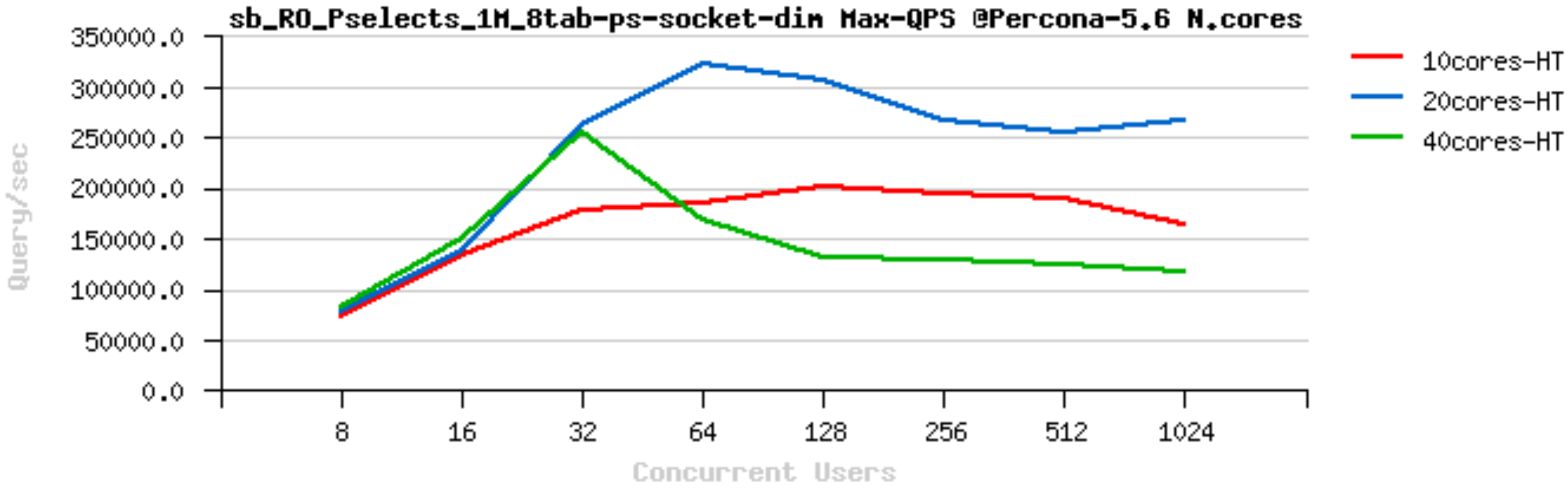
- OLTP_RO Point-selects 8-tables
  - MySQL 5.7 : Max QPS is @40cores (finally! ;-))



ORACLE

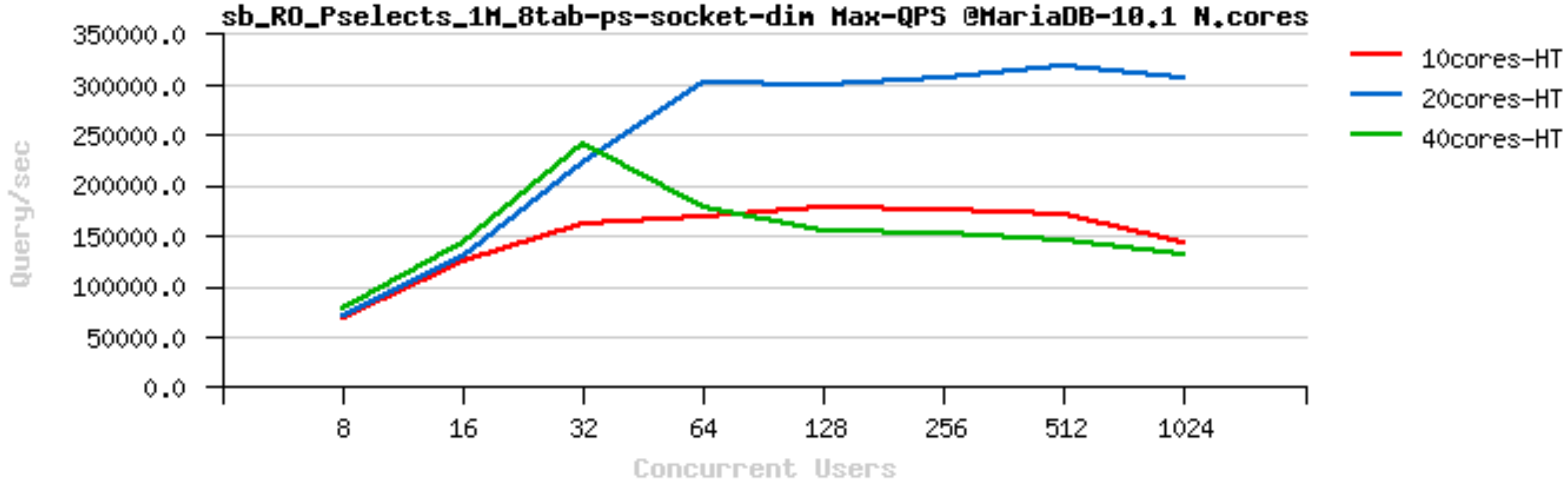
Few words about RO scalability (5)

- OLTP_RO Point-selects 8-tables
  - Percona Server 5.6 : Max QPS is @20cores..



Few words about RO scalability (6)

- OLTP_RO Point-selects 8-tables
  - MariaDB 10.1 : Max QPS is @20cores..



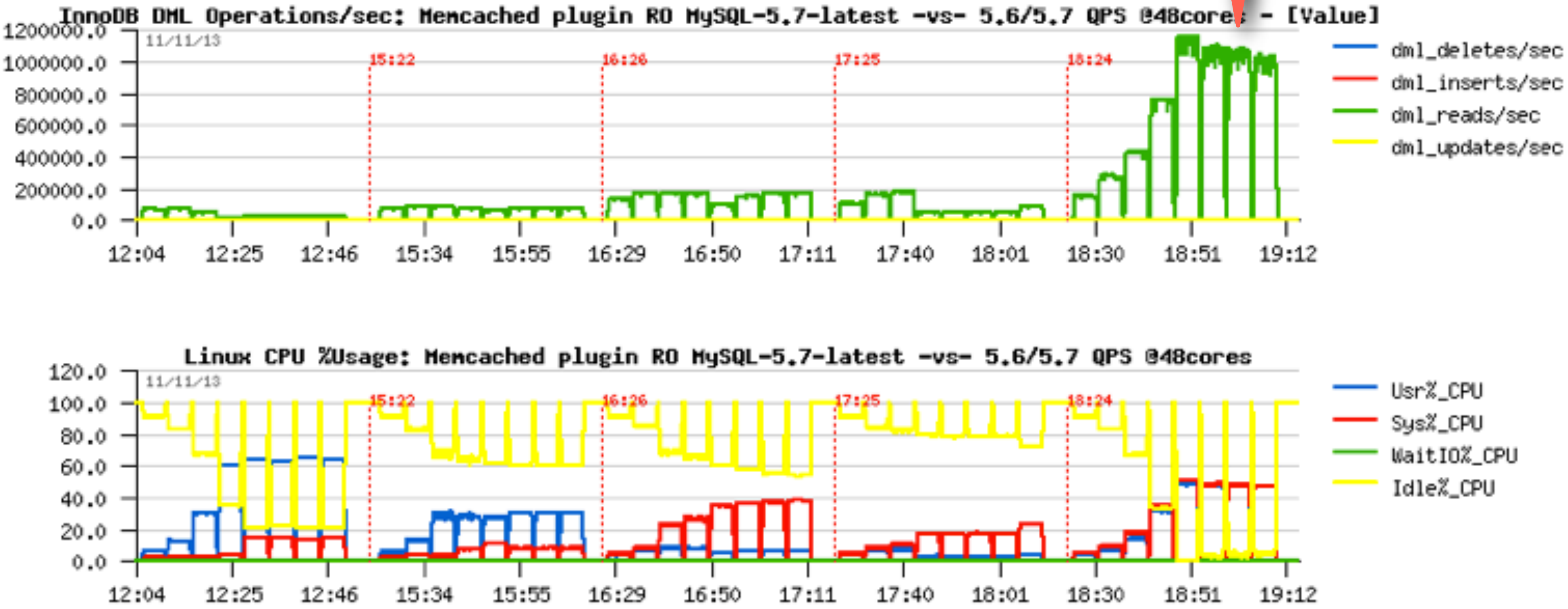
InnoDB Memcached

- MySQL 5.6 :
  - initially introduced
  - QPS : not too much better than SQL..
- MySQL 5.7 :
  - improved TRX list code opened many doors ;-)
  - Facebook => tech talk + test case
  - InnoDB => 1M QPS ;-)
    - 32cores-HT : **900K** QPS
    - 40cores-HT : **1000K** QPS
    - 48cores-HT : **1100K** QPS

InnoDB Memcached @MySQL 5.7

- **Over 1M (!) QPS** on 48cores-HT :

That's it ;-)



ORACLE

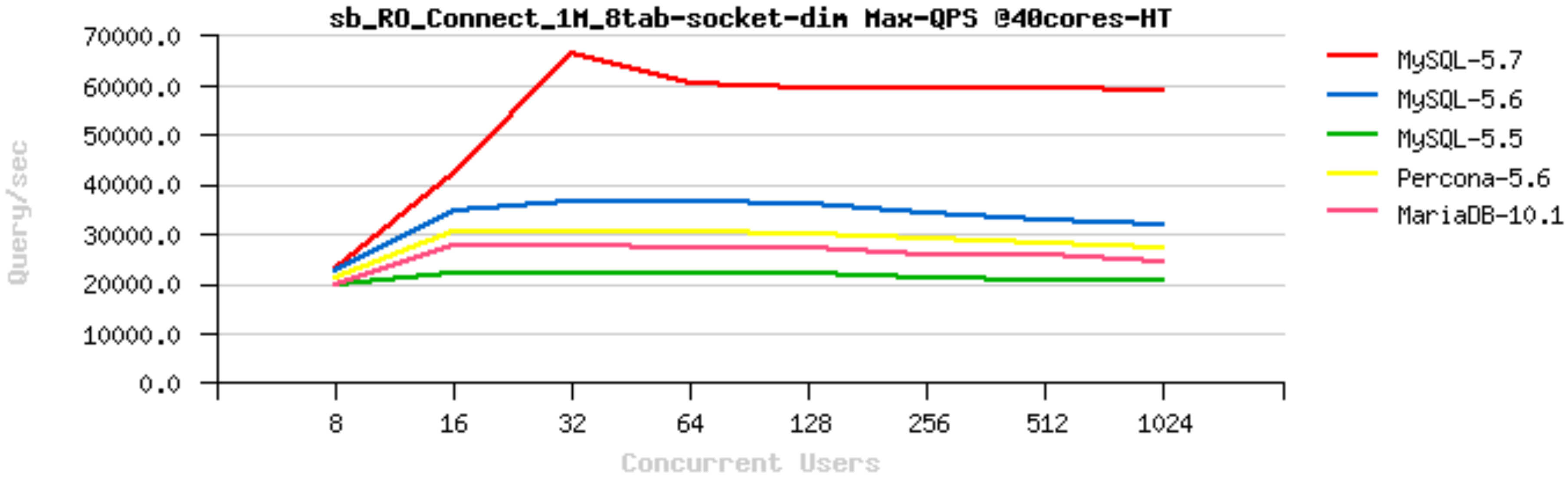
Story #3 : Connect/sec performance

- A true TeamWork :
  - starting with a bug report about PFS overhead on user connect..
  - analyze of PFS issue is pointing also on some hot contentions around connect / disconnect..
  - PFS instrumentation is improved then by ServerGen Team
  - while Runtime Team comes back with yet more ideas for “connect” code
  - the result : x2 times better Connect/sec performance than before! ;-)
- NOTE :
  - the Connect performance was already greatly improved in 5.6 and 5.7
  - this was the next step in Connect speed-up ;-)
- Why Connect/sec performance is important?
  - for many web sites it will be one of the main show-stoppers

ORACLE

OLTP_RO Connect Performance

- 40cores-HT 2.3Ghz : **65K** Connect/sec
  - 1 single point-select per Connect/Disconnect
  - localhost
  - the result is yet more higher on a **faster** CPU



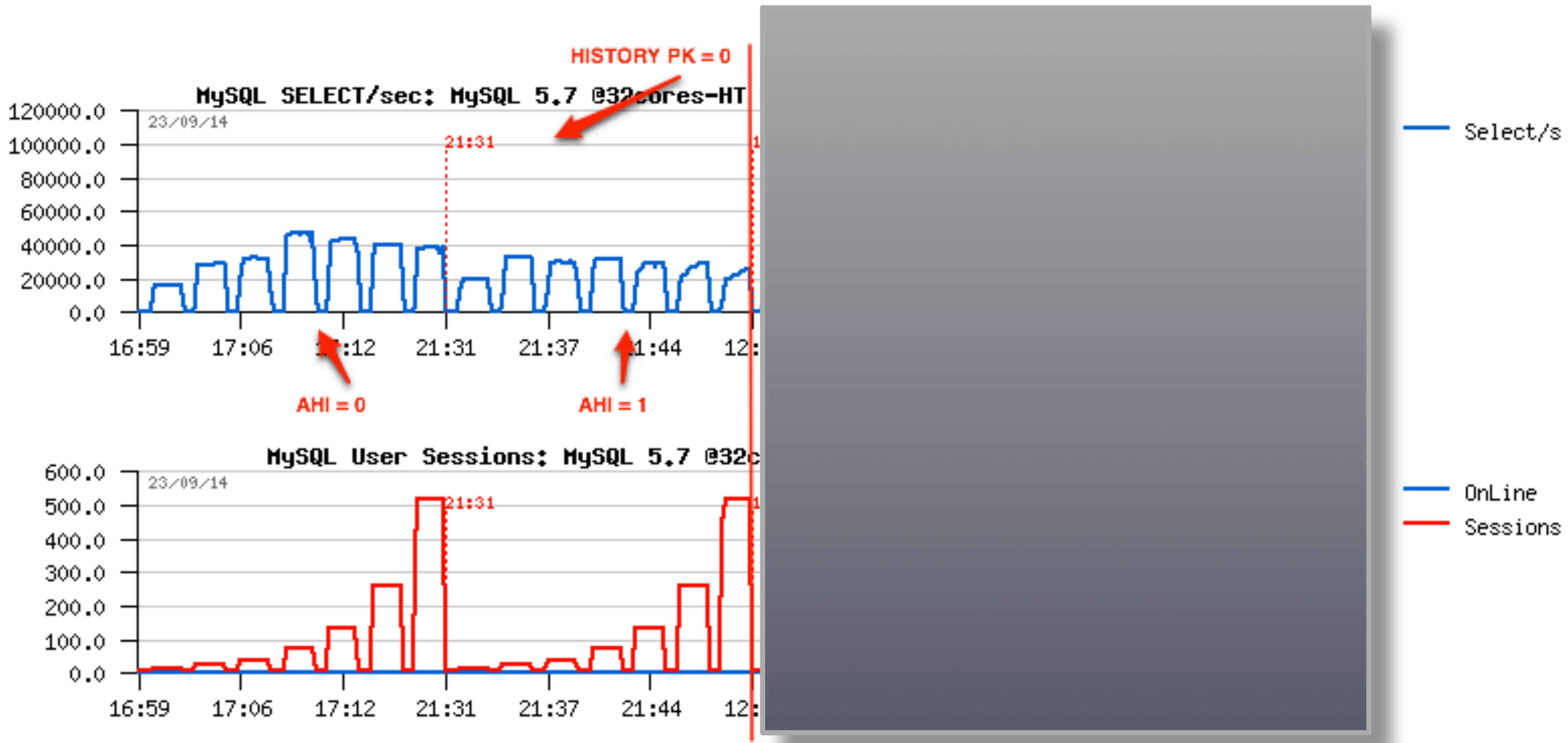
Story #4 : strange scalability issue @dbSTRESS

- Preface:
  - we already observed in the past some strange scalability problems
  - most are gone since “G5 patch” (CPU false caching)
  - but on dbSTRESS the problem remained..
  - lack of needed instrumentation & profiling kept investigation on stand-by
  - finally took some time to analyze it more in detail now ;-)
- Schema:
  - State [1K] <= History [200M] <= Object [10M] => Section[100] =>Zone [10]
  - SEL1 : for Object #id SELECT Object => Section =>Zone
  - SEL2 : for Object #id SELECT all History => State

ORACLE

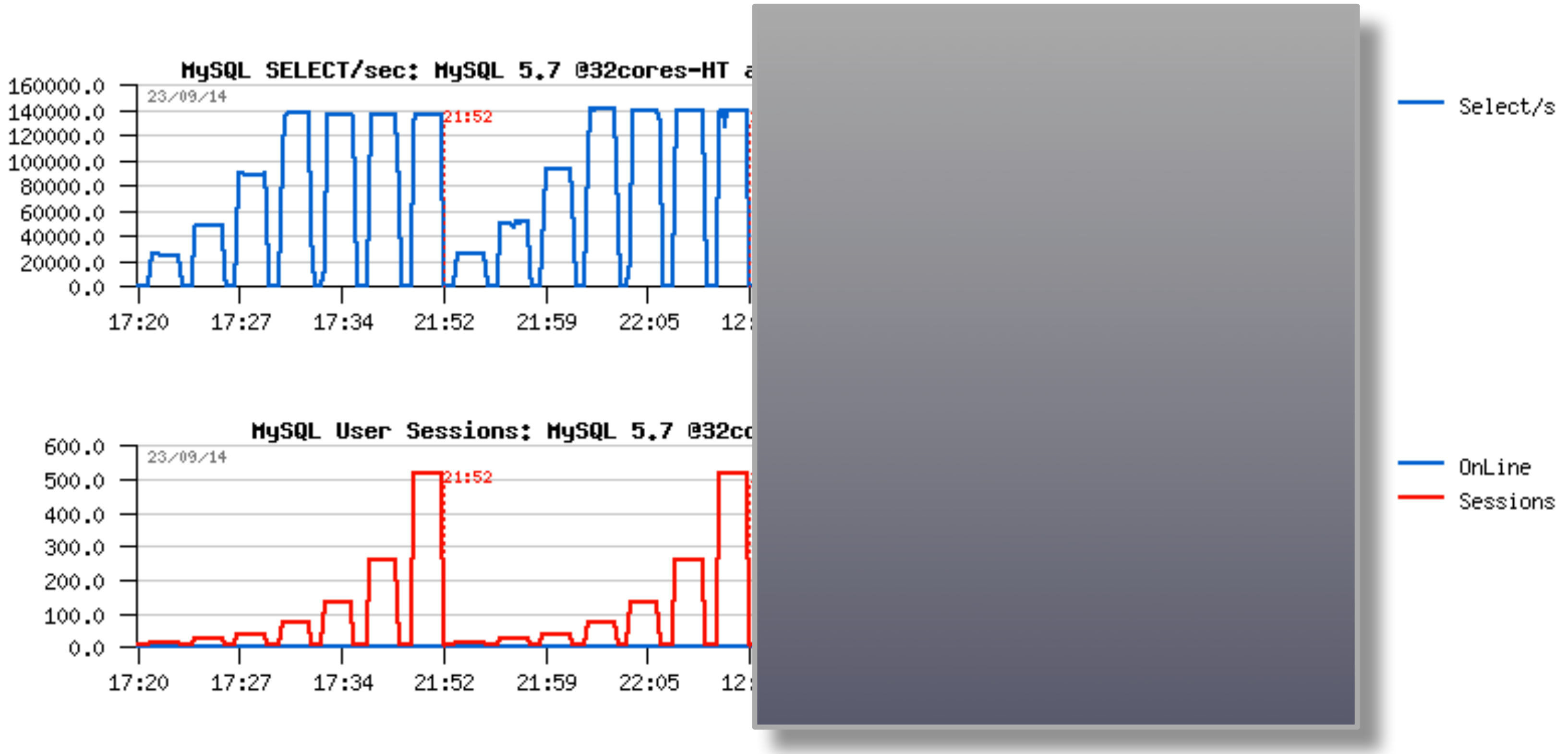
Story #4 : strange scalability issue @dbSTRESS (2)

- 32cores-HT, MySQL 5.7
  - test : dbSTRESS-RO, AHI = off / on



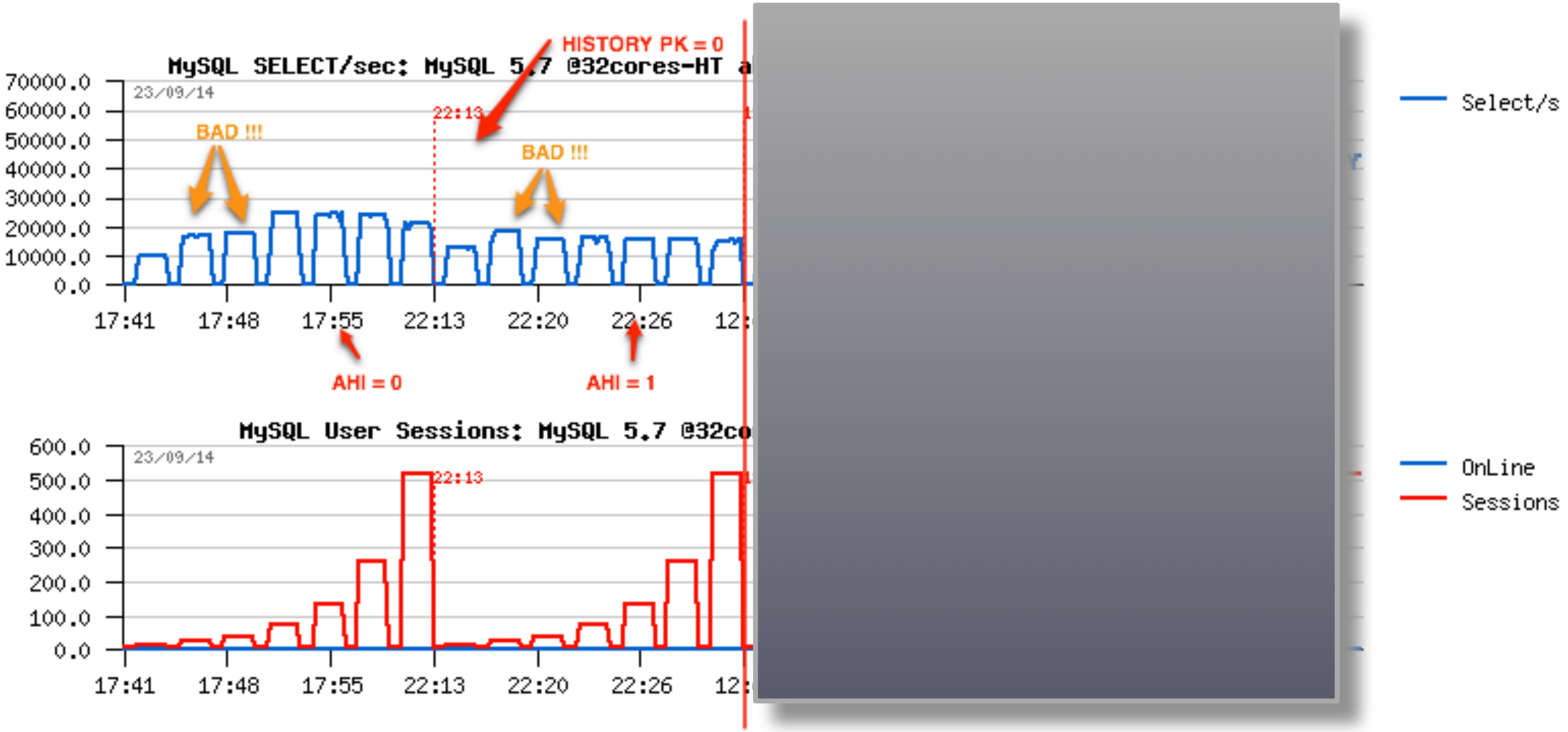
Story #4 : strange scalability issue @dbSTRESS (3)

- 32cores-HT, MySQL 5.7
  - test : dbSTRESS-SEL1, AHI = off / on



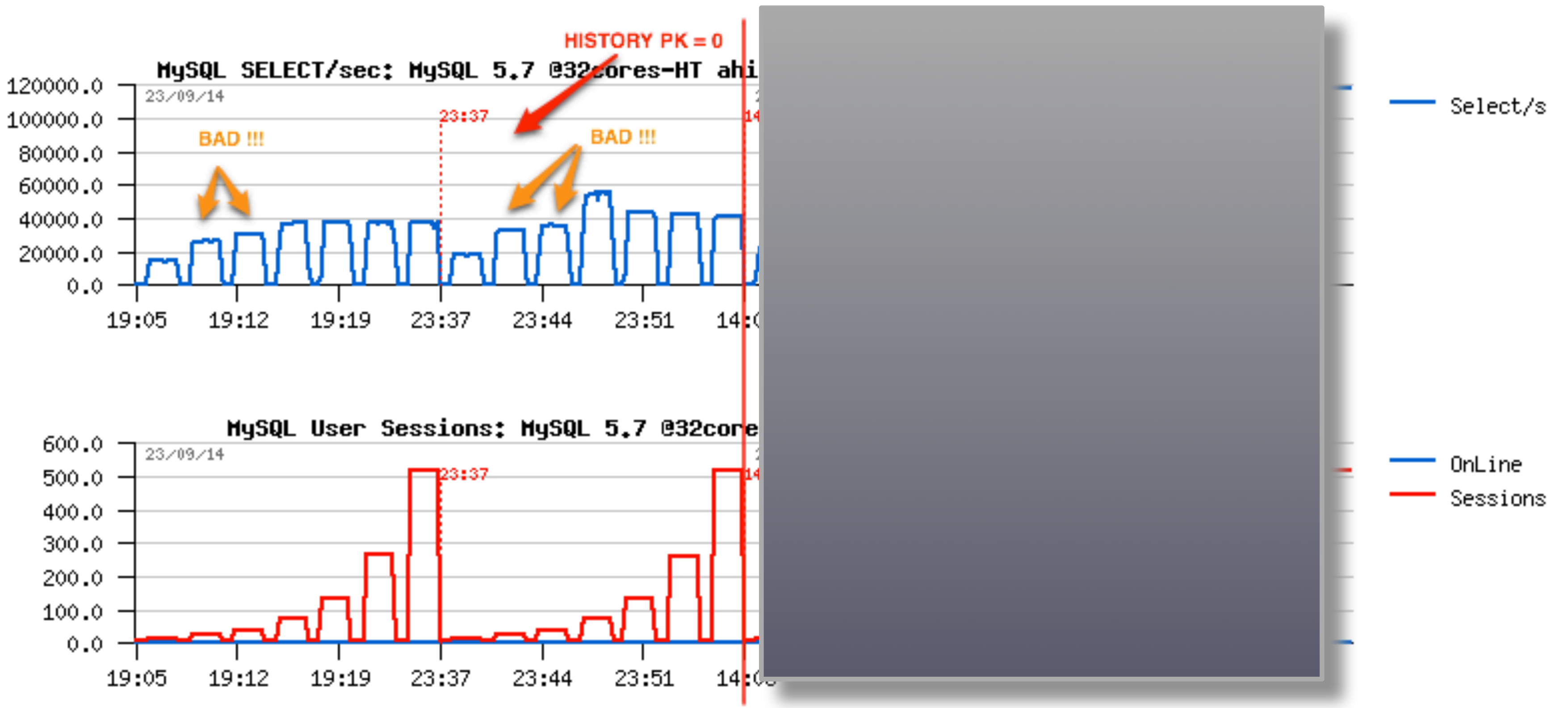
Story #4 : strange scalability issue @dbSTRESS (4)

- 32cores-HT, MySQL 5.7
  - test : dbSTRESS-SEL2, AHI = off / on



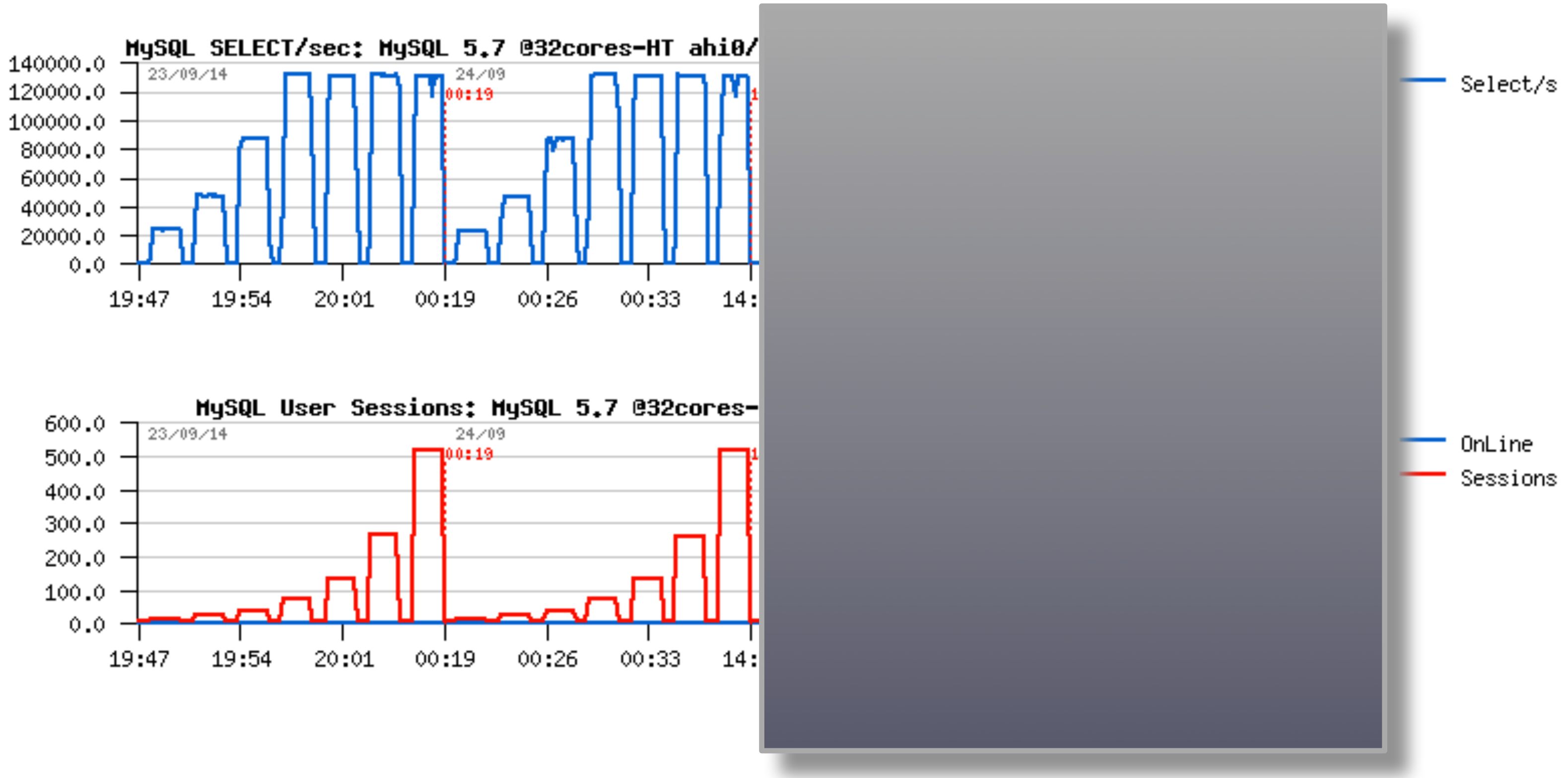
Story #4 : strange scalability issue @dbSTRESS (5)

- 32cores-HT, MySQL 5.7
  - test : dbSTRESS-H1, AHI = off / on



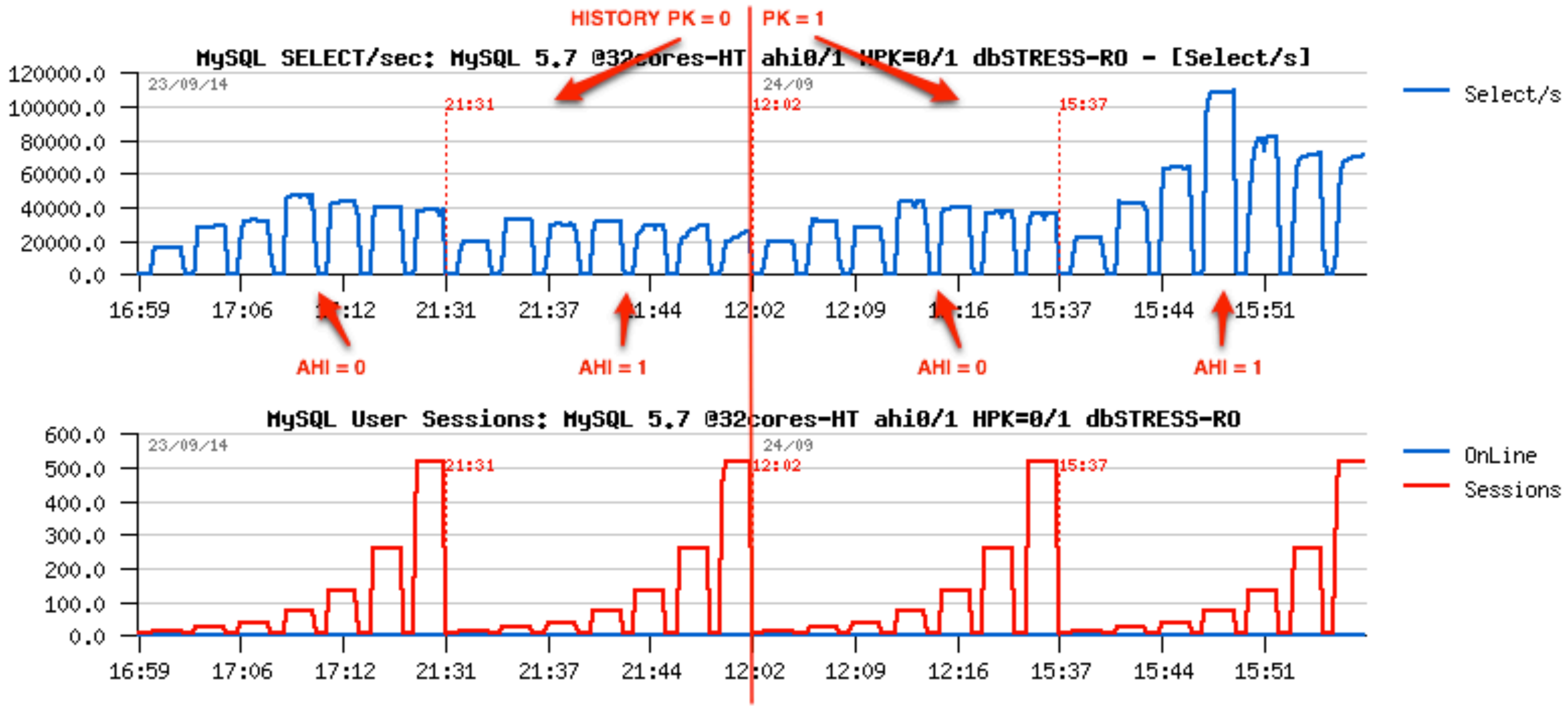
Story #4 : strange scalability issue @dbSTRESS (6)

- 32cores-HT, MySQL 5.7
  - test : dbSTRESS-H1_hord, AHI = off / on



Story #4 : strange scalability issue @dbSTRESS (2/2)

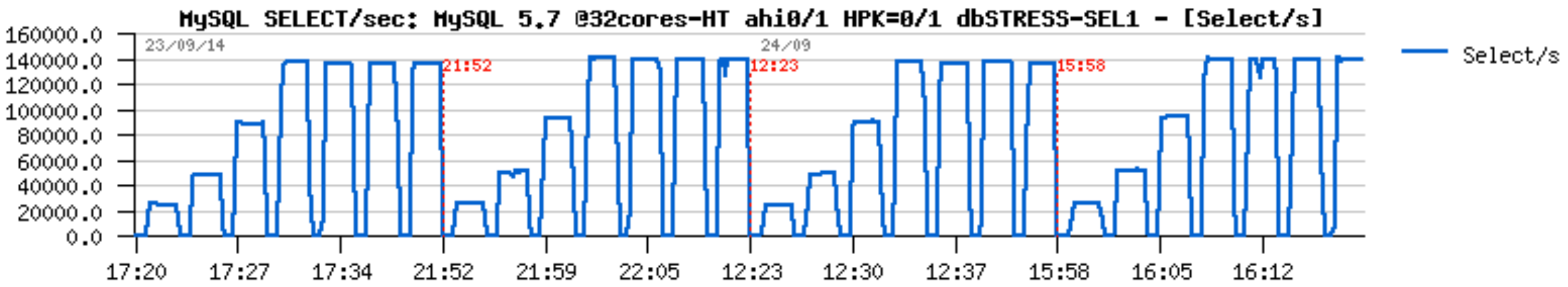
- 32cores-HT, MySQL 5.7
  - test : dbSTRESS-RO, AHI = off / on

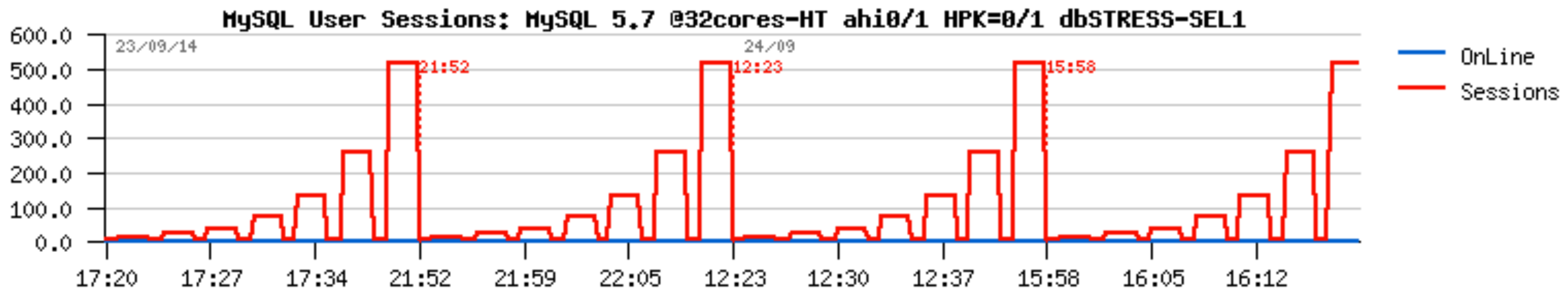


ORACLE

Story #4 : strange scalability issue @dbSTRESS (3/2)

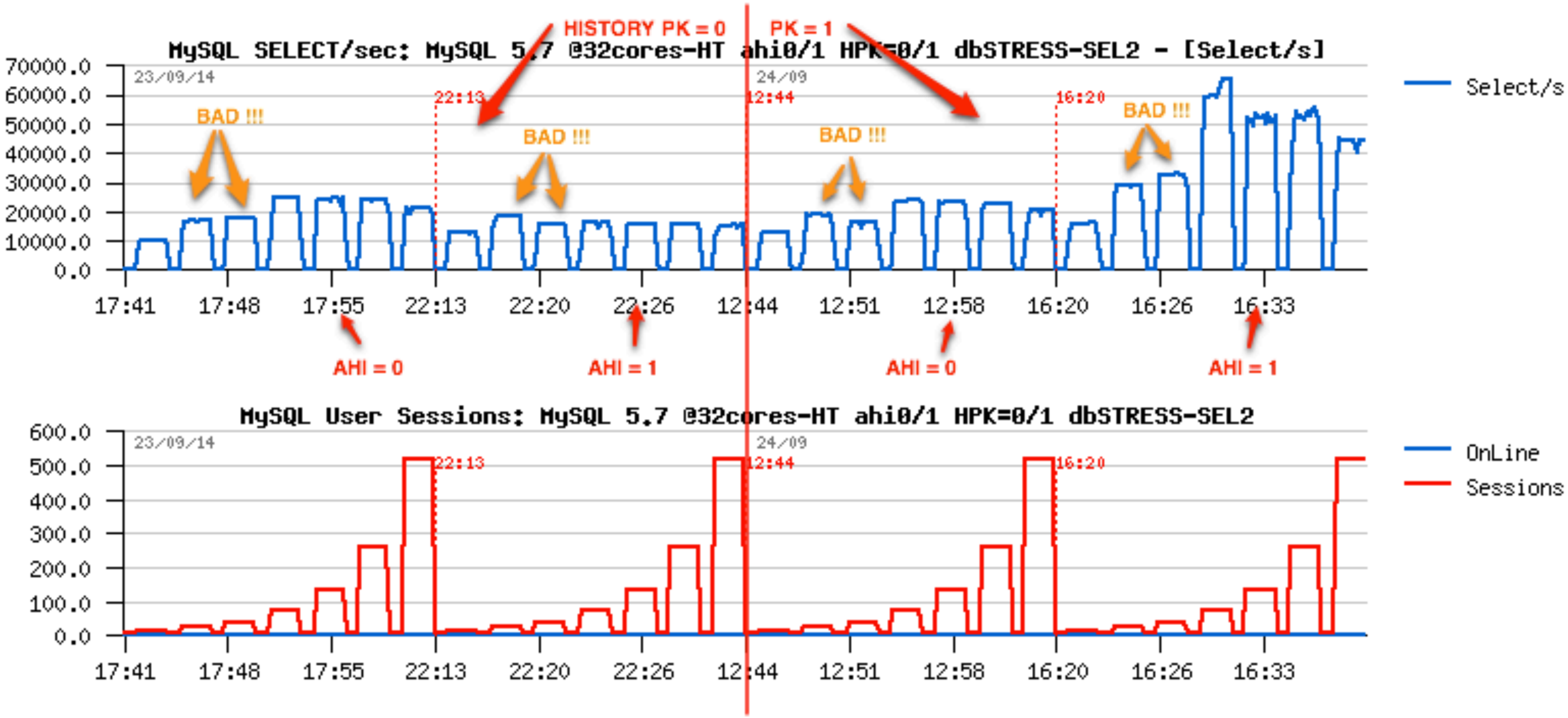
- 32cores-HT, MySQL 5.7
  - test : dbSTRESS-SEL1, AHI = off / on





Story #4 : strange scalability issue @dbSTRESS (4/2)

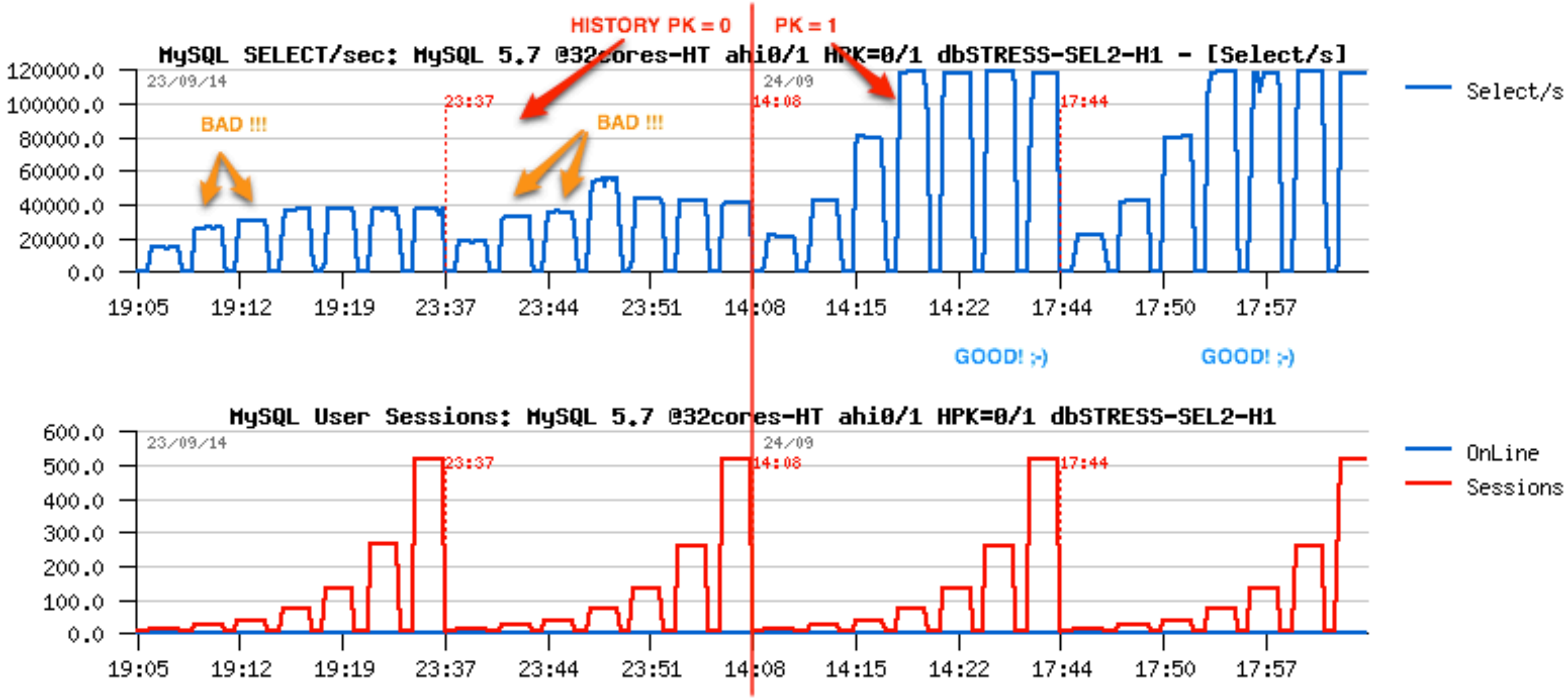
- 32cores-HT, MySQL 5.7
  - test : dbSTRESS-SEL2, AHI = off / on



ORACLE

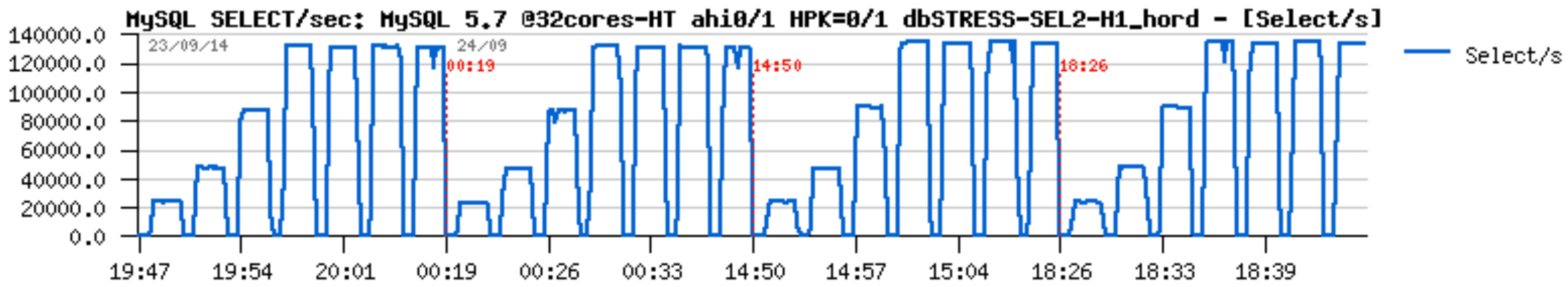
Story #4 : strange scalability issue @dbSTRESS (5/2)

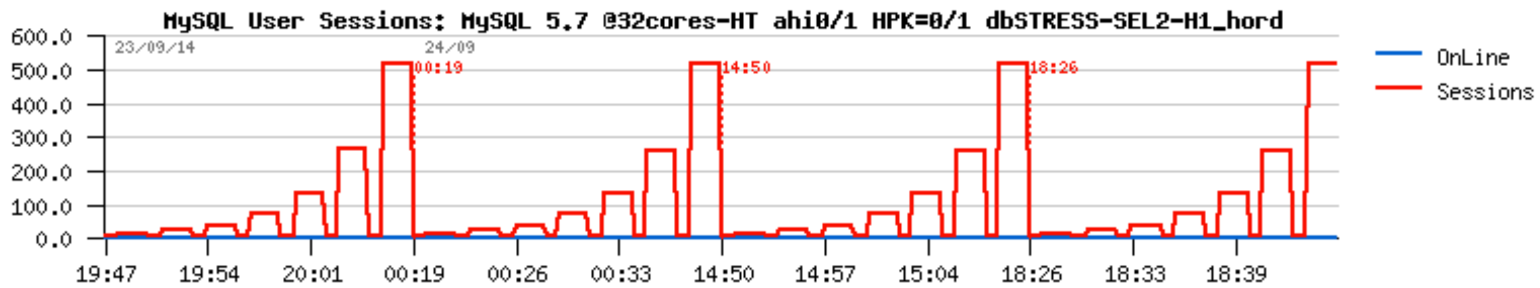
- 32cores-HT, MySQL 5.7
  - test : dbSTRESS-H1, AHI = off / on



Story #4 : strange scalability issue @dbSTRESS (6/2)

- 32cores-HT, MySQL 5.7
  - test : dbSTRESS-H1_hord, AHI = off / on





ORACLE

Story #4 : strange scalability issue @dbSTRESS (7)

- 32cores-HT, MySQL 5.7
  - test case workload dbSTRESS-SEL2 <== the main show-stopper..
  - amazing to see the x3 time performance difference between PK access and secondary index..
  - what do you see in your own workloads and productions systems?
  - the fix is in TODO, but not for tomorrow..
  - NOTE : pfs_* names in profiler are not always related to PFS ;-)

```
24925.00 19.5% pfs_end_rwlock_rdwait_v1              mysqld-575-withPFS-03-Sep
11686.00  9.2% pfs_unlock_rwlock_v1                  mysqld-575-withPFS-03-Sep
 8554.00  6.7% pfs_start_rwlock_wait_v1              mysqld-575-withPFS-03-Sep
 4503.00  3.5% btr_cur_search_to_nth_level(dict_inde mysqld-575-withPFS-03-Sep
 4413.00  3.5% rec_get_offsets_func(unsigned char co mysqld-575-withPFS-03-Sep
 3536.00  2.8% buf_page_get_gen(page_id_t const&, pa mysqld-575-withPFS-03-Sep
 3056.00  2.4% pfs_rw_lock_s_unlock_func(rw_lock_t*) mysqld-575-withPFS-03-Sep
  ...
```

ORACLE

Read+Write Workloads Scalability @MySQL

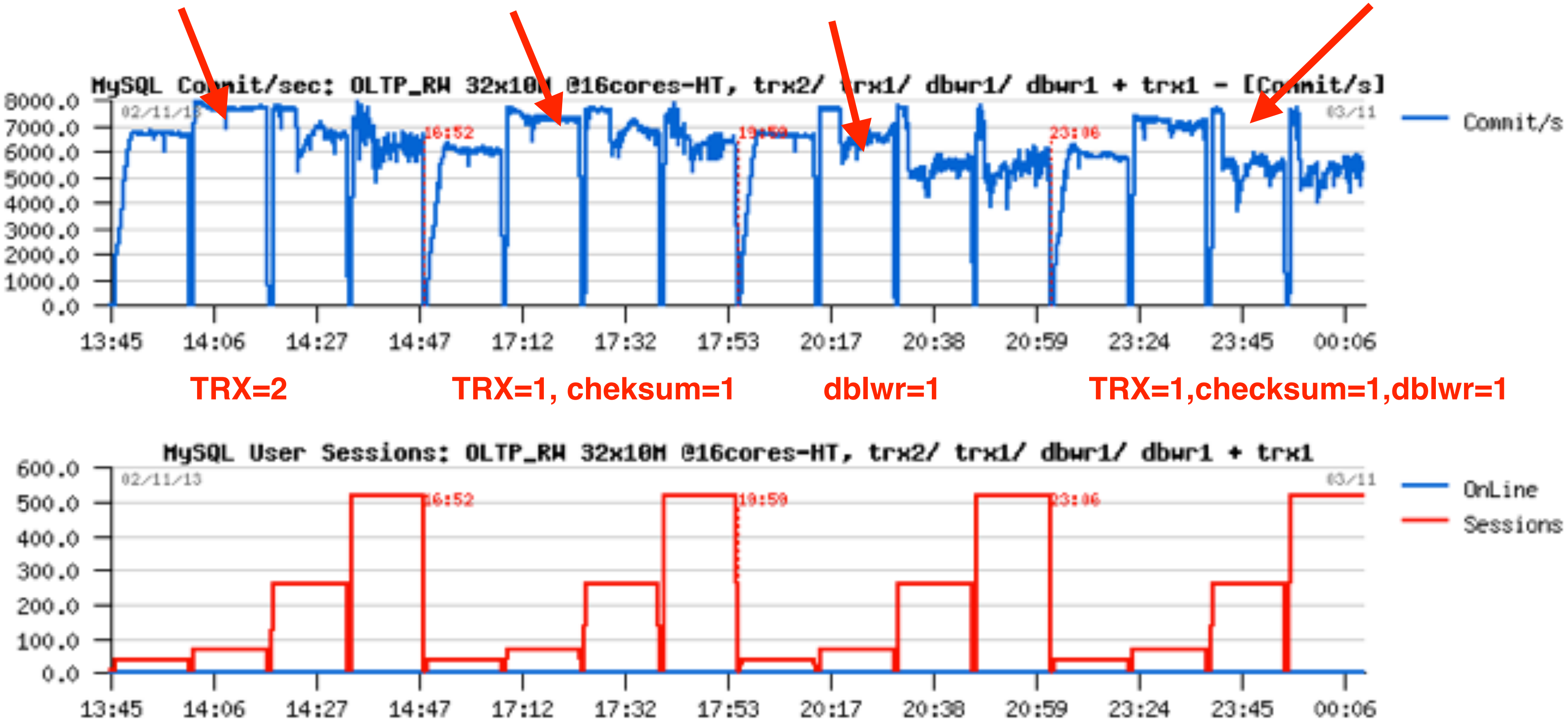
- Huge progress is already here!
- However, not yet as good as Read-Only..
  - Performance continues to increase with more CPU cores
  - But on move from 16 to 32cores-HT you may gain only 50% better
  - Better performance on a faster storage as well
  - But cannot yet use a full power of fast flash for today..
  - Work in progress ;-)
  - Internal contentions & Design limitations are the main issues here..

Read+Write Performance @MySQL / InnoDB

- Transactional processing
  - your CPU-bound transactional processing defines your Max possible TPS
  - with a bigger volume / more IO / etc. => Max TPS will not increase ;-)
- Data Safety
  - binlog : overhead + bottleneck (be sure you have binlog group commit)
  - InnoDB checksums : overhead (reasonable since crc32 is used)
  - innodb_flush_log_at_trx_commit = 1 : overhead + bottleneck
  - InnoDB double write buffer : **KILLER** ! overhead + huge bottleneck..
    - need a fix / re-design / etc. in urgency ;-)
    - Fusion-io atomic writes is one of (**true** support in MySQL 5.7)
    - Using EXT4 with data journal is another one
    - but a true re-design is still preferable ;-)

Impact of “safety” options..

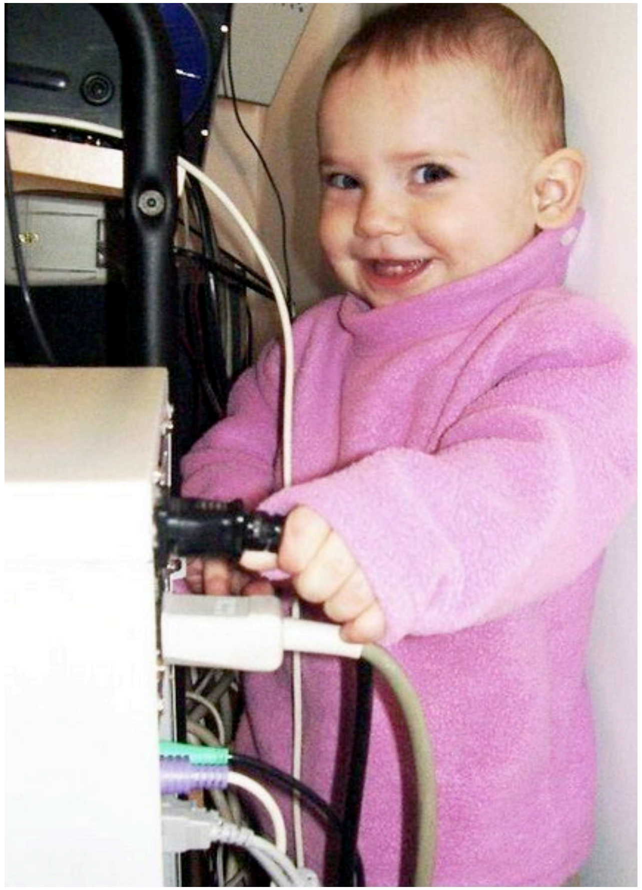
- OLTP_RW 32x10M-tables @Percona-5.6
  - trx=2 | trx=1 + chksum=1 | dblwr=1 | trx=1 + chksum=1 + dblwr=1



RW related starter configuration settings

- my.conf :

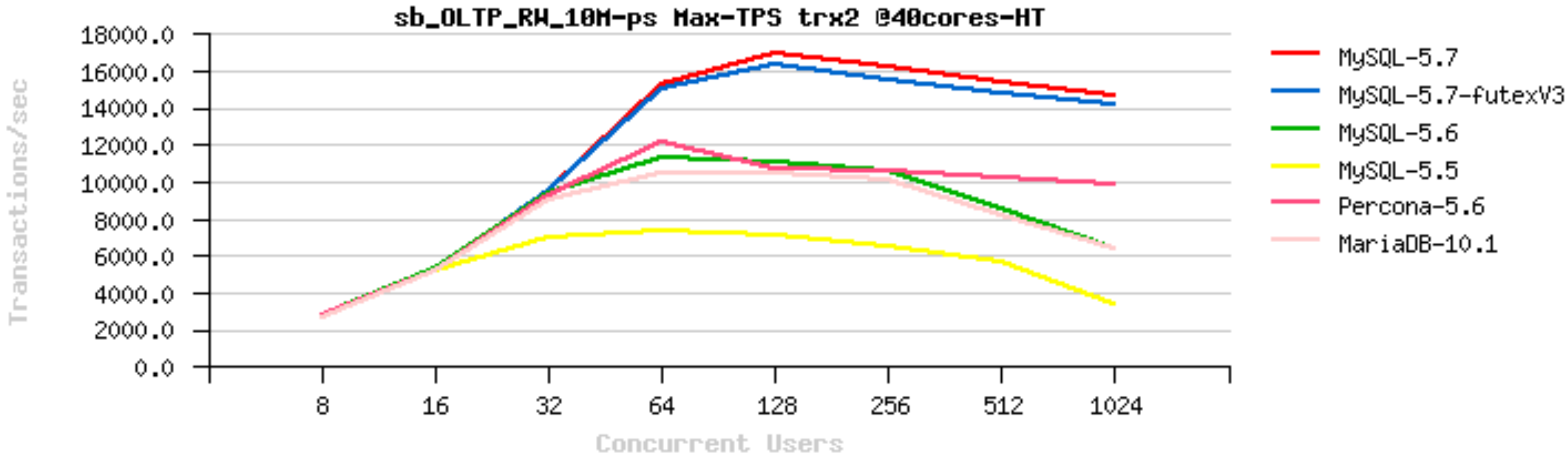
```
innodb_file_per_table
innodb_log_file_size=1024M
innodb_log_files_in_group=3 / 12 / ...
innodb_checksum_algorithm= none / crc32
innodb_doublewrite= 0 / 1
innodb_flush_log_at_trx_commit= 2 / 1
innodb_flush_method=O_DIRECT
innodb_use_native_aio=1
innodb_adaptive_hash_index=0

innodb_adaptive_flushing = 1
innodb_flush_neighbors = 0
innodb_read_io_threads = 16
innodb_write_io_threads = 16
innodb_io_capacity=15000
innodb_max_dirty_pages_pct=90
innodb_max_dirty_pages_pct_lwm=10
innodb_lru_scan_depth=4000
innodb_page_cleaners=4

innodb_purge_threads=4
innodb_max_purge_lag_delay=30000000
innodb_max_purge_lag=1000000

binlog ??
```



ORACLE

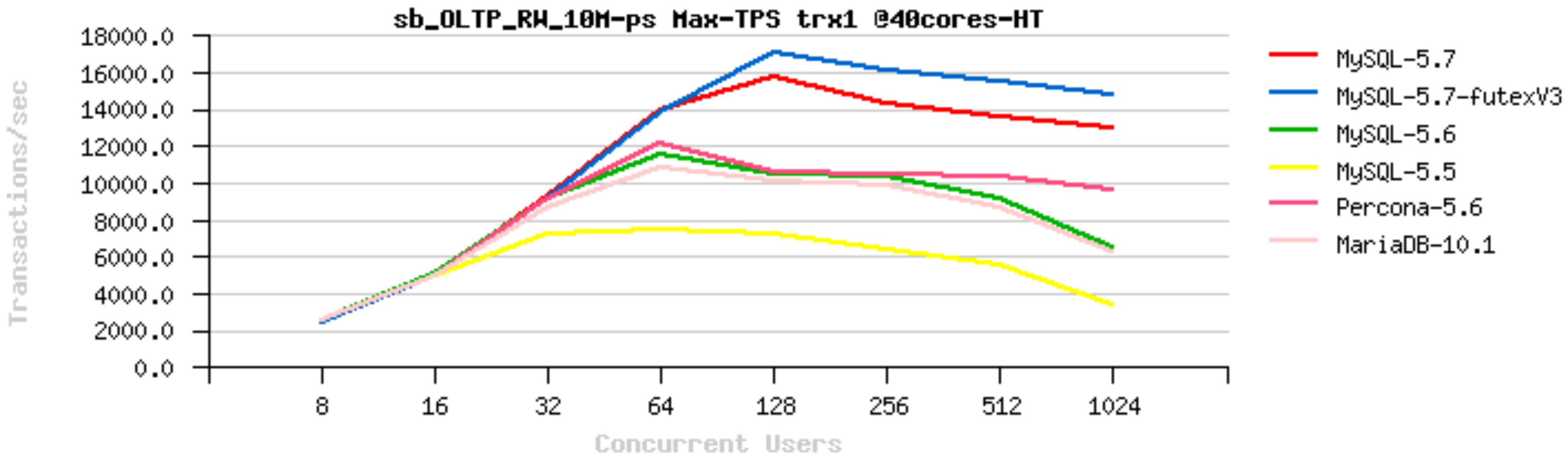
Sysbench OLTP_RW In-Memory

- Sysbench OLTP_RW **1-table** TRX2 @40cores-HT :
  - TRX2 : innodb_flush_log_at_trx_commit = 2



Sysbench OLTP_RW In-Memory

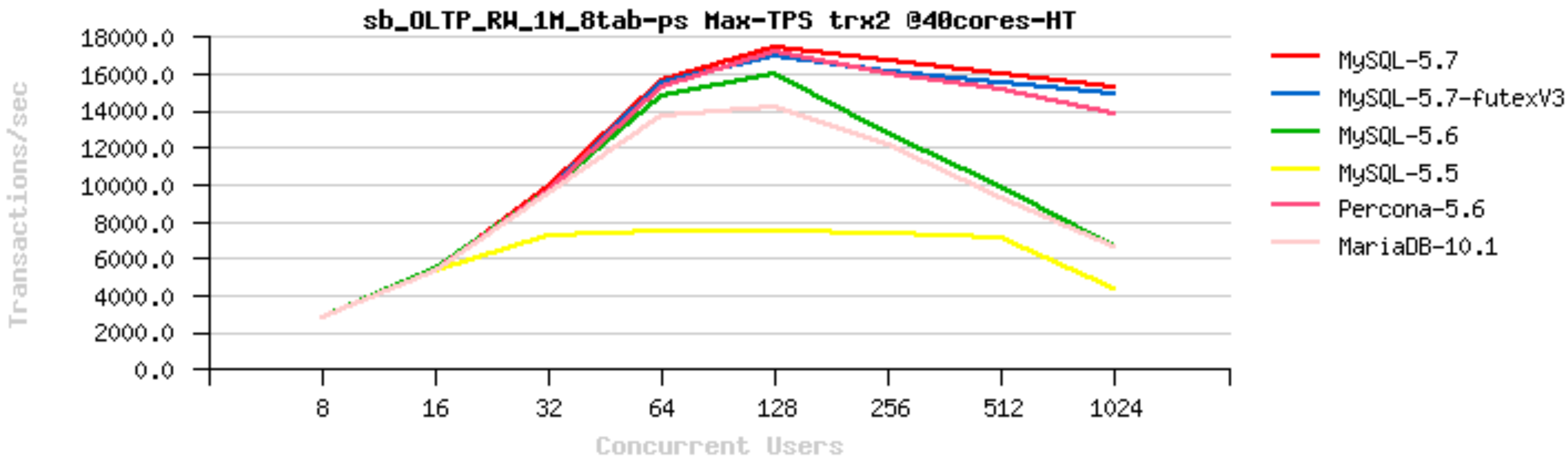
- Sysbench OLTP_RW **1-table** TRX1 @40cores-HT :
  - TRX1 : innodb_flush_log_at_trx_commit = 1



ORACLE

Sysbench OLTP_RW In-Memory

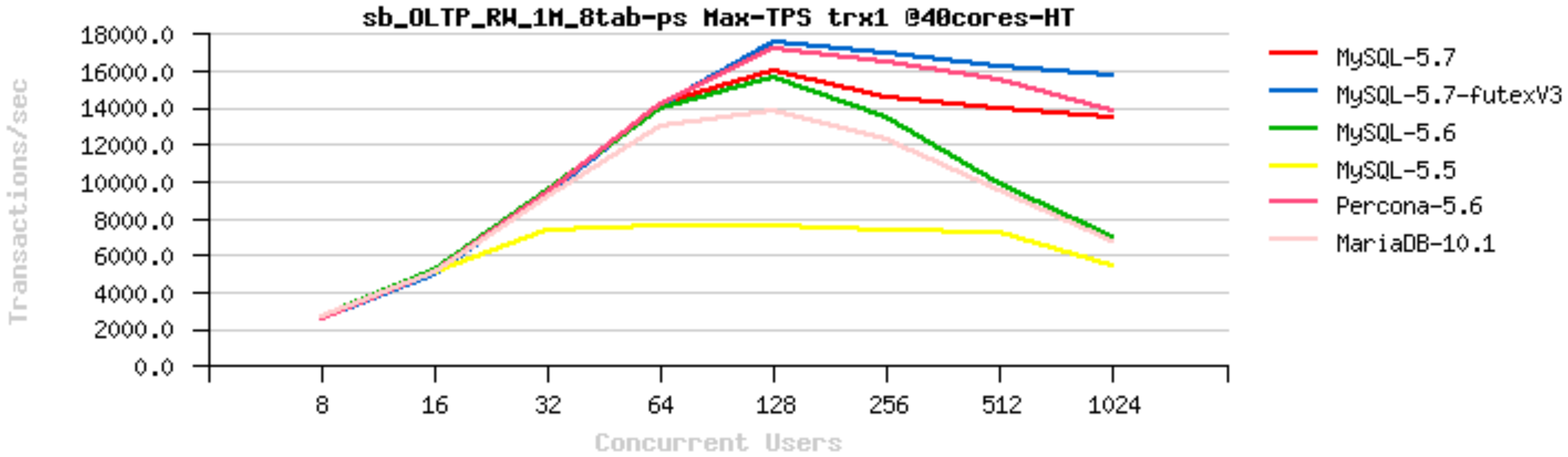
- Sysbench OLTP_RW **8-tables** TRX2 @40cores-HT :
  - TRX2 : innodb_flush_log_at_trx_commit = 2



ORACLE

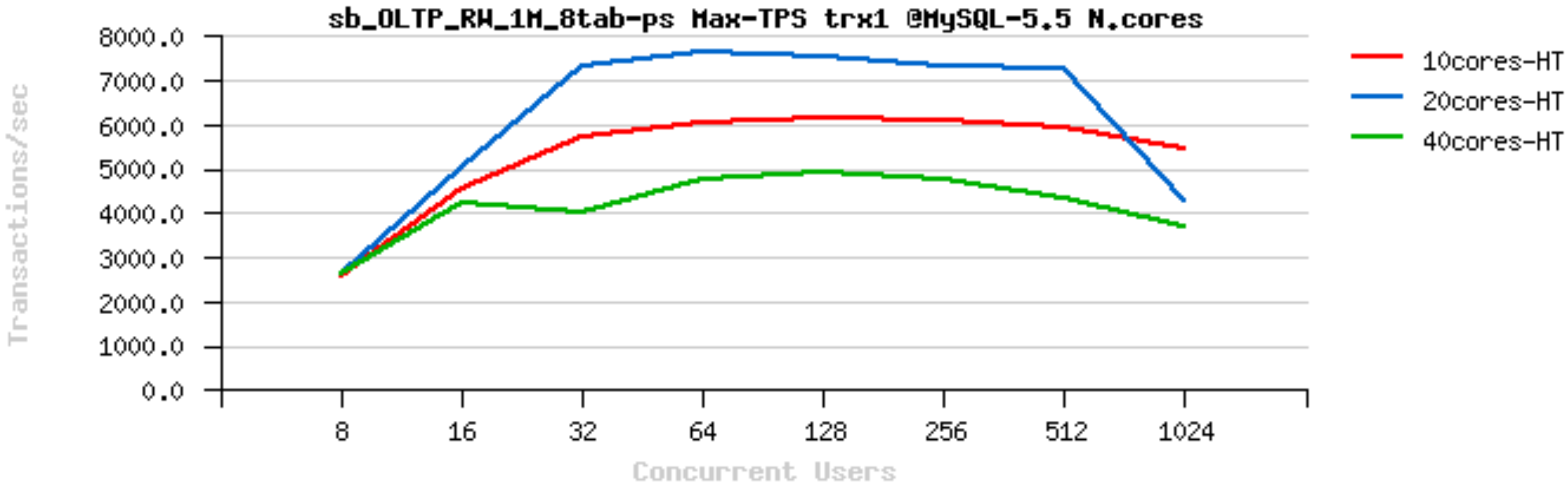
Sysbench OLTP_RW In-Memory

- Sysbench OLTP_RW **8-tables** TRX1 @40cores-HT :
  - TRX1 : innodb_flush_log_at_trx_commit = 1



Sysbench OLTP_RW In-Memory

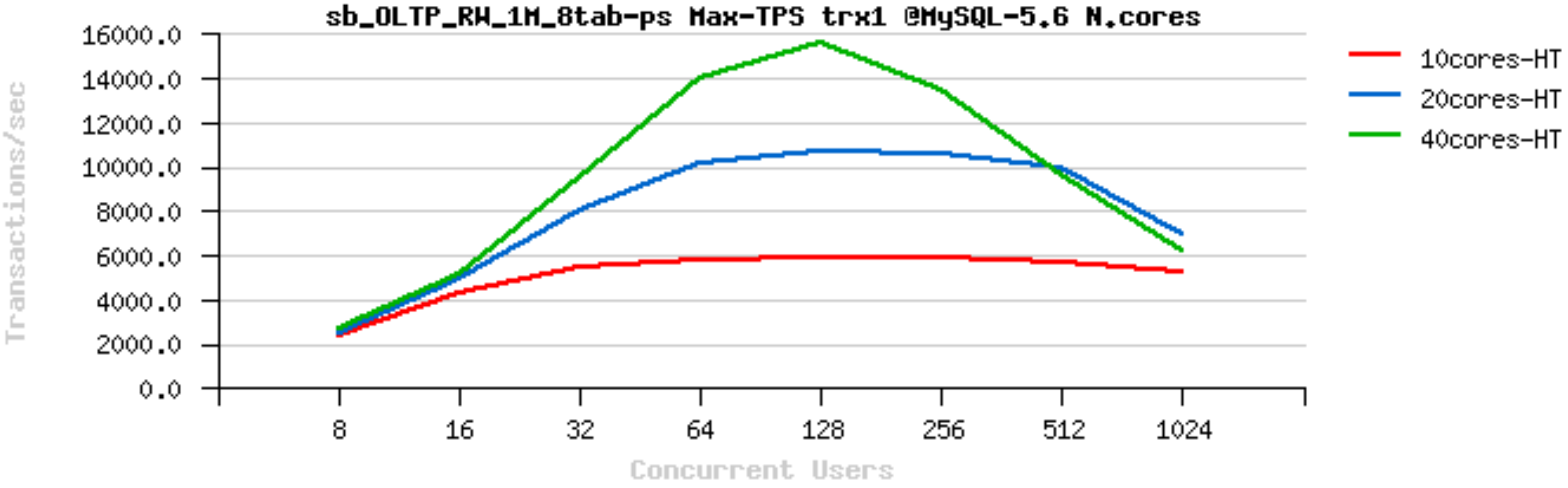
- Sysbench OLTP_RW 8-tables TRX1 @40cores-HT :
  - MySQL 5.5 : Max TPS @20cores



ORACLE

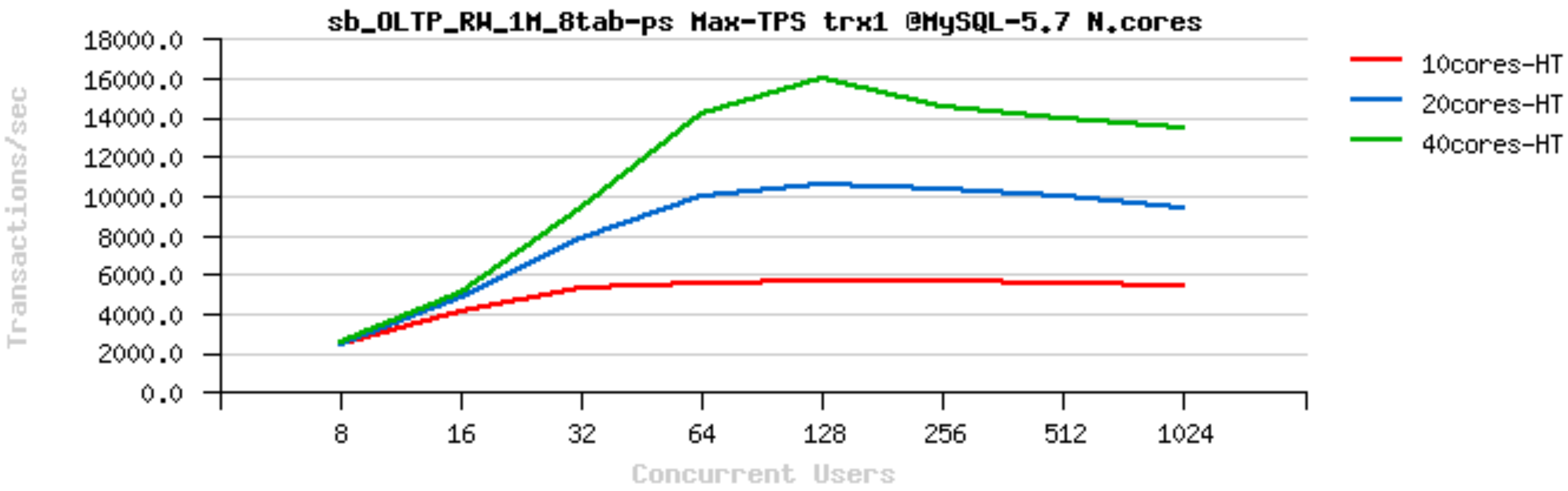
Sysbench OLTP_RW In-Memory

- Sysbench OLTP_RW 8-tables TRX1 @40cores-HT :
  - MySQL 5.6 : Max TPS @40cores



Sysbench OLTP_RW In-Memory

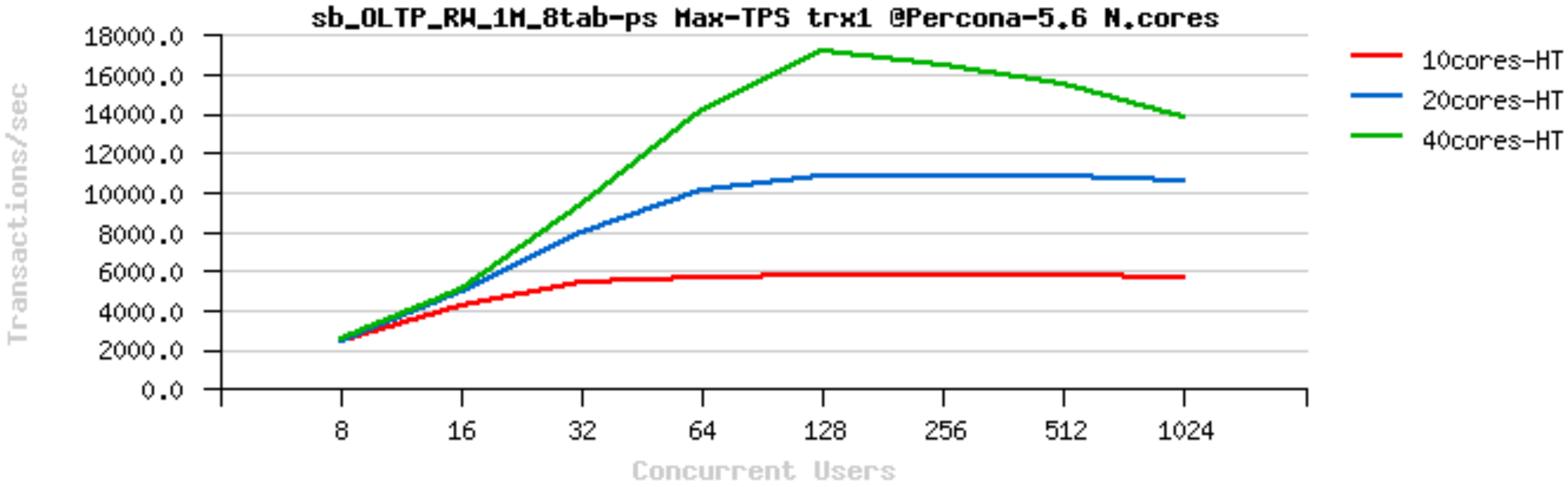
- Sysbench OLTP_RW 8-tables TRX1 @40cores-HT :
  - MySQL 5.7 : Max TPS @40cores



ORACLE

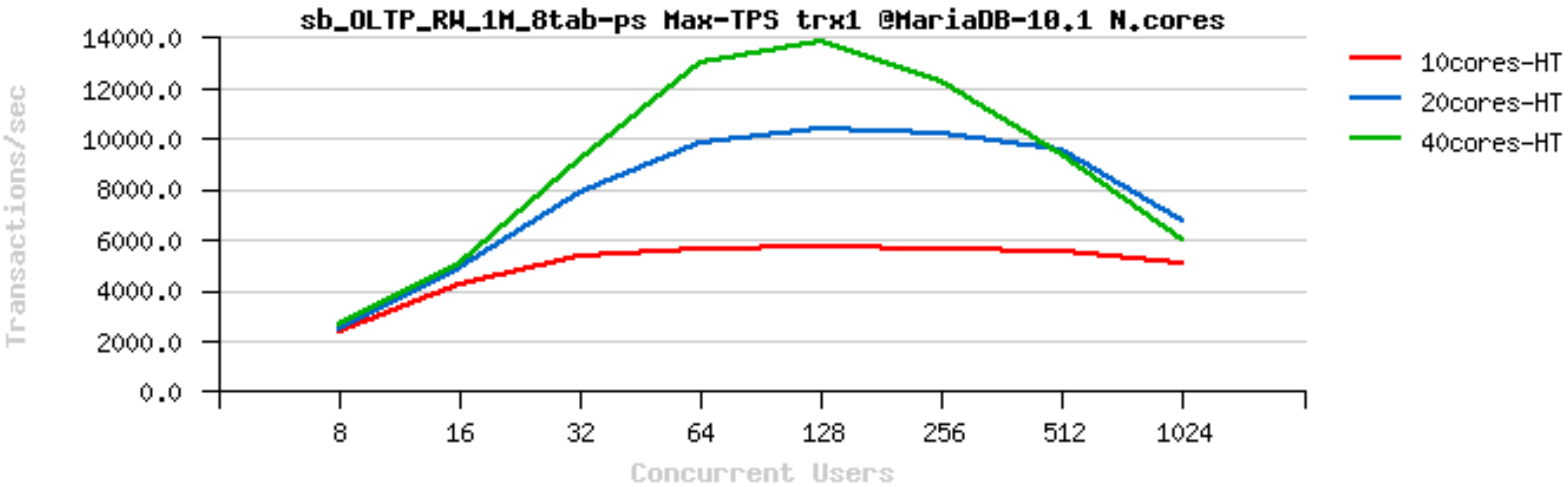
Sysbench OLTP_RW In-Memory

- Sysbench OLTP_RW 8-tables TRX1 @40cores-HT :
  - Percona Server 5.6 : Max TPS @40cores



Sysbench OLTP_RW In-Memory

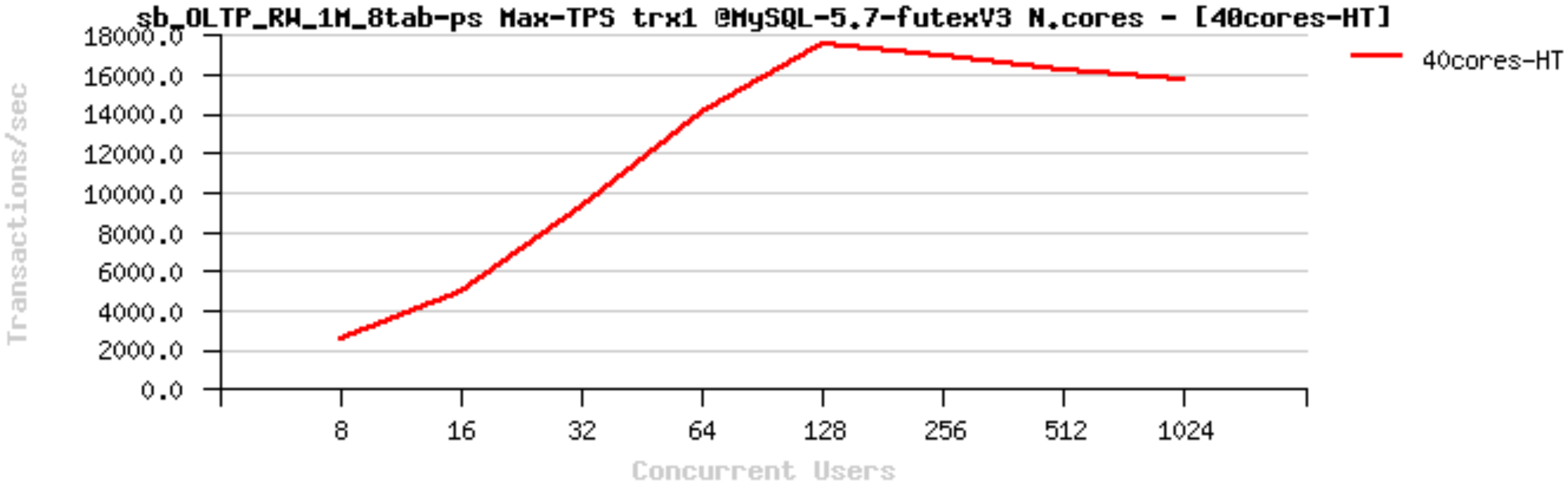
- Sysbench OLTP_RW 8-tables TRX1 @40cores-HT :
  - MariaDB 10.1 : Max TPS @40cores



ORACLE

Sysbench OLTP_RW In-Memory

- Sysbench OLTP_RW 8-tables TRX1 @40cores-HT :
  - MySQL 5.7-dev3 : Max TPS @40cores



ORACLE

Sysbench OLTP_RW In-Memory

- Max TPS configs :

```
mysql> select max(tps), t_engine, t_ccr, spin_delay, trx_commit
          from Bench where t_name = 'sb_OLTP_RW_1M_8tab-ps'
           and t_tag= '575_DMR-RW' and t_cpu like '40cores%'
         group by 2,3,4,5 order by 1 desc  limit 10;

+----------+-------------------+-------+------------+------------+
| max(tps) | t_engine          | t_ccr | spin_delay | trx_commit |
+----------+-------------------+-------+------------+------------+
|    17617 | mysql576_futex_V3 |    64 |         96 |          1 |
|    17497 | mysql576_futex_V3 |     0 |         96 |          1 |
|    17438 | mysql575          |    64 |         96 |          2 |
|    17307 | mysql575          |     0 |         96 |          2 |
|    17231 | percona5620       |    64 |         96 |          1 |
|    17197 | percona5620       |     0 |         96 |          1 |
|    17168 | percona5620       |     0 |         96 |          2 |
|    17113 | percona5620       |    64 |         96 |          2 |
|    16963 | mysql576_futex_V3 |    64 |         96 |          2 |
|    16780 | mysql576_futex_V3 |     0 |         96 |          2 |
+----------+-------------------+-------+------------+------------+
10 rows in set (0.03 sec)

mysql>
```

RW IO-bound

- Still data In-Memory, but much bigger volume :
  - more pages to flush for the **same** TPS rate
- Data bigger or much bigger than Memory / cache / BP :
  - the amount of free pages becomes short very quickly..
  - and instead of mostly IO writes only you’re starting to have IO reads too
  - these reads usually mostly random reads
  - if your storage is slow - reads will simply kill your TPS ;-)
  - if your storage can follow - once you’re hitting fil_sys mutex you’re done
  - as well LRU flushing may become very heavy..
- NOTE:
  - using **AIO + O_DIRECT** seems to be the most optimal for RW IO-bound
  - but always check yourself ;-)

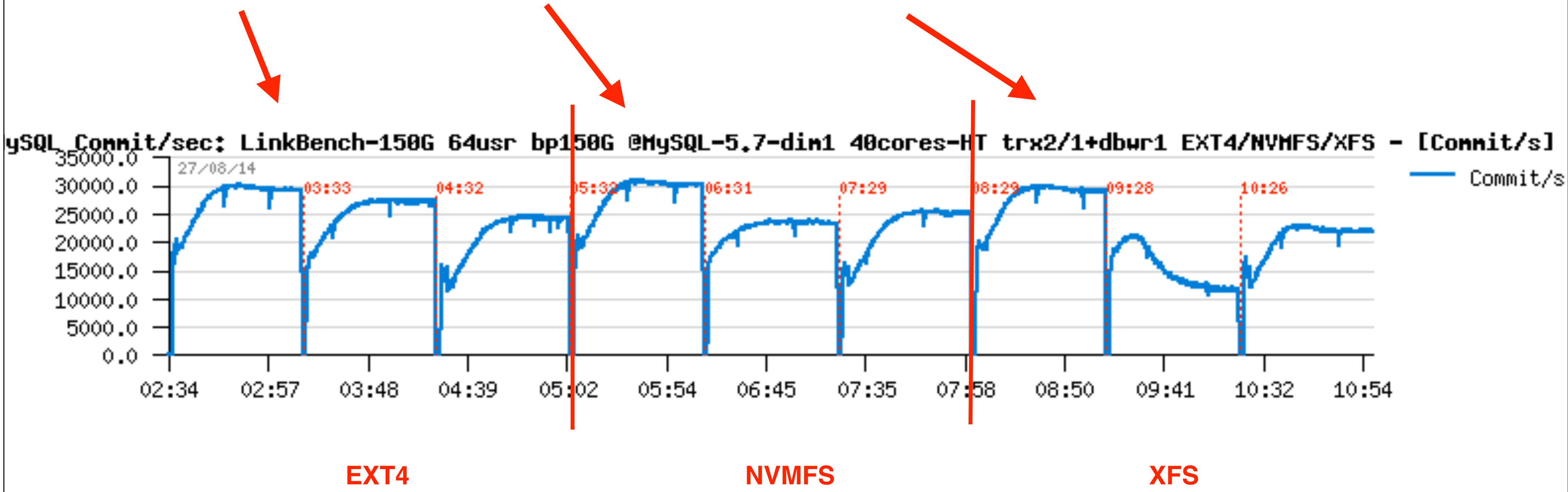
RW IO-bound "In-Memory"

- Impact of the database size
  - with a growing db size the TPS rate may be only the same or worse ;-)
  - and required Flushing rate may only increase..
- ex.: DBT2 workload :
  - 64 users, db volume: 50W, 100W, 250W, 500W

ORACLE

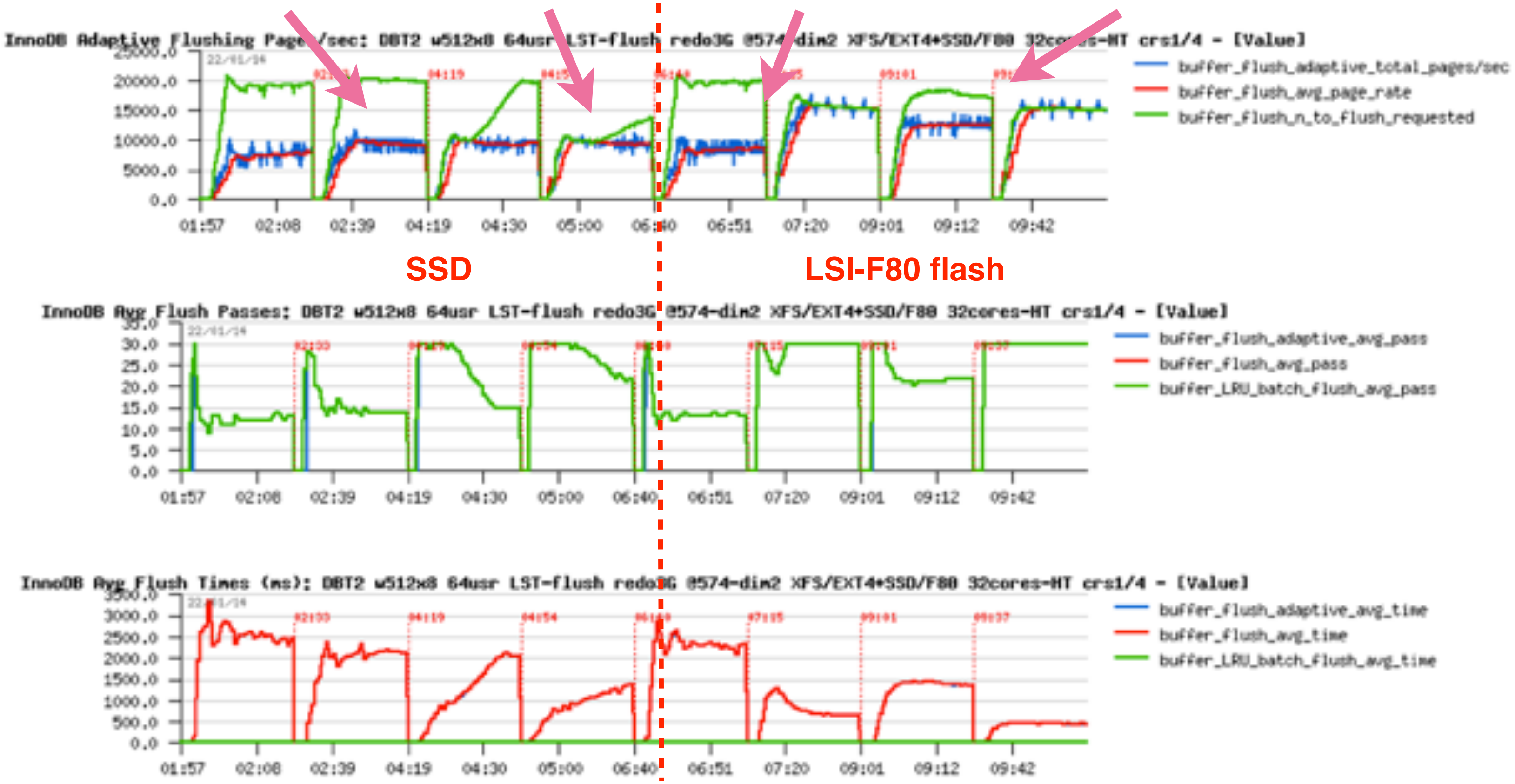
RW IO-Bound : Test your Filesystem before to deploy

- LinkBench 150G
  - safety options on 64usr, Fusion-io
  - EXT4 -vs- NVMFS -vs- XFS



RW IO-Bound : Consider a fast storage

- InnoDB Flushing in MySQL 5.7 & storage:
  - DBT2 512Wx8, 64usr, each test first with 1 then with 4 cleaners
  - XFS@SSD | EXT4@SSD | XFS@LSI-F80 | EXT4@LSI-F80



ORACLE

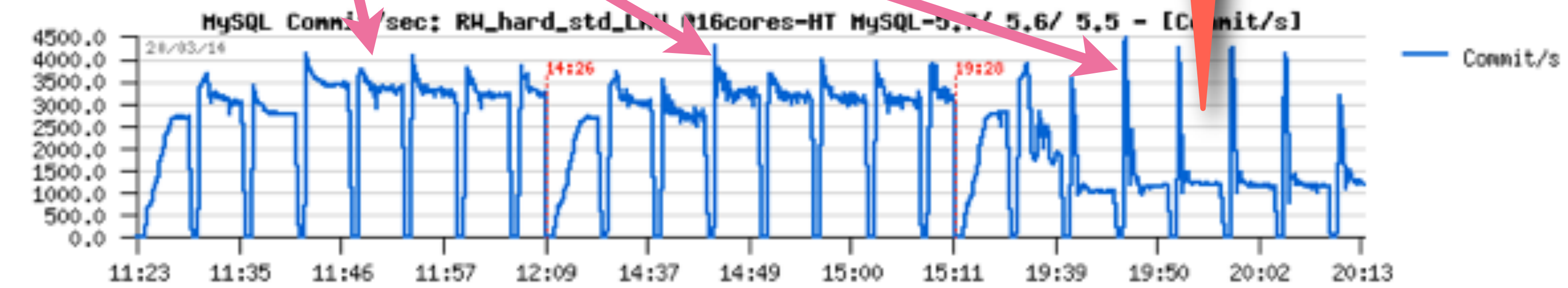
RW IO-bound "Out-of-Memory"

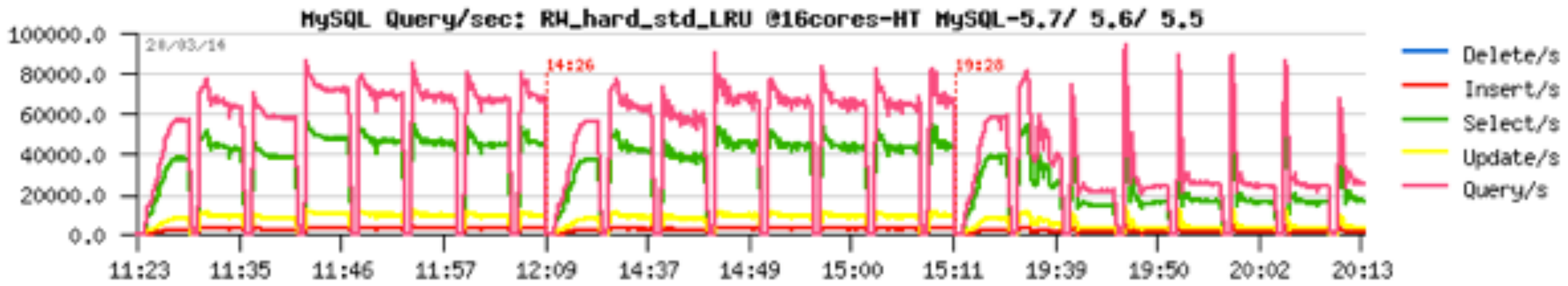
- The "entry" limit here is storage performance
  - as you'll have a lot of IO reads..
- Once storage is no more an issue :
  - you may hit internal contentions (ex. InnoDB file_sys mutex)
  - or other engine design limitations..
  - sometimes a more optimal config settings may help..
  - but sometimes not ;-)

RW LRU-bound : 5.5 is out of the game..

- Sysbench OLTP_RW 10M x32-tables
  - Users: 8, 16, 32 .. 1024
  - MySQL : 5.7 / 5.6 / 5.5

Please, upgrade me to 5.6 !!!

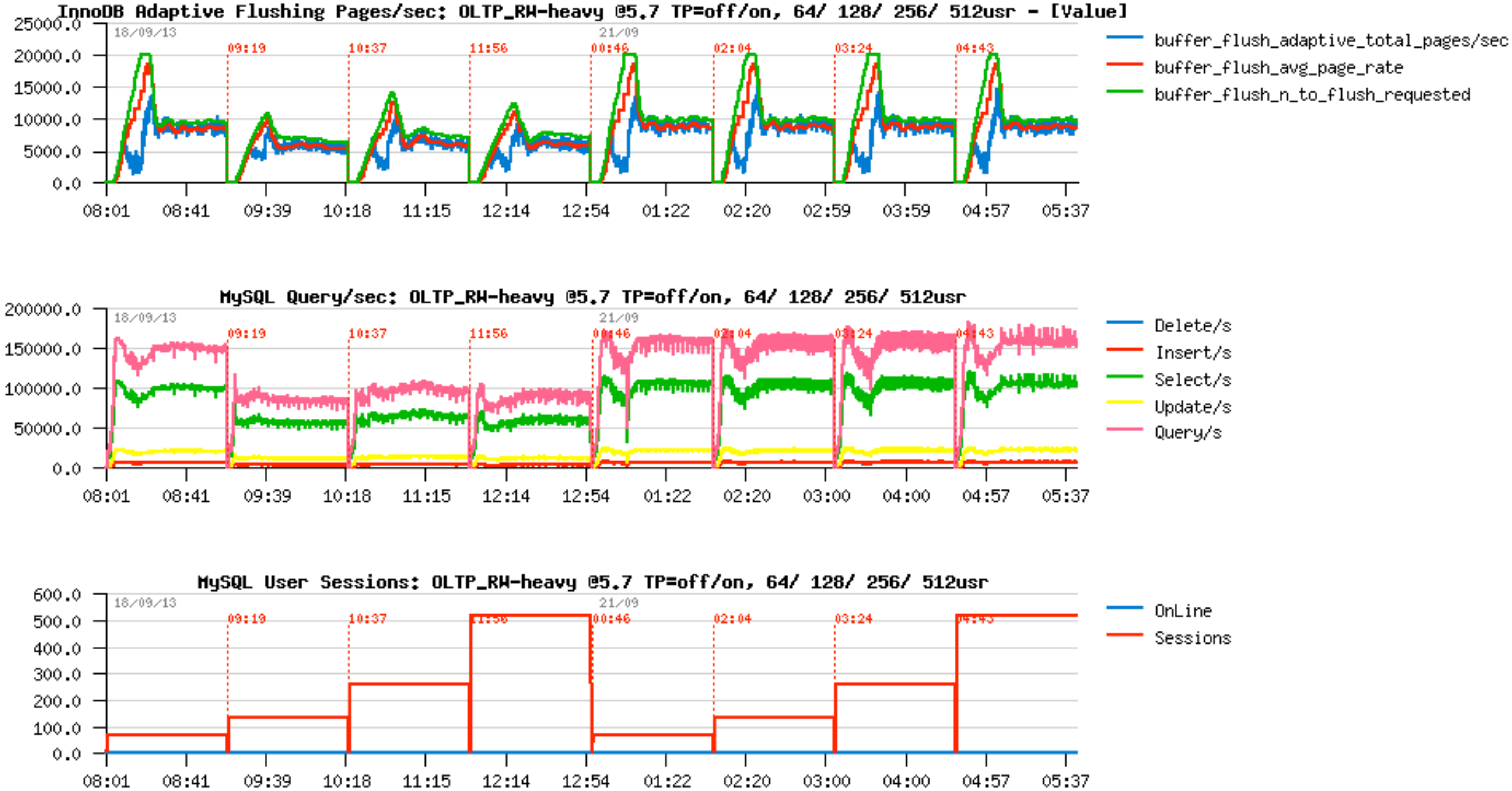




High Concurrency Tuning

- If bottleneck is due a concurrent access on the same data (due application design) – ask your 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 ;-)
  - tune InnoDB spin wait delay to improve your Max QPS (dynamic)
  - innodb_thread_concurrency=N to avoid QPS drop on usr++ (dynamic)
  - CPU taskset / prcset (Linux / Solaris, both dynamic)
  - Thread Pool
  - NOTE:
    - things with contentions may radically change since 5.7, so stay tuned ;-)
    - InnoDB thread concurrency feature was **improved** in 5.6 and 5.7
    - the best working in 5.7, and using innodb_thread_concurrency=64 by default now makes sense..

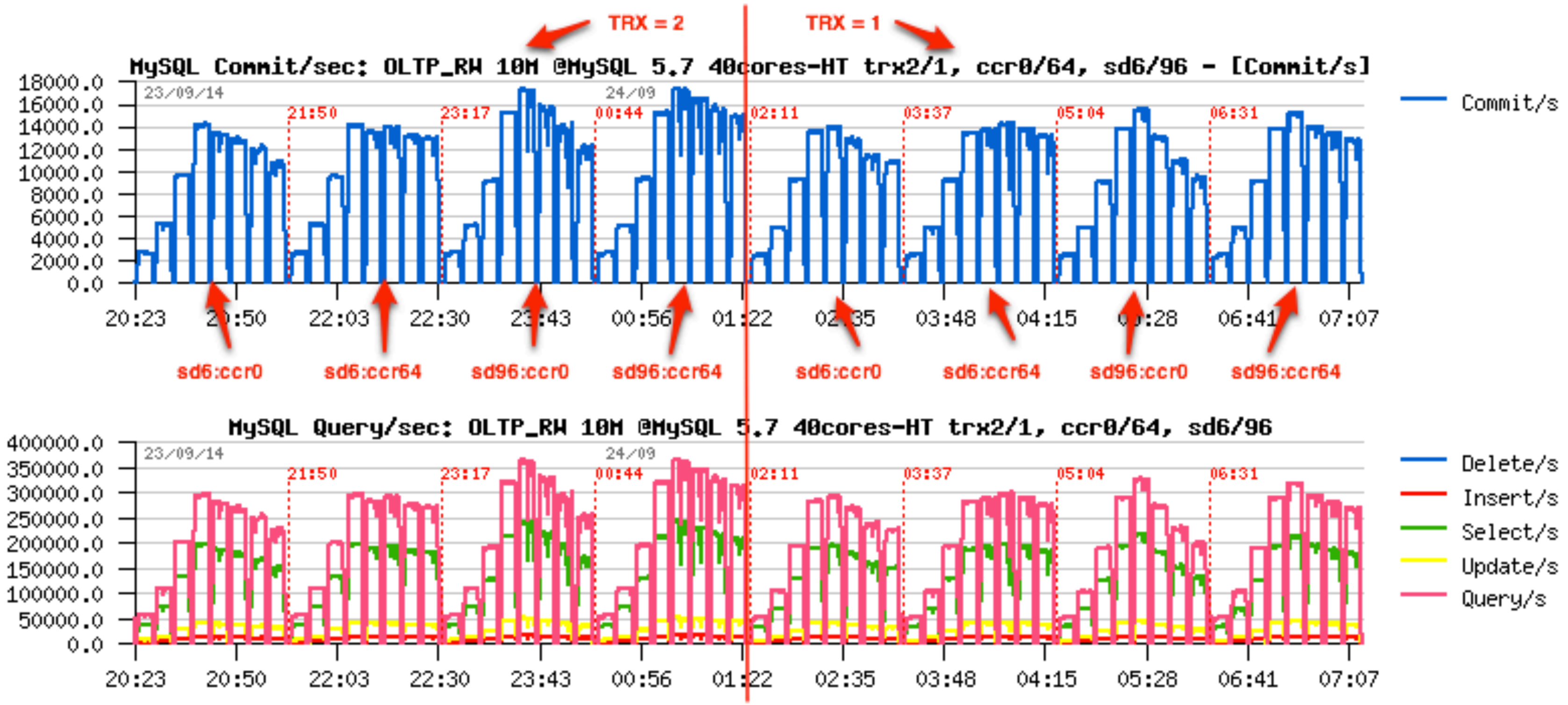
Thread Pool in old MySQL 5.7 @Heavy OLTP_RW



ORACLE

Concurrency tuning on OLTP_RW @MySQL 5.7

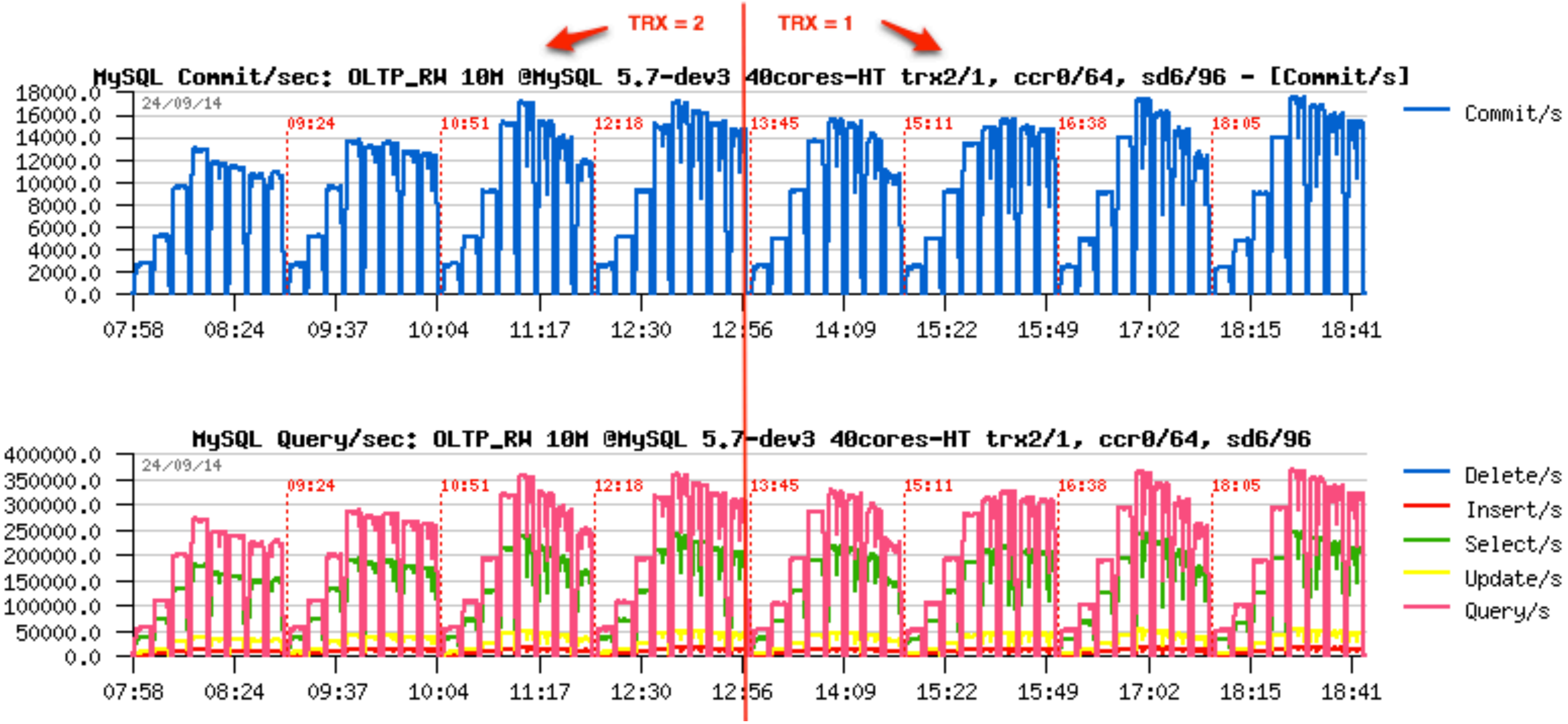
- OLTP_RW 10M @MySQL 5.7 40cores-HT :
  - load conditions: TRX = 2 -vs- TRX = 1
  - cooking receipt : concurrency (ccr) & spin wait delay (sd)



ORACLE

Concurrency tuning on OLTP_RW @MySQL 5.7-dev3

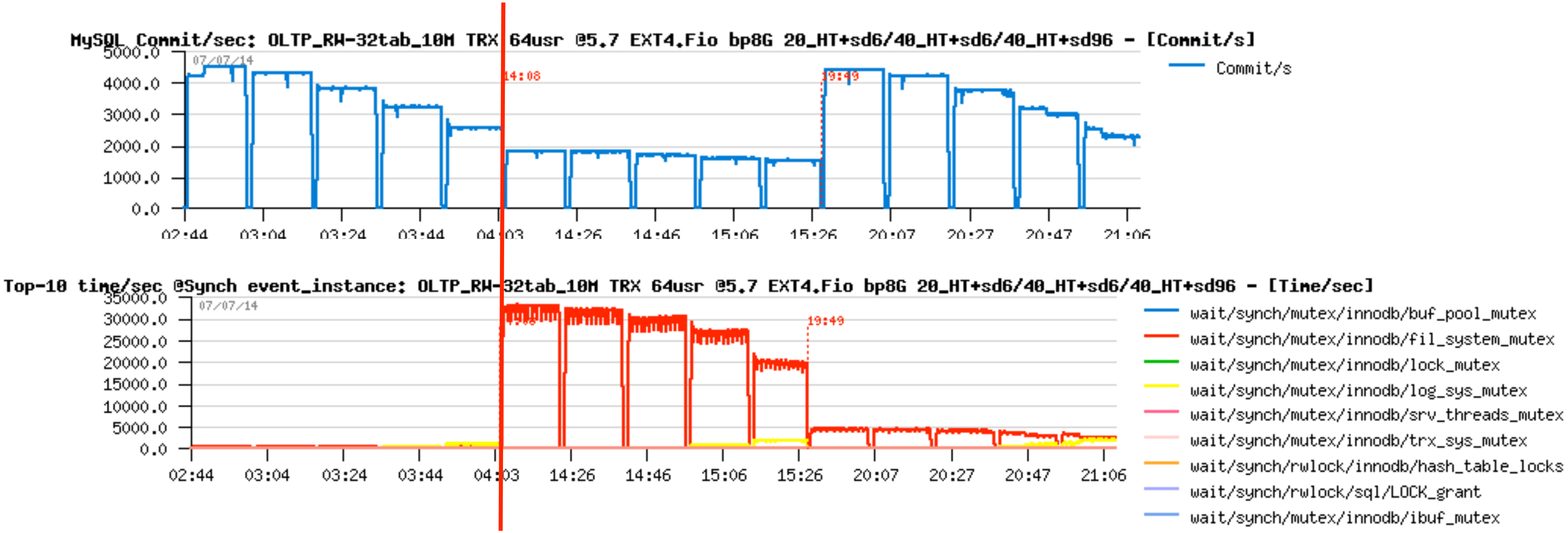
- OLTP_RW 10M @MySQL 5.7-dev3 40cores-HT :
  - load conditions: TRX = 2 -vs- TRX = 1
  - cooking receipt : concurrency (ccr) & spin wait delay (sd)

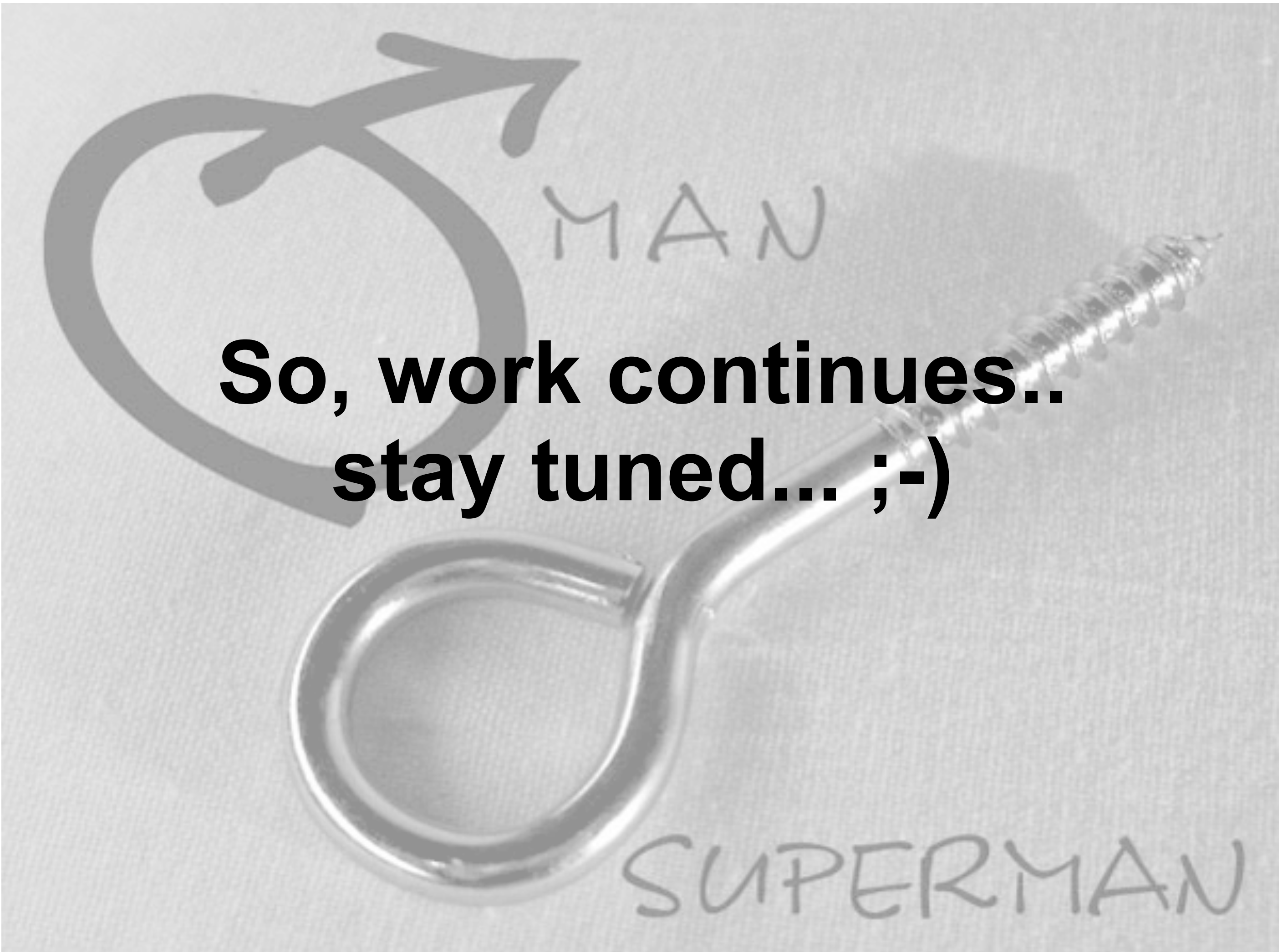


ORACLE

Impact of “Write” queries on MySQL Performance

- IO Bound “out-of-memory”
  - 64usr, growing R/W ratio, near x2 times TPS drop at the end..
  - 20cores | 40cores | 40cores & spin delay = 96





So, work continues..
stay tuned... ;-)

Few words about dim_STAT (if you're asking ;-))

- 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! :-)

ORACLE

THANK YOU !!!

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