



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

Why Benchmarking MySQL ?...

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..



www.freeuniverse4all.com

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 ;-)



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!

- **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!

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?..

ORACLE MySQL Enterprise Monitor

22 22 0 248 0 admin Refresh: Off

Dashboards Events Query Analyzer Reports & Graphs Configuration

Group Overview: All

Database Statistics

Database Availability

Day 100%
Week 100%
Month 100%

Connections - All MySQL Instances

Database Activity - All MySQL Instances

Query Response Time Index

Current Problem MySQL Instances

ID	Status	Emergency	Critical	Warning
bur05:33030	Up	0	2	11
tyr55:33300	Up	0	2	13
tyr58:3399	Up	0	1	17
tyr52:33030	Up	0	1	12

Showing 1 to 4 of 4 entries

Current Problem Hosts

ID	Status	Emergency	Critical	Warning
bur05	Up	0	1	0

Showing 1 to 1 of 1 entries

Emergency & Critical Events

Show 5 entries

	Subject	Topic	Time	Actions
<input type="checkbox"/>	bur05, MEM Built-in Agent	Agent CPU Usage Excessi...	about a minute ago	✘
<input type="checkbox"/>	bur05, bur05:33030	Table Cache Not Optimal	about a minute ago	✘
<input type="checkbox"/>	tyr52, tyr52:33030	Table Cache Not Optimal	2 minutes ago	✘
<input type="checkbox"/>	bur05, bur05:33030	Attempted Connections T...	3 minutes ago	✘
<input type="checkbox"/>	tyr58, tyr58:3399	Table Cache Not Optimal	4 minutes ago	✘

Showing 1 to 5 of 7 entries

Copyright © 2005, 2013, Oracle and/or its affiliates. All rights reserved. 3.0.2.7154 - bur05 (10.172.161.65) - Sep 16, 2013 1:38:02 pm (Up Since: 1 day, 18 hours ago) - About



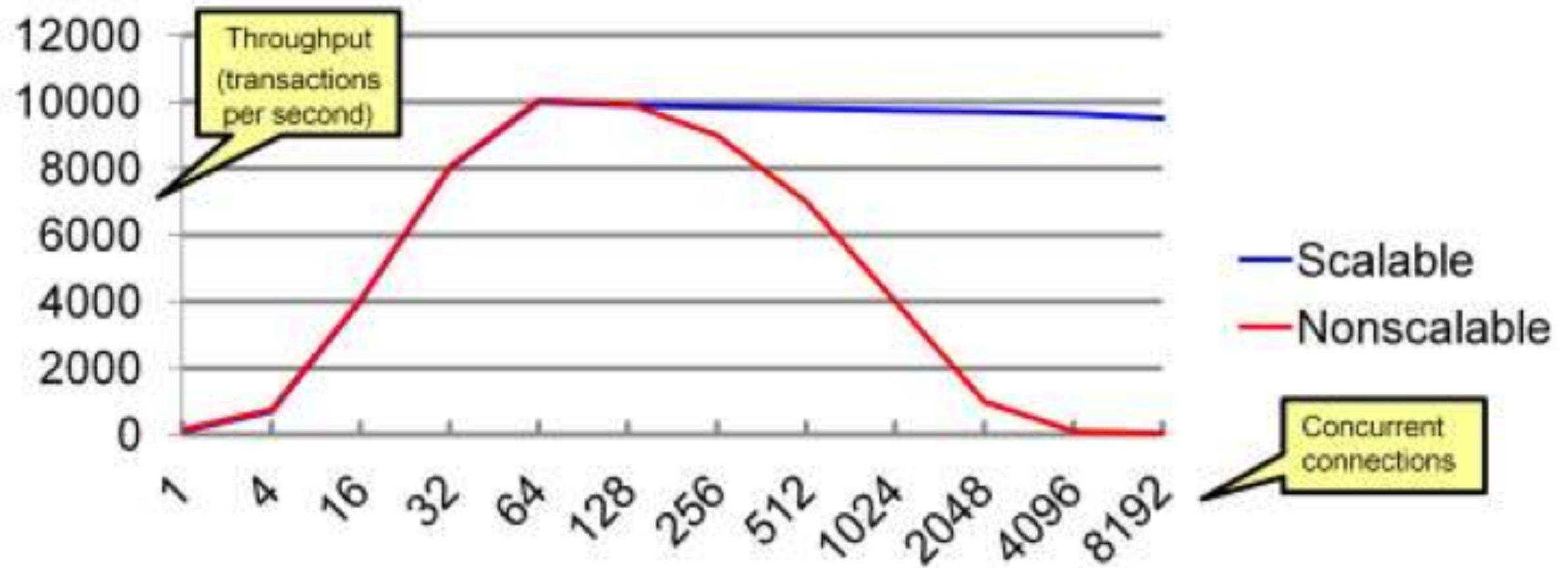
Other Monitoring Tools

- 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..



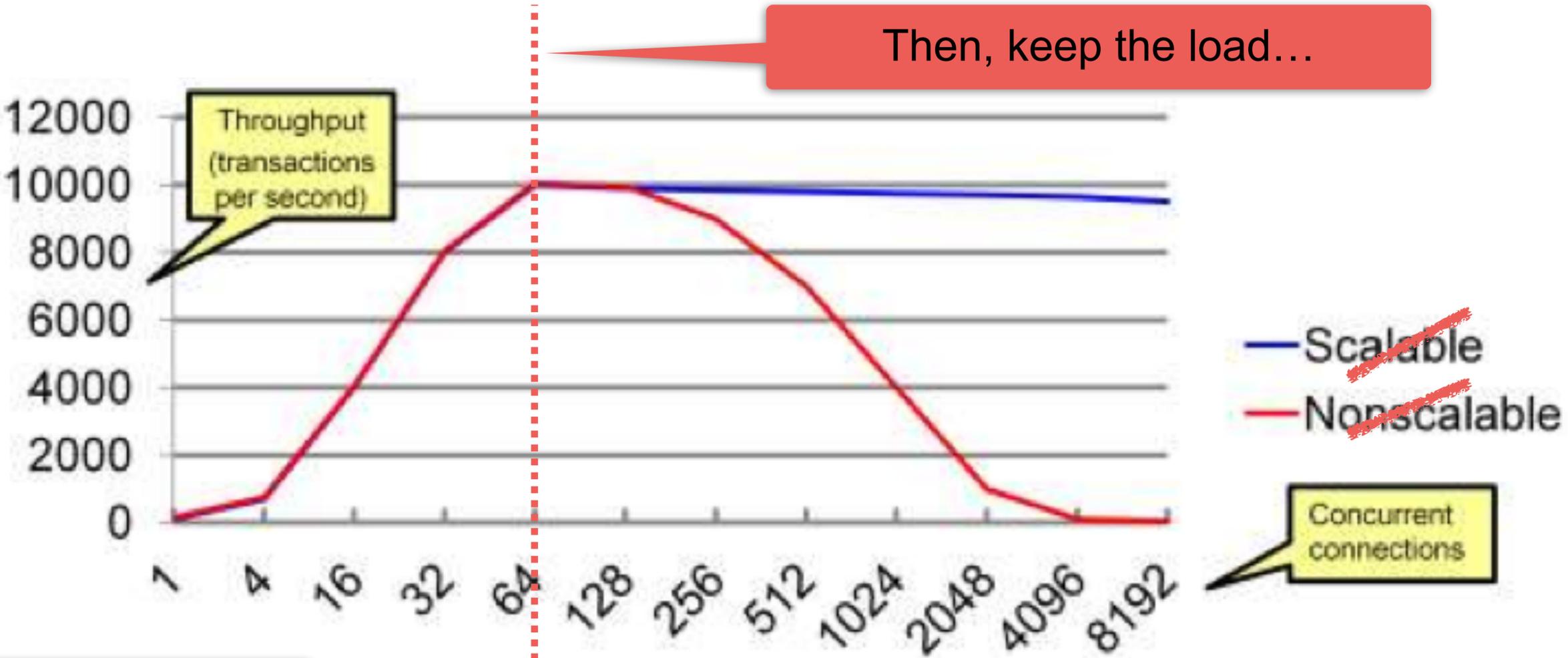
A B-shit Slide...

- Odd interpretation of Scalability...



A B-shit Slide... (2)

- Odd interpretation of Scalability...



Scale up to N connections

Both are scaling up to 64 connections, but only one is able to keep a higher load..

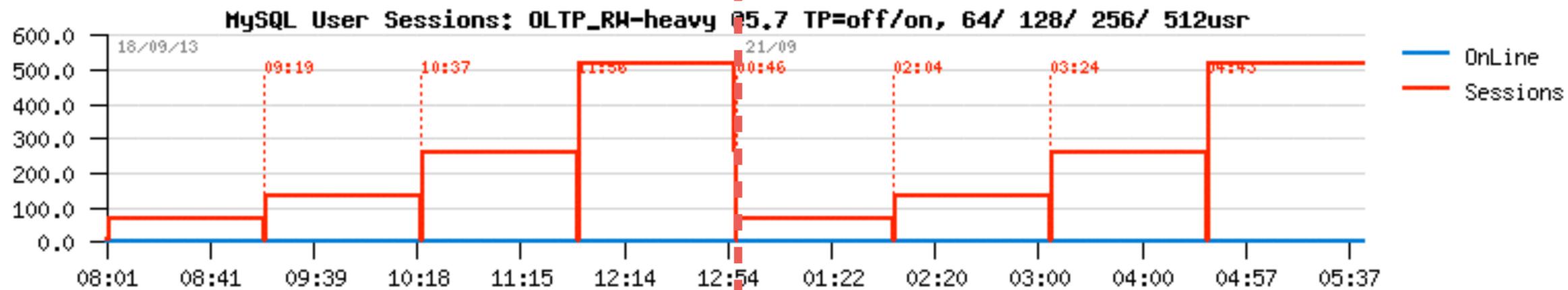
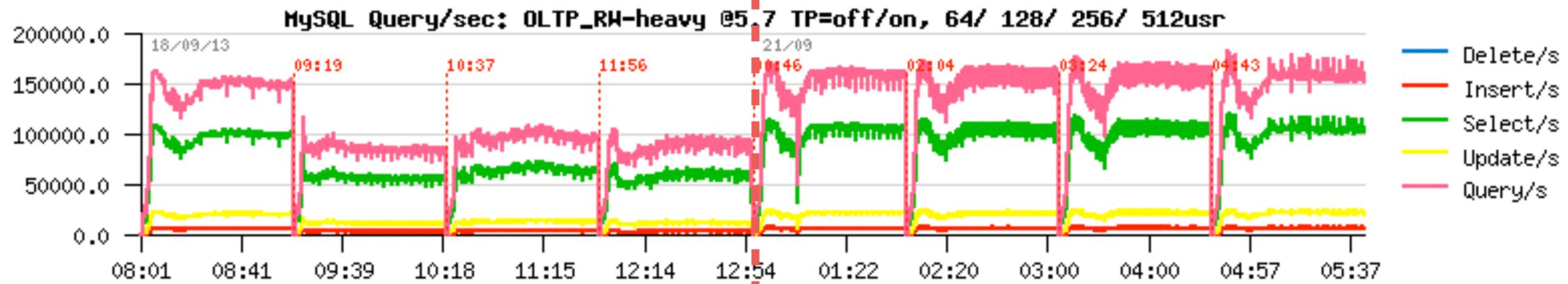
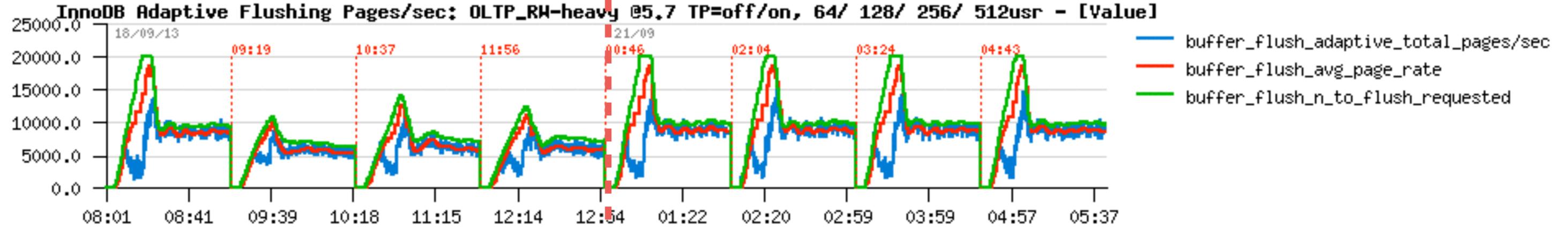
MySQL Scalability Limits...

- **Before to suspect your database engine is not scaling, be sure :**
 - you're not rather hitting your storage I/O limits (test before you say "NO" ;-))
 - network limits? (test!)
 - do you see your CPU 100% busy?
 - a poor application?
 - NOTE: a single poorly written SQL query may show you that your database server is scaling only up to 1 user ;-))
- **How long I'm scaling on a growing load?**
 - you're continuing to scale as long as your TPS continue to grow with a growing load..
 - however, it's another story if you can consider it as "good", "not too bad" or "poor" ;-)
 - ideal scalability: your response time remains the same from 1 to N **concurrent** users
 - NOTE: never forget what is your main target!
 - e.g. if a response time on a single user is already higher than "acceptable", don't expect it becomes better with a higher number of concurrent users ;-))

MySQL on High Load

- Once you've reached your Max TPS on your system :
 - try to understand first what is limiting you? (I/O, CPU, Network, MySQL internals?)
 - the next goal then: to avoid a TPS "regression" on a higher load
- How to keep your Max TPS on a higher load too?
 - the dumb rule : avoid to have a higher load! ;-)
 - seriously :
 - usually all you need is to find a way to do not let you workload concurrency out-pass the levels your reaching on the TPS Max, that's all..
 - InnoDB thread concurrency helps here (yet more improved in MySQL 5.7)
 - InnoDB spin wait delay tuning helps to lower mutexes / rw-locks waits impact
 - ThreadPool
 - **NOTE : there is no "magic" for response time :**
 - if your Max TPS you're reaching on N users
 - and able to keep the same Max TPS on N x2 users (or x3, x4, etc.)
 - your response time may only grow! (and be x2 times bigger (or x3, or x4, etc.))

Thread Pool in old MySQL 5.7 @Heavy OLTP_RW



MySQL & CPU Usage

- CPU chips progress:
 - CPU = 1 CPU (1 vcpu)
 - CPU = N cores (N vcpu)
 - CPU = N cores, M core threads (NxP vcpu)
 - ...
- How many **really** parallel tasks your CPU is able to execute??
 - as many as how many vcpu are **really** able to run in parallel!
 - ex. 32cores-HT :
 - only 32 concurrent MySQL threads may be executed on the same time
 - is HT helping? - yes
 - is HT makes 32cores be equal to 64cores? - no
 - if my system is reporting to have CPU 50% busy on my MySQL workload, does it mean I have a 50% marge in CPU usage? — **NO!**.. ;-)
 - my workload is pure CPU bound, I'm reaching N TPS on 64 users and I'm claiming I'm getting x5 higher (Nx5) TPS on 512 users! — well, you're lying somewhere ;-))

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..” ;-))

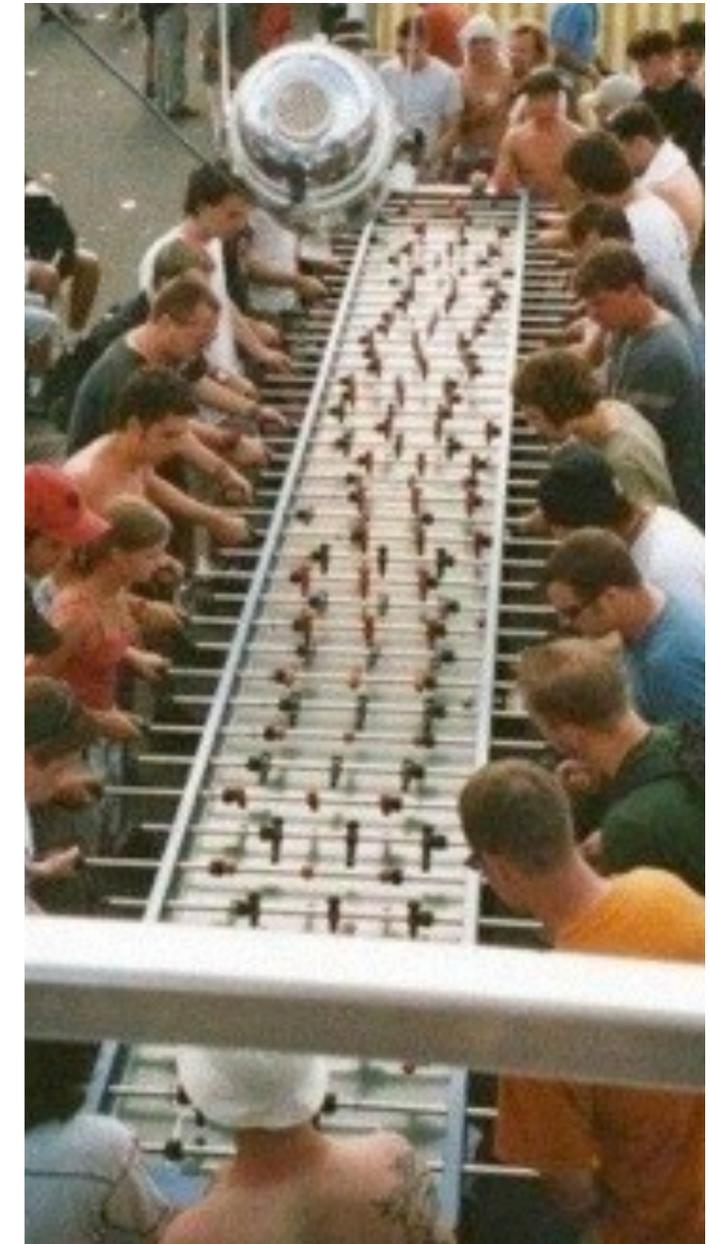


“Generic” Test Workloads @MySQL

- **Sysbench**
 - OLTP, RO/RW, N-tables, lots test workload load options, deadlocks
- **DBT2 / TPC-C-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 ;-)

Be sure you can trust your Benchmark results ;-)

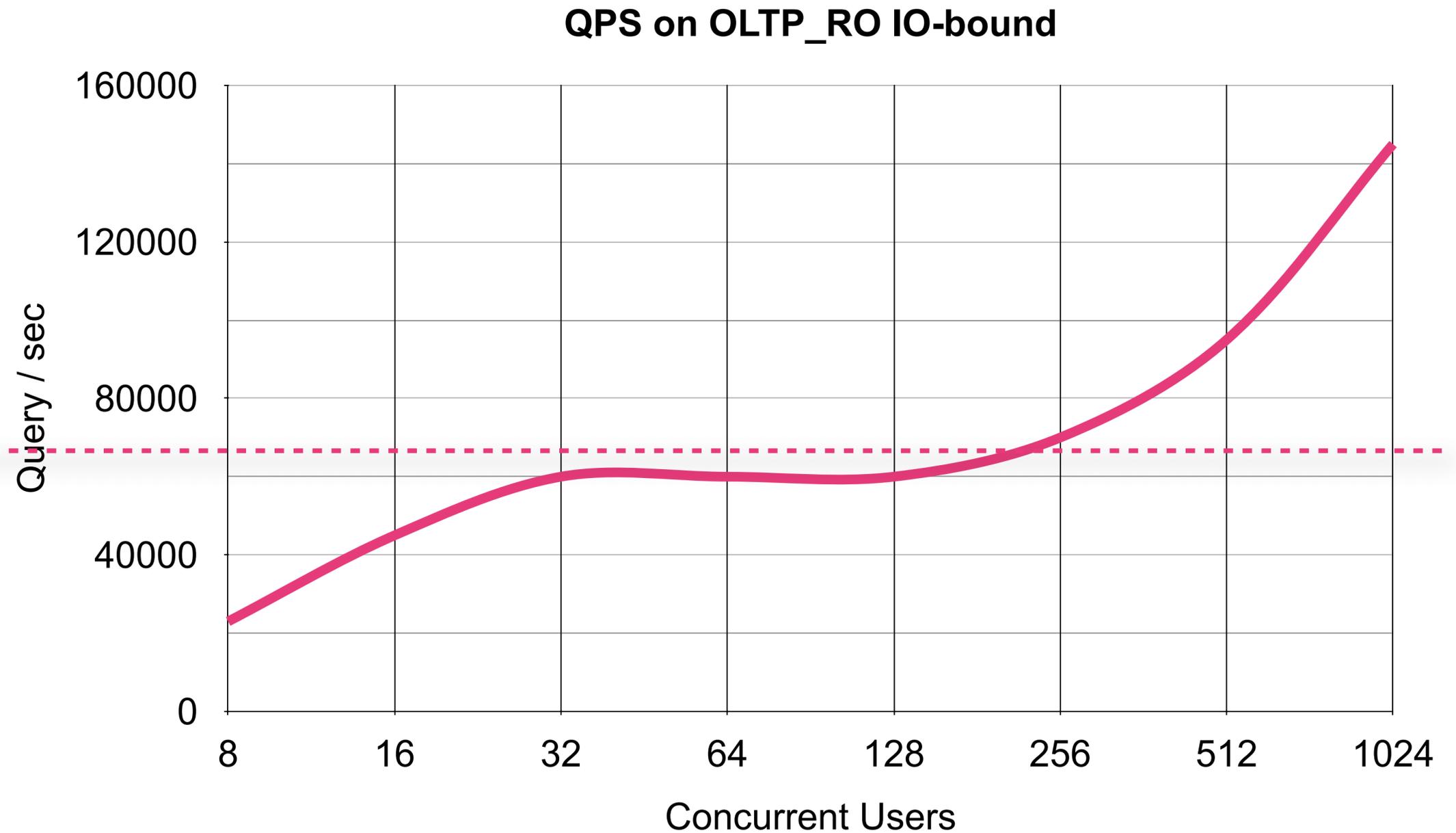
- Know your HW platform **limits**
- Understand what your **Workload** is doing
- Keep in mind MySQL Server **internals**
- Be sure your test case has no self-limits either!!!
- Once again : There is **NO** “Silver Bullet” !!!
 - Think about the #1 MySQL Performance Best Practice ;-))



Let's analyze the following benchmark result..

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

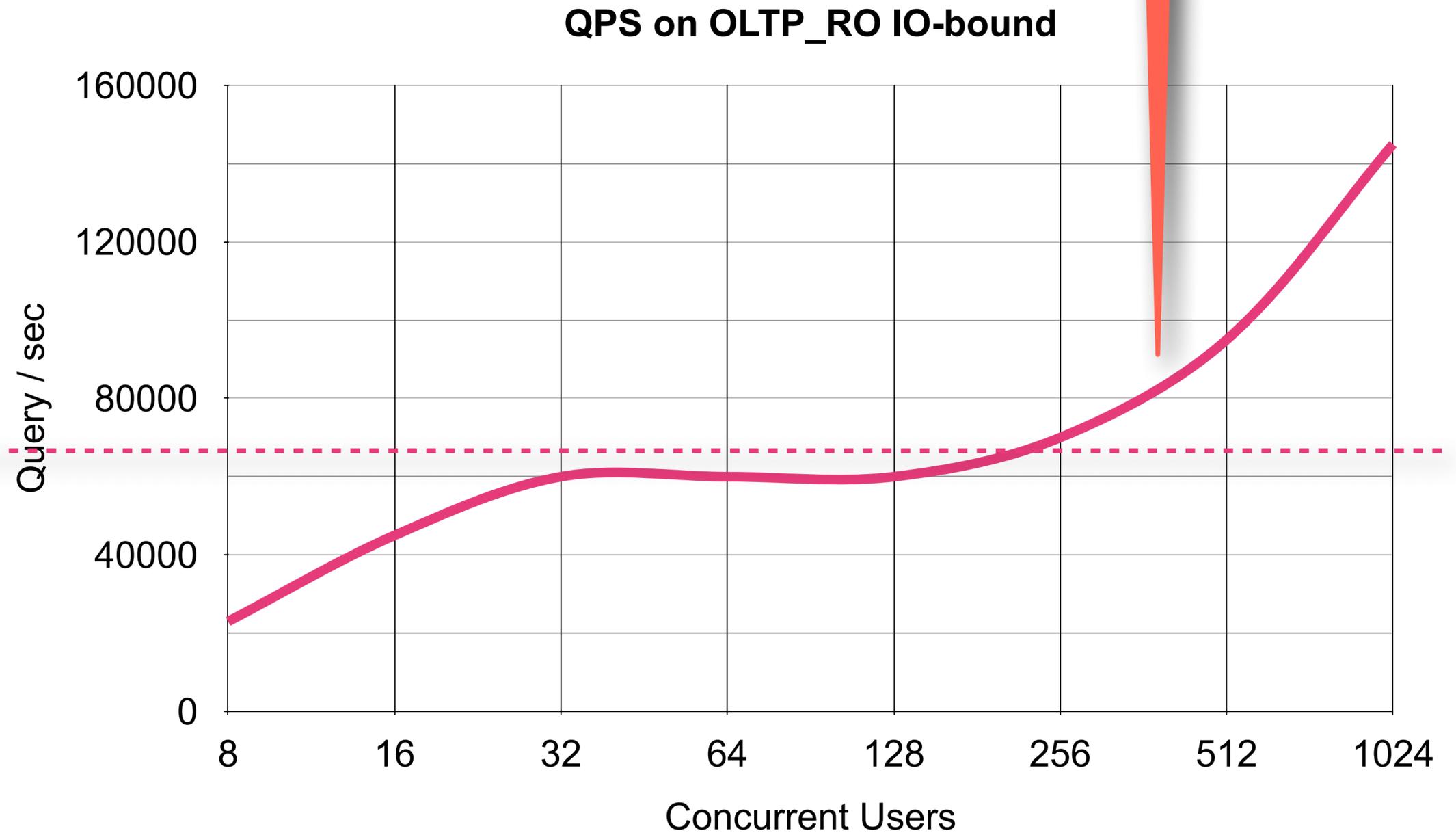
I/O limit



Let's analyze the following benchmark result..

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

I/O limit



Cached !!!

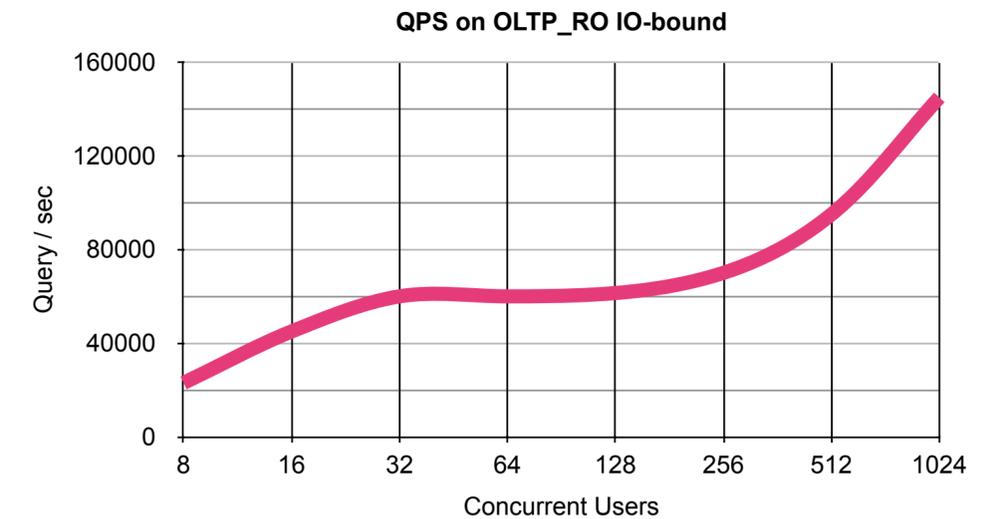
Let's analyze the following benchmark result..

- **Test : fully random IO-bound OLTP_RO**

- Storage limit : 60K reads/sec max
- 150K QPS ??
- WTF?... ;-)

- **The issue:**

- the random ID for a row access 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)..



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 !
- **Note: RW performance cannot be good if your RO is poor! ;-)**

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 64 CPU cores for sure, reported on more cores too..
 - Note : remind my “scalability” notes ;-))
 - 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

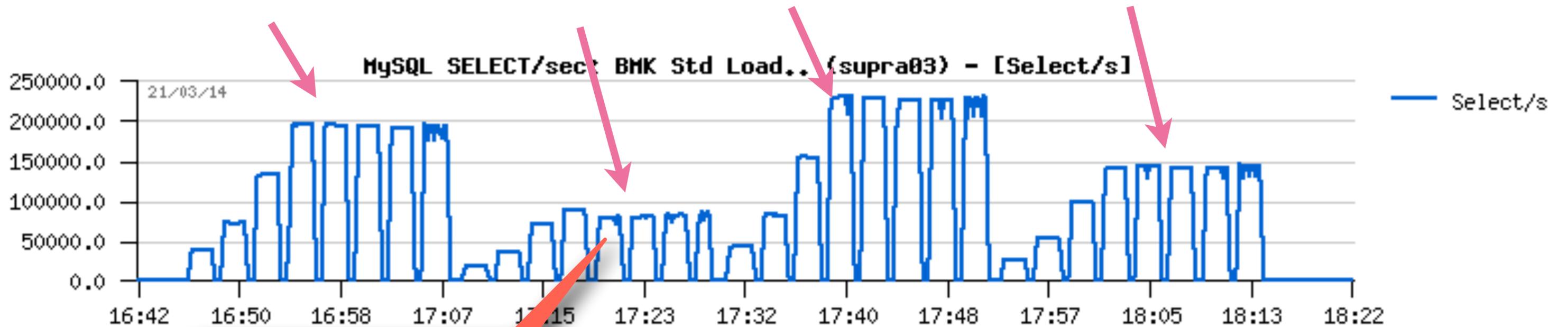
innodb_stats_persistent = 1
innodb_adaptive_hash_index= 0 / 1
innodb_monitor_enable = '%'
```

Sysbench OLTP_RO Workloads

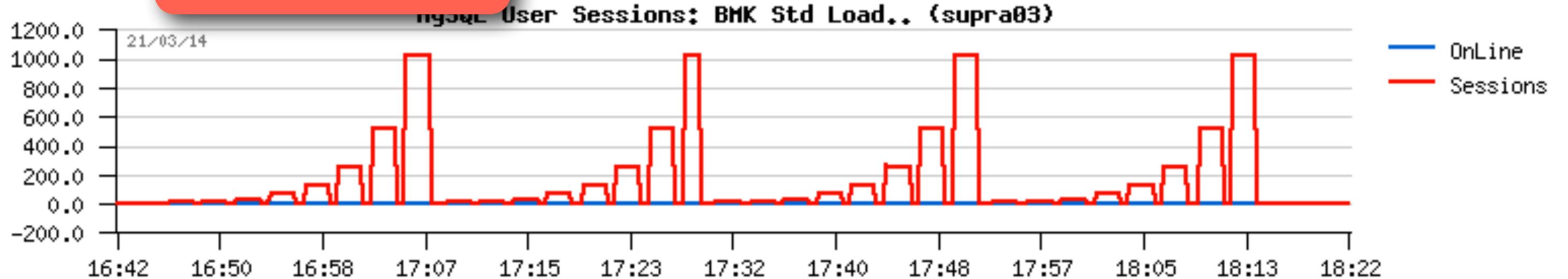
- **Available Test Workloads :**
 - **Point-Select** : a row read by PK id (most aggressive workload, extremely fast queries)
 - **Simple-Ranges** : read N rows via PK range (hot on memcpy() and hash)
 - **Order-Ranges** : as Simple-Ranges, but ordered by non-indexed column (hot on the same)
 - **SUM-Ranges** : read SUM value from N rows in PK range (hot on the same)
 - **Distinct-Ranges** : as Order-Ranges, but DISTINCT values from non-indexed column (extremely hot on in-memory temp tables create/drop)..
- **OLTP_RO :**
 - composed of :
 - x10 Point-Selects
 - x1 Simple-Range, N=100
 - x1 Order-Range, N=100
 - x1 SUM-Range, N=100
 - x1 Distinct-Range, N=100
 - most impacted by: memcpy() (kidding ;-) — ALL test cases are important!

Sysbench OLTP_RO Workloads @MySQL 5.7 - Apr.2014

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

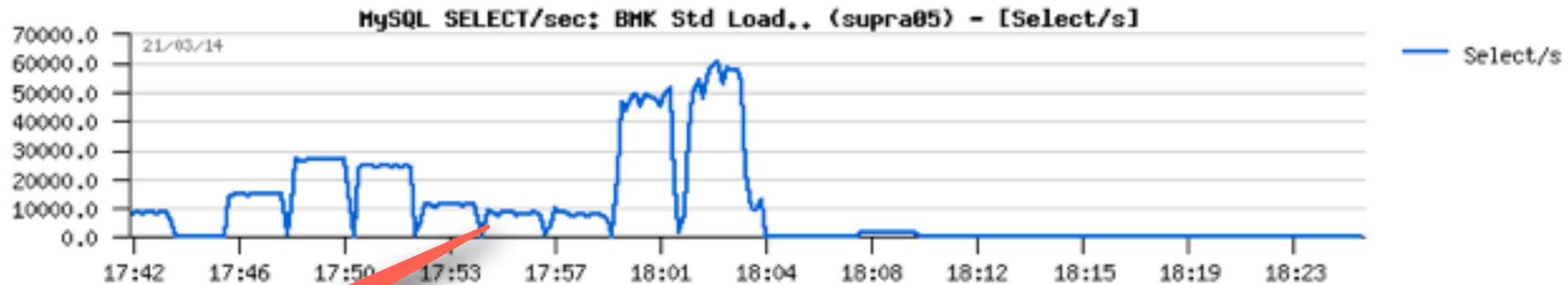


Was not ok...

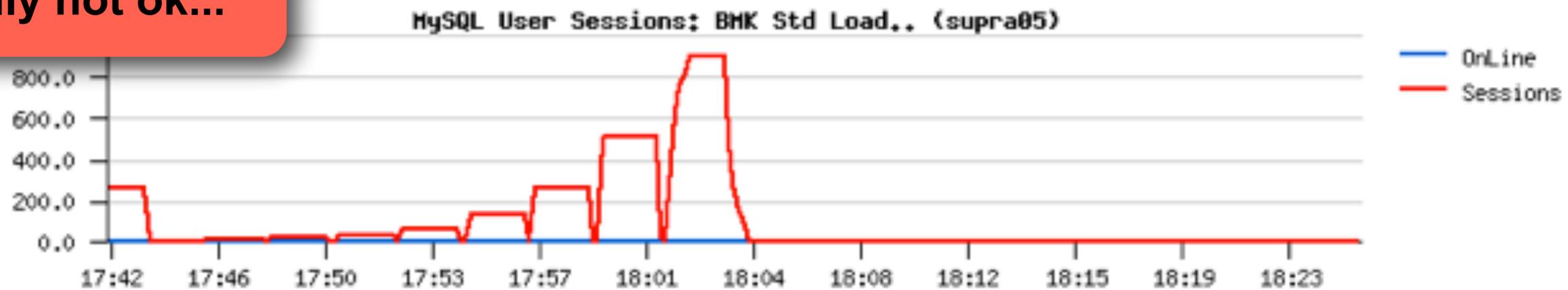


Story #1 : mysterious kernel contention

- Sysbench RO Distinct Selects - Apr.2014
 - 40cores-HT server



Really not ok...



Story #1 : mysterious kernel contention (2)

- Sysbench RO Distinct Selects - Apr.2014
 - 40cores-HT server
- Profiler:

Really not ok...

```
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
...
```

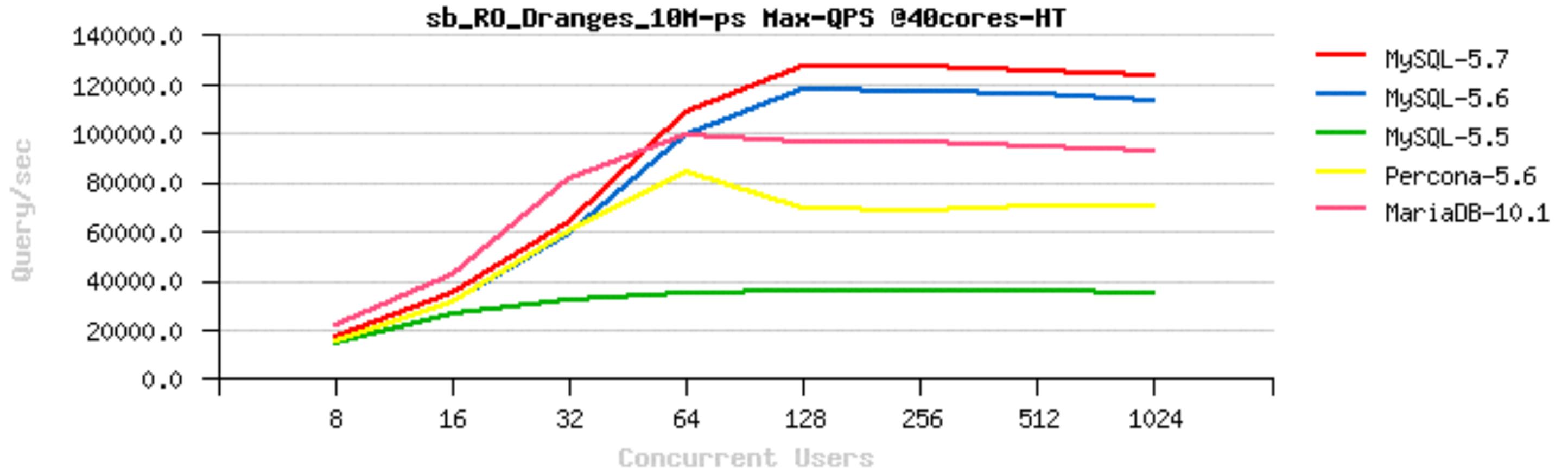

Story #1 : mysterious kernel contention (4)

- Sysbench RO Distinct Selects - Apr.2014
 - 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
```

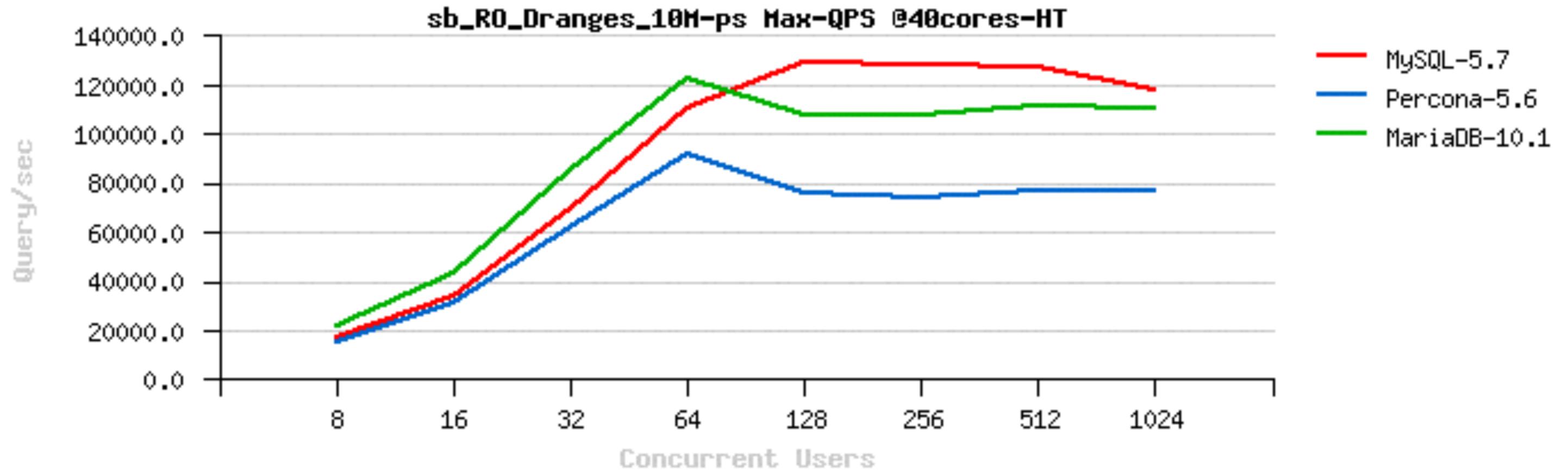
OLTP_RO Distinct Selects with “fixed” jemalloc - Oct.2014

- Max QPS @40cores-HT :



OLTP_RO Distinct Selects with “fixed” jemalloc - Apr.2015

- Max QPS @40cores-HT :

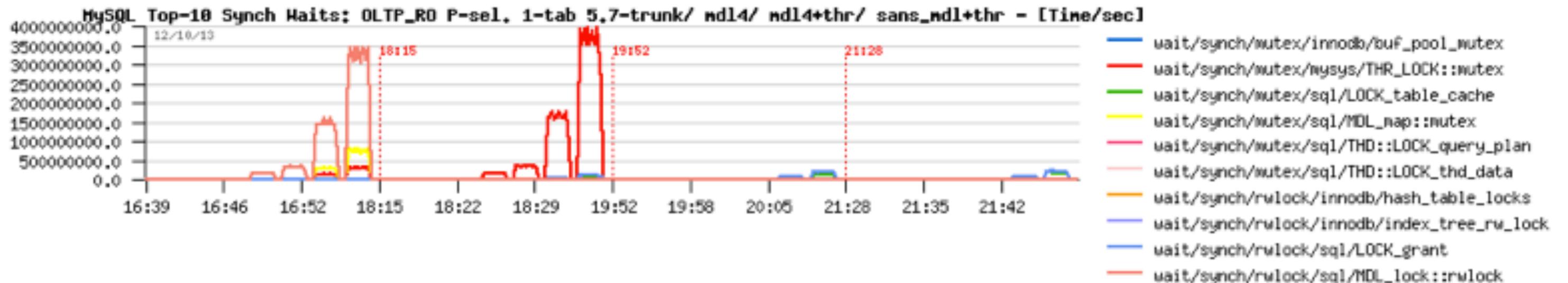
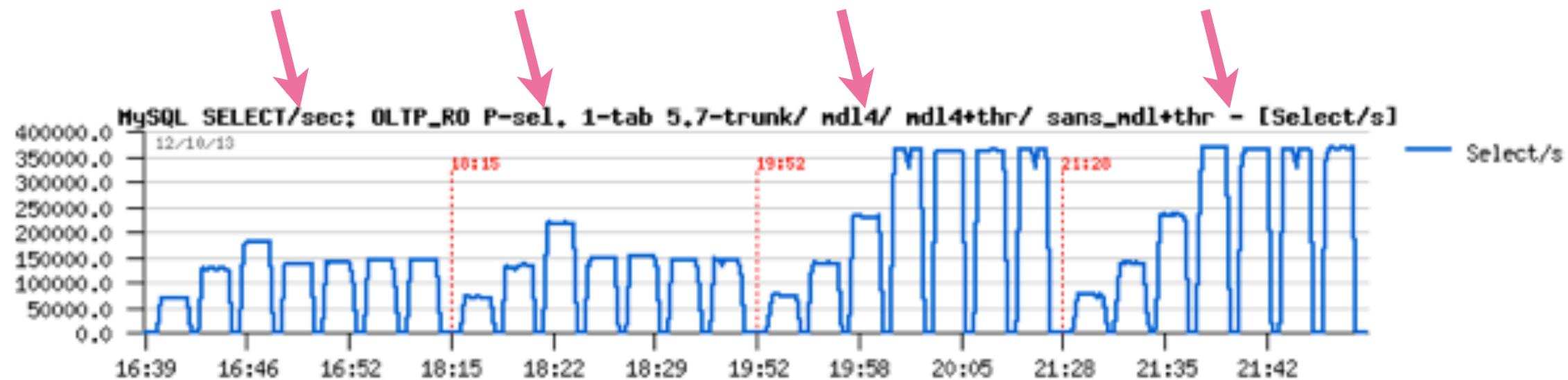


Story #2 : contentions around a hot table

- Once again a game of contentions :
 - improved TRX list ==> more hot MDL..
 - more hot MDL ==> regression on all single-table workloads..
 - Note: on this stage 5.7 became even slower than 5.6 ;-)
 - 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!
 - next-level: LOCK_grant, LOCK_plugin ==> **fixed!!!**
 - MySQL 5.7 today : the only internal lock contentions you could see now are from InnoDB only (normally).. ;-)

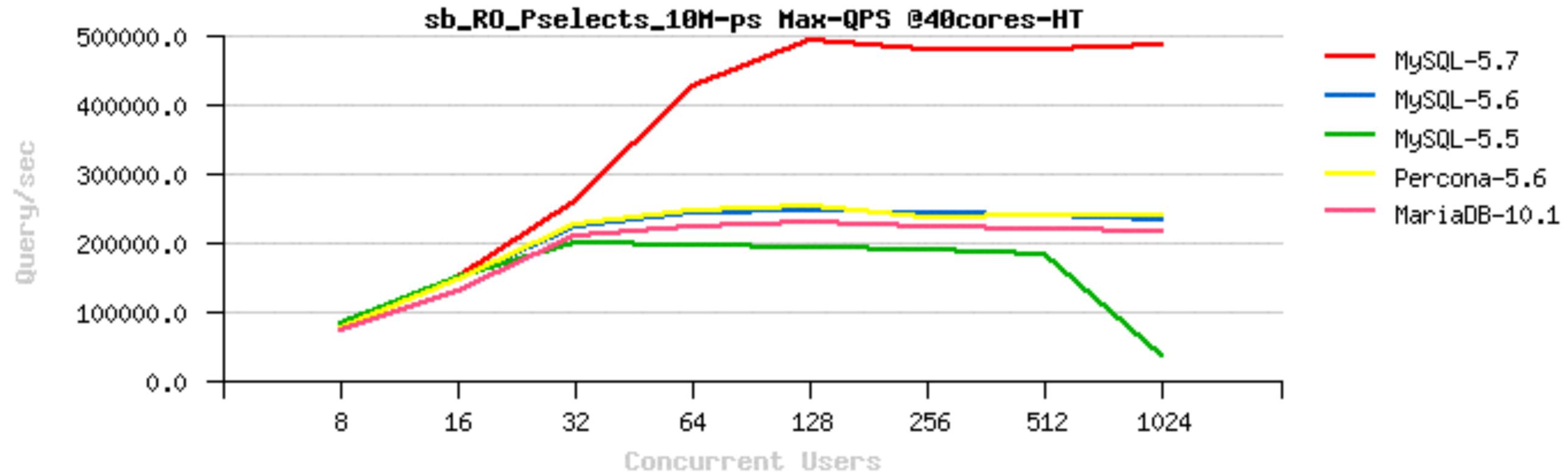
Story #2 : contentions around a hot table (2)

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



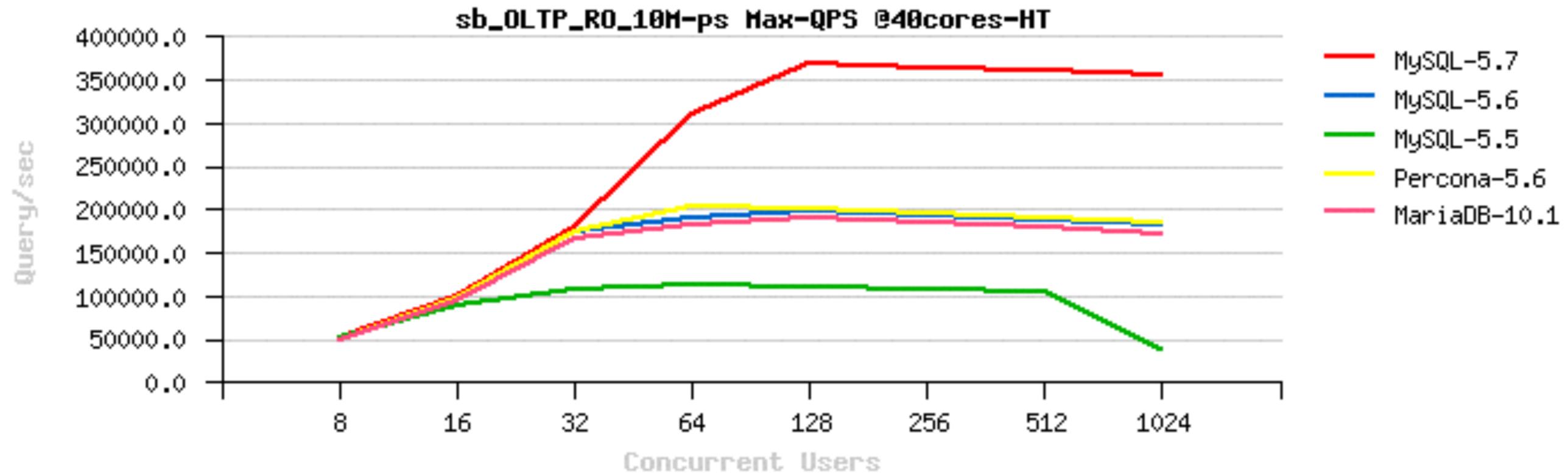
Sysbench OLTP_RO Point-Selects single-table

- Max QPS @40cores-HT :



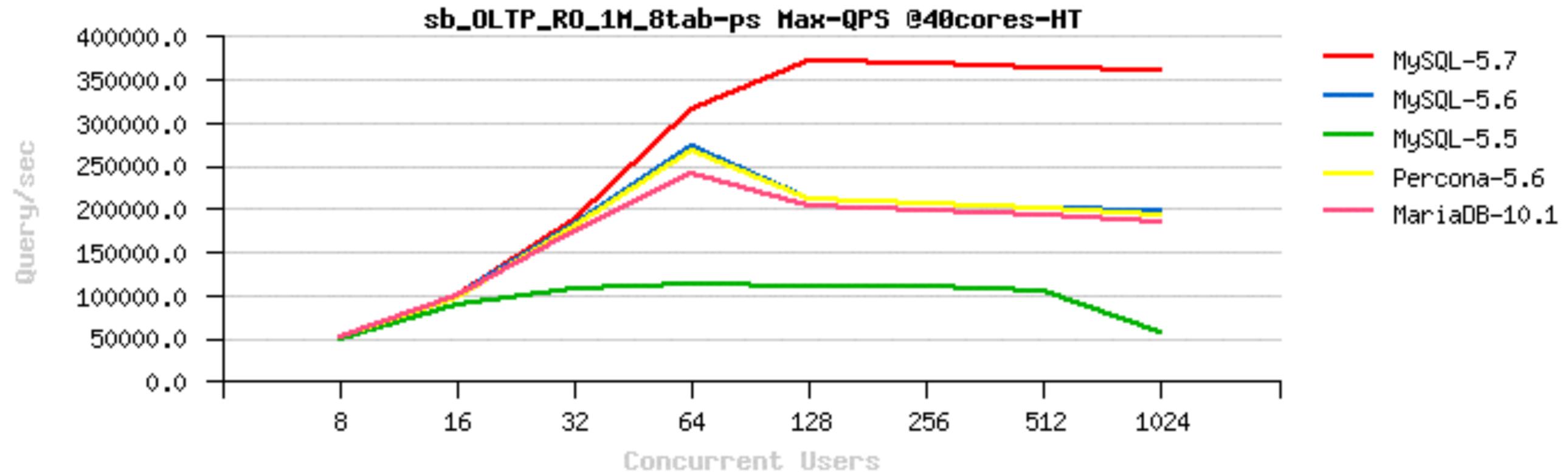
Sysbench OLTP_RO Single-table

- Max QPS @40cores-HT :



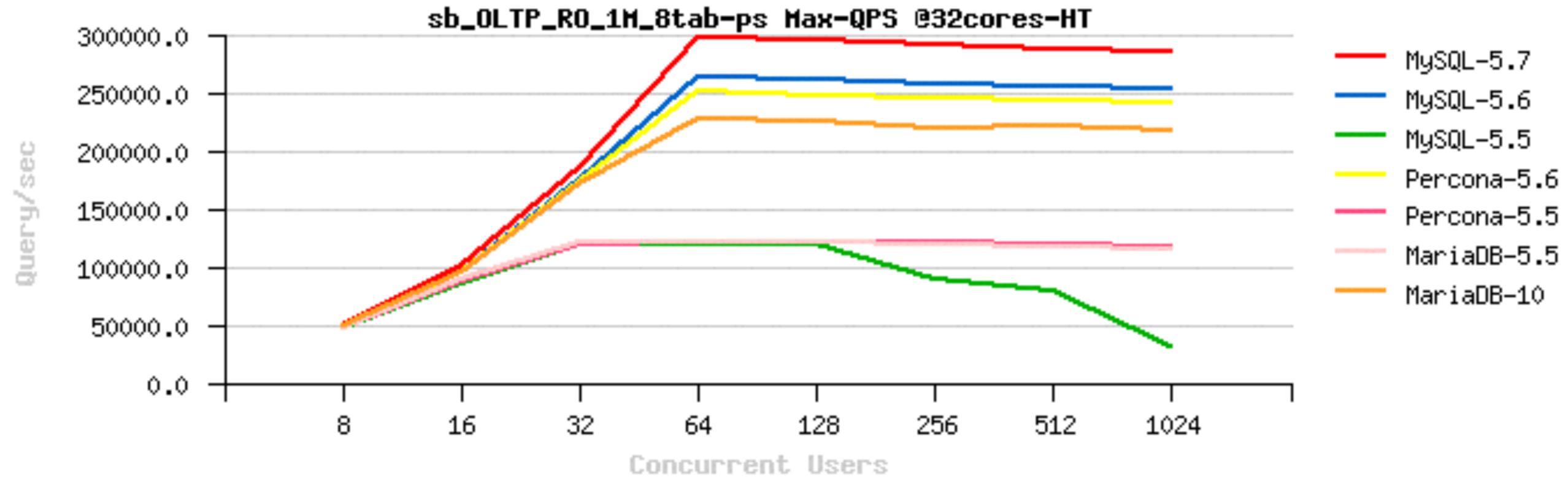
RO In-Memory @MySQL 5.7

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



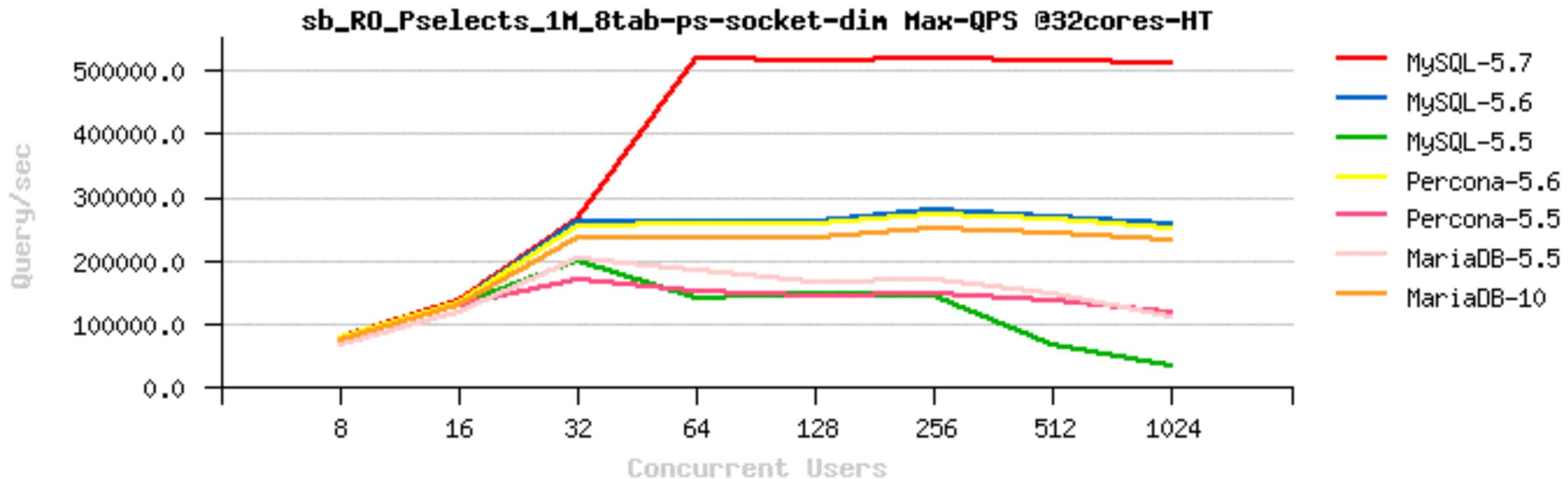
RO In-Memory @MySQL 5.7

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



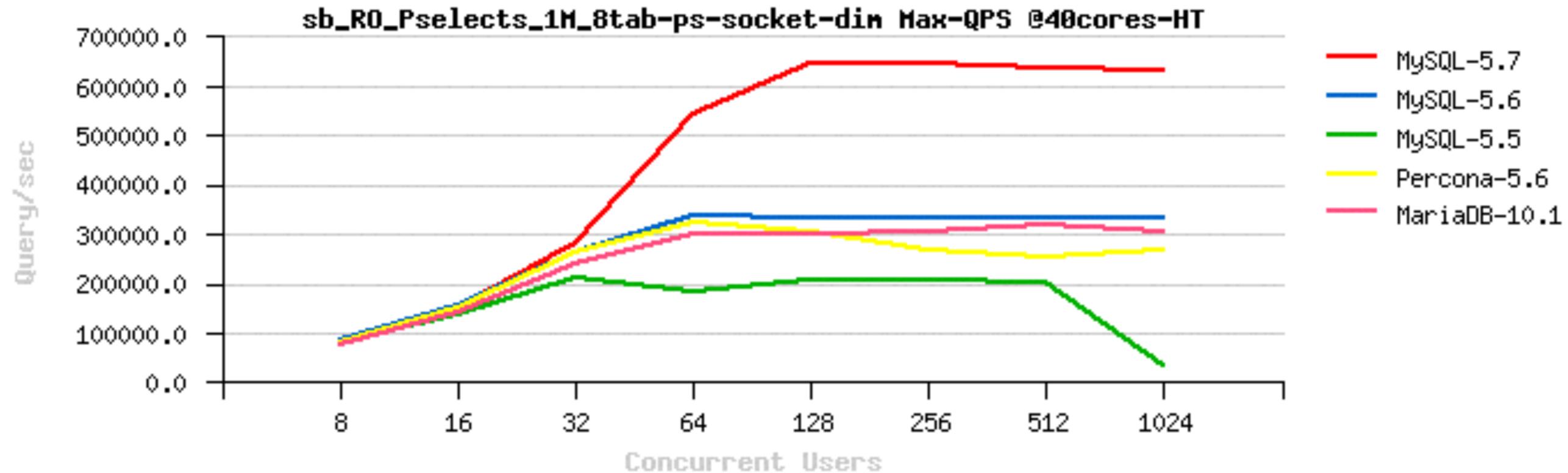
RO In-Memory @MySQL 5.7

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



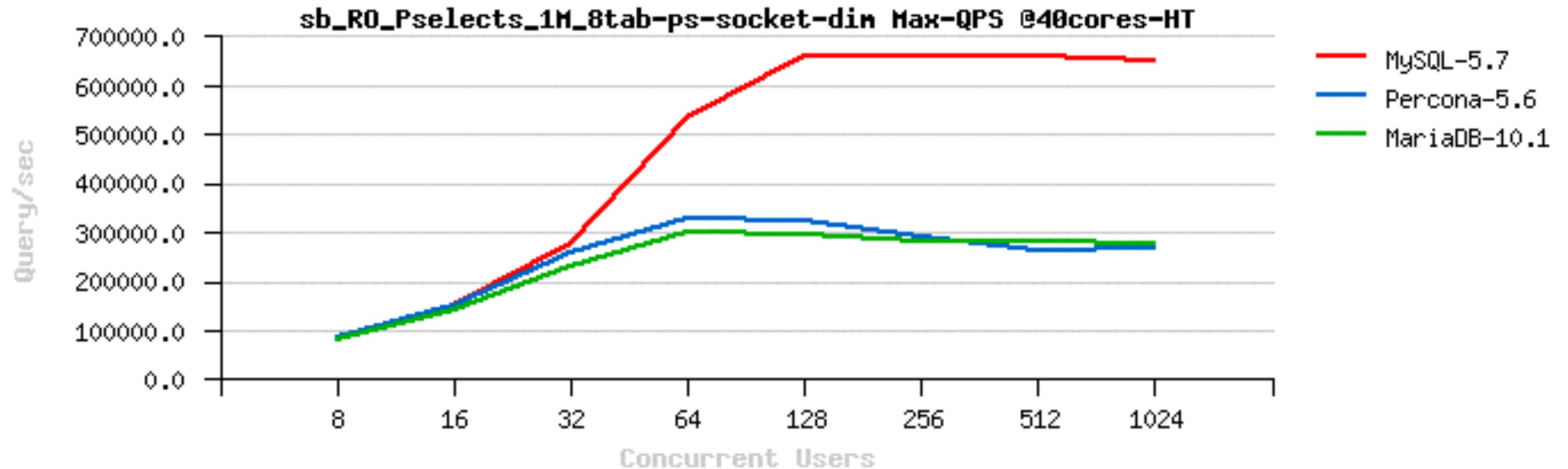
RO In-Memory @MySQL 5.7

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



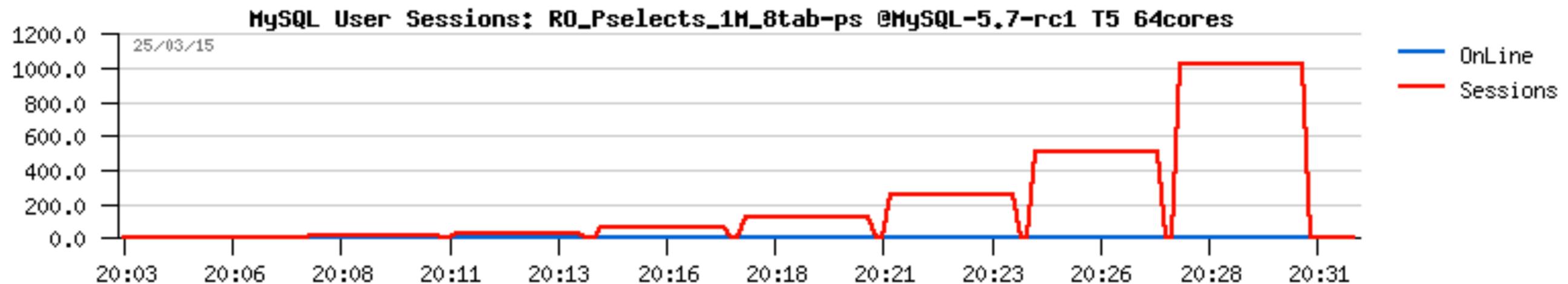
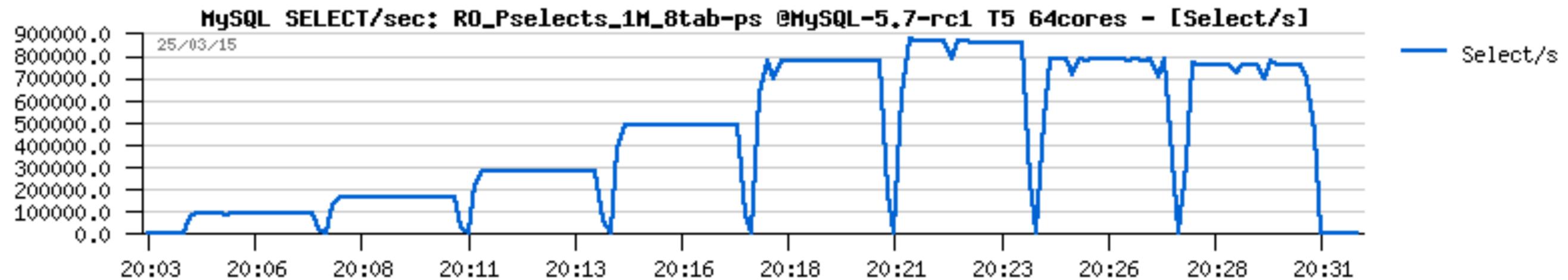
RO In-Memory @MySQL 5.7 - Apr.2015

- **655K QPS** Sysbench Point-Selects 8-tab, 40cores-HT :
 - And we can do over 700K QPS and more if will remove many new features we've added to MySQL 5.7 — but this is not our goal ;-))



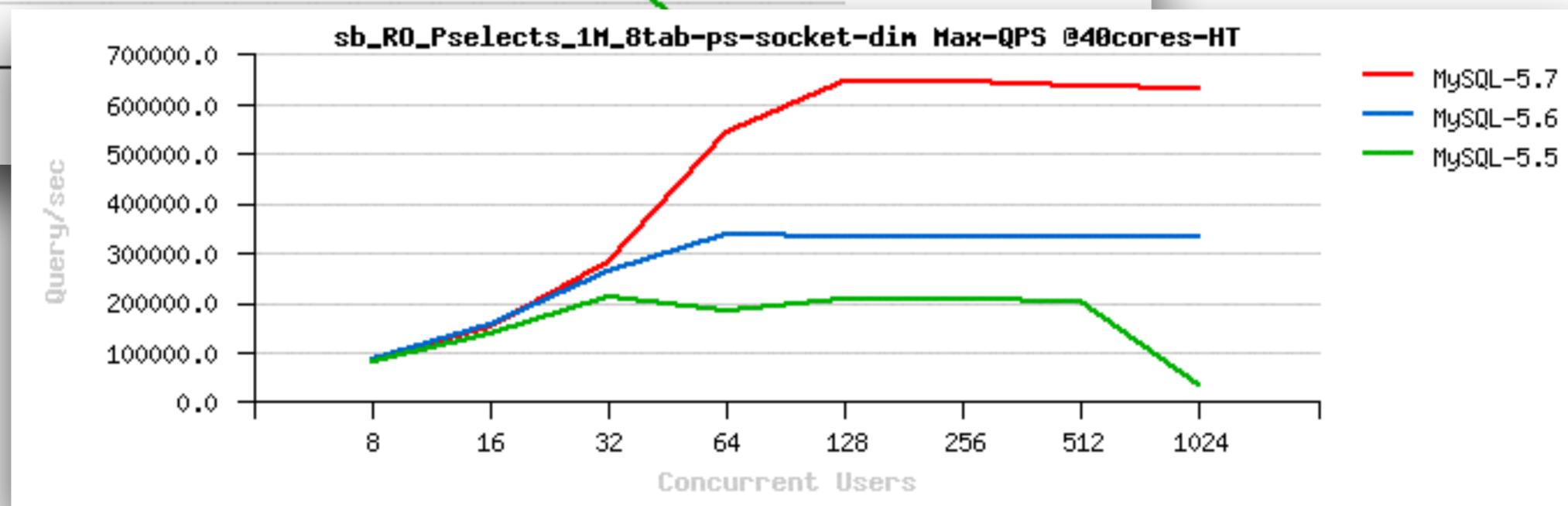
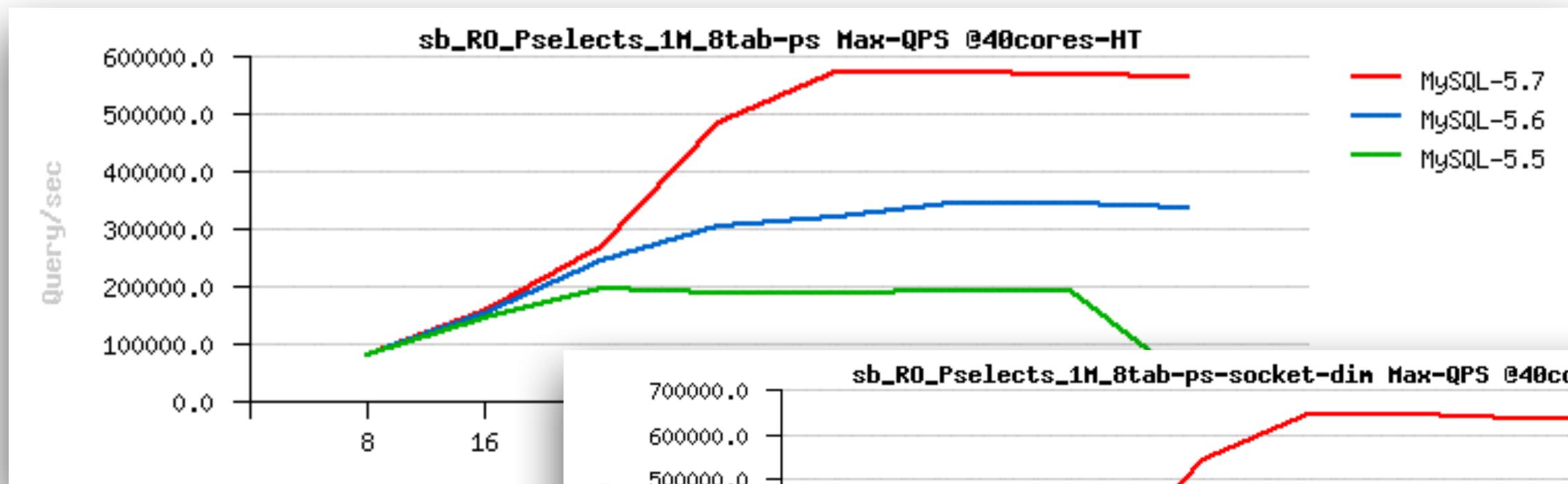
RO In-Memory @MySQL 5.7 - Apr.2015

- **Around 900K QPS** Sysbench Point-Selects 8-tab, 64cores SPARC T5 :
 - Just a probe test with zero efforts ;-)



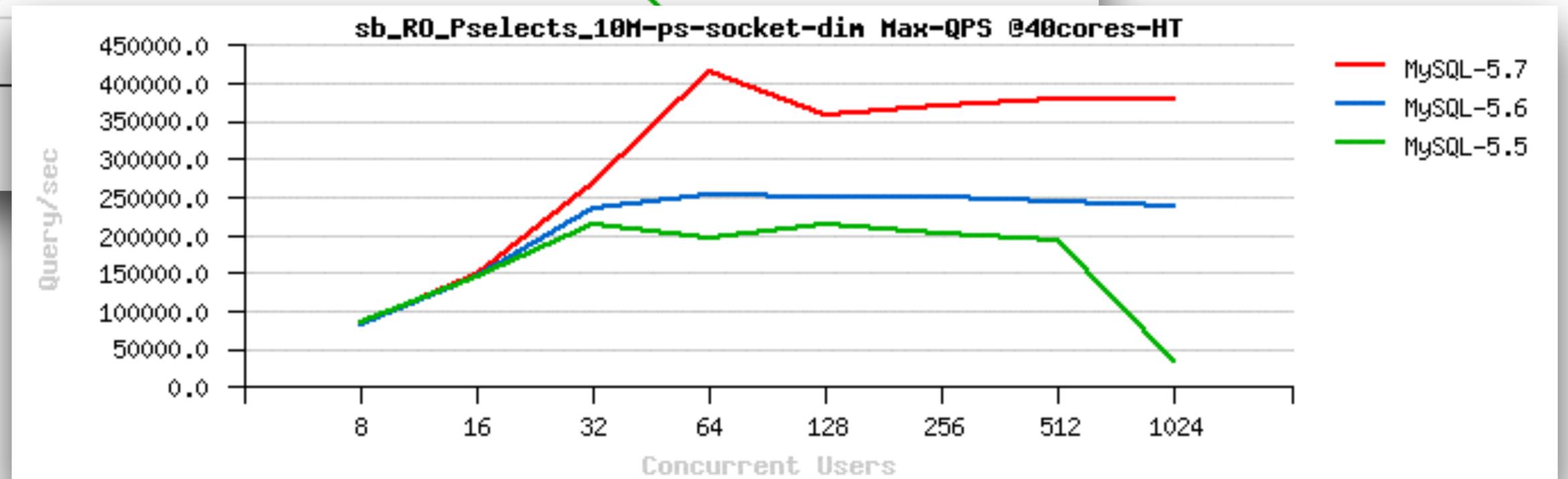
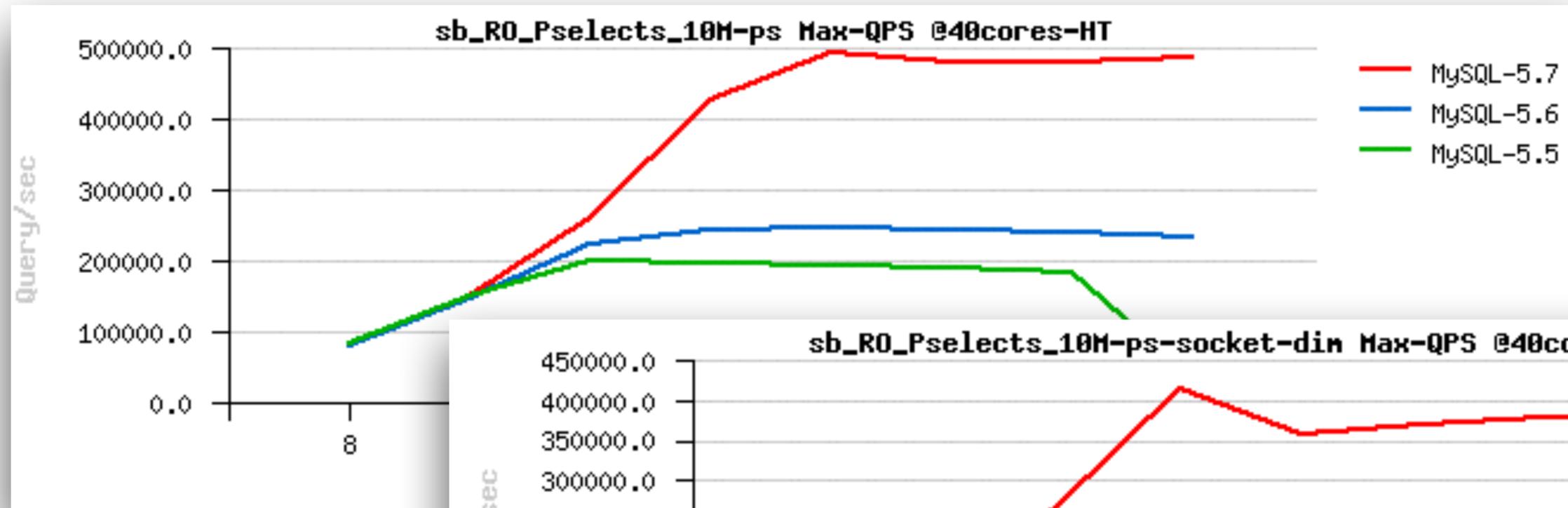
Few words about RO scalability

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



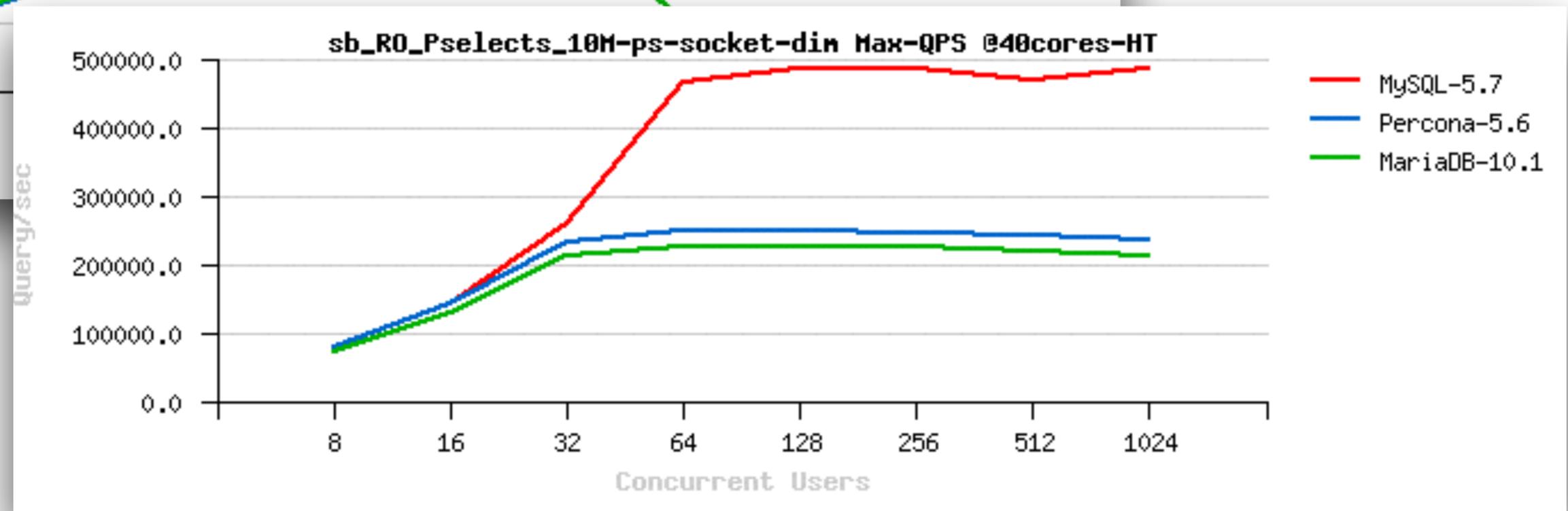
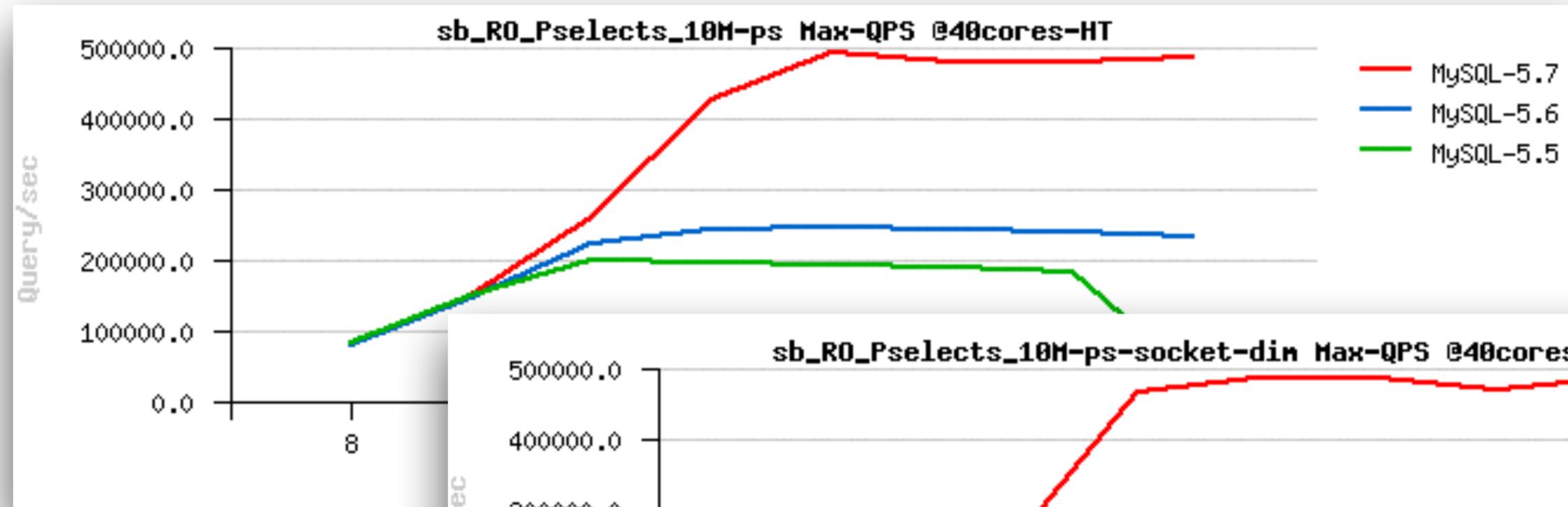
Few words about RO scalability (bis) - Oct.2014

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



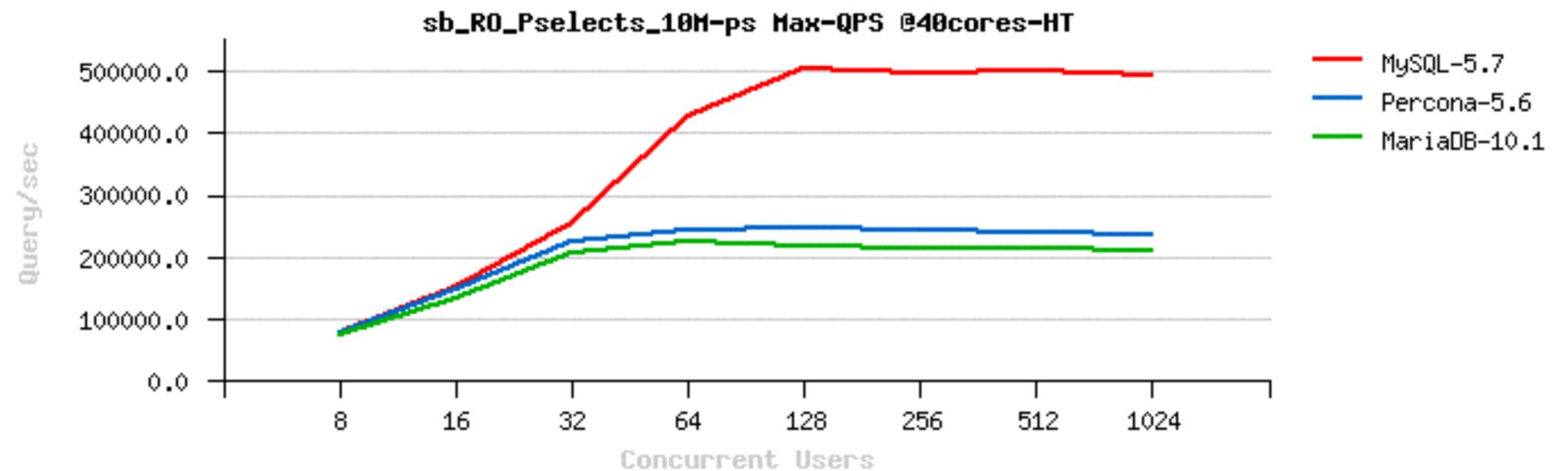
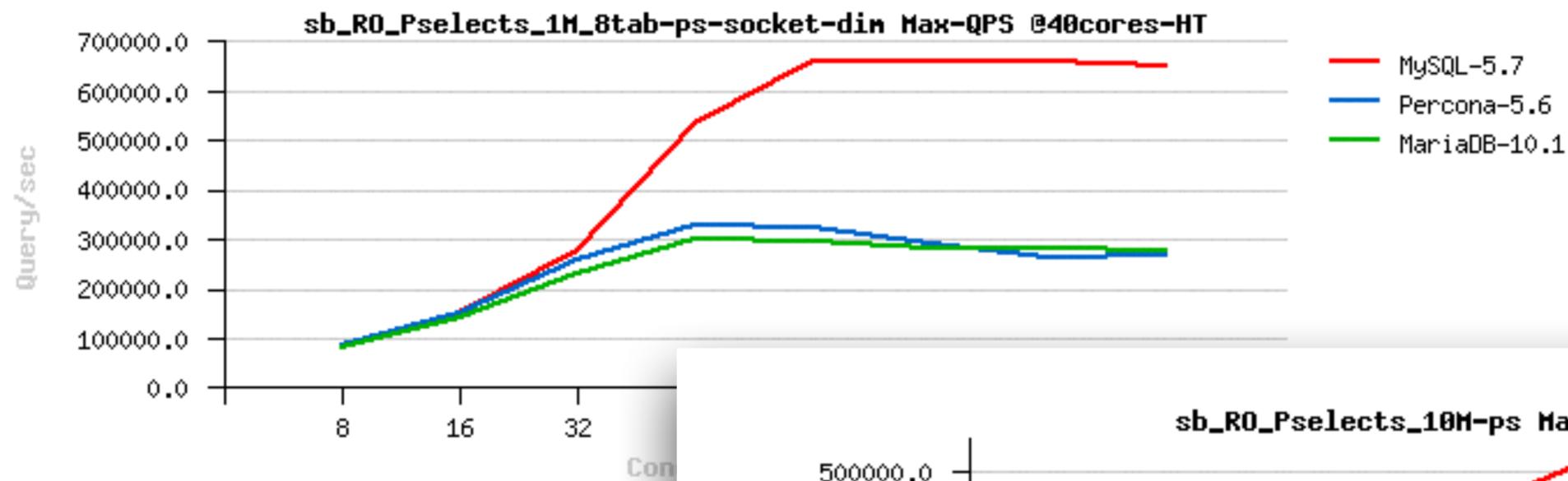
Few words about RO scalability (bis) - Apr.2015

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



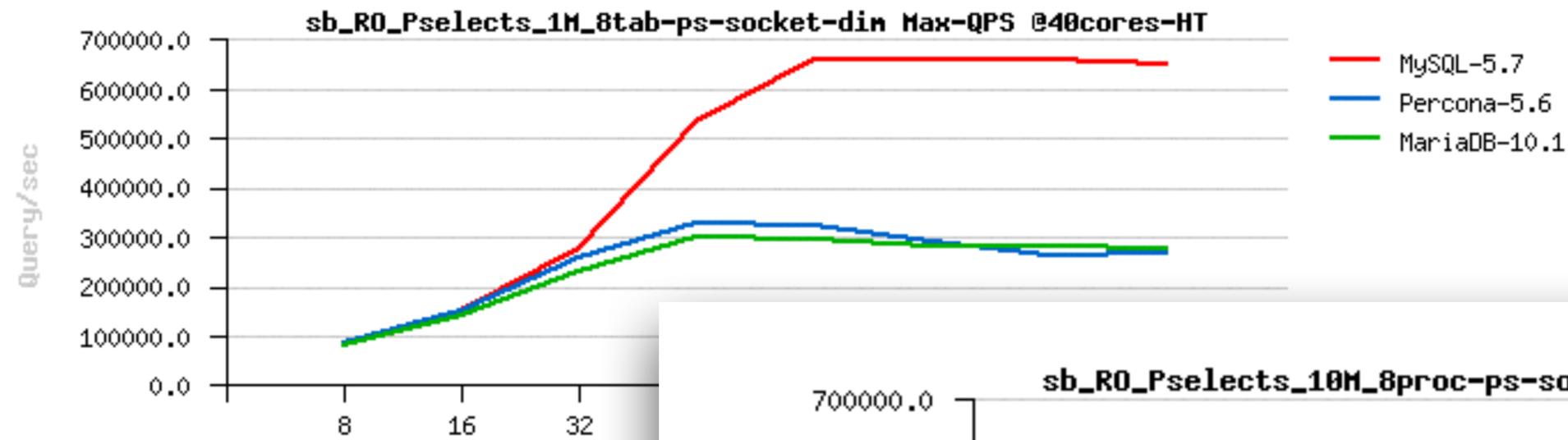
Few words about RO scalability (bis2) - Apr.2015

- OLTP_RO Point-selects **8-tables -vs- 1-table**, the same 40cores host
 - Note : running 8 sysbench processes on 8-tables

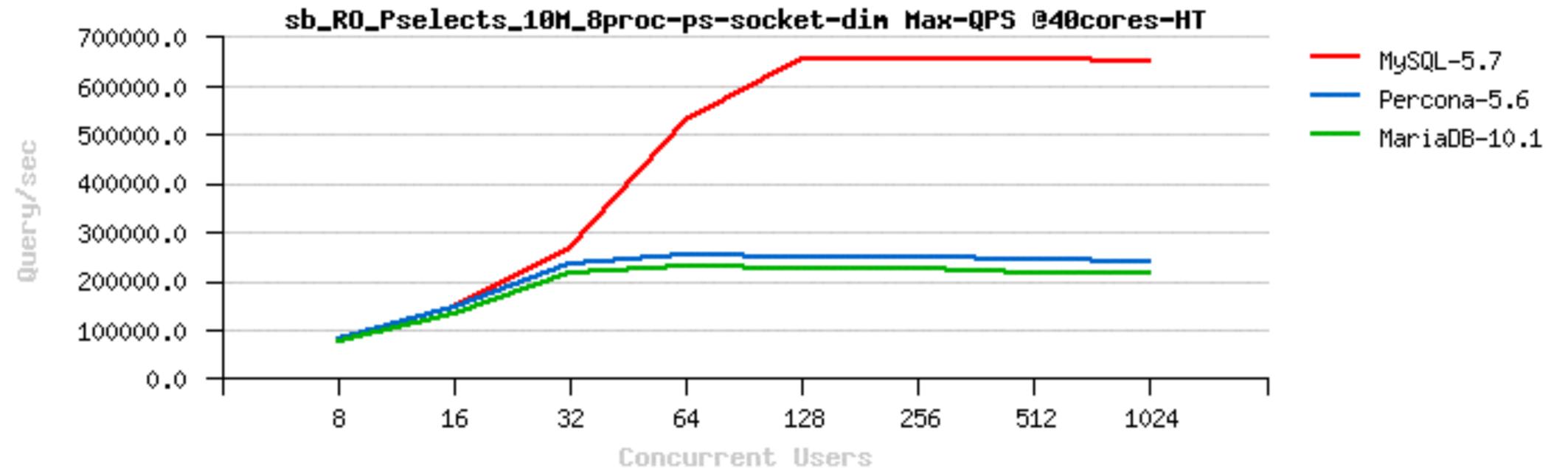


Few words about RO scalability (bis2) - Apr.2015

- OLTP_RO Point-selects 8-tables -vs- 1-table, the same 40cores host
 - Note : running now 8 sysbench processes on 1-table test too

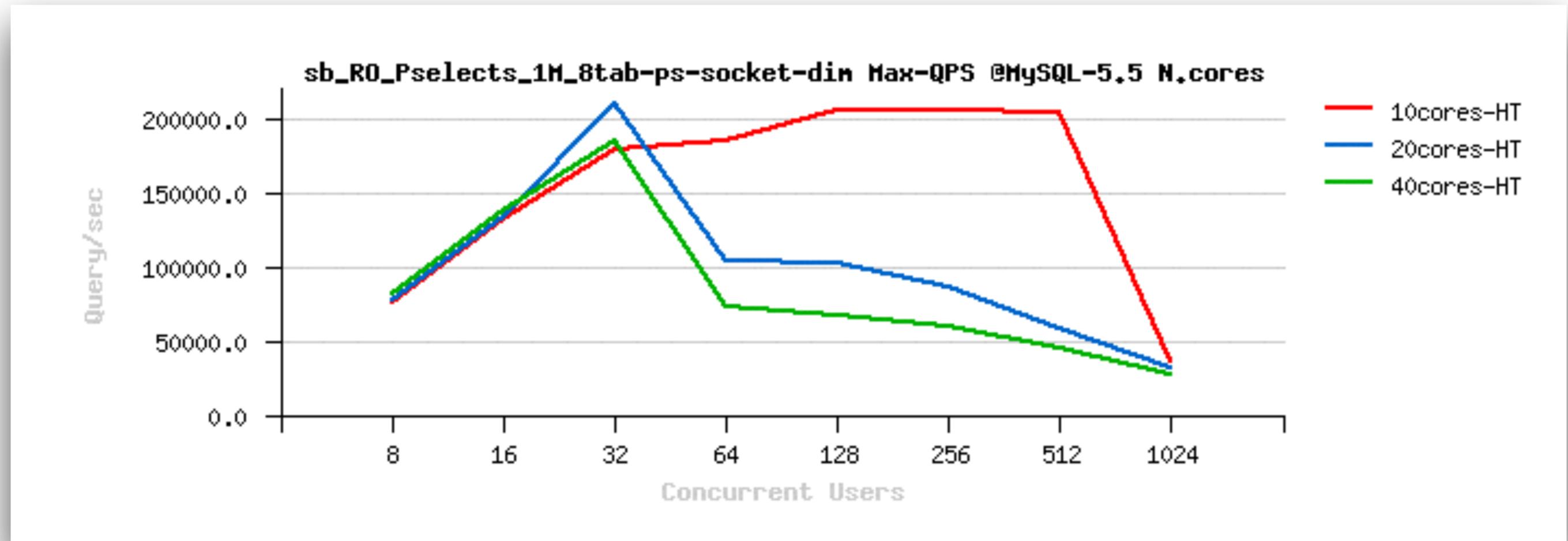


TODO: fix Sysbench ;-)



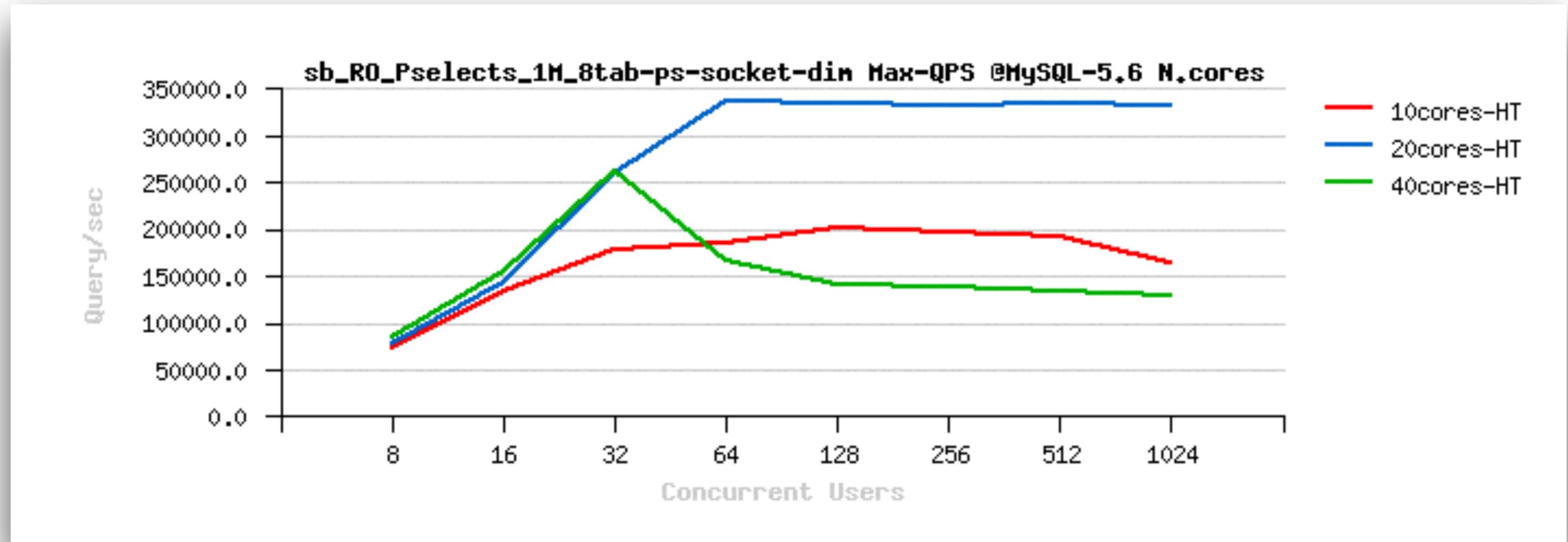
Few words about RO scalability (2)

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



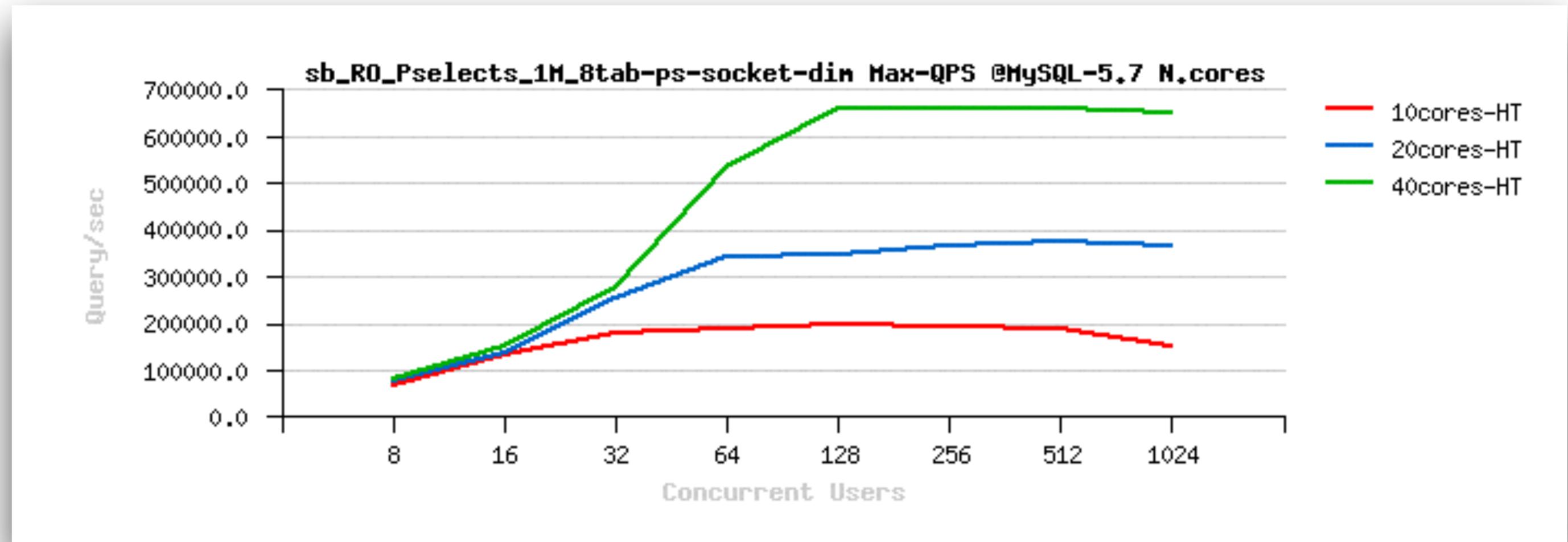
Few words about RO scalability (3)

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



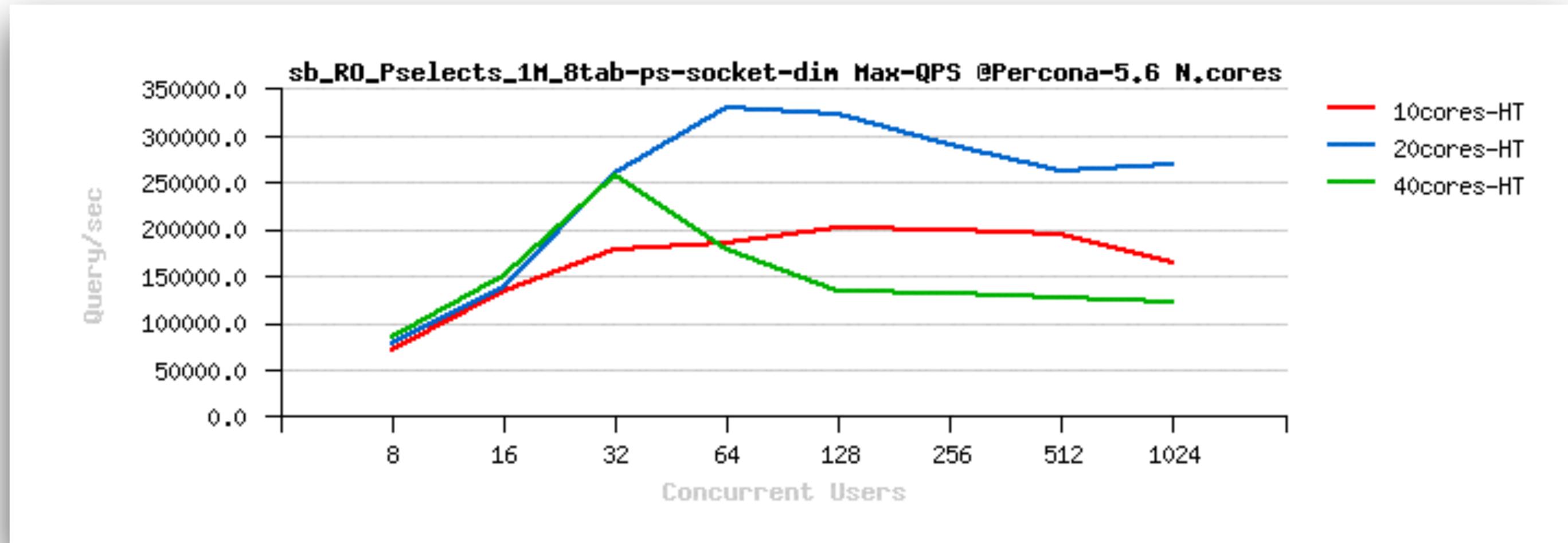
Few words about RO scalability (4) - Apr.2015

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



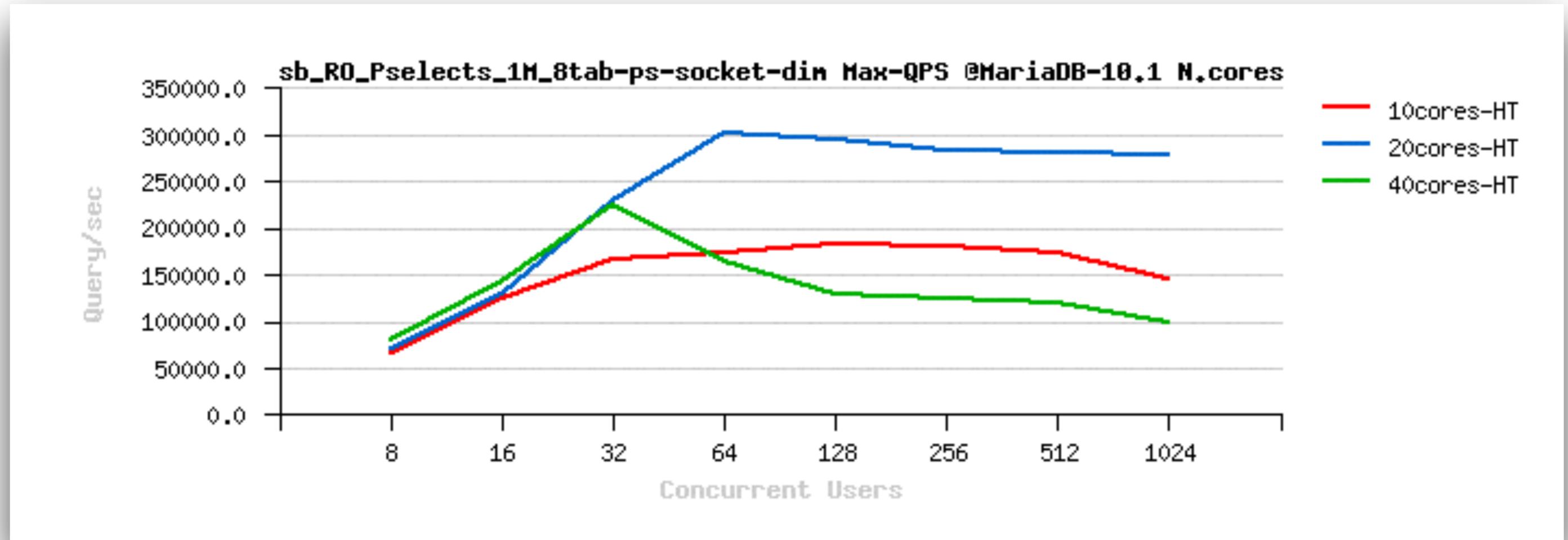
Few words about RO scalability (5) - Apr.2015

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



Few words about RO scalability (6) - Apr.2015

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



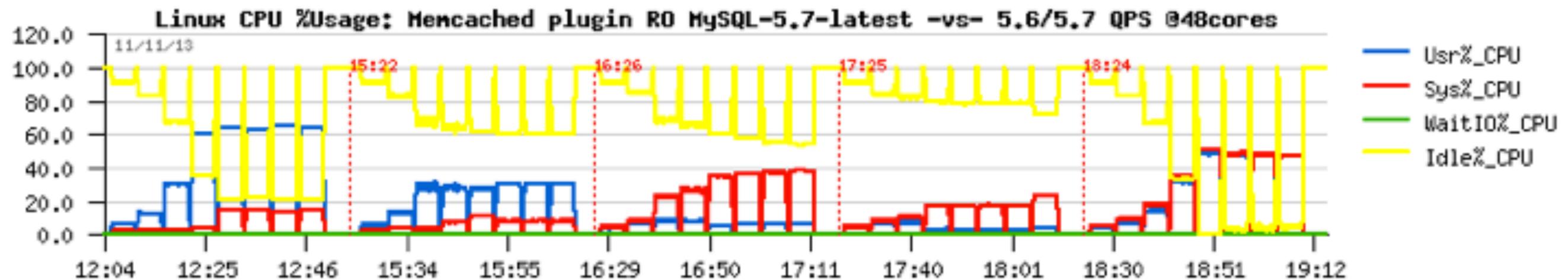
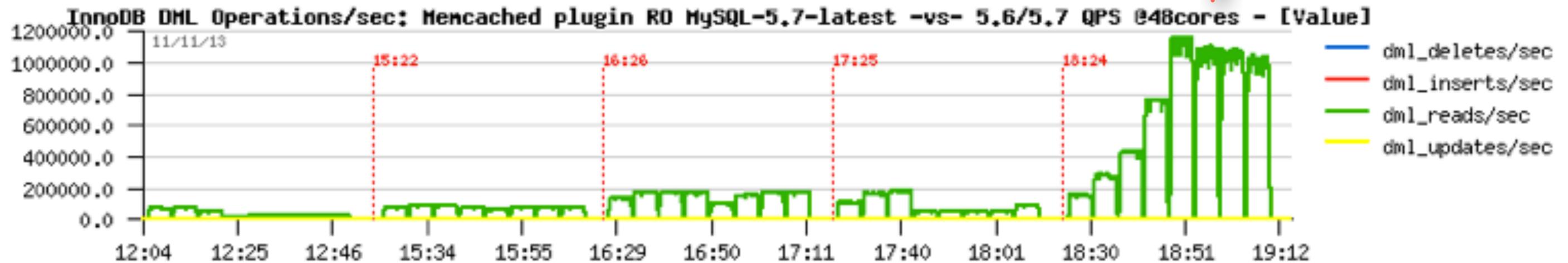
InnoDB Memcached Plugin

- **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 ;-)

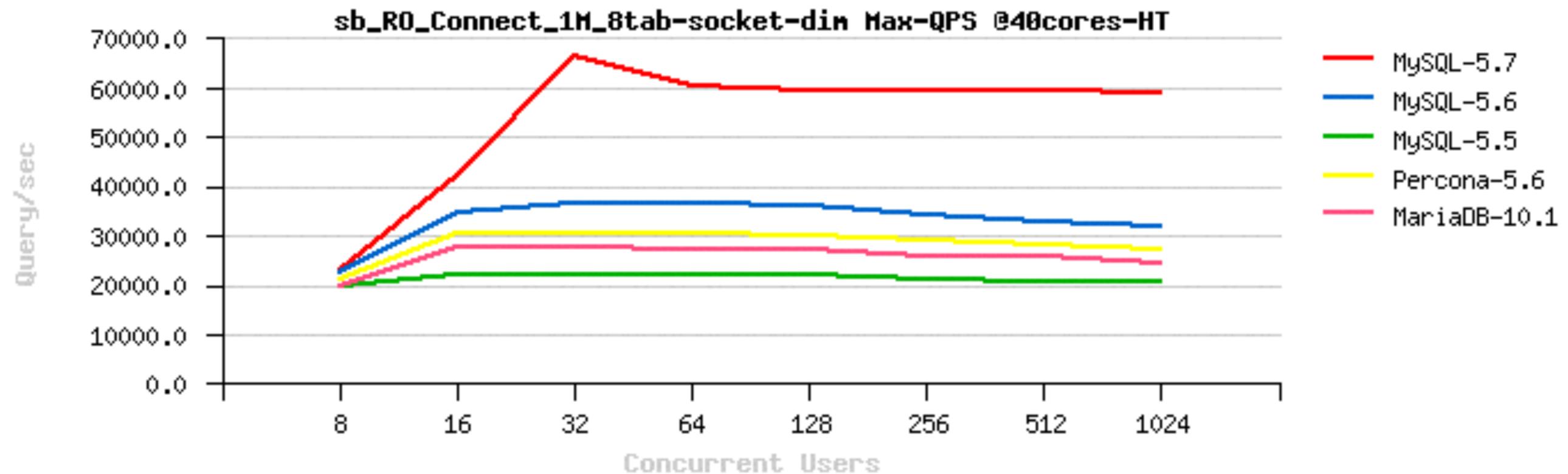


Story #3 : Connect/sec performance

- **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

OLTP_RO Connect Performance - Oct.2014

- 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

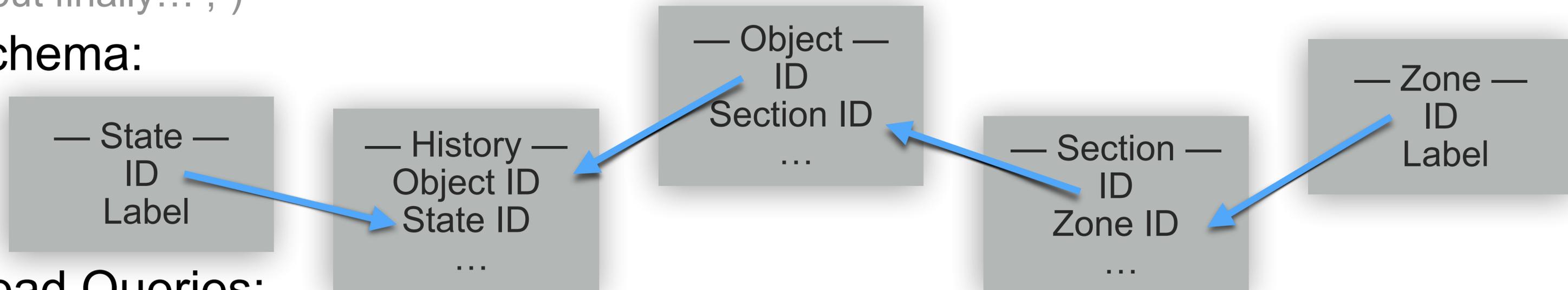


Story #4 : strange scalability issue @dbSTRESS

- Preface:

- we already observed in the past some strange RO 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..
- but finally... ;-)

- Schema:

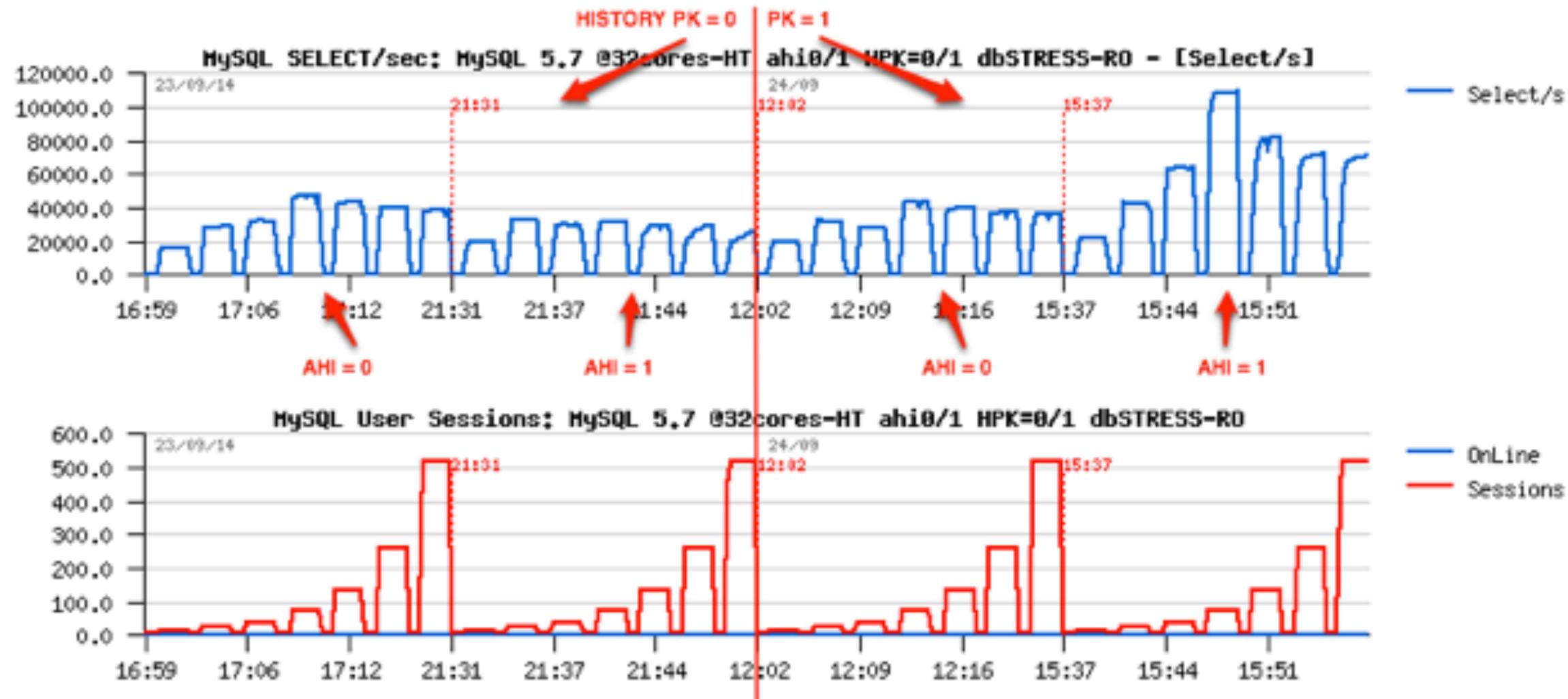


- Read Queries:

- SEL1 : SELECT * from Object, Section, Zone where Object_ID = \$(ID) ;
- SEL2 : SELECT * from History, State where Object_ID = \$(ID) ;
- H1 (debug) : SELECT * from History where Object_ID = \$(ID) ;

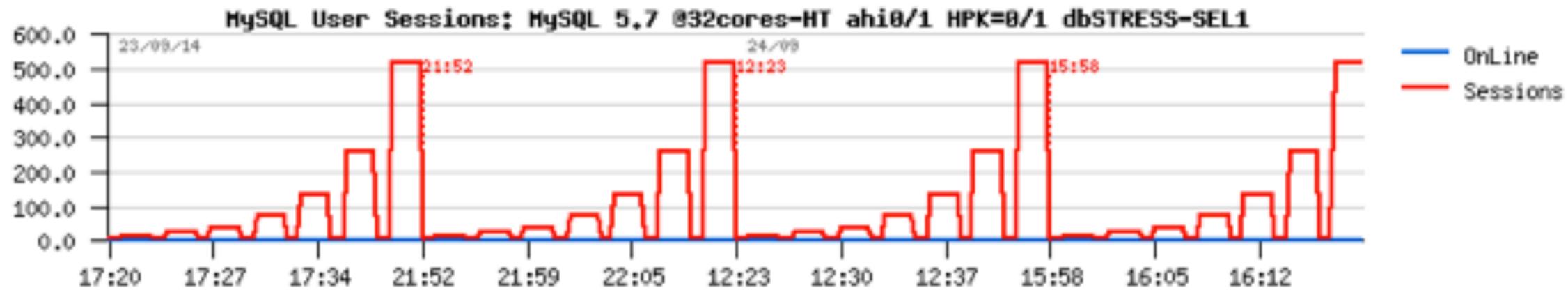
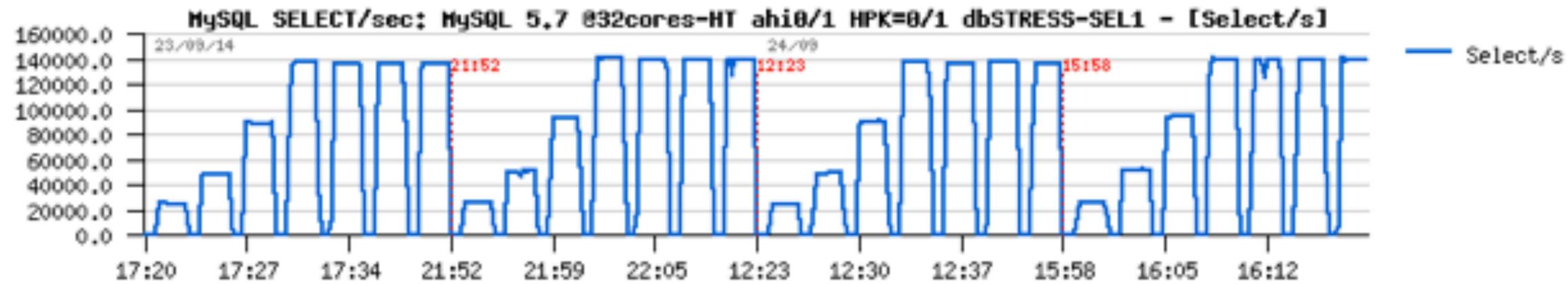
Story #4 : strange scalability issue @dbSTRESS (2)

- 32cores-HT, MySQL 5.7
 - test : dbSTRESS-RO (SEL1+SEL2), AHI = off / on



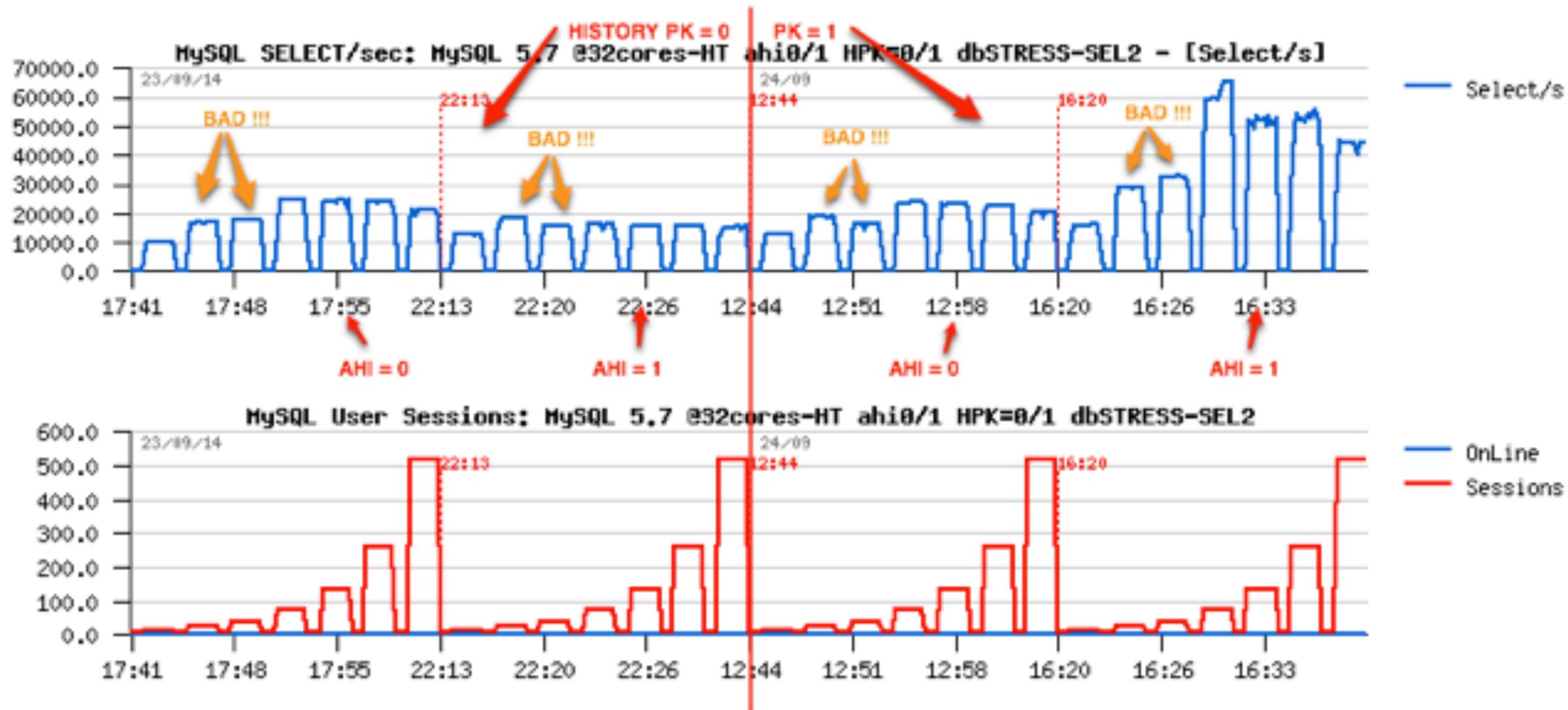
Story #4 : strange scalability issue @dbSTRESS (3)

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



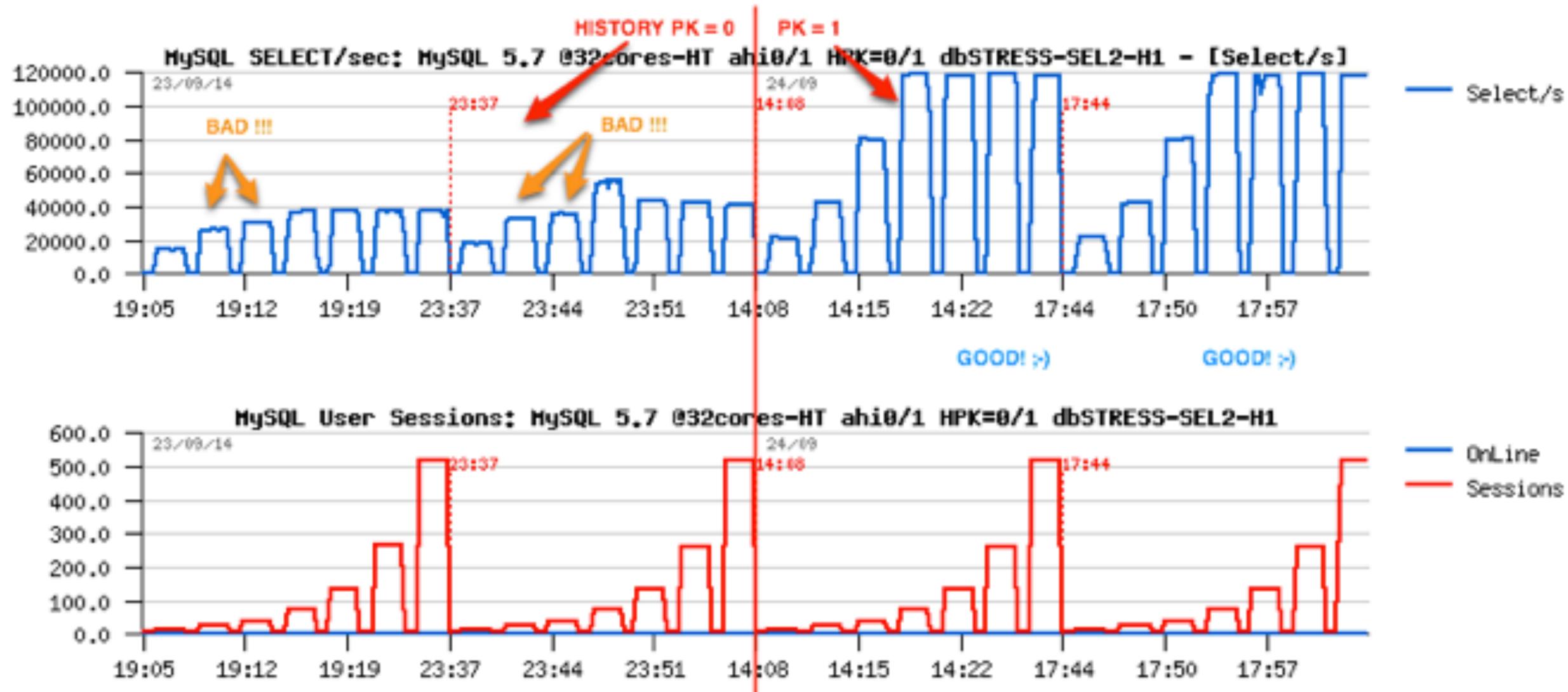
Story #4 : strange scalability issue @dbSTRESS (4)

- 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 - Oct.2014
 - test case workload dbSTRESS-SEL2 <== the main show-stopper..
 - amazing to see a x3 times perf. 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 yet for tomorrow..
 - Top CPU time from Profiler report :

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

- NOTE : pfs_* names in profiler are not always related to PFS ;-)

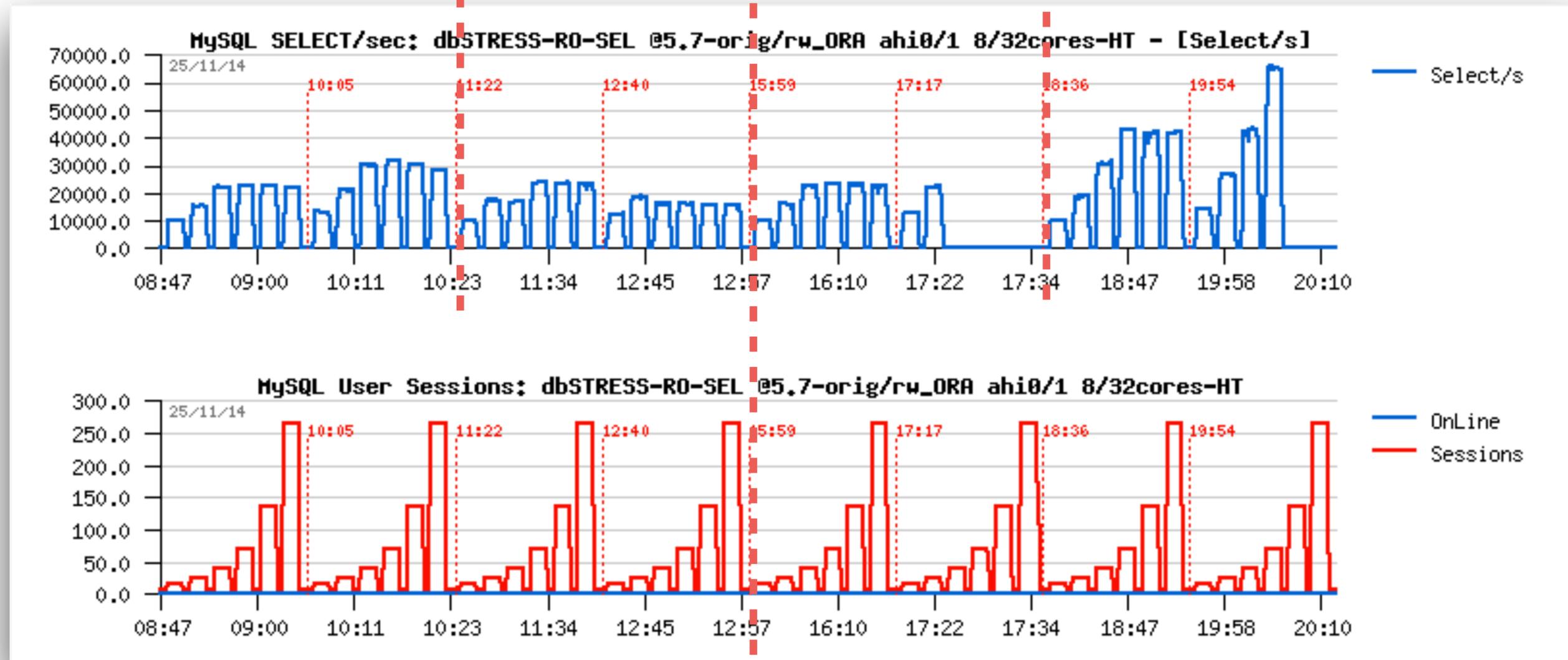
Story #4 : strange scalability issue @dbSTRESS (7)

- 32cores-HT, MySQL 5.7 - Apr.2015
 - test case workload dbSTRESS-SEL2 <== the main show-stopper..
 - InnoDB RW-lock design is the main issue here..
 - InnoDB RW-locks are not aware about CPU cache line sync/miss..
 - which is involving inter-CPU-sockets cache line “contention”..
 - and killing all the AHI benefit..
 - (once again HPC related problems in Database world ;-))
 - a new “scalable” RW-lock design is required!
 - a more advanced AHI design too!
 - (and generally to have a more advanced mutex code would be nice as well ;-))

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

- 32cores-HT, MySQL 5.7 - Apr.2015
 - test case workload dbSTRESS-SEL2 <== the main show-stopper..
 - result on a new RW-lock prototype :

8cores, AHI=0/1 | 32cores, AHI=0/1 | 8cores, AHI=0/1 | 32cores, AHI=0/1



Read+Write Workloads Scalability @MySQL 5.7

- Huge progress is already here too!
 - improved index locking
 - reduced lock_sys mutex contention
 - parallel flushing + improved flushing design
 - much better observability of internals
 - etc..
- 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..
 - still many things are in pipe & prototype..

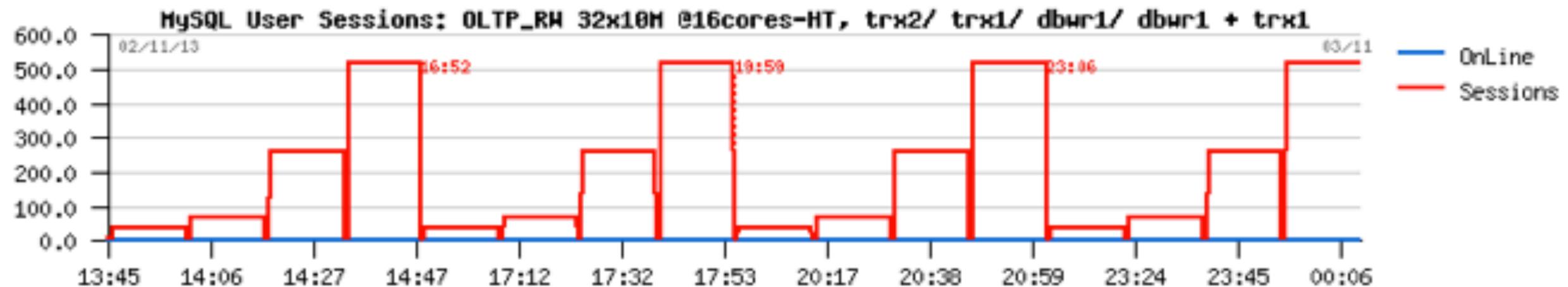
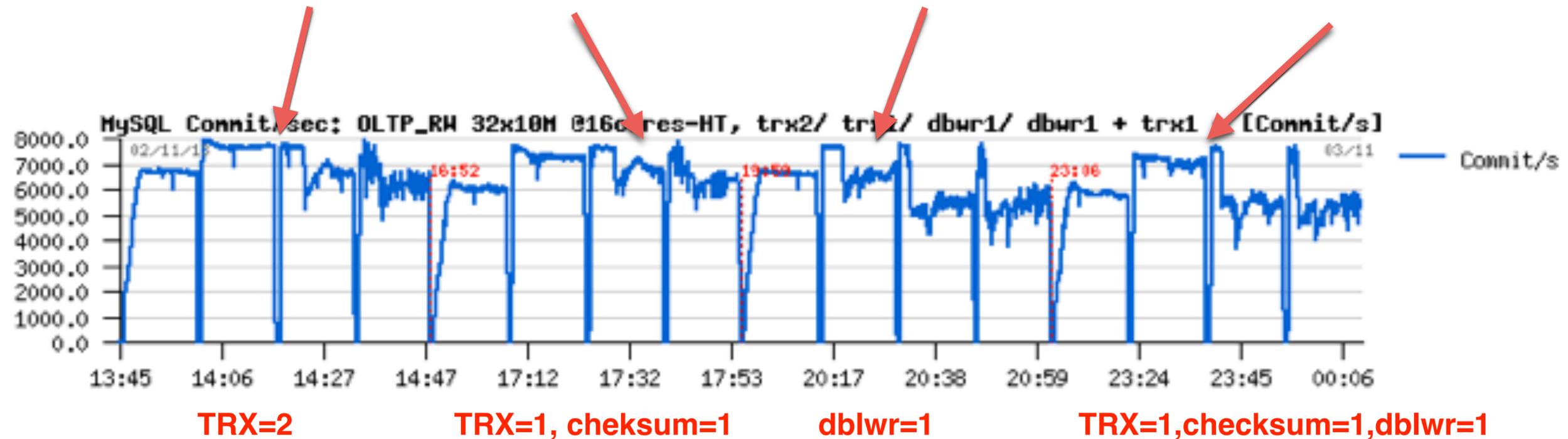
Read+Write Performance @MySQL / InnoDB

- 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

- test cases: trx=2 | trx=1 + checksum=1 | dblwr=1 | trx=1 + checksum=1 + dblwr=1



RW related starter configuration settings

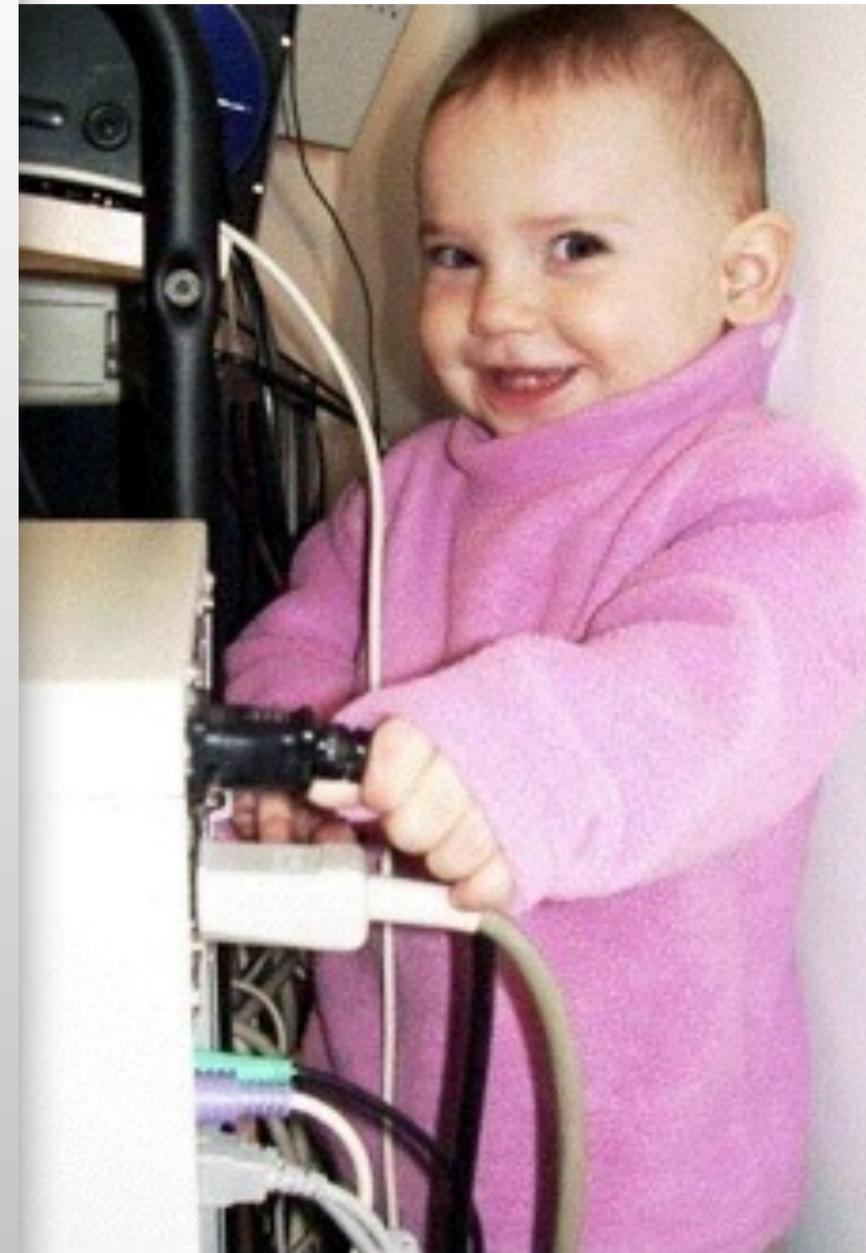
- my.conf :

```
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=0_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

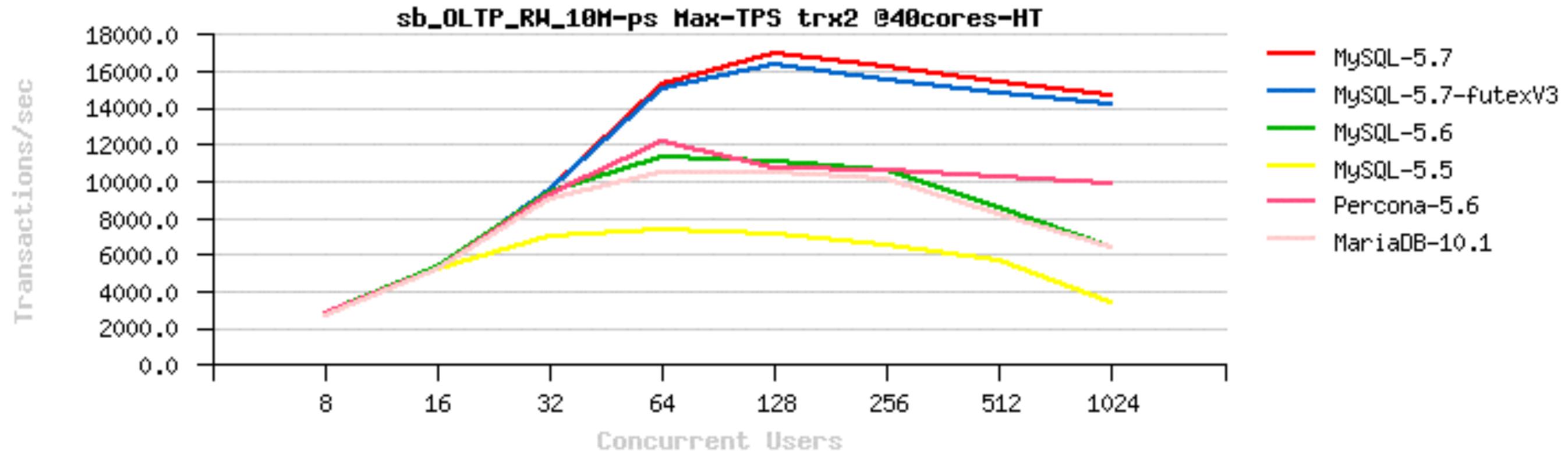
innodb_purge_threads=4
innodb_max_purge_lag_delay=30000000
innodb_max_purge_lag= 0 / 1000000

binlog ??
```



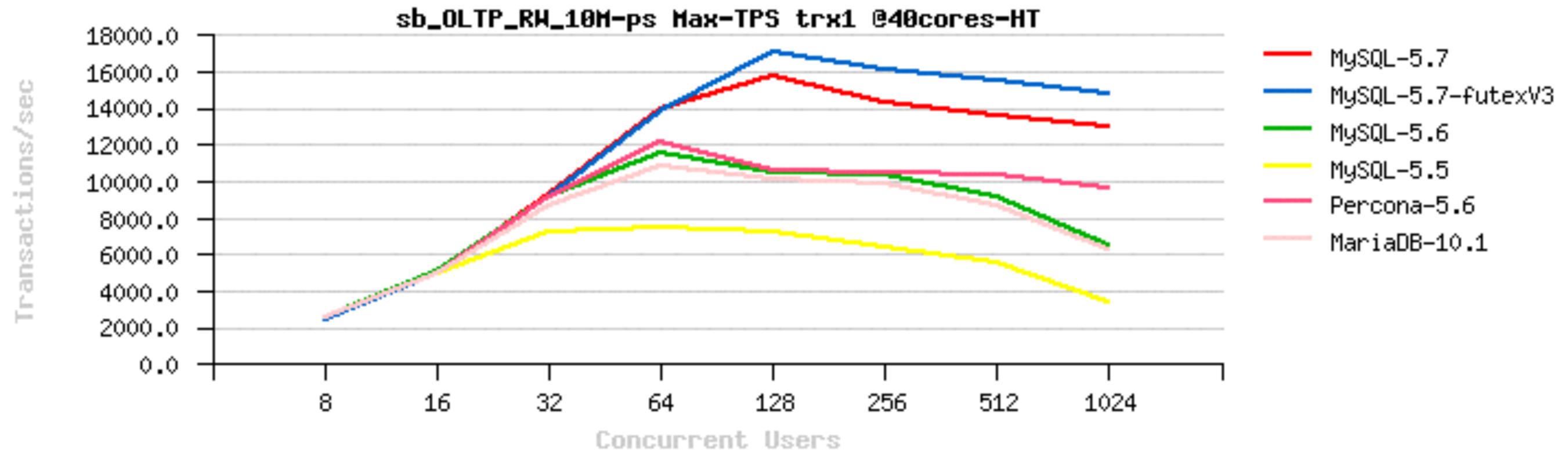
Sysbench OLTP_RW In-Memory - Oct.2014

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



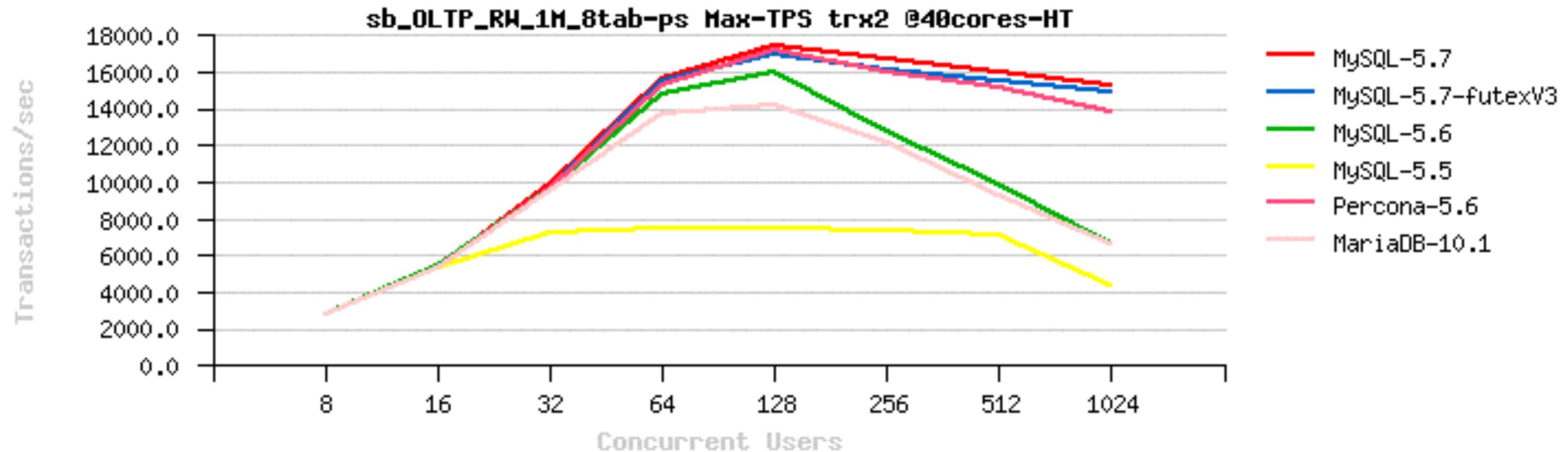
Sysbench OLTP_RW In-Memory - Oct.2014

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



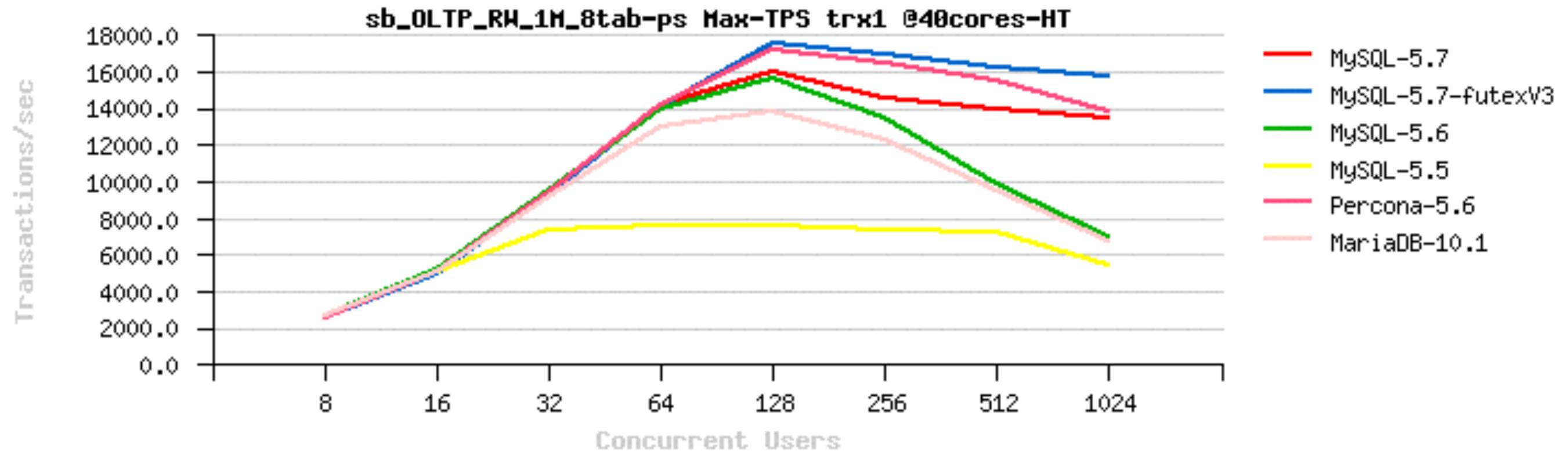
Sysbench OLTP_RW In-Memory - Oct.2014

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



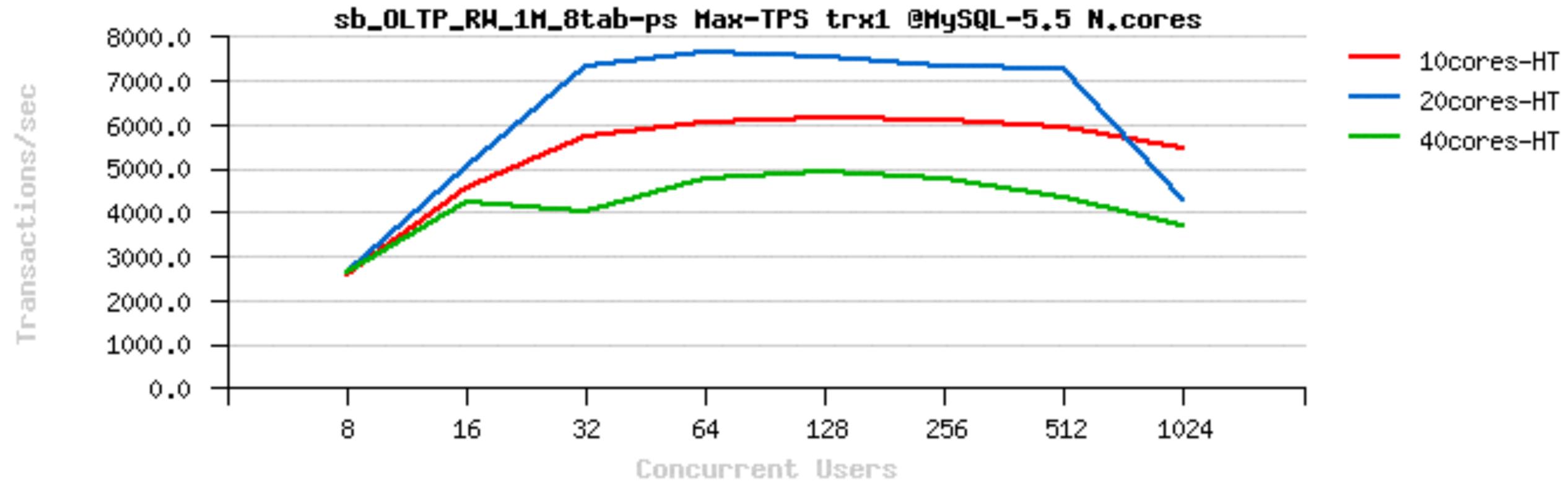
Sysbench OLTP_RW In-Memory - Oct.2014

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



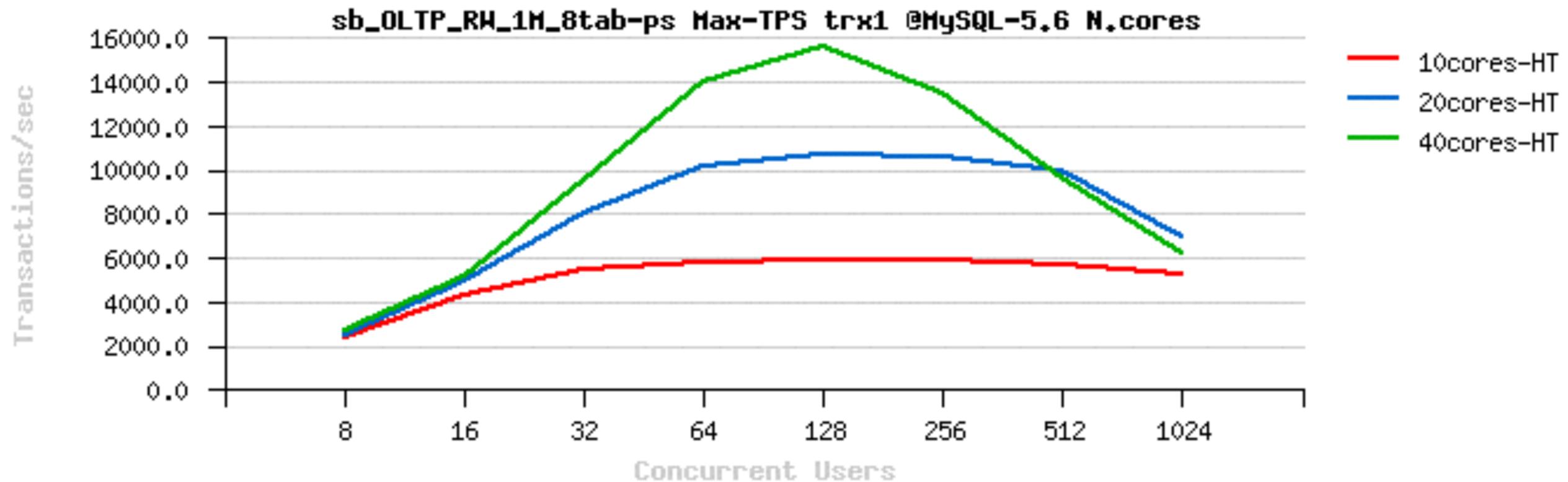
Sysbench OLTP_RW In-Memory - Oct.2014

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



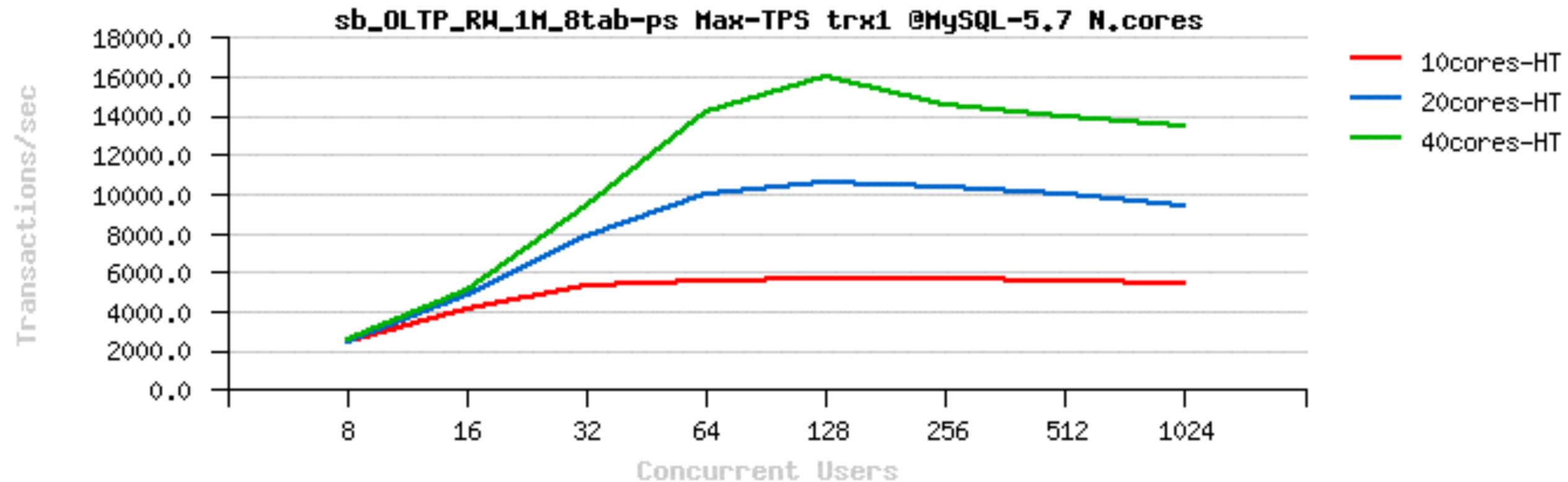
Sysbench OLTP_RW In-Memory - Oct.2014

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



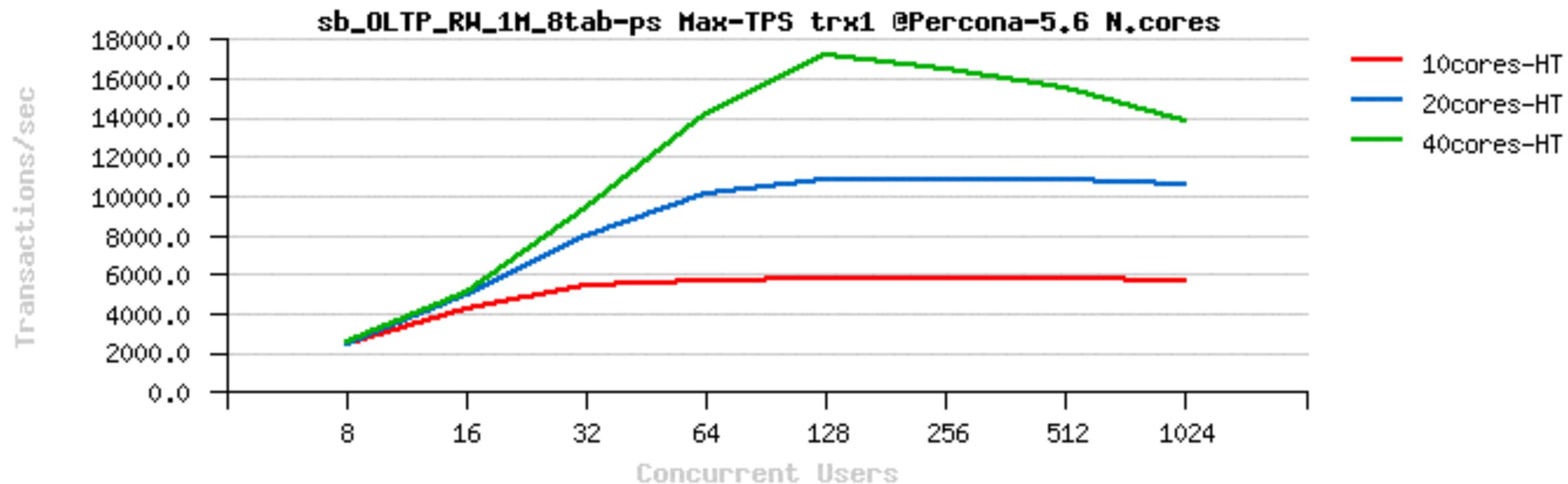
Sysbench OLTP_RW In-Memory - Oct.2014

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



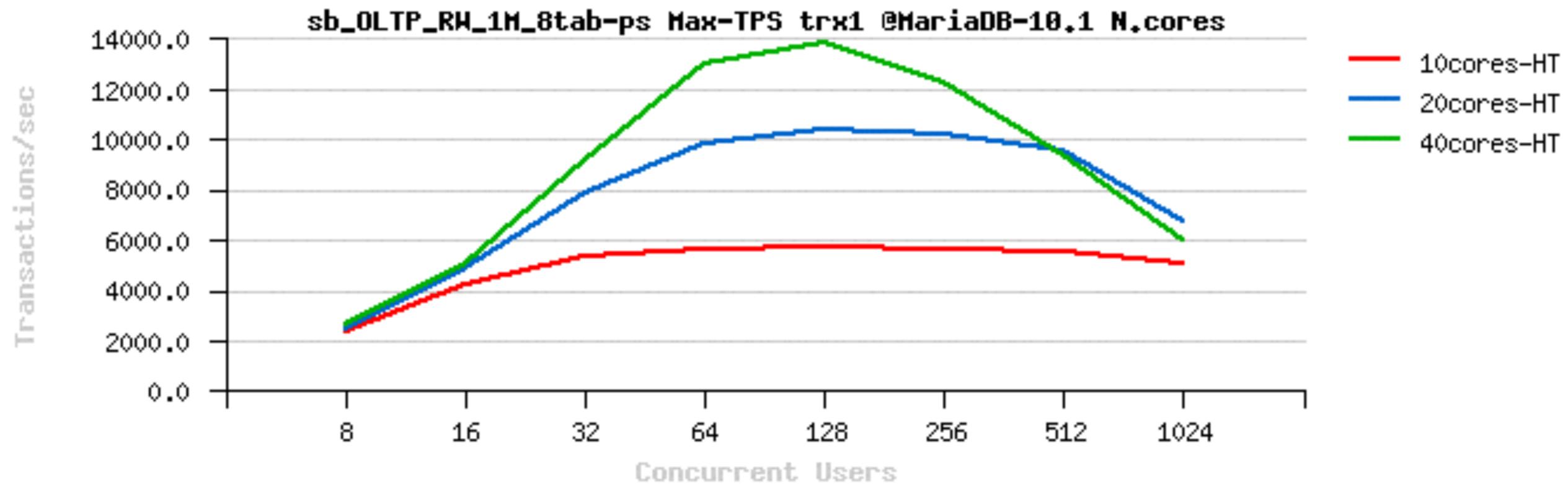
Sysbench OLTP_RW In-Memory - Oct.2014

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



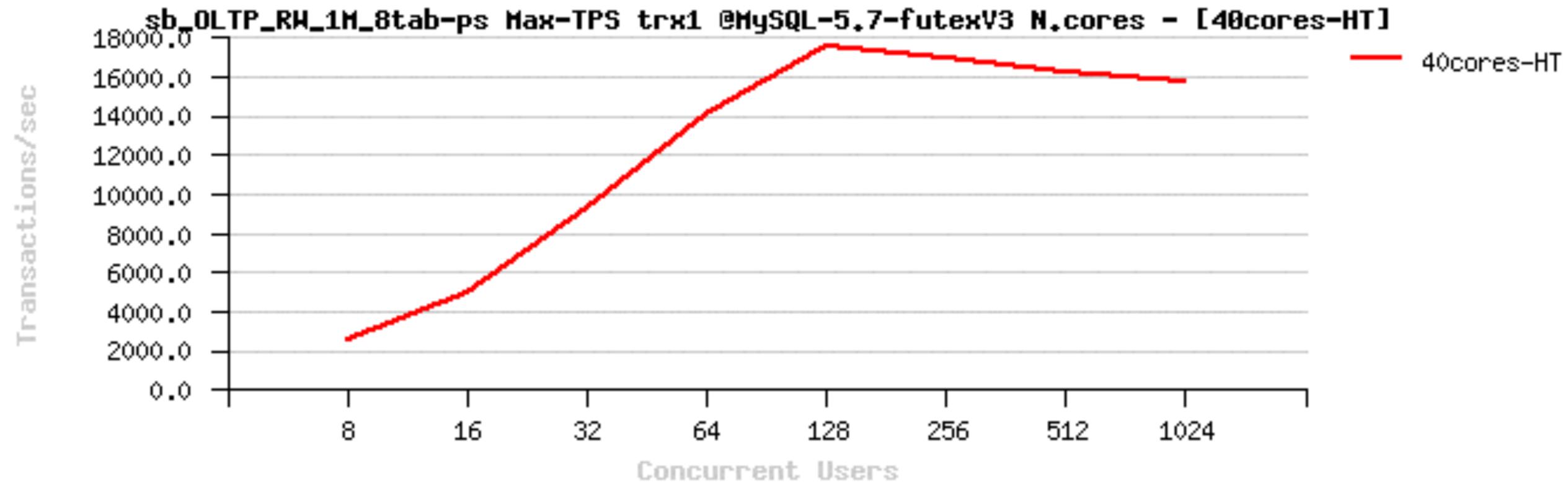
Sysbench OLTP_RW In-Memory - Oct.2014

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



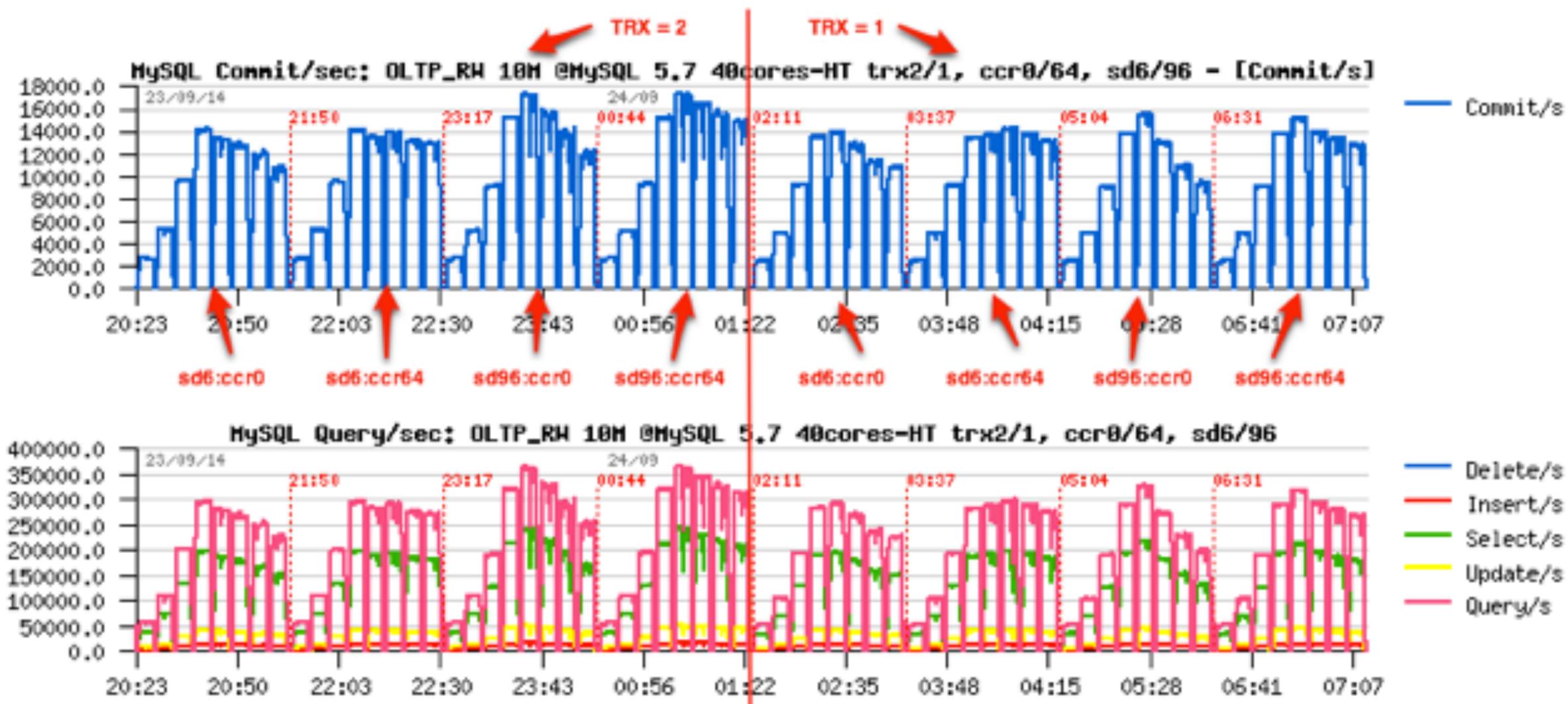
Sysbench OLTP_RW In-Memory - Oct.2014

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



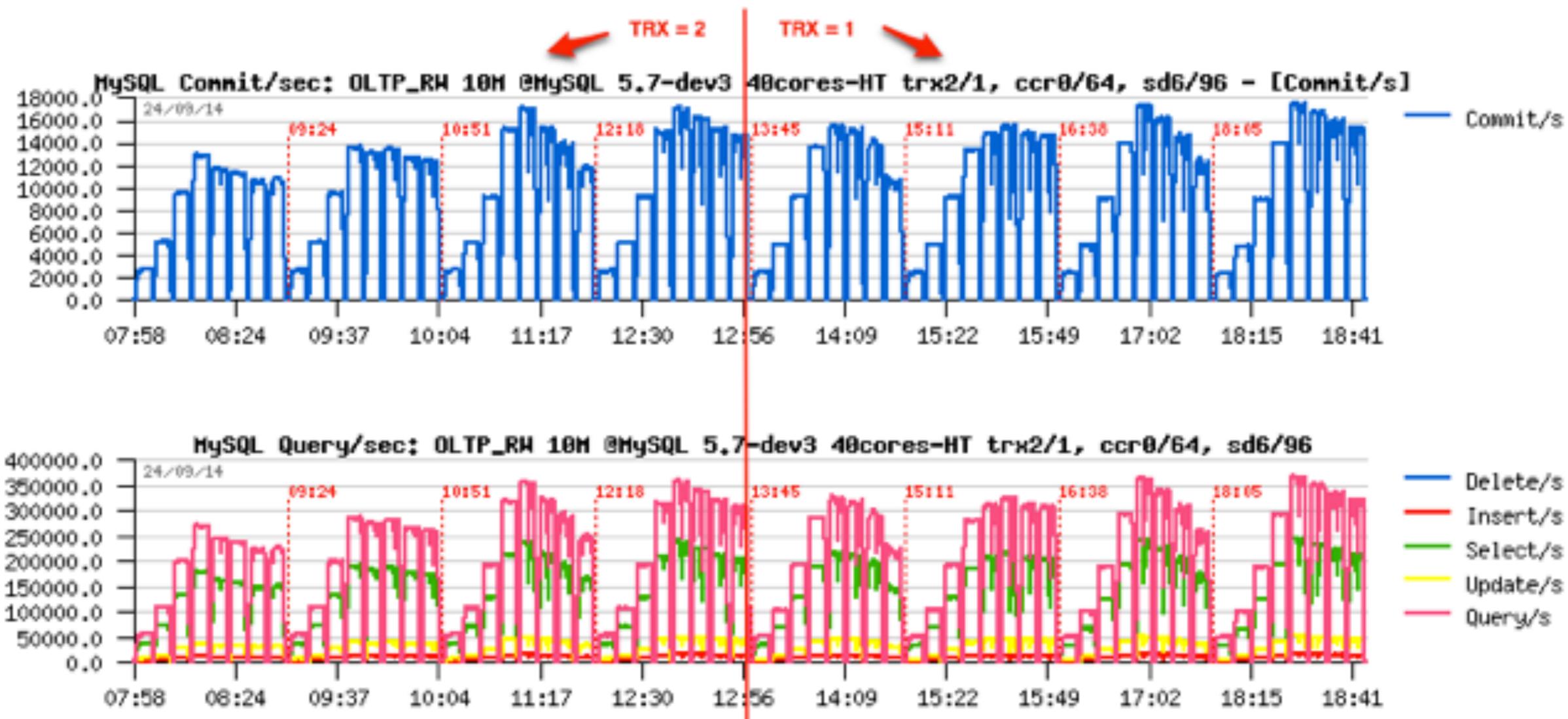
Concurrency tuning on OLTP_RW @MySQL 5.7

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



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)

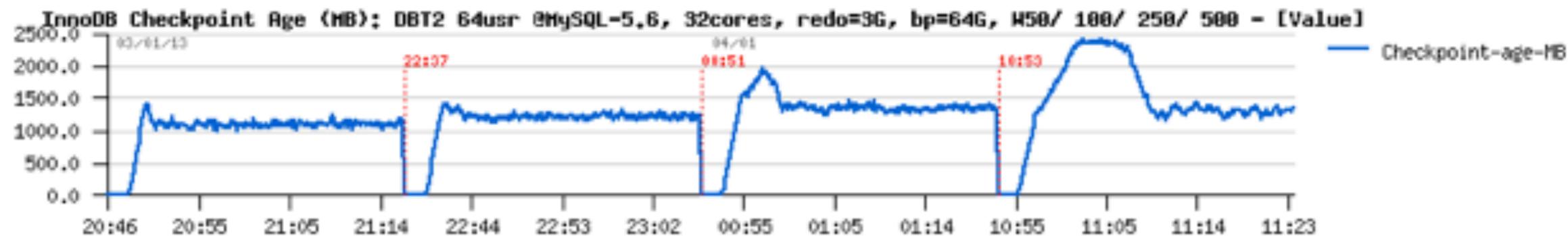
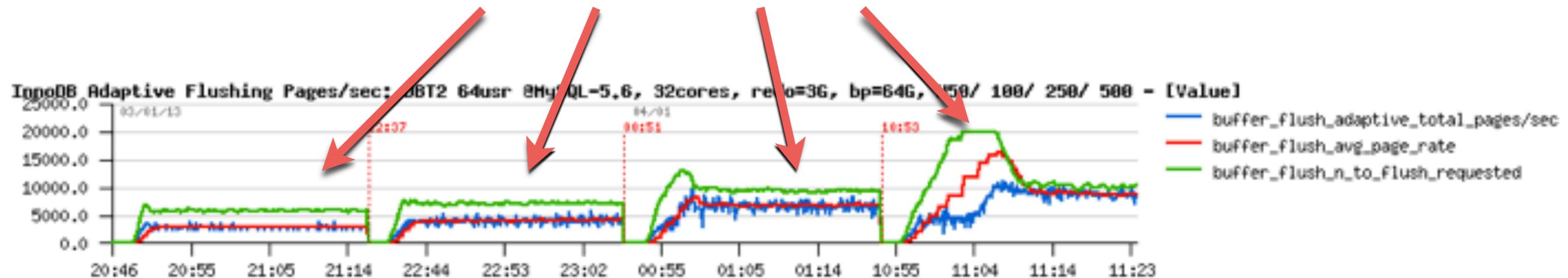


RW IO-bound

- **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 ;-)

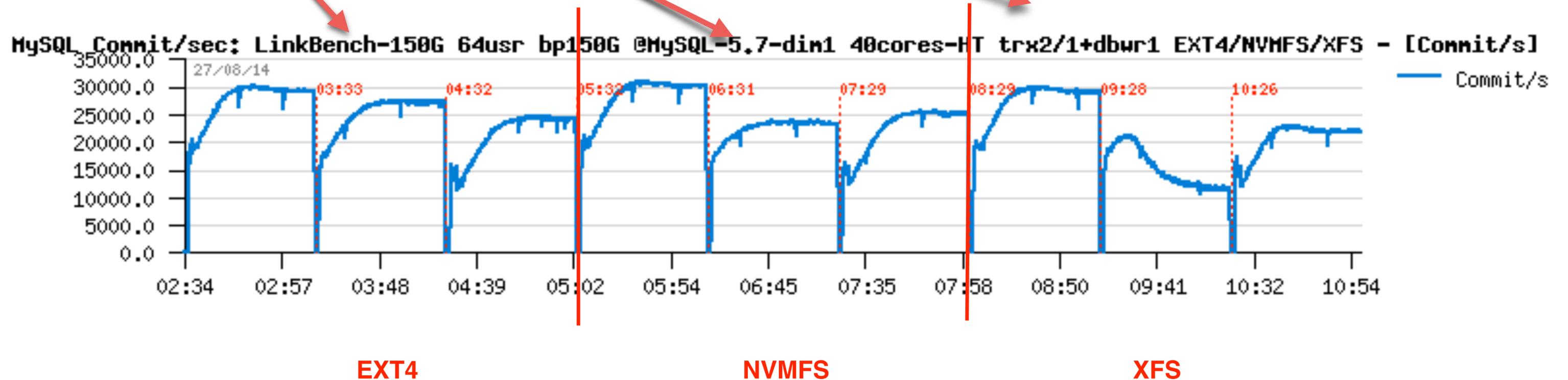
RW IO-bound “In-Memory”

- 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



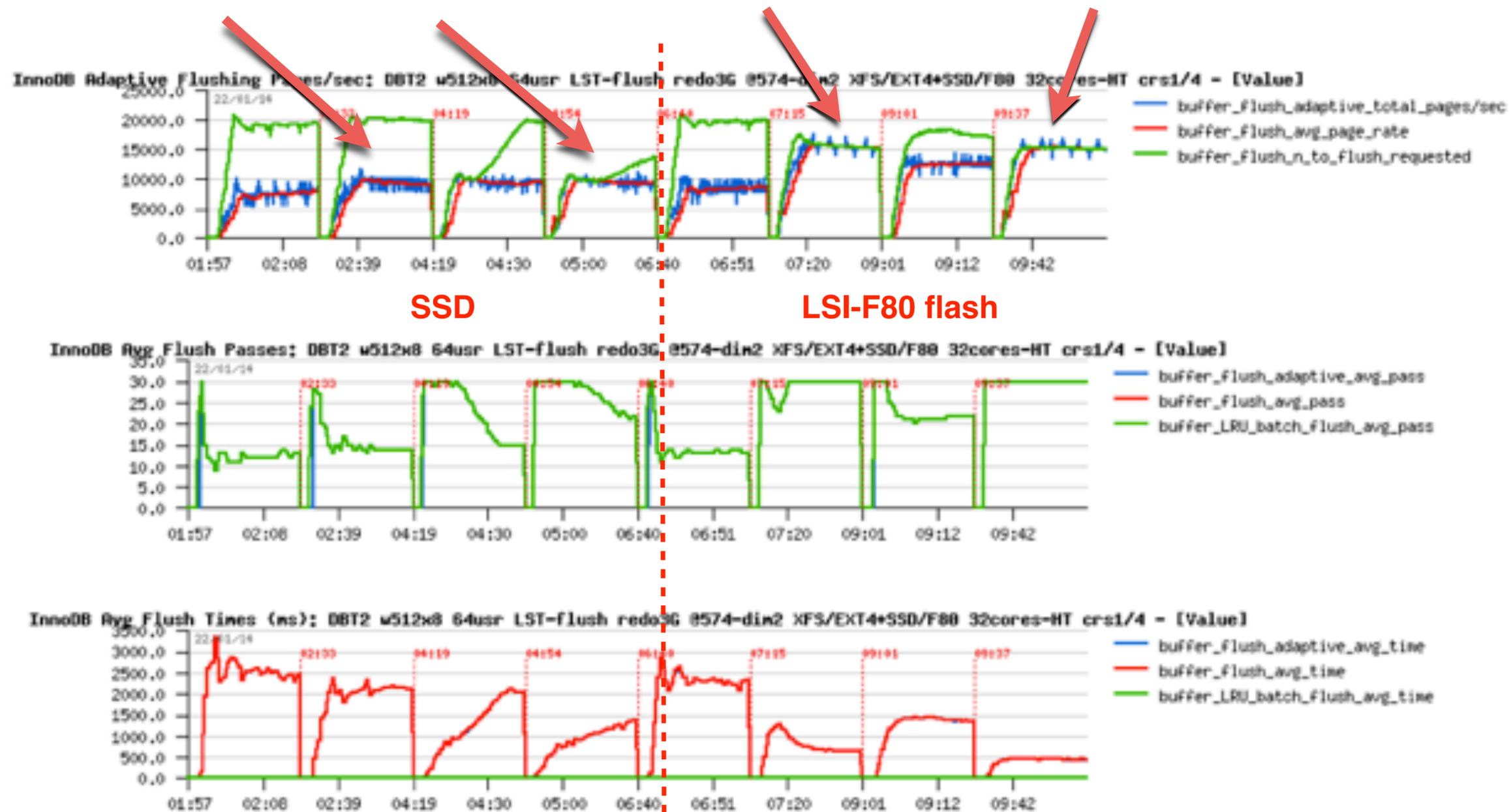
RW IO-Bound : Test your Filesystem before to deploy

- LinkBench 150G workload
 - test cases : “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



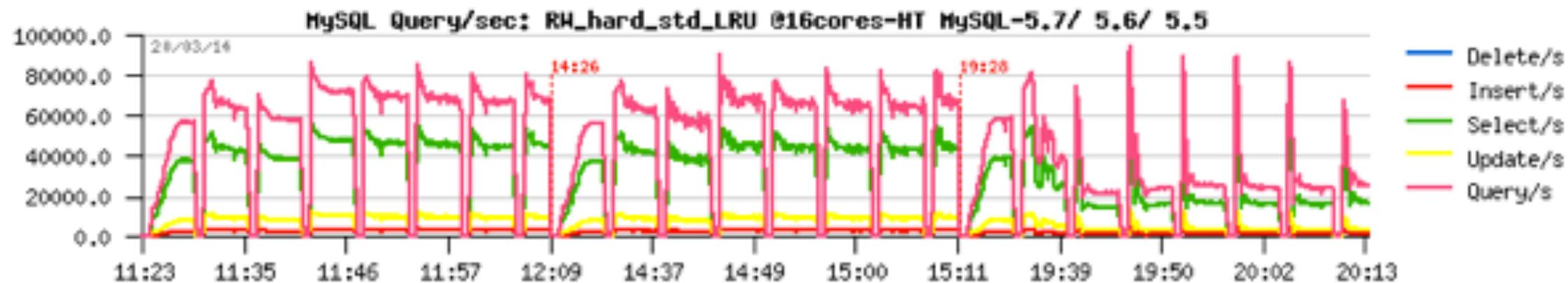
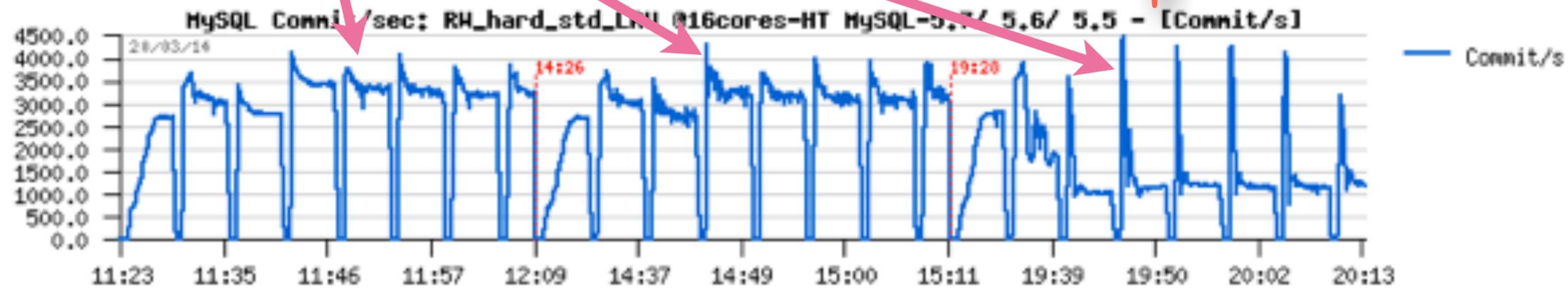
RW IO-bound “Out-of-Memory”

- 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 !!!

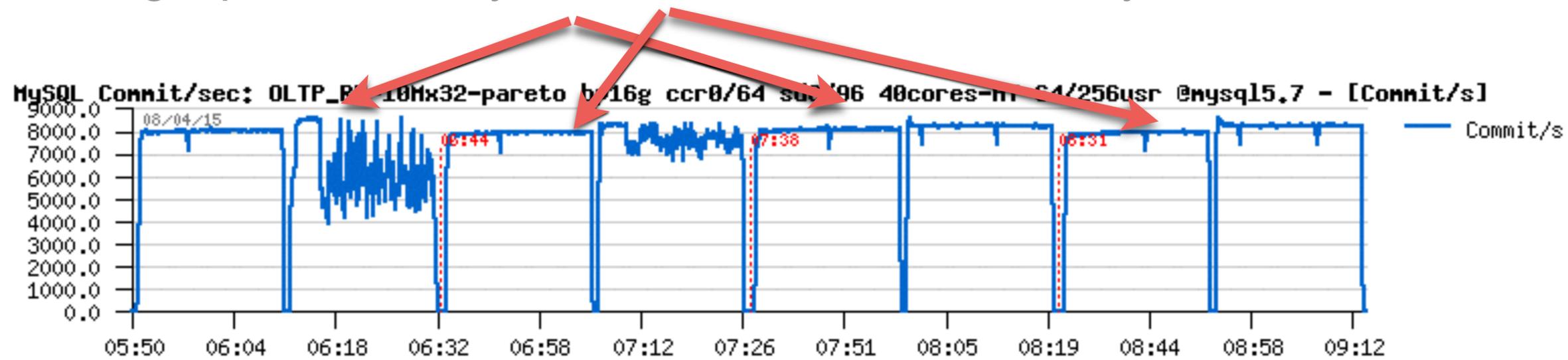


Analyzing OLTP_RW-32x10M Workload @40cores-HT

- Mostly IO-bound (100G database)
 - so, storage layer: Fusion-io flash, EXT4
- Test cases :
 - engines: Percona Server 5.6.23 / MySQL 5.7-rc1 / MariaDB 10.1.4
 - access pattern: pareto / uniform
 - concurrent user sessions: 64, 256
 - Buffer Pool size: 16G (LRU-bound) / 96G (Flushing-bound)
 - LRU depth = 4000
 - IO capacity = 15000
 - IO_DIRECT_NO_FSYNC + native AIO
 - REDO log size = 3 x 1GB
 - InnoDB thread concurrency = 0 / 64
 - InnoDB spin wait delay = 6 / 96
 - ...

OLTP_RW-32x10M-“pareto” Workload @40cores-HT

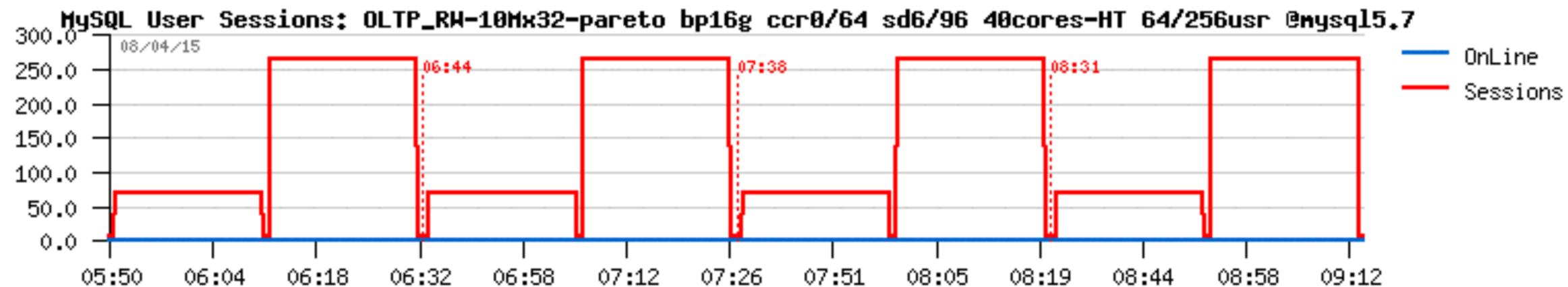
- LRU-bound (BP=16G): Seeking for the most optimal tuning
 - engine: MySQL 5.7
 - tuning: spin wait delay= 6 / 96 + thread concurrency= 0 / 64



concurrency = 0

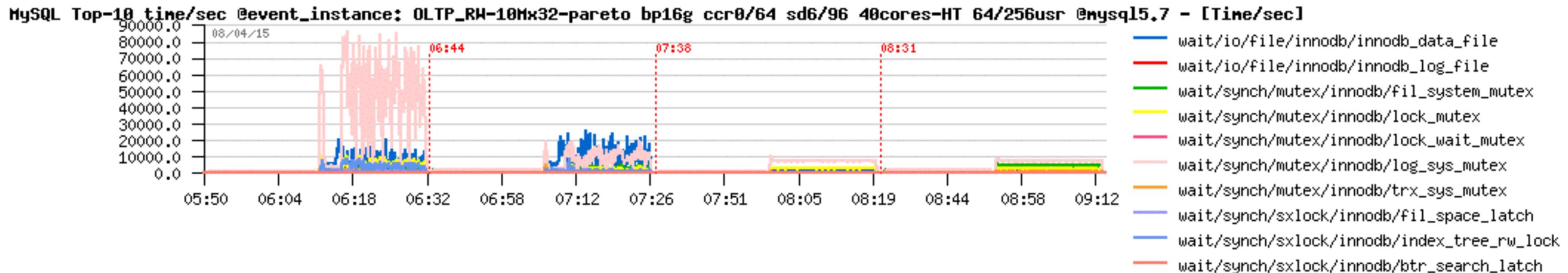
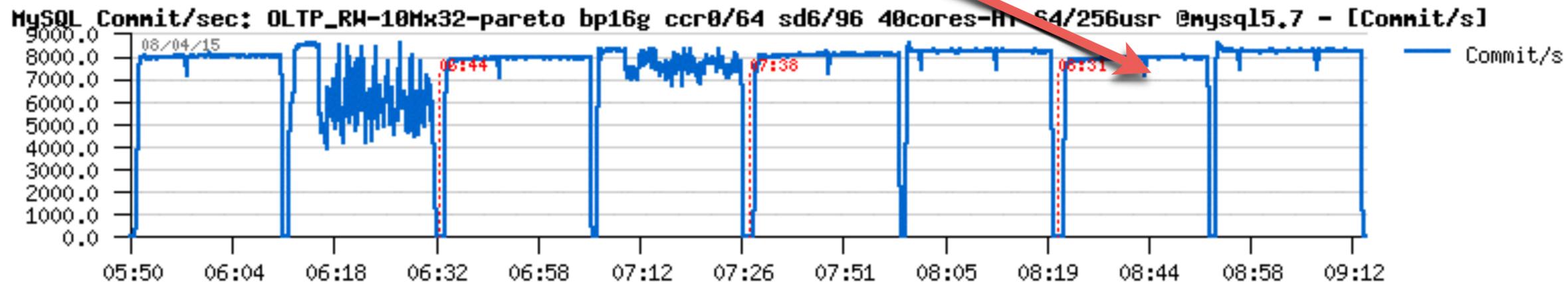
|

concurrency = 64



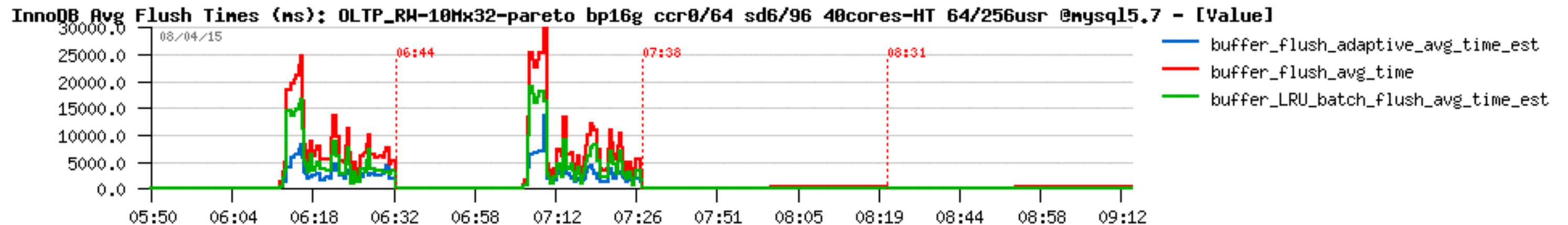
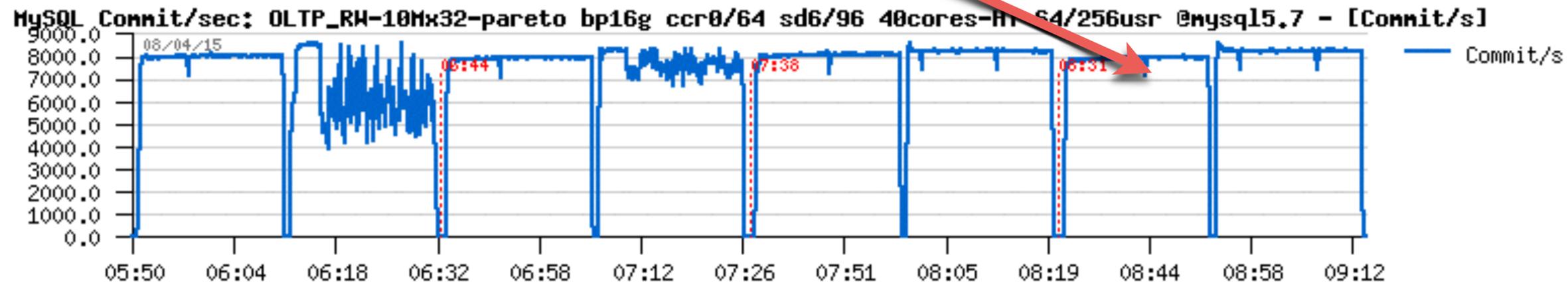
OLTP_RW-32x10M-“pareto” Workload @40cores-HT

- LRU-bound (BP=16G): Seeking for the most optimal tuning
 - engine: MySQL 5.7
 - tuning winner: spin wait delay= 96 + thread concurrency= 64



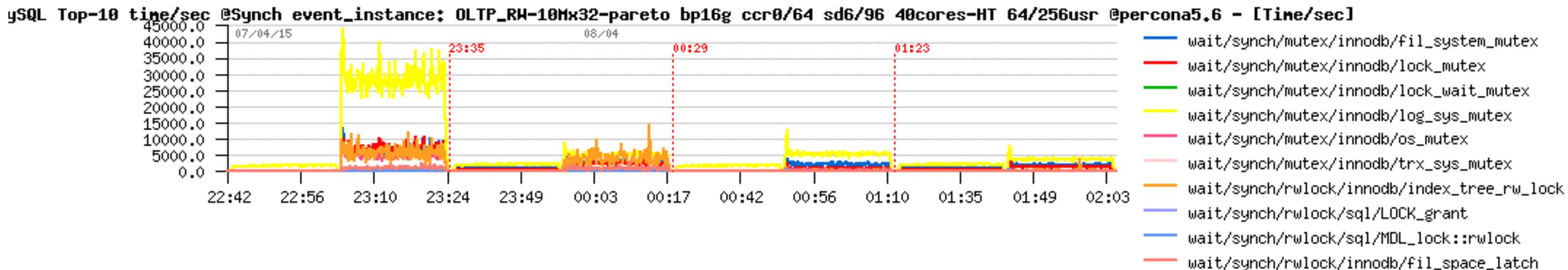
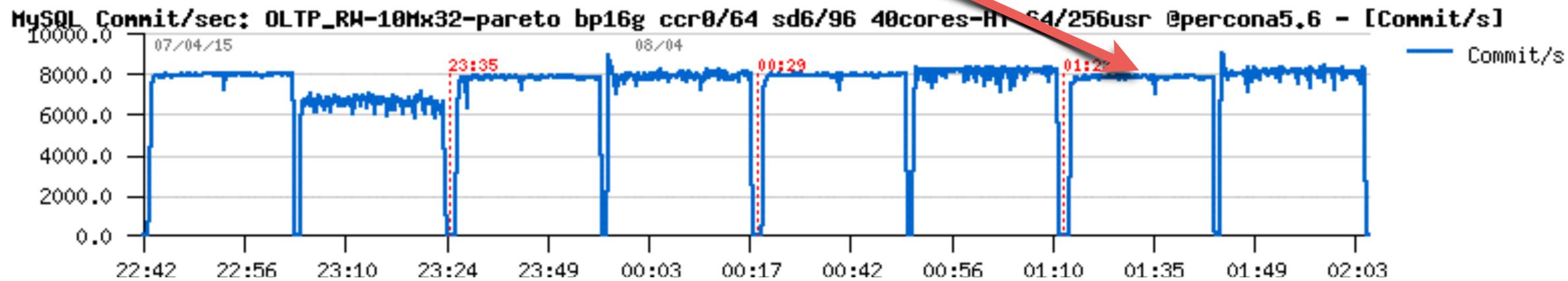
OLTP_RW-32x10M-“pareto” Workload @40cores-HT

- LRU-bound (BP=16G): Seeking for the most optimal tuning
 - engine: MySQL 5.7
 - tuning winner: spin wait delay= 96 + thread concurrency= 64



OLTP_RW-32x10M-“pareto” Workload @40cores-HT

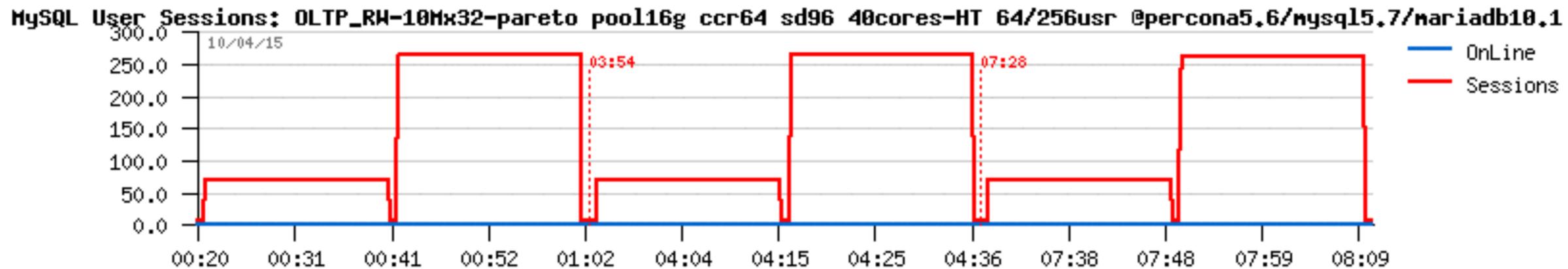
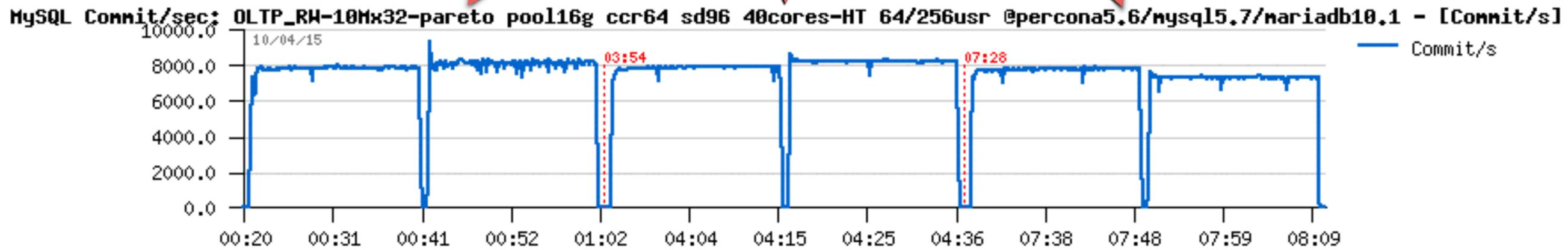
- LRU-bound (BP=16G): Seeking for the most optimal tuning
 - engine: Percona Server 5.6
 - tuning winner: spin wait delay= 96 + thread concurrency= 64



OLTP_RW-32x10M-“pareto” Workload @40cores-HT

- LRU-bound (BP=16G): Final Results

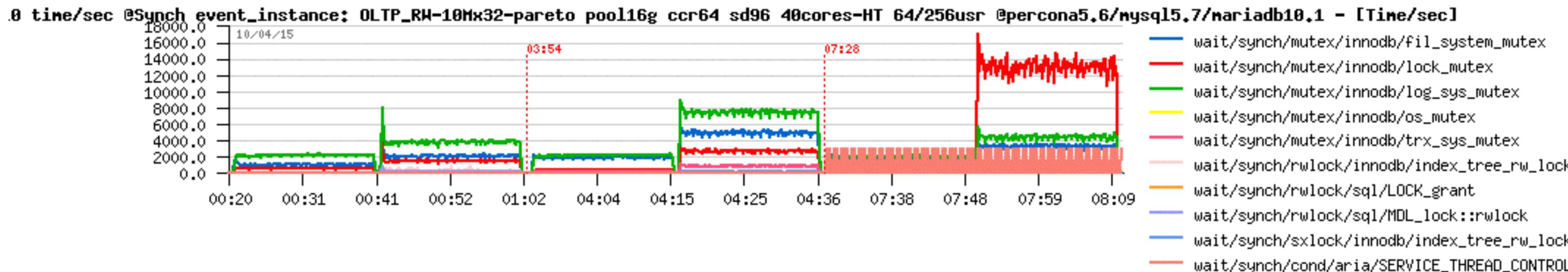
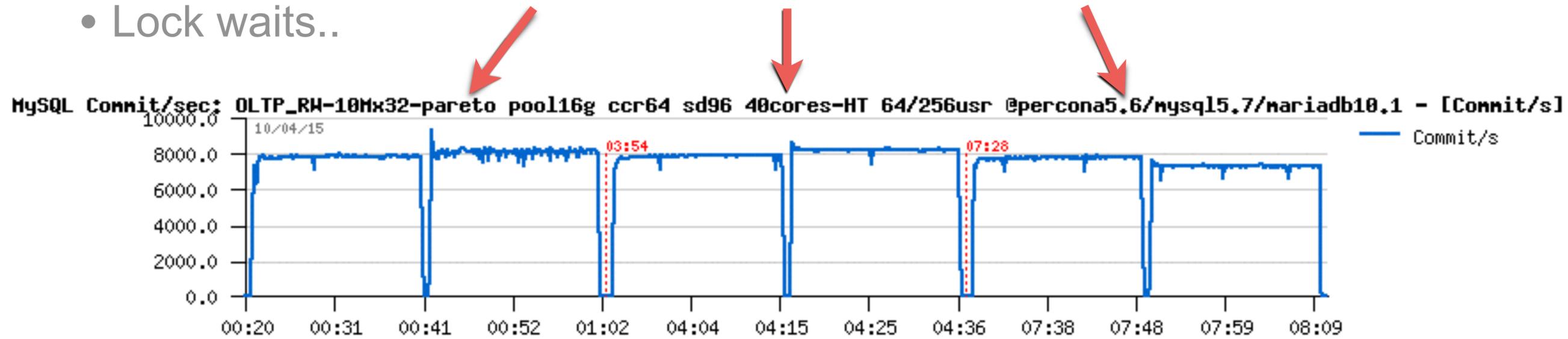
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- TPS: Commit/sec



OLTP_RW-32x10M-“pareto” Workload @40cores-HT

- LRU-bound (BP=16G): Final Results

- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- Lock waits..

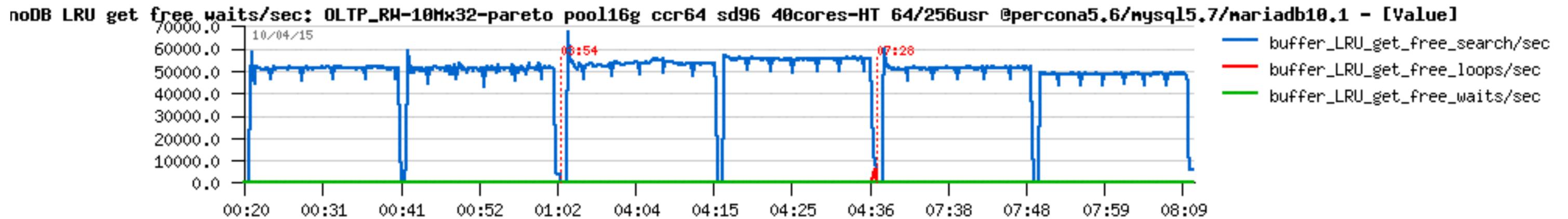
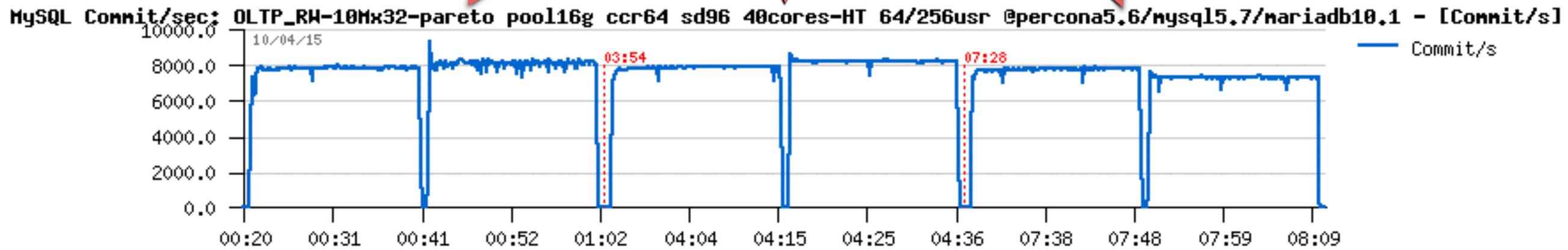


OLTP_RW-32x10M-“pareto” Workload @40cores-HT

- LRU-bound (BP=16G): Final Results

- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1

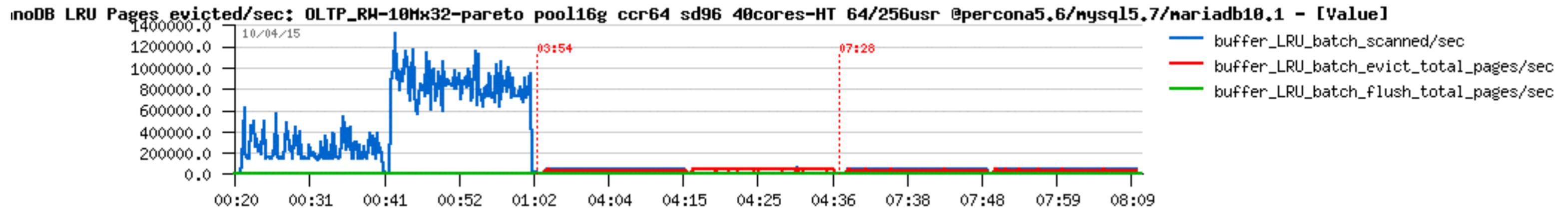
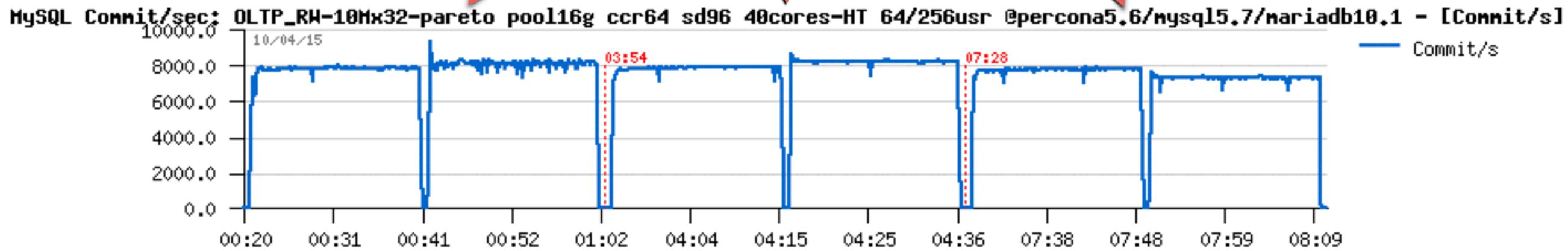
- **50K+ page reads/sec !!!**



OLTP_RW-32x10M-“pareto” Workload @40cores-HT

- LRU-bound (BP=16G): Final Results

- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- page scans (lower is better)..

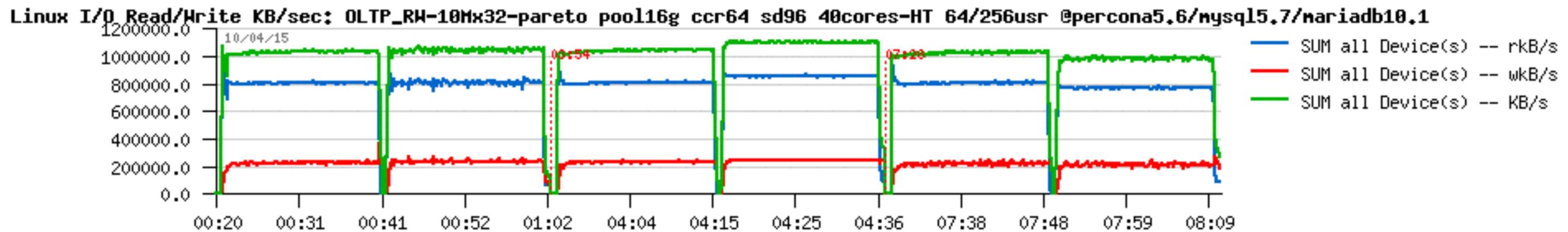
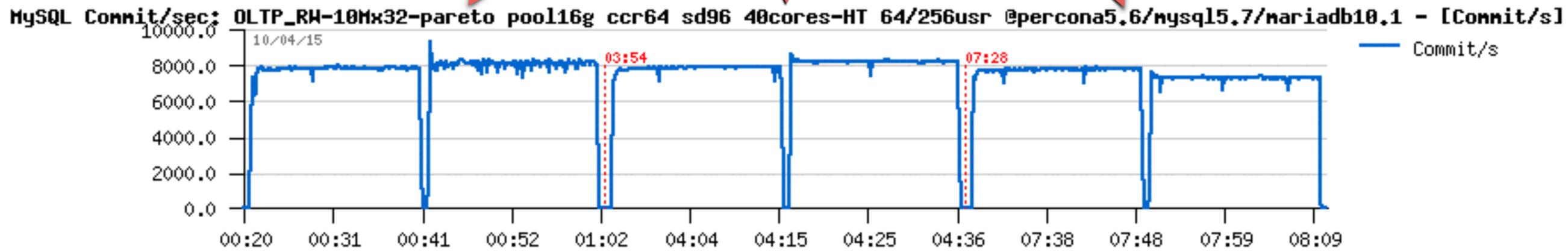


OLTP_RW-32x10M-“pareto” Workload @40cores-HT

- LRU-bound (BP=16G): Final Results

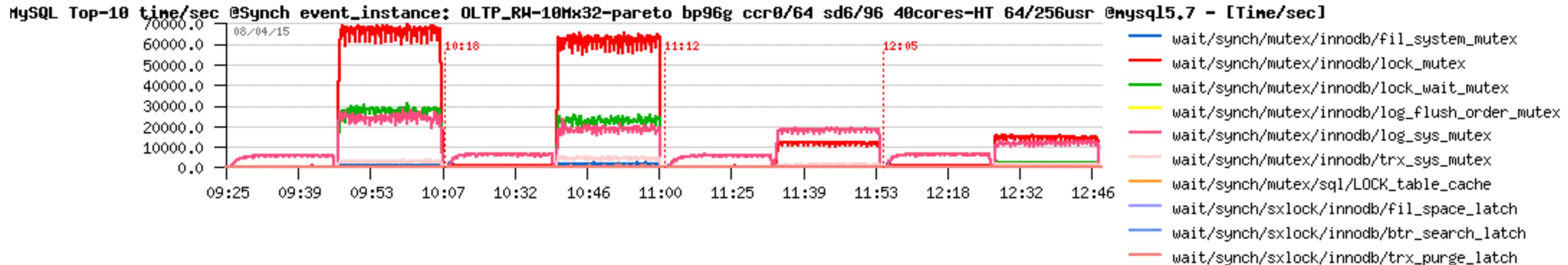
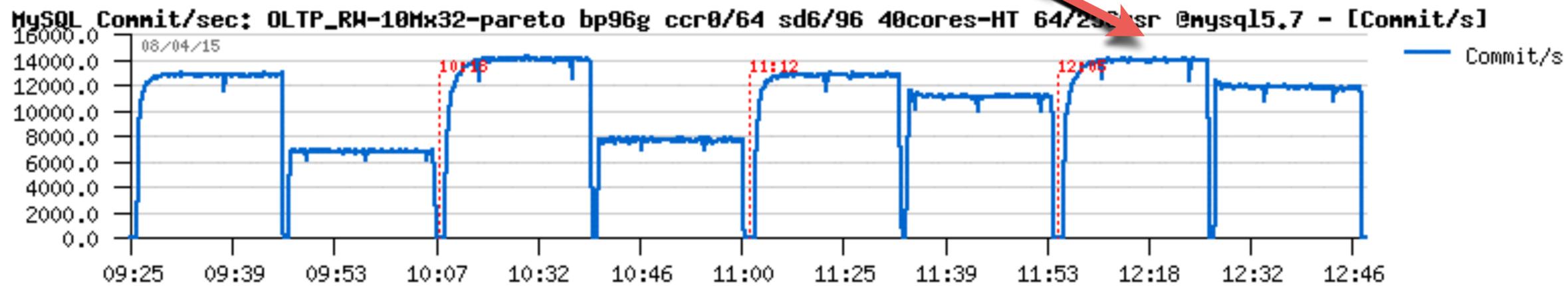
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1

- 1100MB/sec I/O traffic !!!



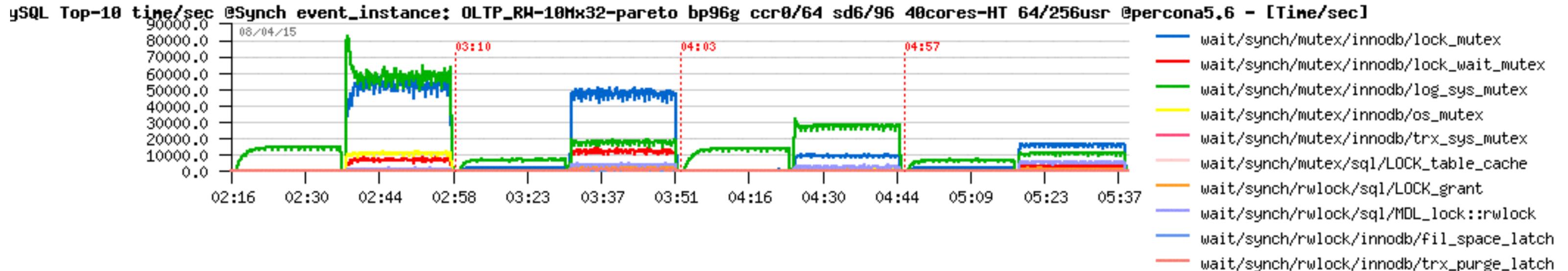
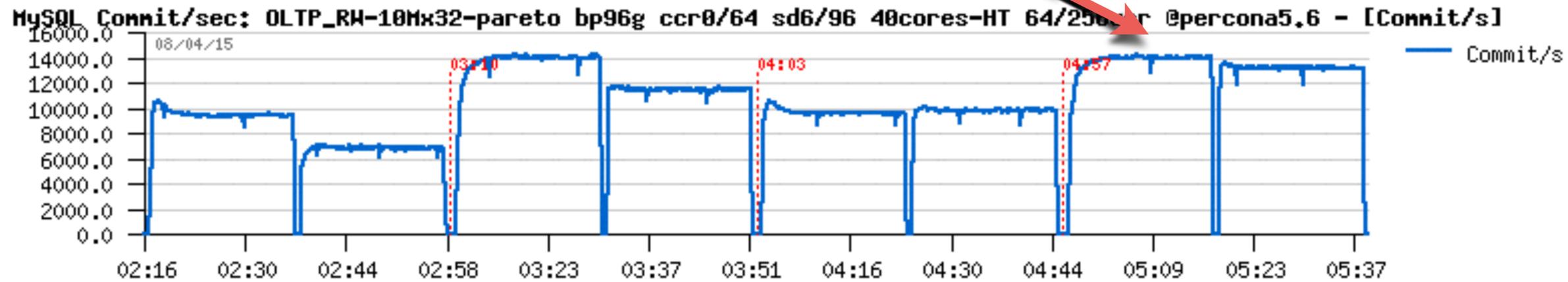
OLTP_RW-32x10M-“pareto” Workload @40cores-HT

- Flushing-bound (BP=96G): Seeking for the most optimal tuning
 - engine: MySQL 5.7
 - tuning winner: spin wait delay= 96 + thread concurrency= 64



OLTP_RW-32x10M-“pareto” Workload @40cores-HT

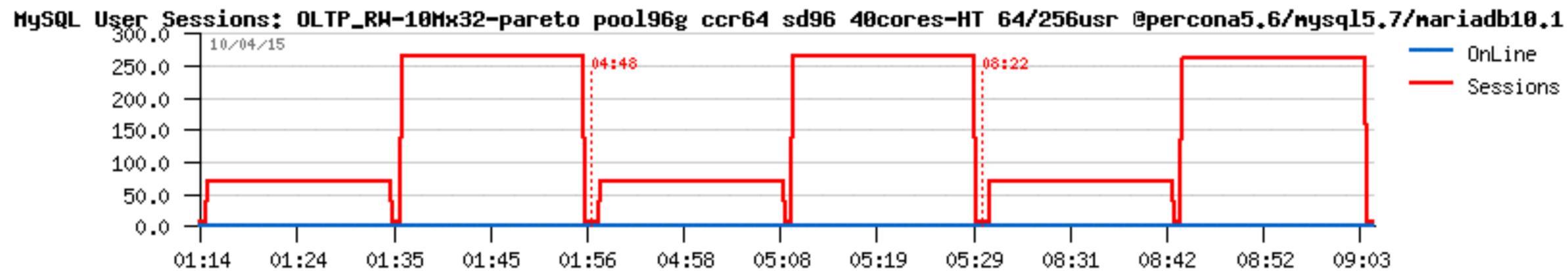
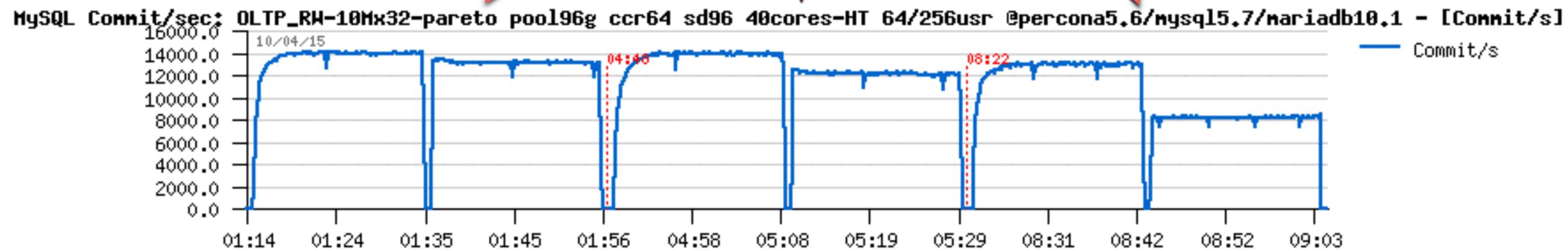
- Flushing-bound (BP=96G): Seeking for the most optimal tuning
 - engine: Percona Server 5.6
 - tuning winner: spin wait delay= 96 + thread concurrency= 64



OLTP_RW-32x10M-“pareto” Workload @40cores-HT

- Flushing-bound (BP=96G): Final Results

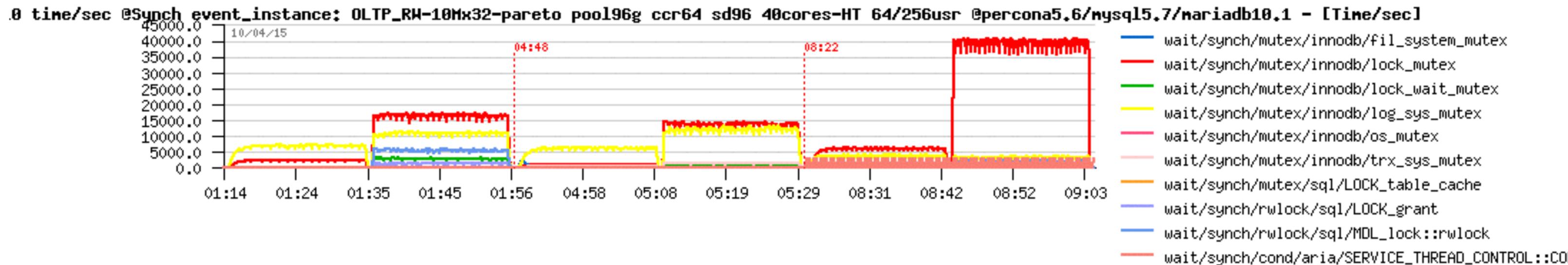
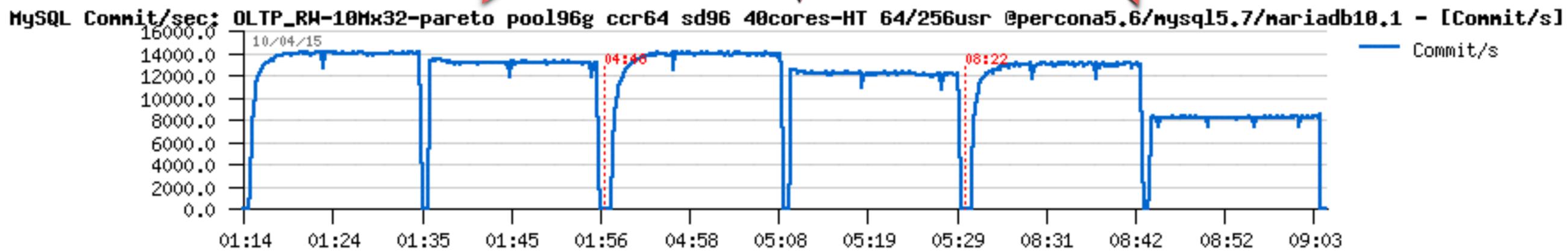
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- TPS: Commit/sec



OLTP_RW-32x10M-“pareto” Workload @40cores-HT

- Flushing-bound (BP=96G): Final Results

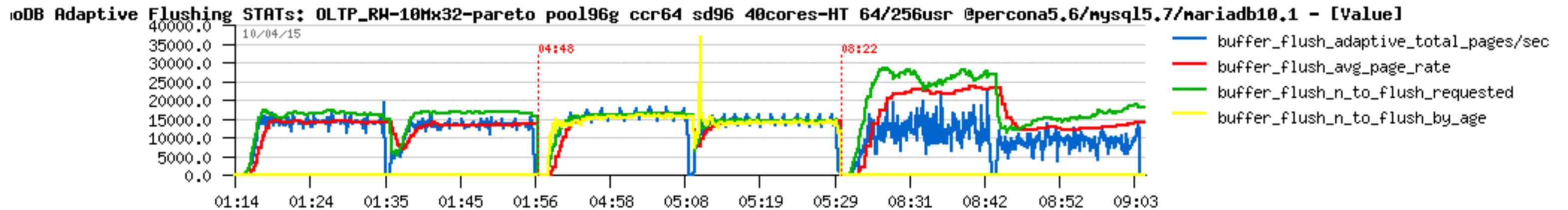
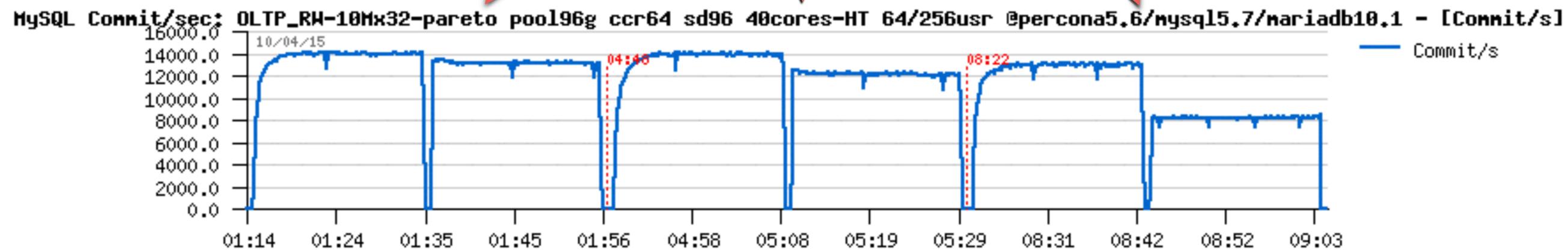
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- Lock waits



OLTP_RW-32x10M-“pareto” Workload @40cores-HT

- Flushing-bound (BP=96G): Final Results

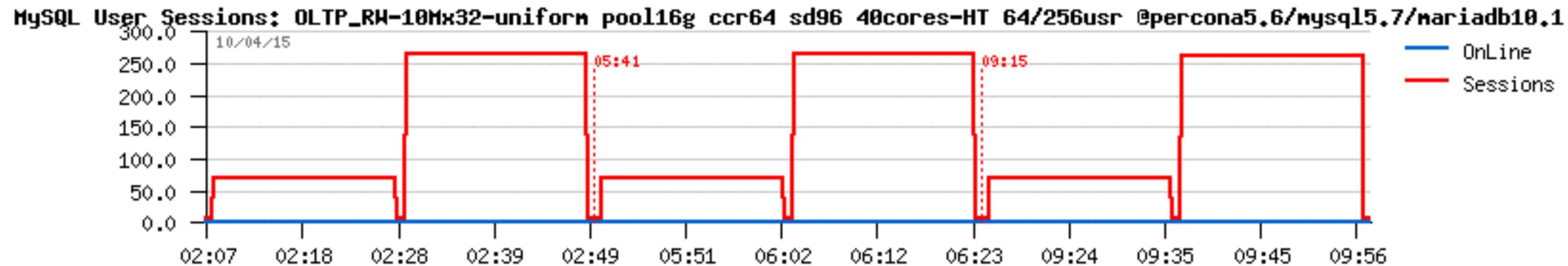
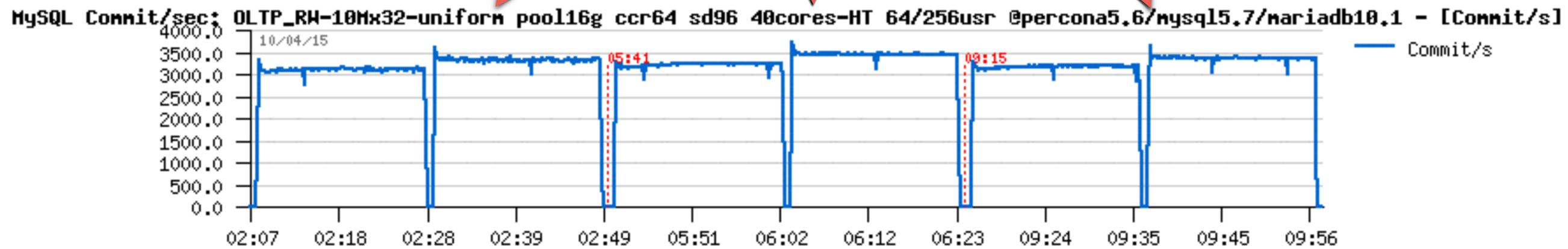
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- Flushing activity



OLTP_RW-32x10M-“uniform” Workload @40cores-HT

- LRU-bound (BP=16G): Final Results

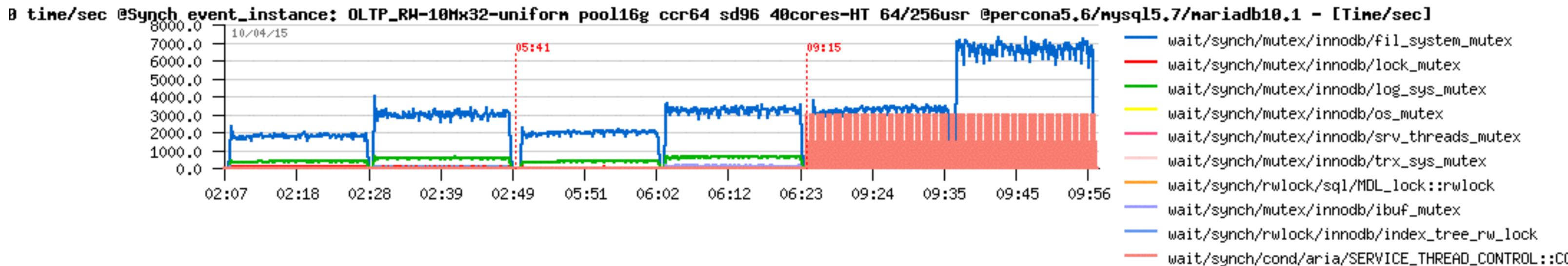
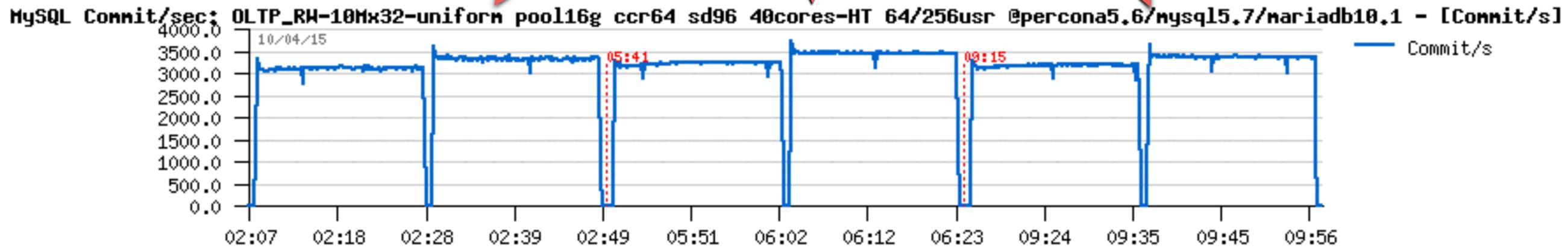
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- TPS: Commit/sec



OLTP_RW-32x10M-“uniform” Workload @40cores-HT

- LRU-bound (BP=16G): Final Results

- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- Lock waits: fil_sys mutex contention is blocking higher reads..

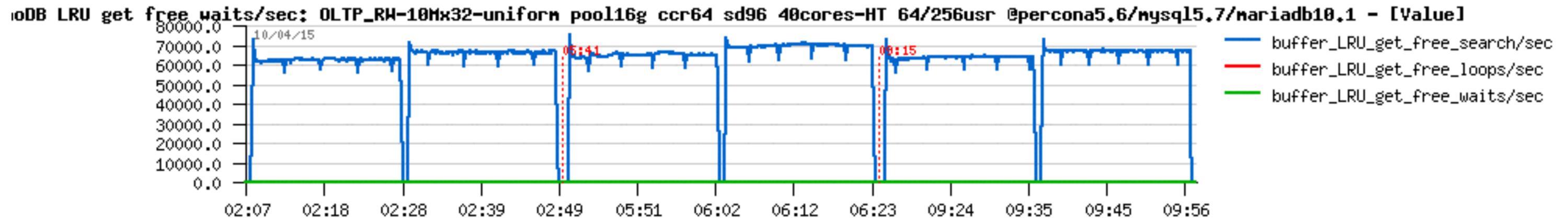
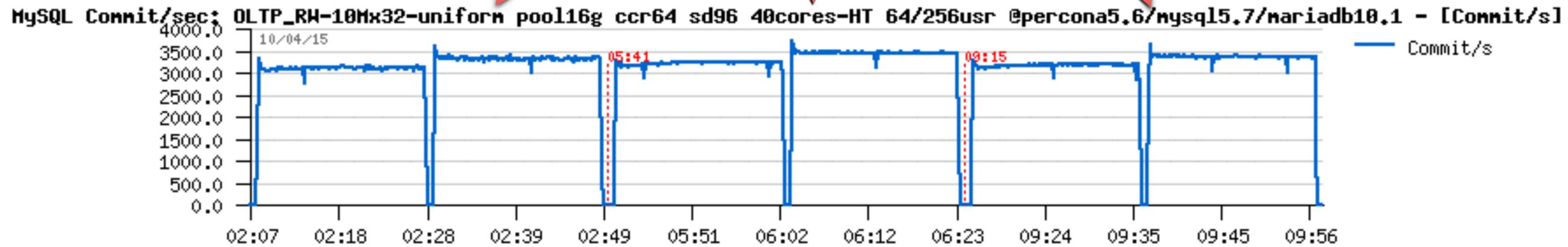


OLTP_RW-32x10M-“uniform” Workload @40cores-HT

- LRU-bound (BP=16G): Final Results

- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1

- 70K+ page reads/sec !!!

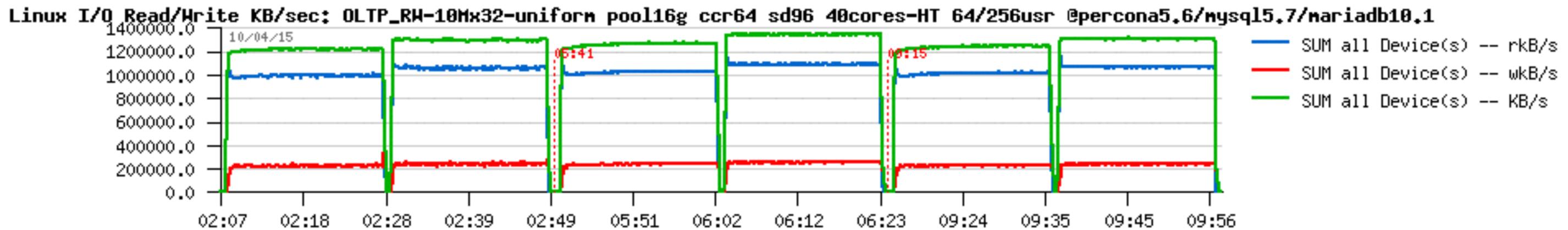
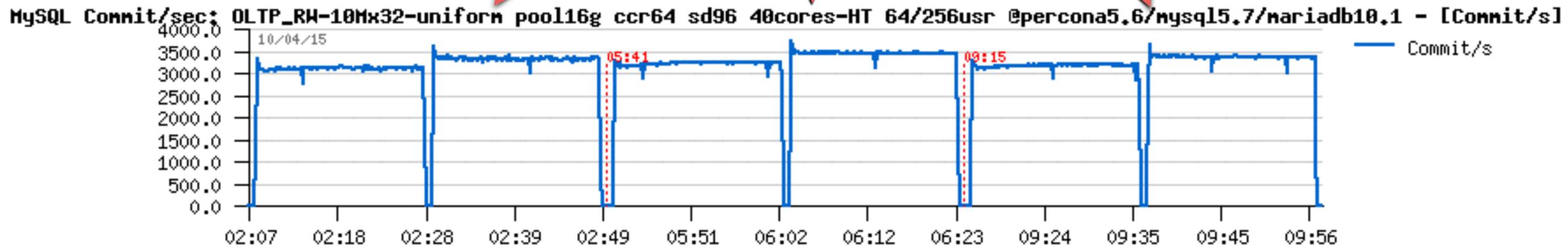


OLTP_RW-32x10M-“uniform” Workload @40cores-HT

- LRU-bound (BP=16G): Final Results

- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1

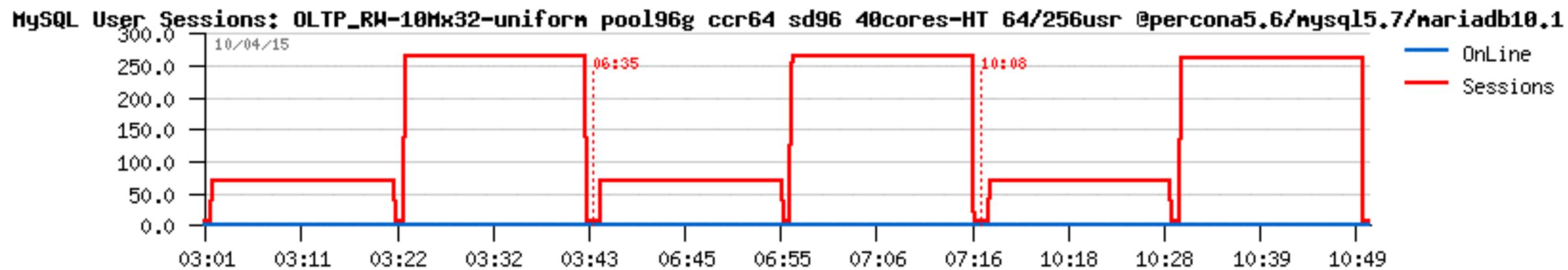
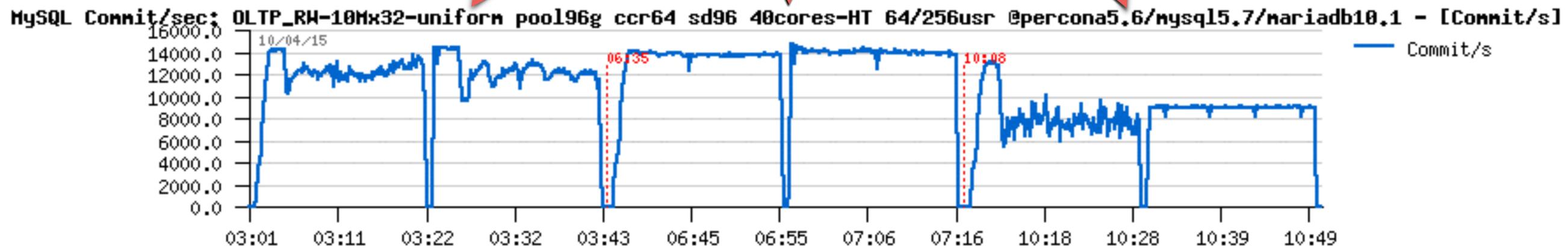
- **1400 MB/sec I/O traffic !!!**



OLTP_RW-32x10M-“uniform” Workload @40cores-HT

- Flushing-bound (BP=96G): Final Results

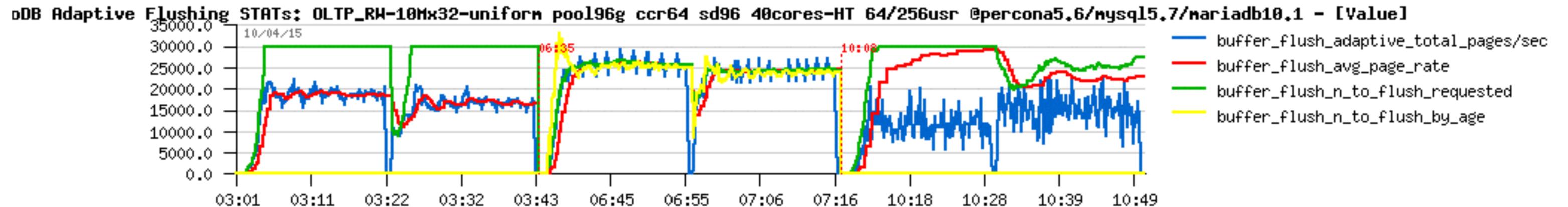
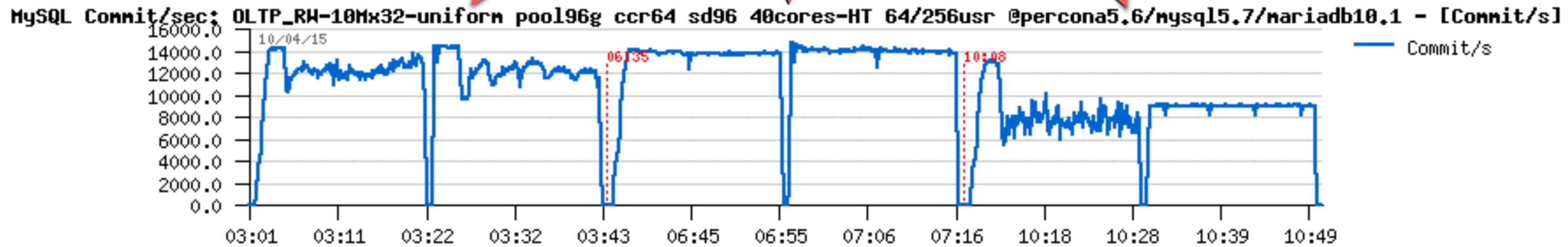
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- TPS: Commit/sec



OLTP_RW-32x10M-“uniform” Workload @40cores-HT

- Flushing-bound (BP=96G): Final Results

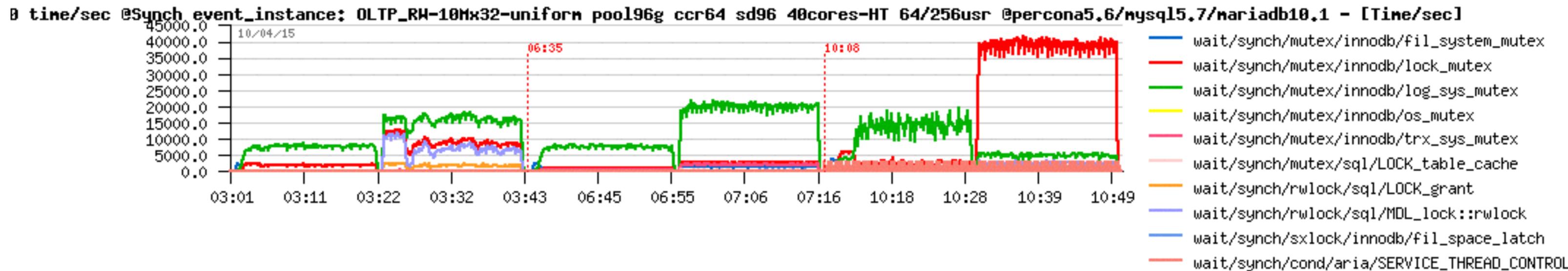
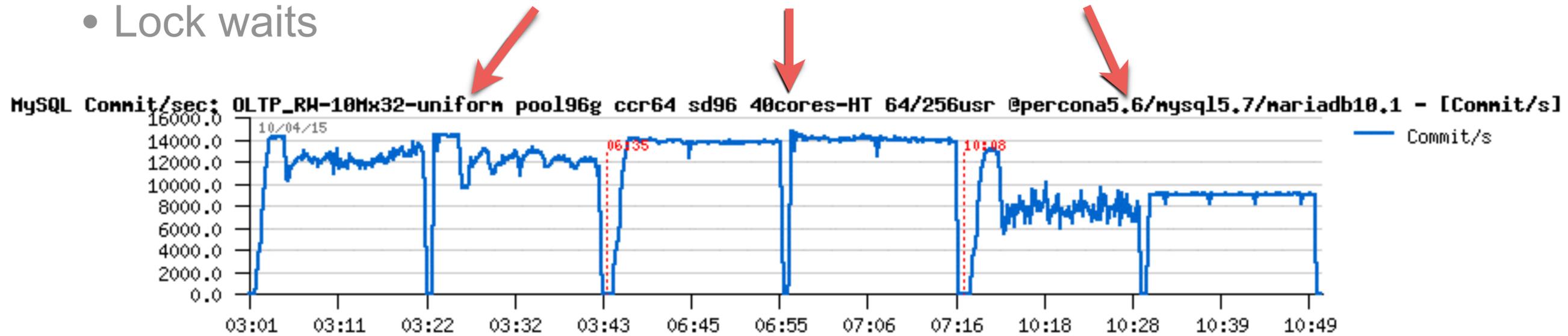
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- MySQL 5.7 Improved InnoDB Flushing in action!!!



OLTP_RW-32x10M-“uniform” Workload @40cores-HT

- Flushing-bound (BP=96G): Final Results

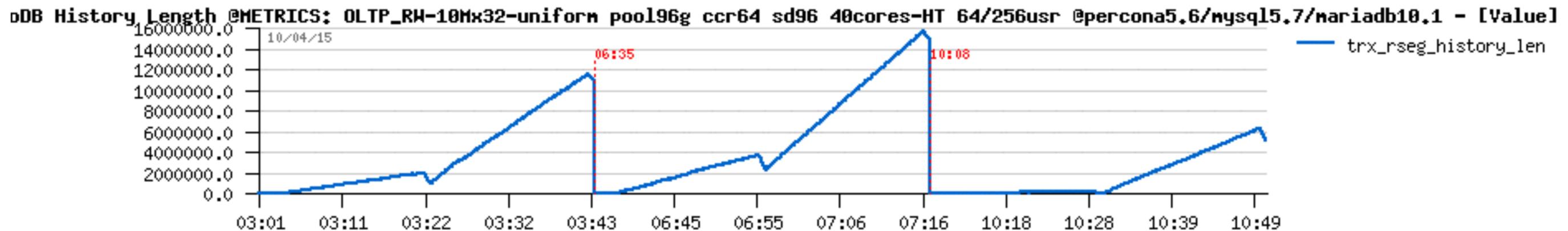
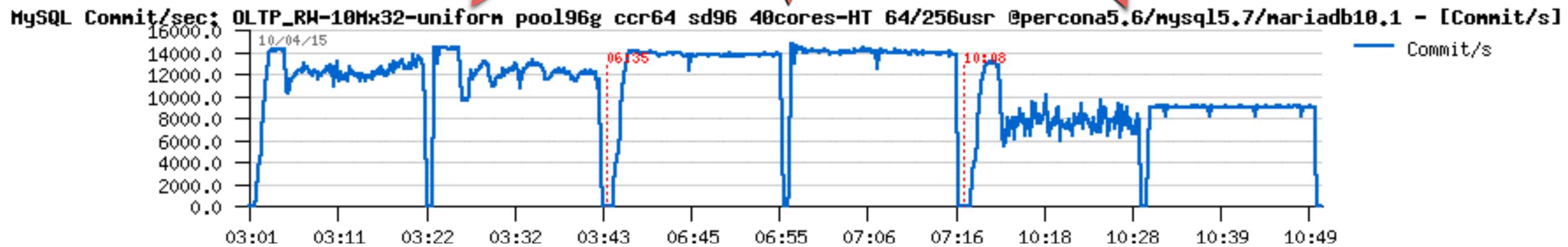
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- Lock waits



OLTP_RW-32x10M-“uniform” Workload @40cores-HT

- Flushing-bound (BP=96G): Final Results

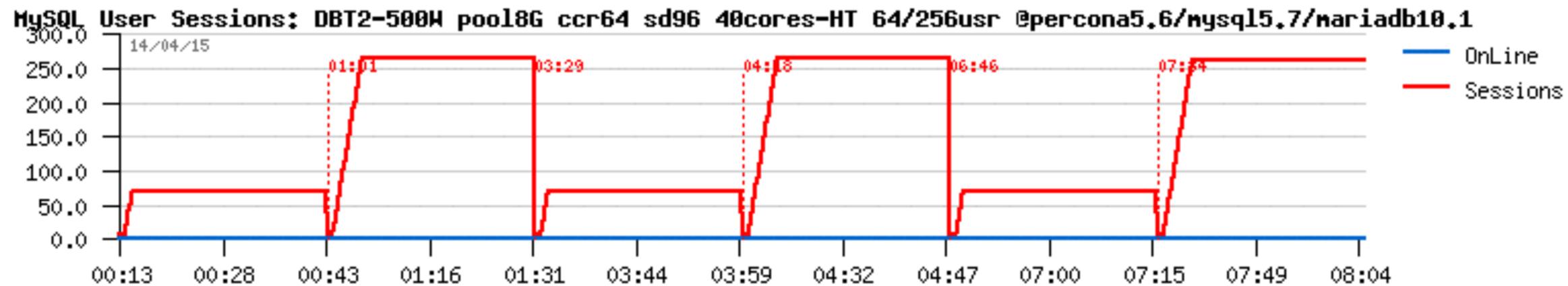
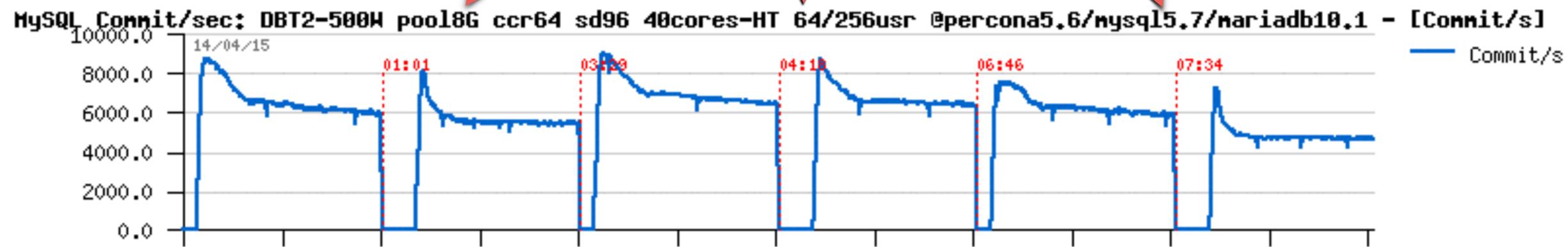
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- Purge remains a big problem!..



DBT2-500W Workload @40cores-HT

- LRU-bound (BP=8G): Final Results

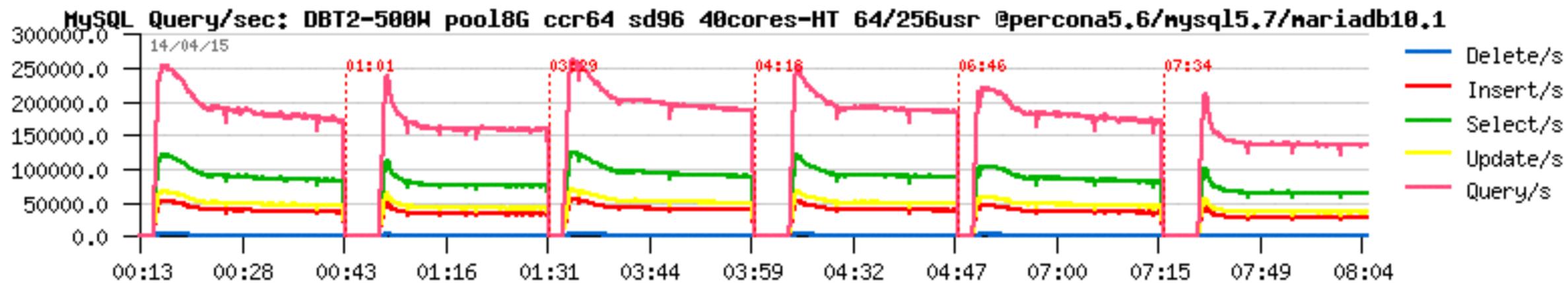
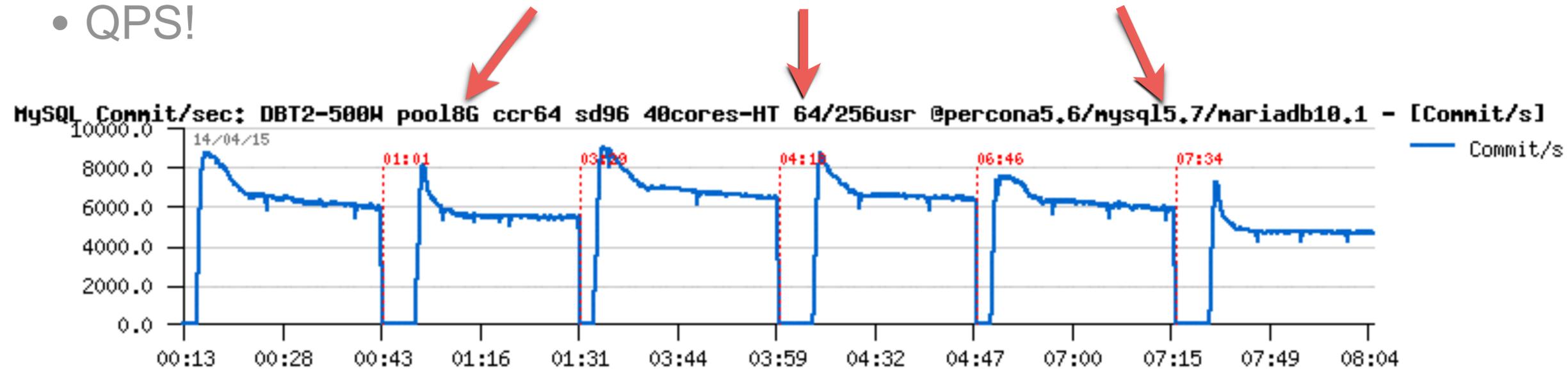
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- TPS: Commit/sec



DBT2-500W Workload @40cores-HT

- LRU-bound (BP=8G): Final Results

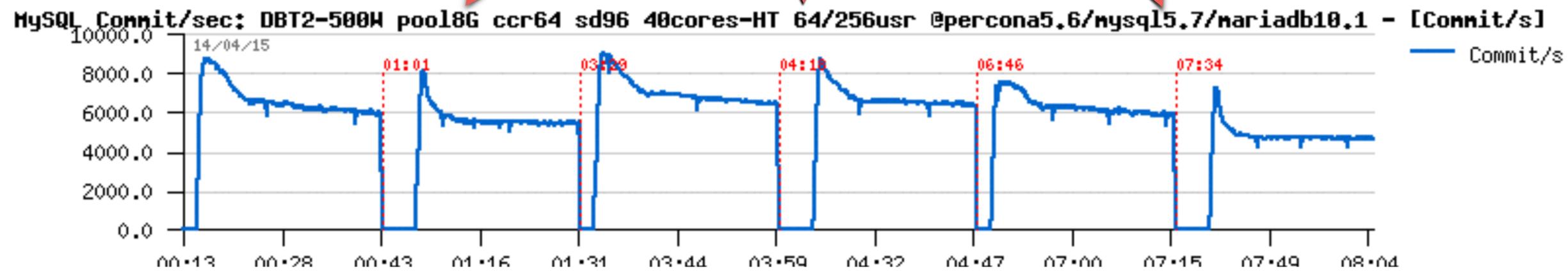
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- QPS!



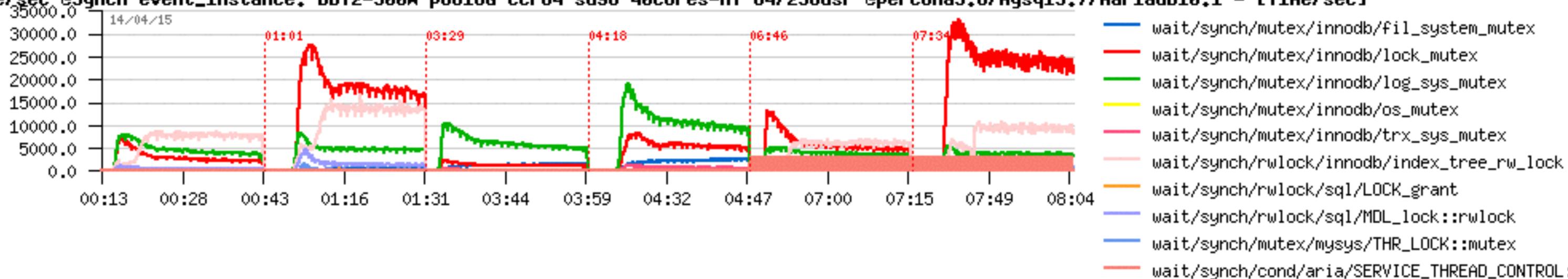
DBT2-500W Workload @40cores-HT

- LRU-bound (BP=8G): Final Results

- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- Lock waits: index lock impact...



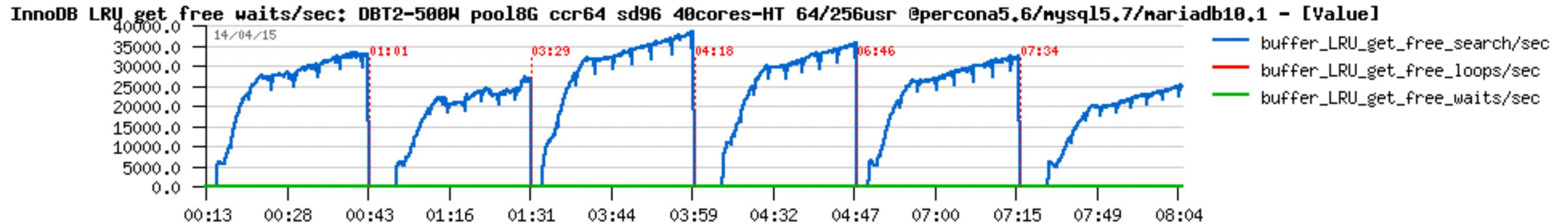
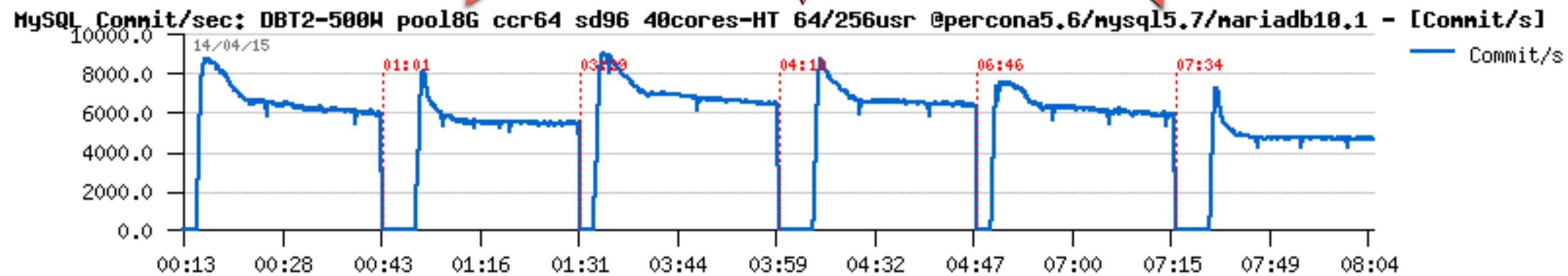
L Top-10 time/sec @Synch event_instance: DBT2-500W pool8G ccr64 sd96 40cores-HT 64/256usr @percona5.6/mysql5.7/mariadb10.1 - [Time/sec]



DBT2-500W Workload @40cores-HT

- LRU-bound (BP=8G): Final Results

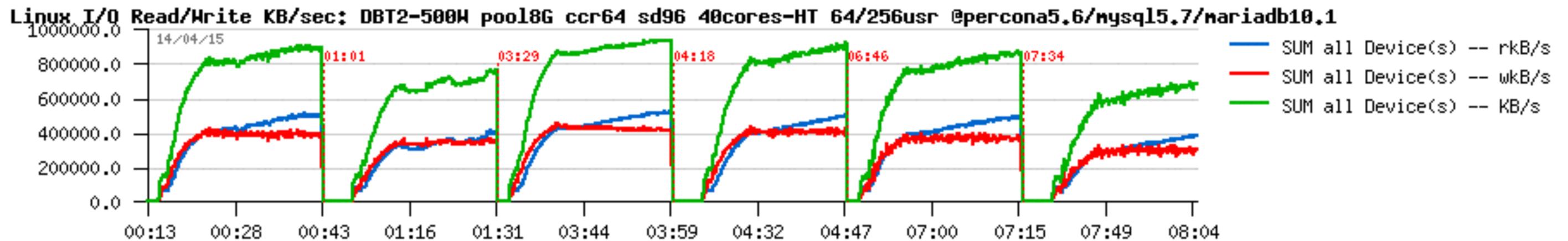
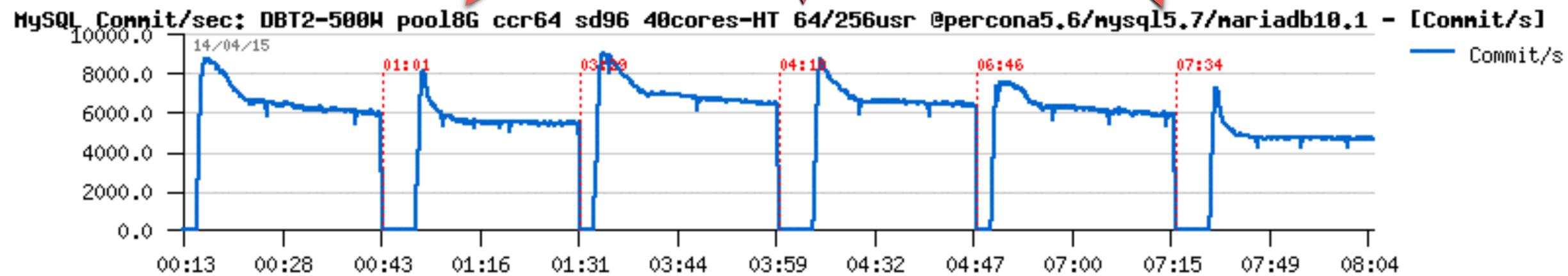
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- up to 40K page reads/sec rate



DBT2-500W Workload @40cores-HT

- LRU-bound (BP=8G): Final Results

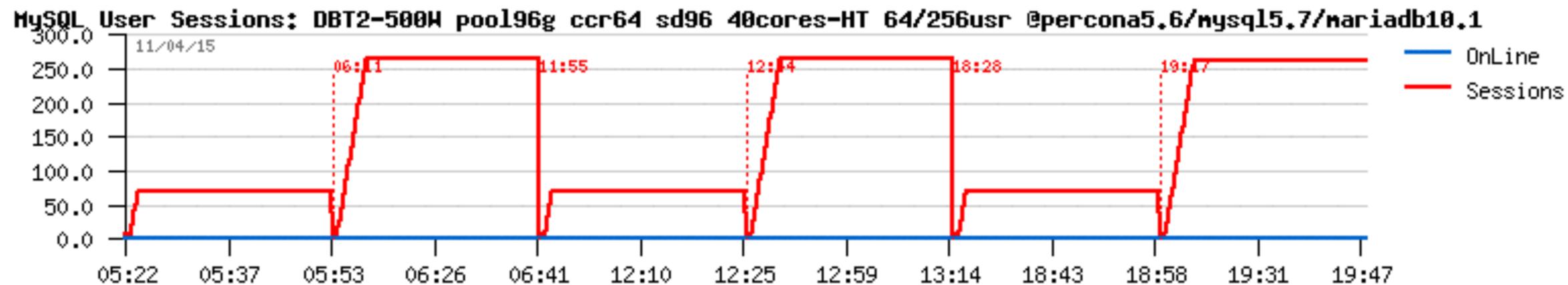
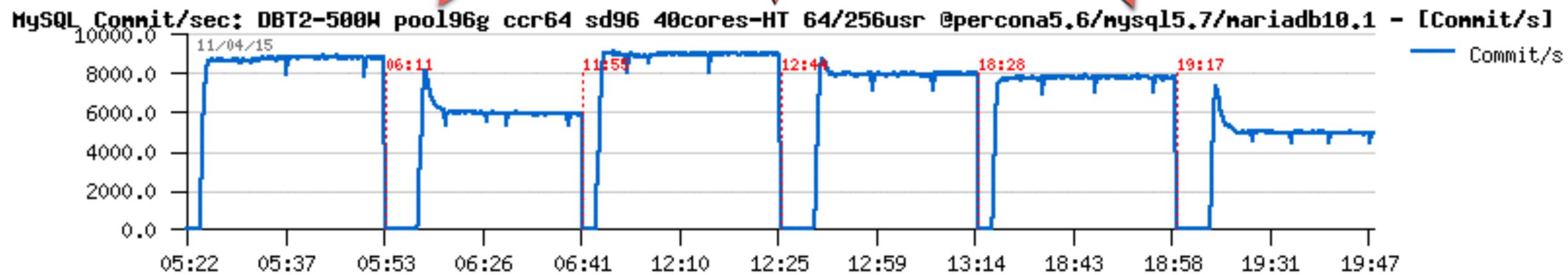
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- 900MB/sec I/O traffic



DBT2-500W Workload @40cores-HT

- Flushing-bound (BP=96G): Final Results

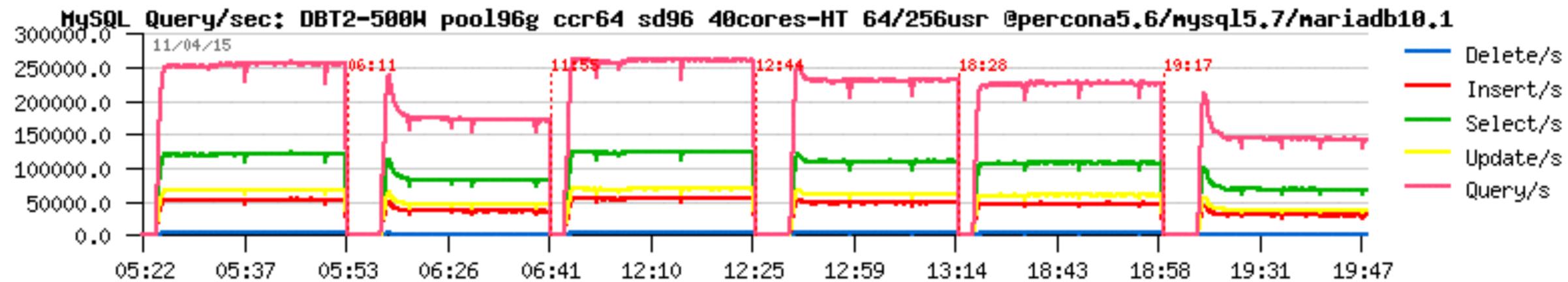
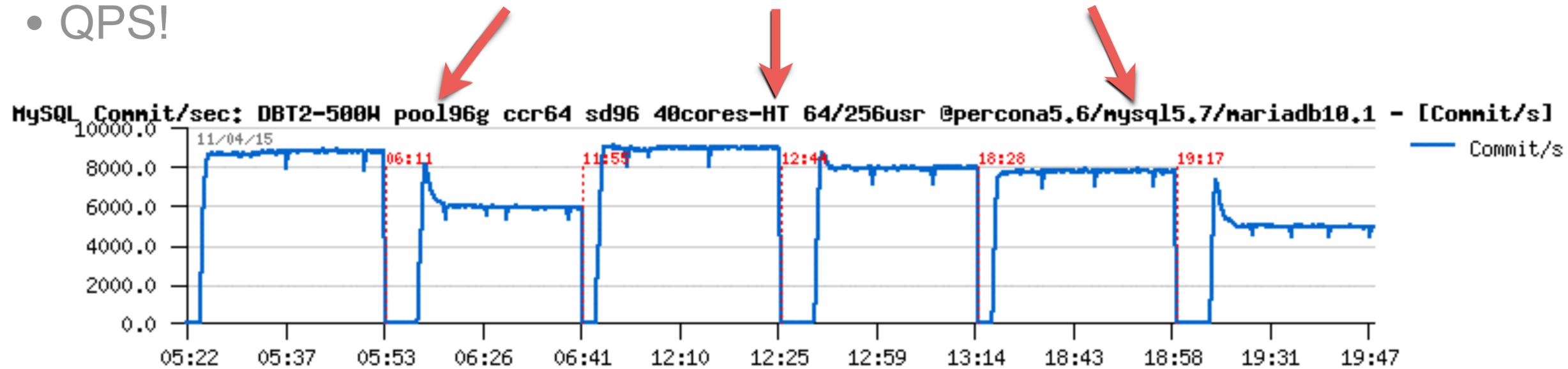
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- TPS: Commit/sec



DBT2-500W Workload @40cores-HT

- Flushing-bound (BP=96G): Final Results

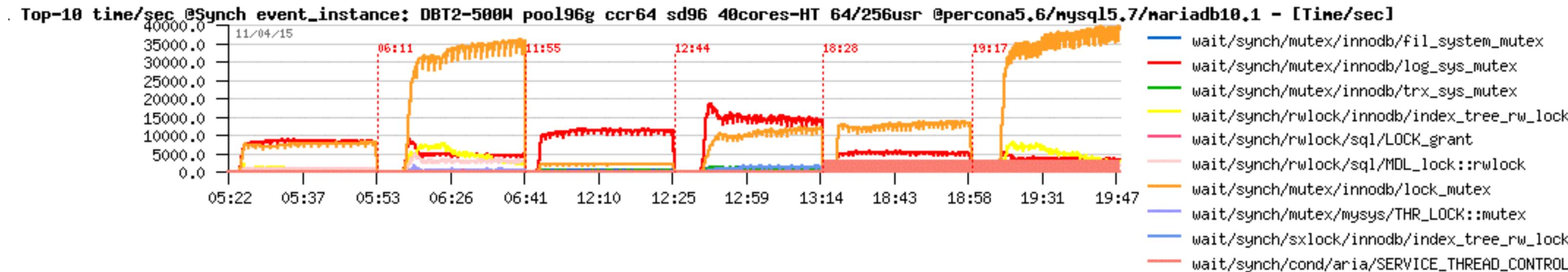
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- QPS!



DBT2-500W Workload @40cores-HT

- Flushing-bound (BP=96G): Final Results

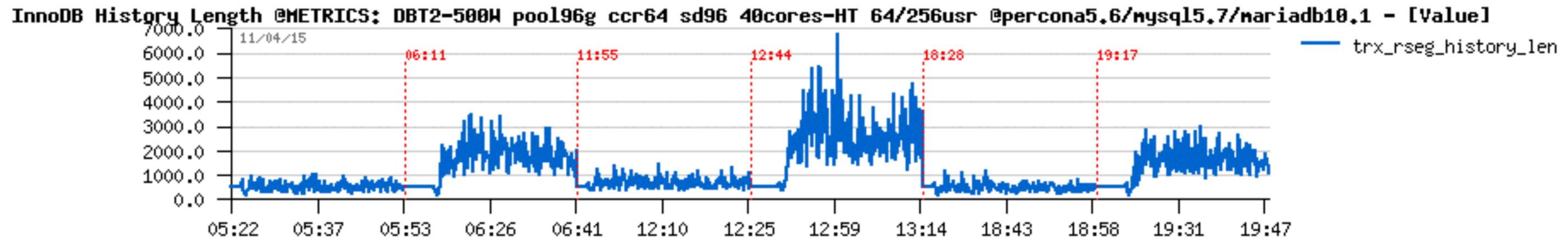
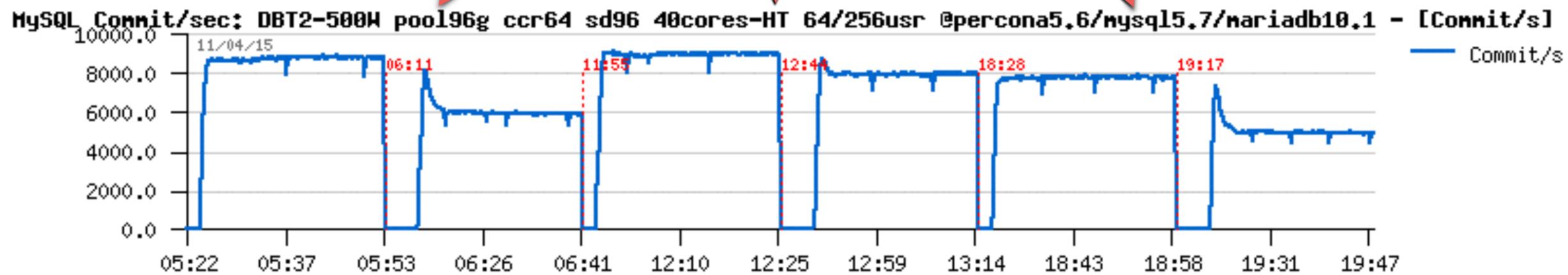
- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- Lock waits: lock_sys_mutex impact...



DBT2-500W Workload @40cores-HT

- Flushing-bound (BP=96G): Final Results

- engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
- No problem with Purge on this workload..



**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! :-)

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.