MySQL

ORACLE

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
- As well may be not exactly in the proposed order ;-)

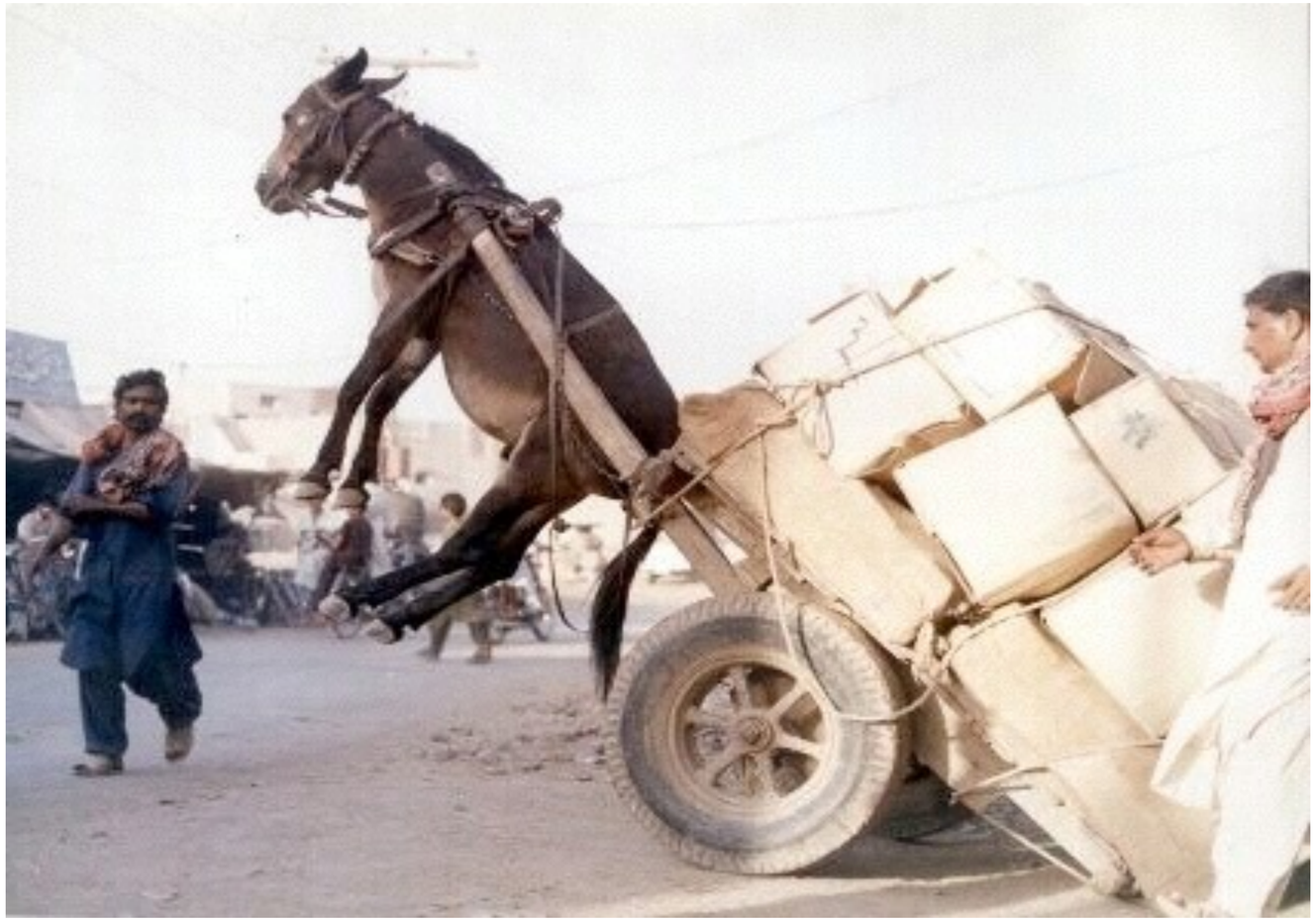
Why Benchmarking MySQL ?...

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..

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 Best Practice #1 is... ???..

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

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

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

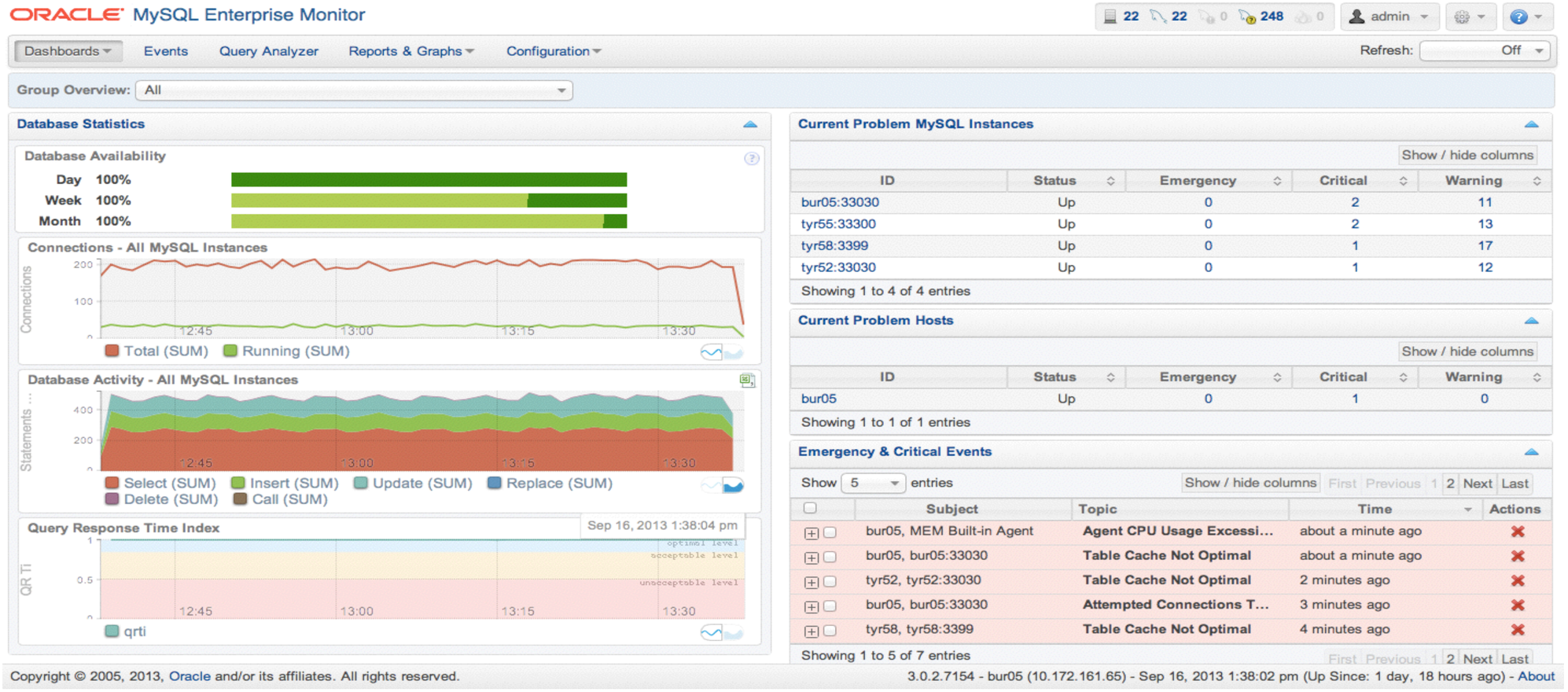
USE YOUR BRAIN !!!... ;-)

Monitoring is THE MUST !

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

MySQL Enterprise Monitor

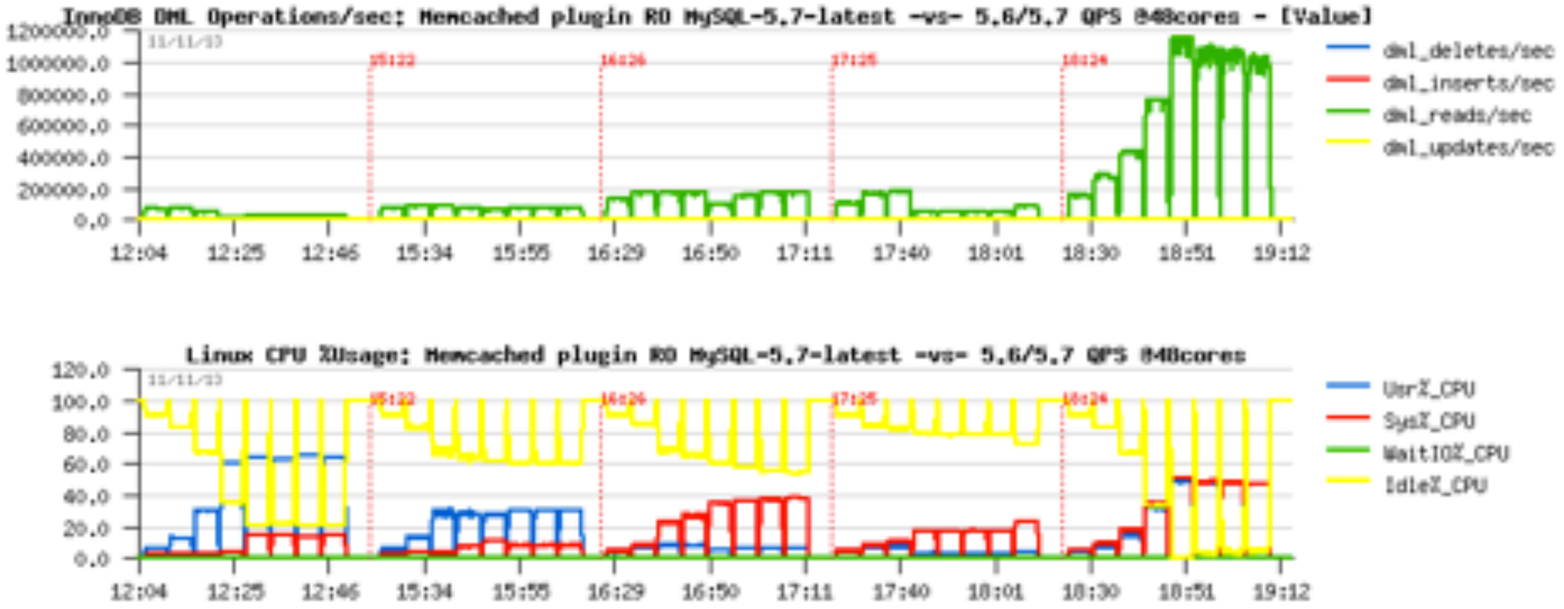
- Fantastic tool!
  - Did you already try it?.. Did you see it live?..



ORACLE

Other Monitoring Tools

- Cacti, Zabbix, Nagios, Solarwinds, 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 Word about Monitoring…

- **always** validate the impact of your Monitoring on your Production ;-)
- taking 1sec measurements is fine, except :
  - if it's eating 100% CPU time on one or more CPU cores..
  - reducing your network traffic / latency..
  - eats your RAM, etc.
- avoid to be too much intrusive on MySQL/InnoDB internals..
  - you may easily create an additional overhead
  - as well you may add artificial locks on your workflow
    - ex: in 5.6 run in loop "show processlist", etc..
- well, think about what you're doing (#1 best practice once again ;-))

Benchmarking & Tuning

- there is no Benchmarking without Tuning ;-)
- as there is no Tuning without Benchmarking ;-)
- depending on the MySQL version :
  - some things you “may tune”
  - some things you “may just accept” ;-)
  - (e.g. you need 5.6 to have binlog group commit, etc.)
- so, you need to have a clear idea about :
  - which situation you can always solve by tuning, so no worry..
  - which situation you may only avoid, so have to consider and take care about..
  - don’t be creating artificial limitations (e.g. if 32GB REDO is allowed - use it!)
  - be sure what is really important for you!

The following materials are about…

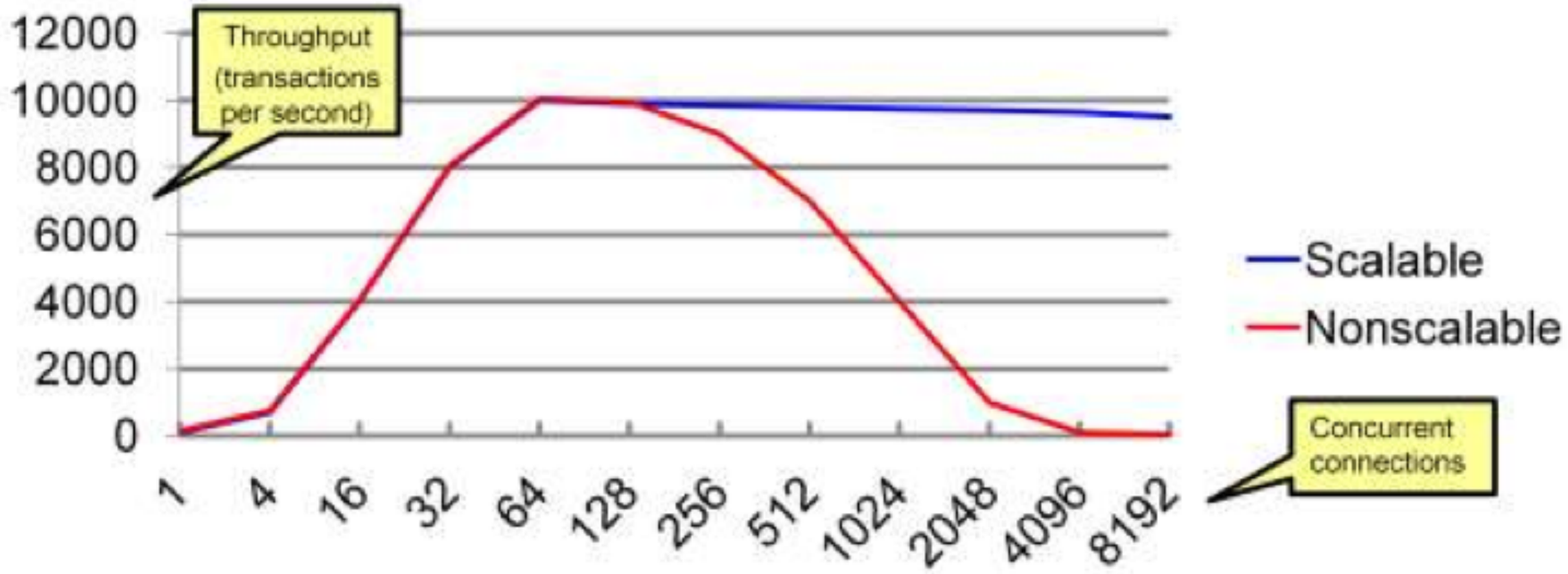
- Single MySQL Instance Performance & Scalability
  - single HW host
  - no replication
  - just to understand how far MySQL Server may scale..
  - what are the limits
  - what to care about ahead
  - which situations are absolutely to avoid..

Why Scalability ?..

- CPU Speed : no more "free lunches" ;-)
  - will x2 times faster CPU increase your performance by x2 ?..
- CPU cores : more and more over year-to-year..
  - Intel 2CPU : 8cores-HT
  - Intel 2CPU : 12cores-HT
  - Intel 2CPU : 16cores-HT
  - Intel 2CPU : 20cores-HT
  - Intel 2CPU : 36cores-HT (2015)
  - …
- Scalability In Few Words :
  - your software is able to deliver a **higher** throughput if more HW resources are available..
  - (then, scaling it well or not is another story ;-))

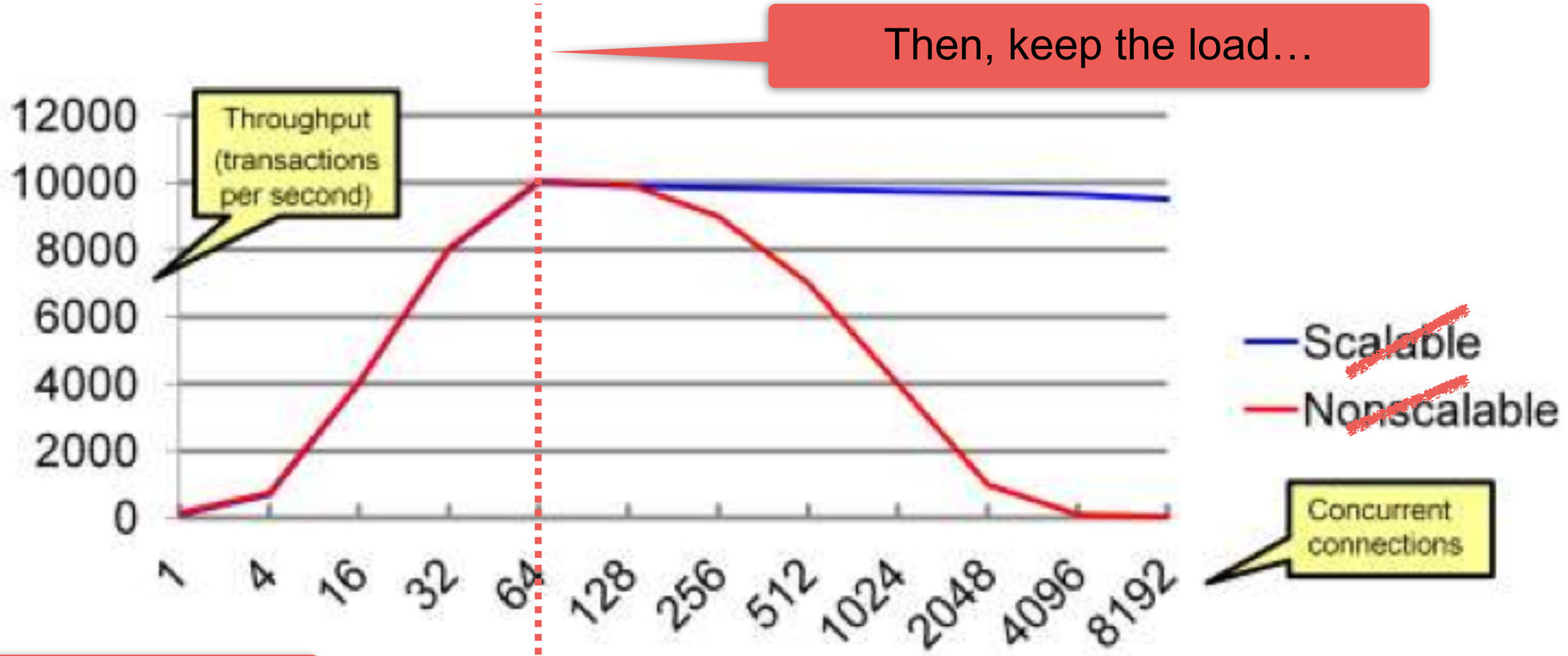
A B-shit Slide…

- Odd interpretation of Scalability…



A B-shit Slide… (2)

- Odd interpretation of Scalability…

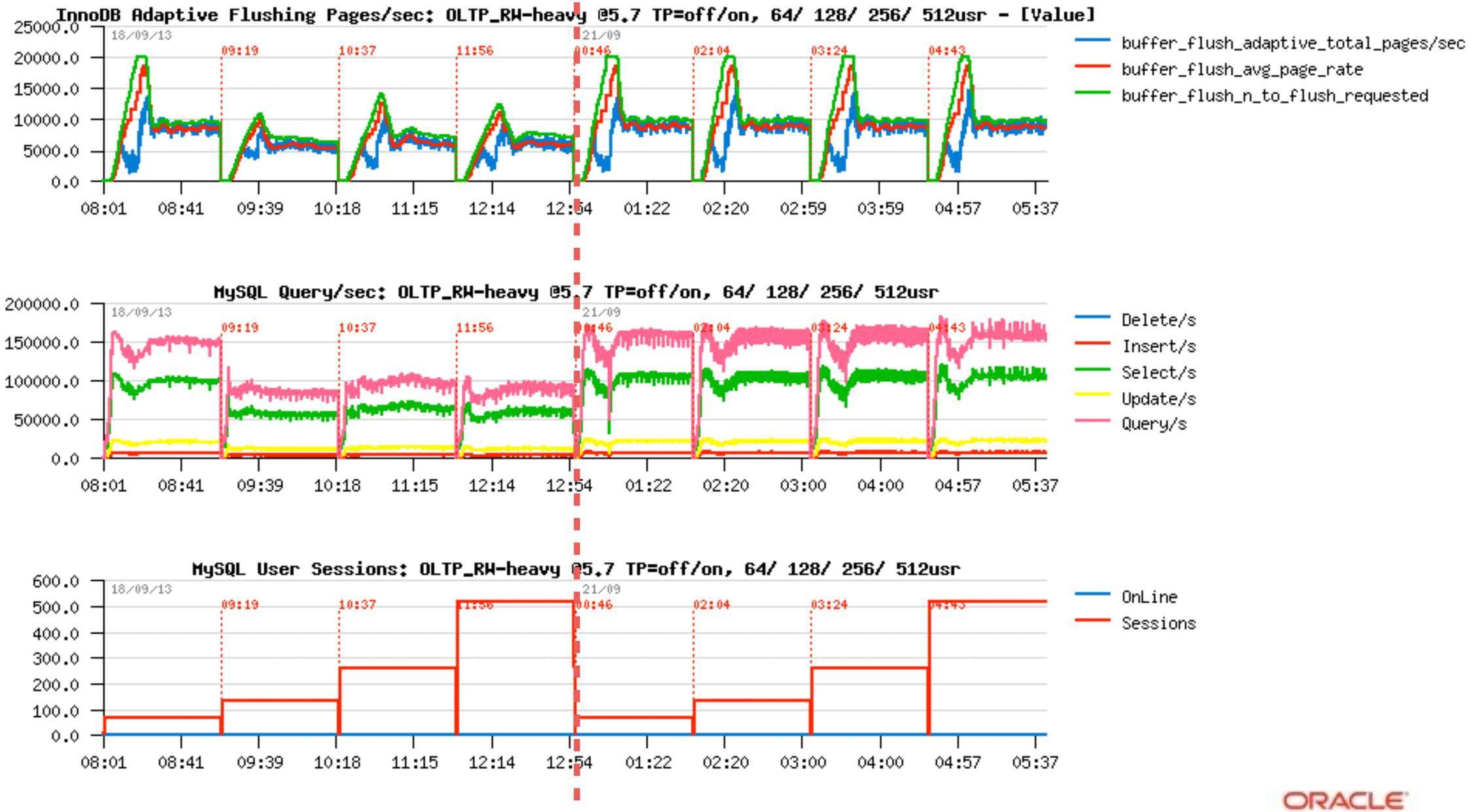


Both are scaling up to 64 connections,
but only one is able to keep a higher load..

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

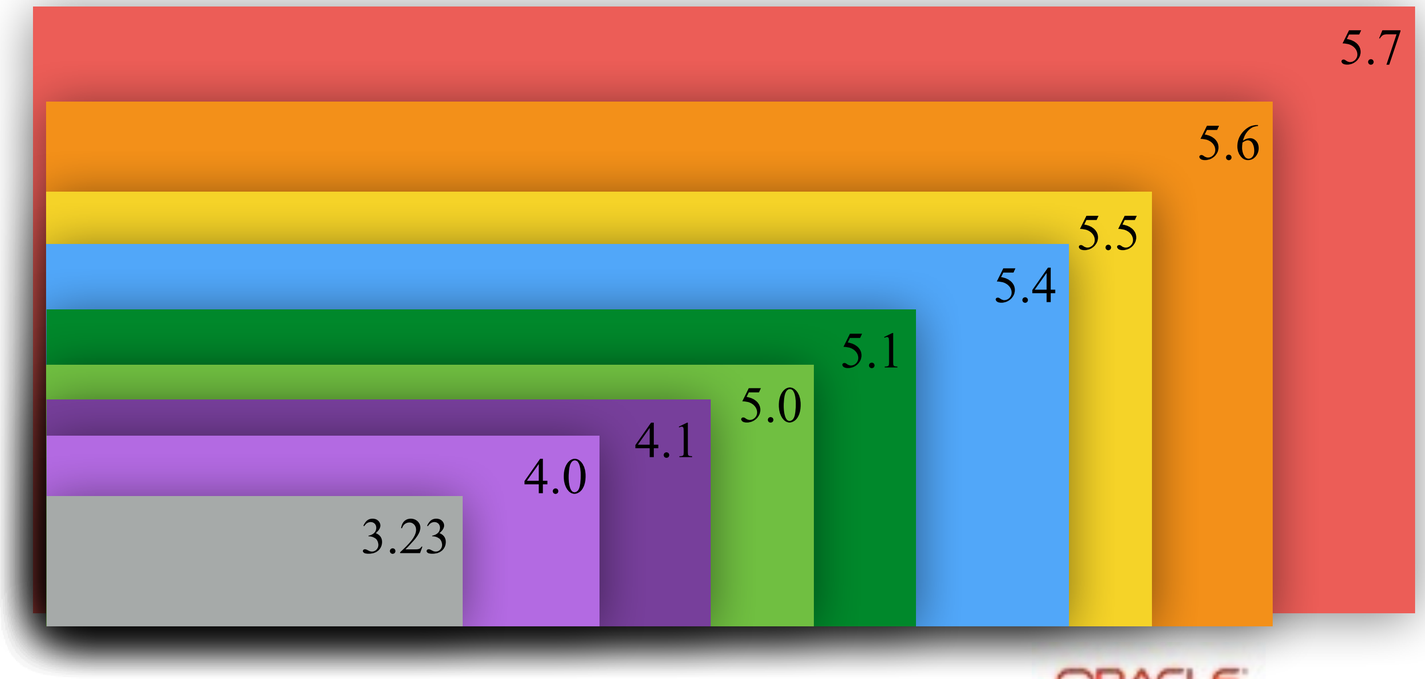


MySQL & CPU Usage

- 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!
  - **for ex. you have 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 ;-))

MySQL Performance Evolution

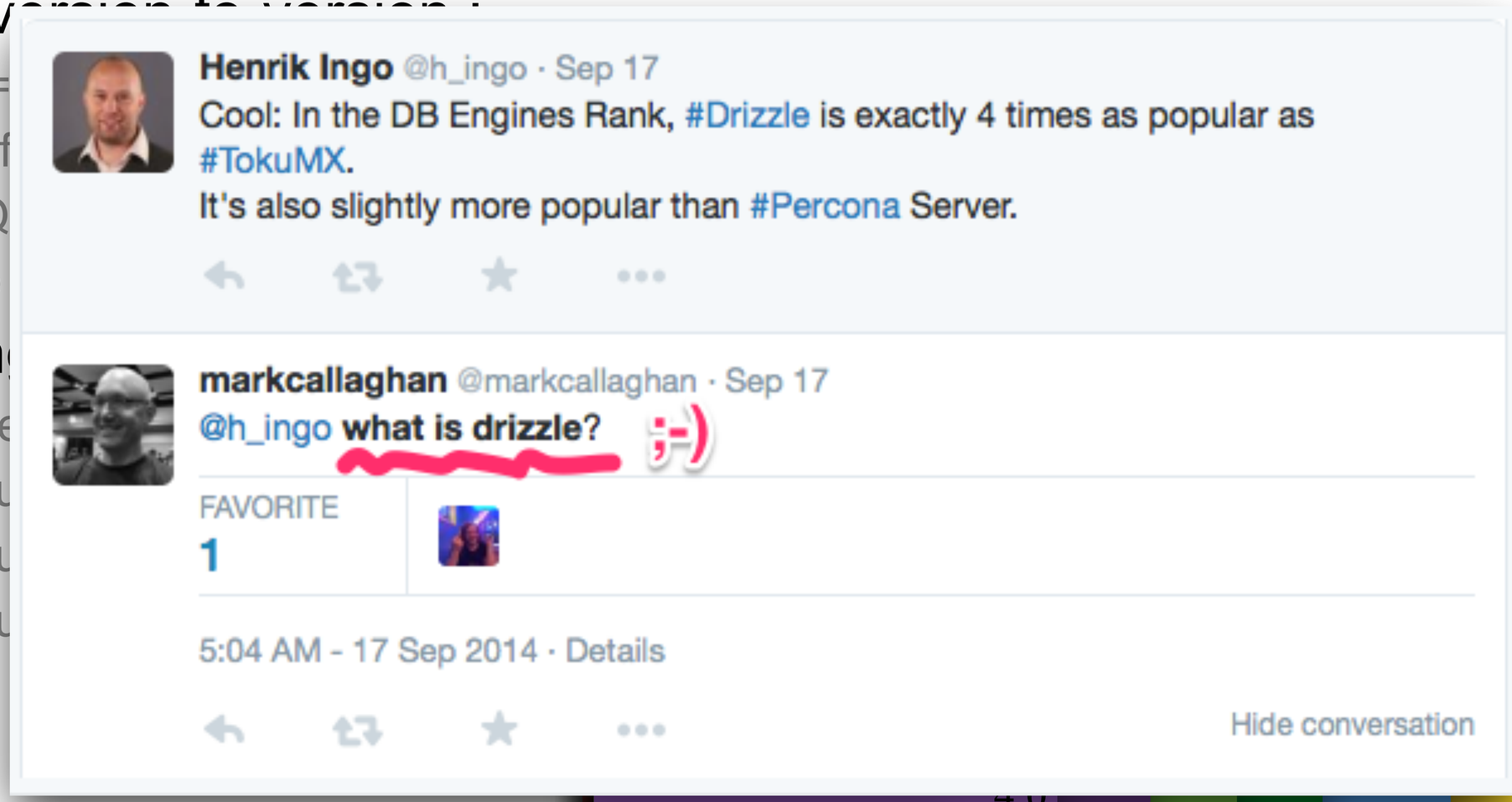
- From version-to-version :
  - 3.23 => 4.0 => 4.1 => 5.0 => 5.1 => 5.4 => 5.5 => 5.6 => 5.7 …
  - more features => longer code path.. (see: “What is new in MySQL 5.7” by Geir this AM)
  - MySQL/InnoDB code is very sensible to CPU cache(s)..
  - single user / low load => going slower..
- Looking back :
  - Drizzle !
  - do you know Drizzle ?
  - do you use Drizzle ?
  - do you run your production on ?



MySQL Performance Evolution

- From v
  - 3.23 =
  - more f
  - MySQ
  - single
- Lookin
  - Drizzle
  - do you
  - do you
  - do you

this AM)



Performance Investigation Efforts (relative)

- report a problem..
- point on the source of the problem..
- suggest what should be fixed..
- suggest **how** it shoud be fixed…
- **implement** the final fix…

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

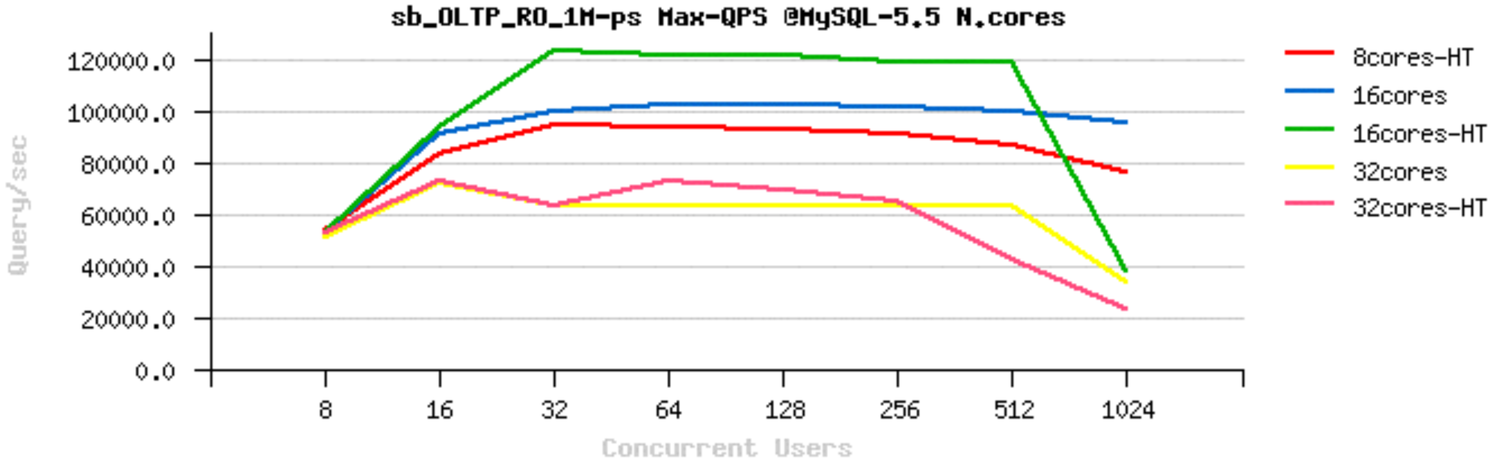
- Sysbench
  - OLTP, RO/RW, N-tables, lots test workload load options, deadlocks
- DBT2 / TPCC-like
  - OLTP, RW, very complex, growing db, no options, deadlocks
  - In fact using mostly only 2 tables! (thanks Performance Schema ;-))
- dbSTRESS
  - OLTP, RO/RW, several tables, one most hot, configurable, no deadlocks
- iiBench
  - pure INSERT (time series) + SELECT
- LinkBench (Facebook)
  - OLTP, RW, very intensive, IO-hungry..
- DBT3
  - DWH, RO, complex heavy query, loved by Optimizer Team ;-)

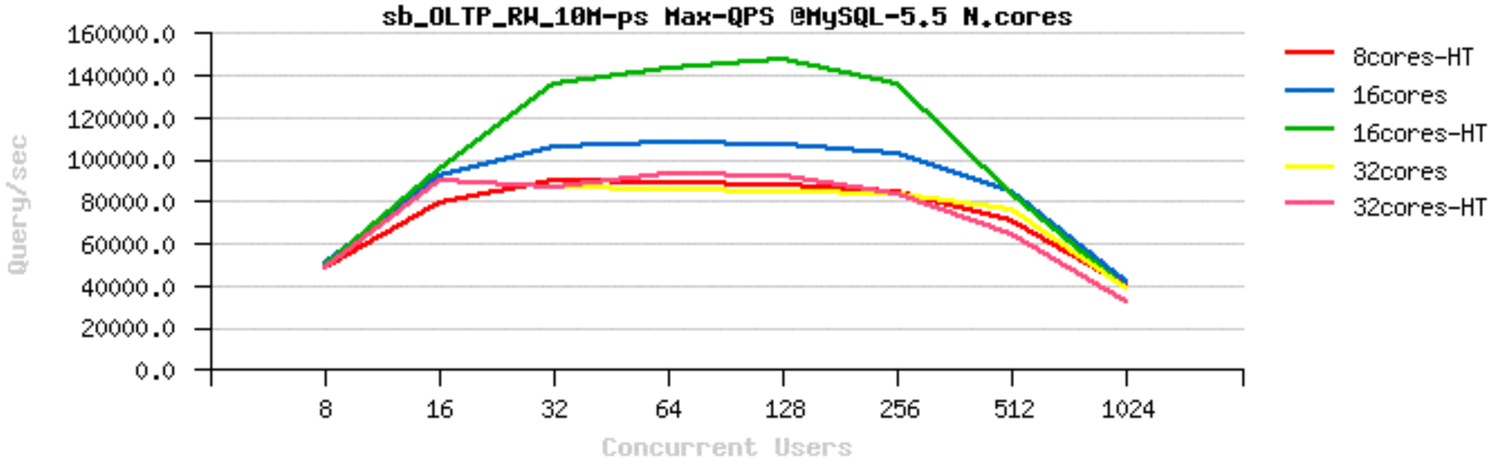
Analyzing Workloads: RO -vs- RW

- 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 !
- NOTE : Random Read (RR) operation is the main IO-bound killer !!!

From where we're coming with MySQL 5.7 ?..

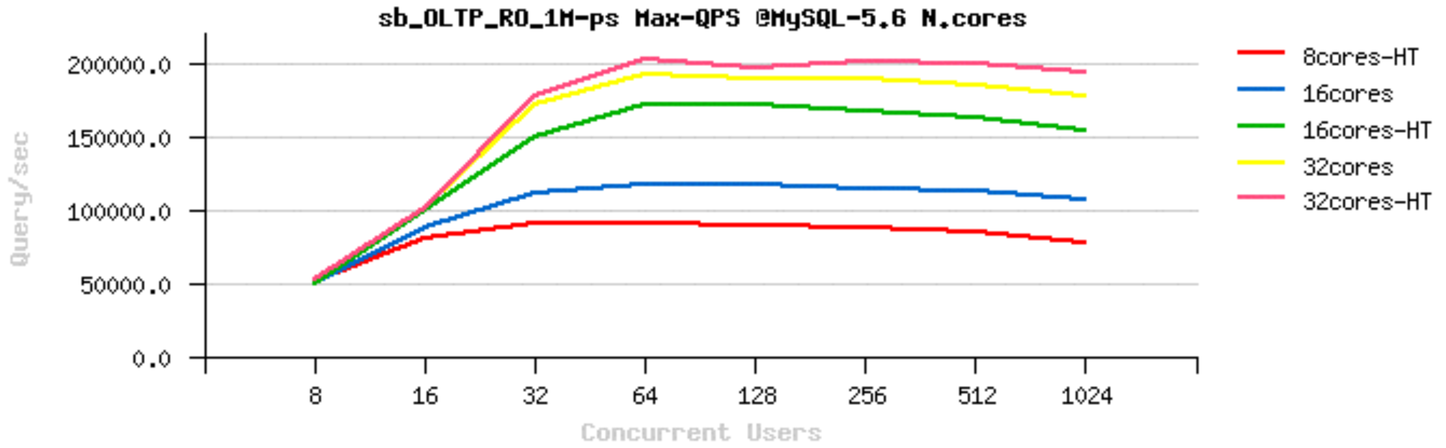
- MySQL 5.5 : RO & RW
  - QPS Max on 16cores
  - worse on 32cores
  - Note: RW out-pass RO!

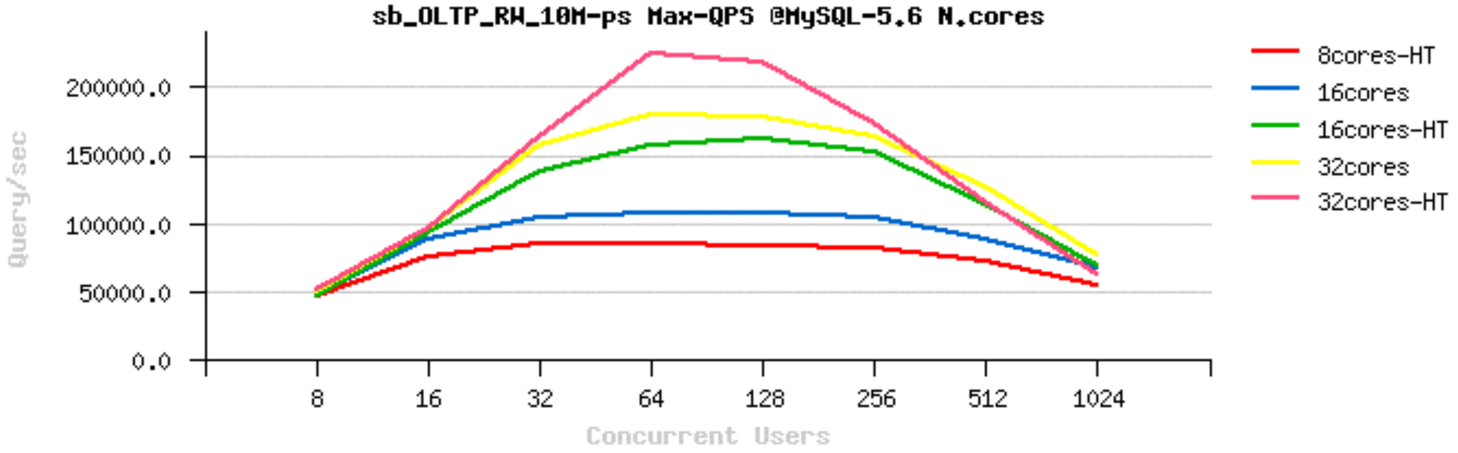




From where we're coming with MySQL 5.7 ?..

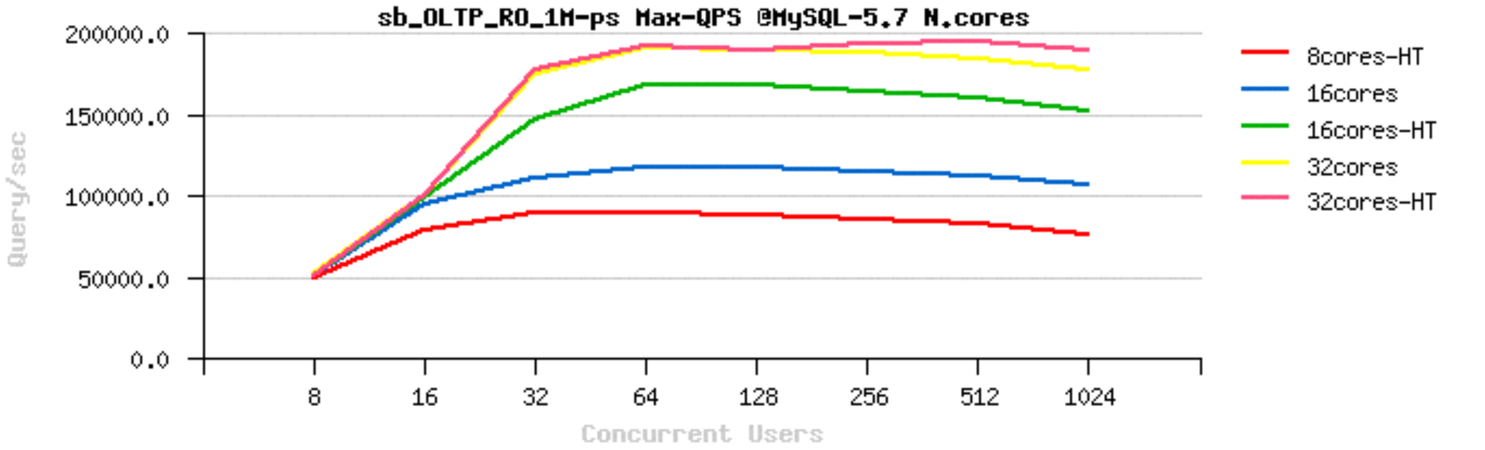
- MySQL 5.6 : RO & RW
  - not lower on 32cores!! ;-)
  - RW out-pass RO !!..??

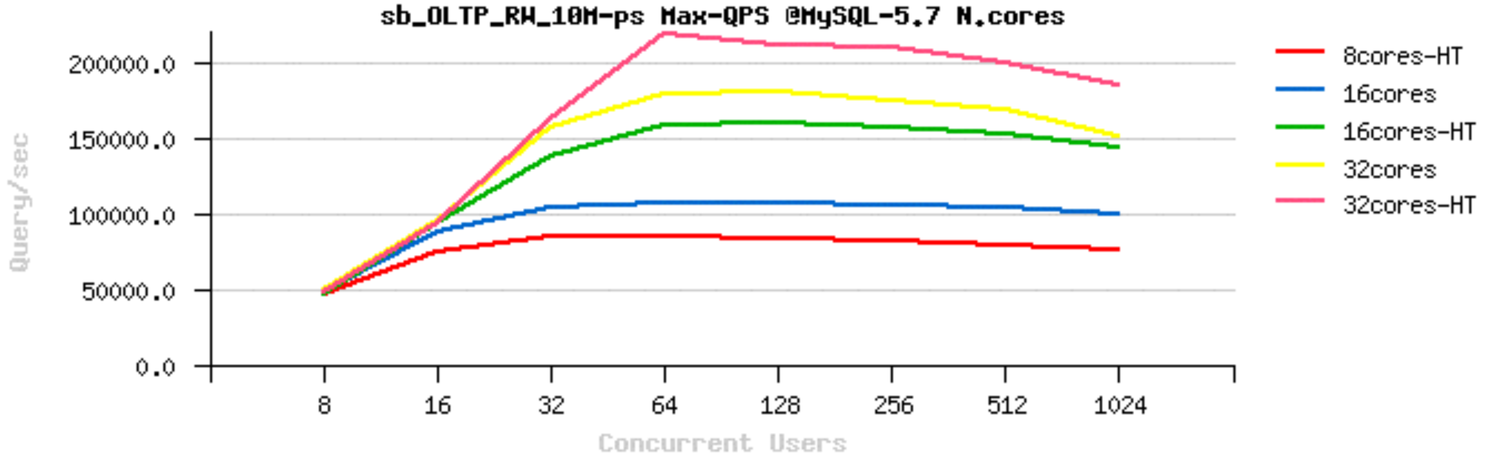




From where we're coming with MySQL 5.7 ?..

- MySQL 5.7.1 : RO & RW
  - more stable than 5.6
  - **RW** out-pass RO !!..





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 :

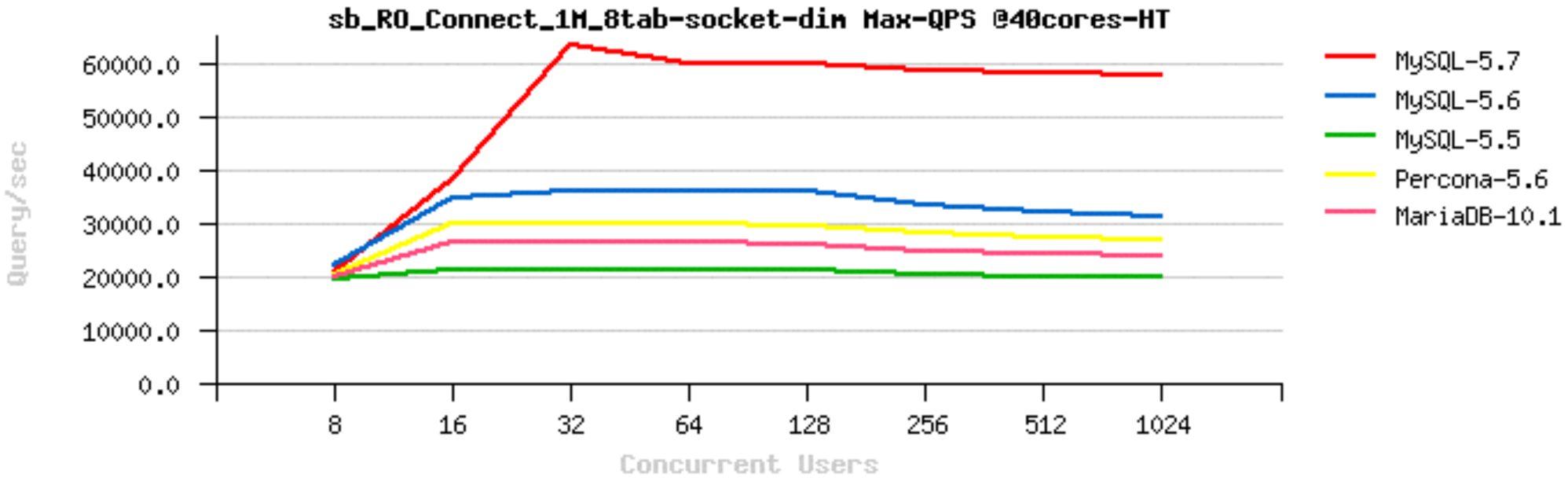
```
join_buffer_size=32K
sort_buffer_size=32K

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)..
  - **RO_Connect** : a single Point-Select with re-connect
- 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

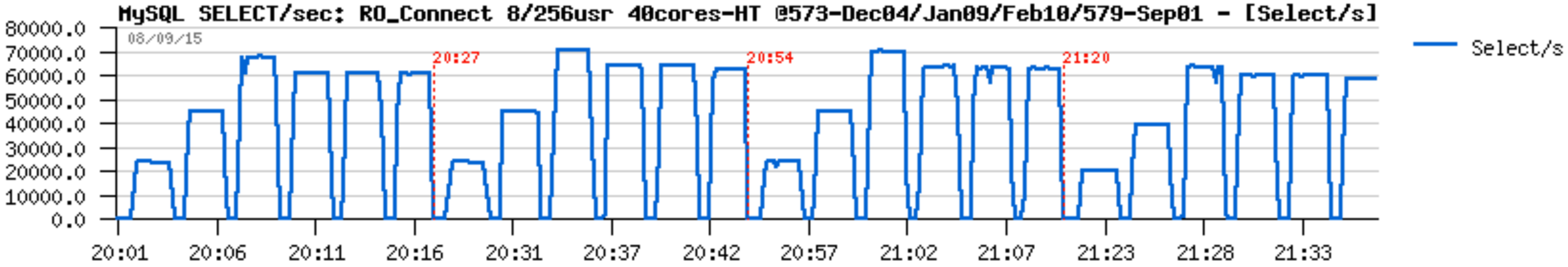
Entry Ticket : RO_Connect

- Many web apps cannot use persistent connections
  - connect => Query(s) => disconnect



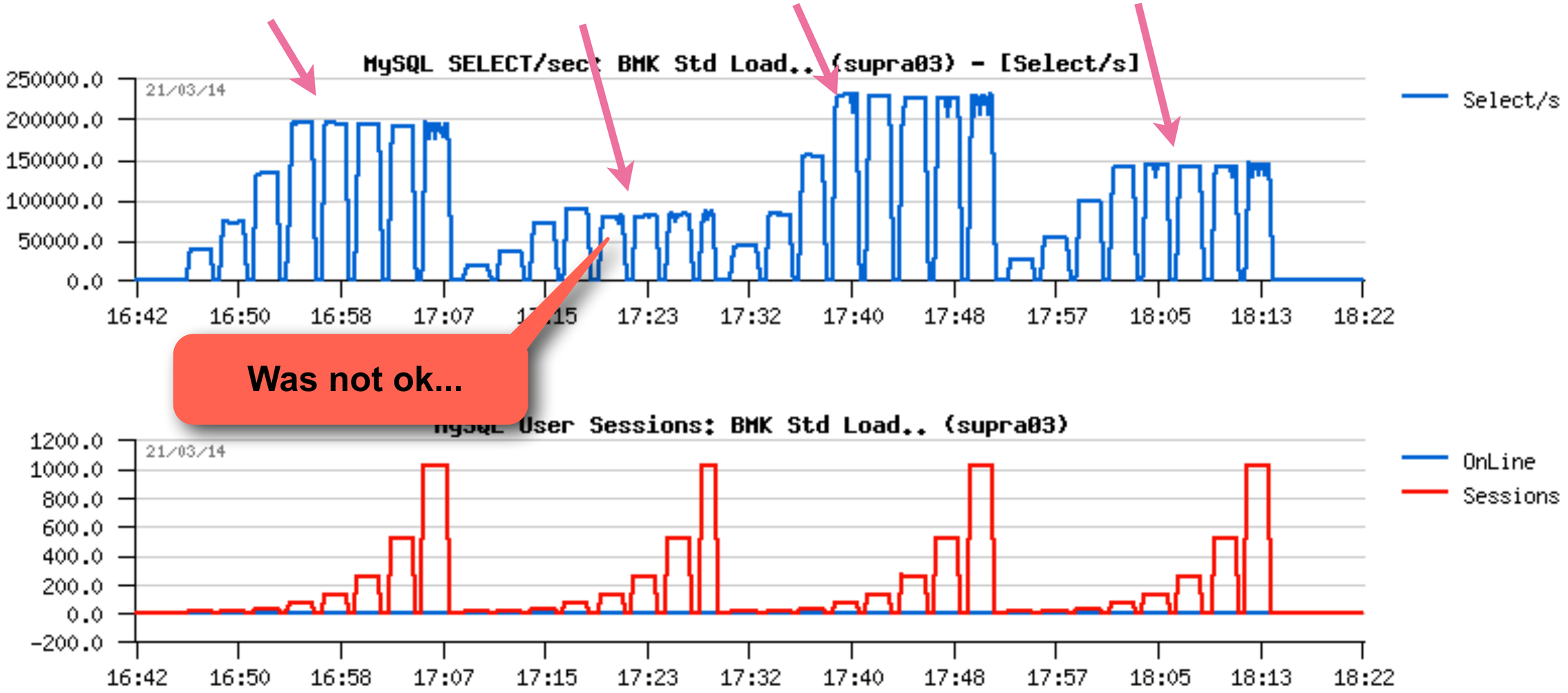
Entry Ticket : RO_Connect in 5.7

- Many web apps cannot use persistent connections
  - connect => Query(s) => disconnect
  - there was even 70K Connect/sec, but new features over 2 years..
  - on newer HW : over 100K Connect/sec
  - 5.8 expectations : to do much more than this ;-)



Sysbench OLTP_RO Workloads @MySQL 5.7 - Apr.2014

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



ORACLE

RO_Dranges : “kernel” contention

- Sysbench RO Distinct Ranges - Apr.2014
  - 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%
   0.76%
   0.69%
   0.53%
   ...
```

```
  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 ?...

RO_Dranges : “kernel” contention

- Sysbench RO Distinct Ranges - 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 ;-)

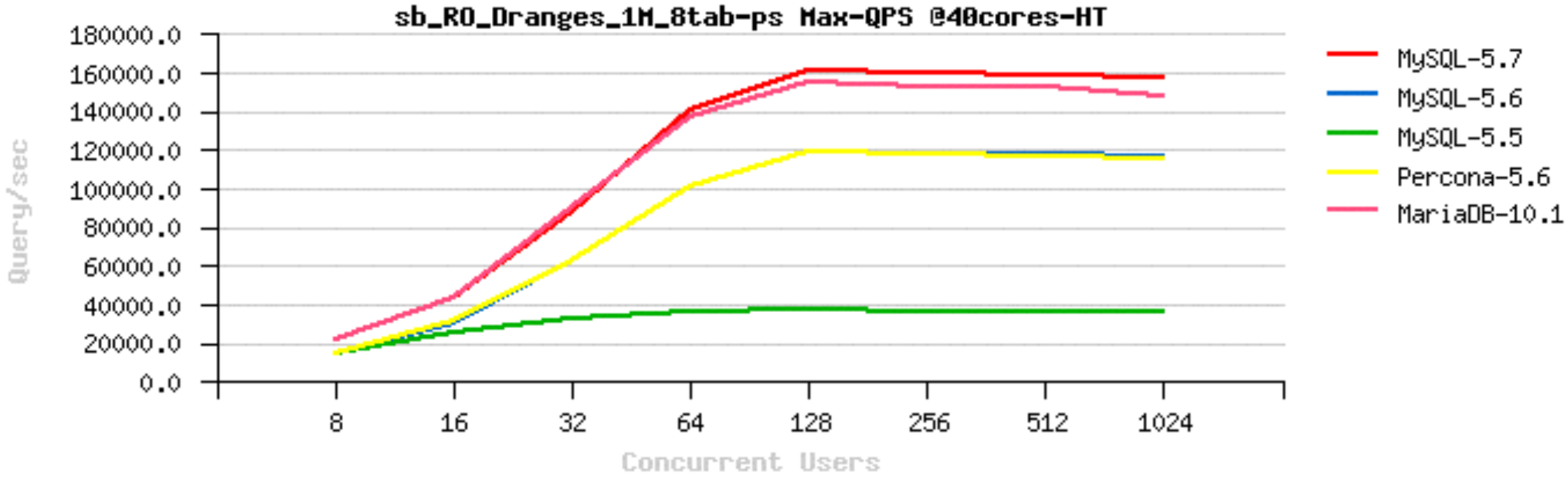
```
LD_PRELOAD=/apps/lib/libjemalloc.so  ; export LD_PRELOAD
MALLOC_CONF=lg_dirty_mult:-1         ; export MALLOC_CONF
```

RO_Dranges : “hot” hash

- Sysbench RO Distinct Ranges - 2015
  - 40cores-HT (32/12cores-HT as well)
  - my_hash_sort_simple() takes over 20% CPU time..
- A long story short :
  - on DISTINCT optimizer is doing INSERT into a Memory table with unique hash index
  - (same the processing of the GROUP BY as well)
  - however, Memory table (HEAP engine) is calling 4 times the my_hash_sort_simple() function per INSERT..
  - fixed in 5.7 : 20% more performance ;-)

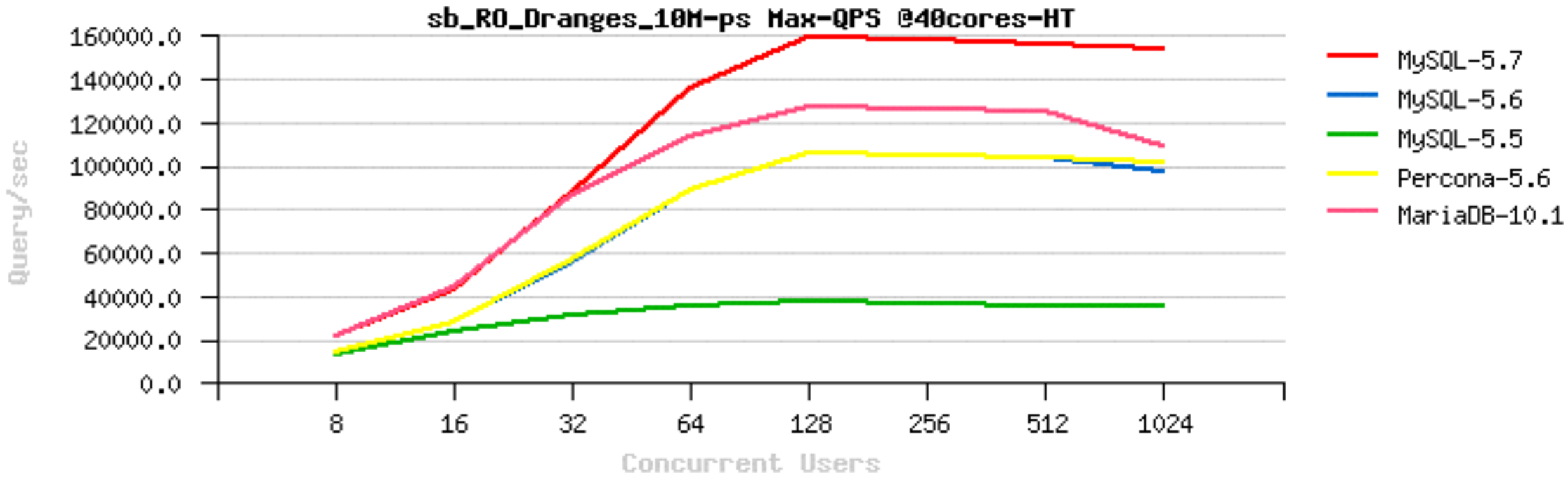
RO_Dranges : 8-tables

- Sysbench RO Distinct Ranges 1Mx8-tables - Sep.2015
  - 40cores-HT



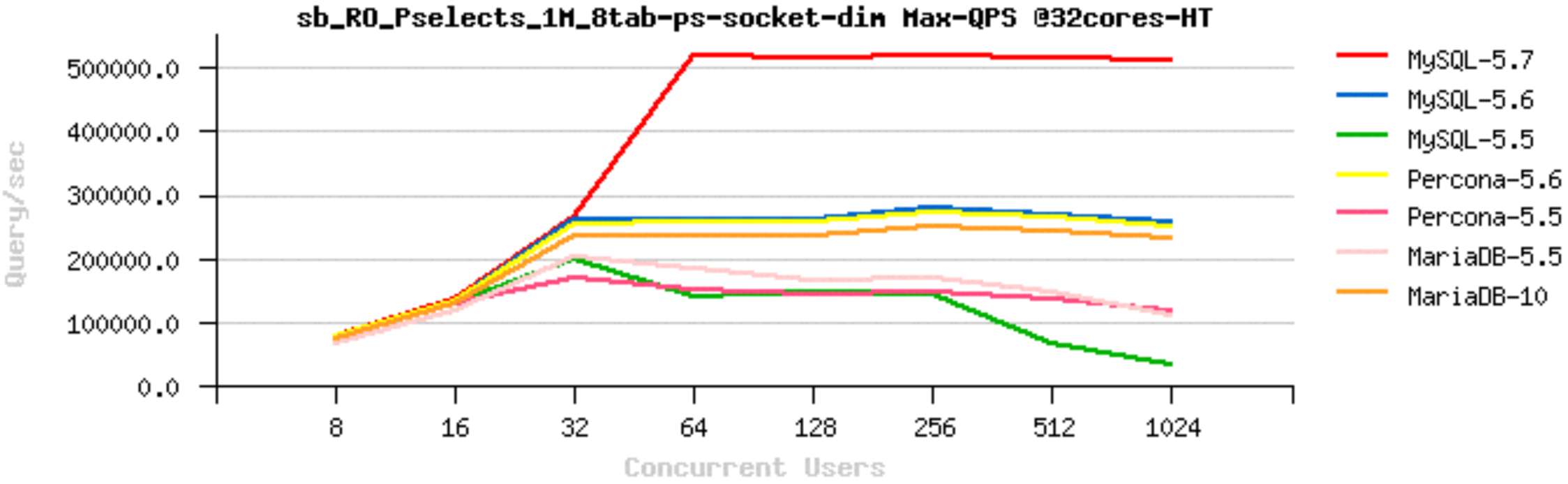
RO_Dranges : 1-table

- Sysbench RO Distinct Ranges 10M - Sep.2015
  - 40cores-HT



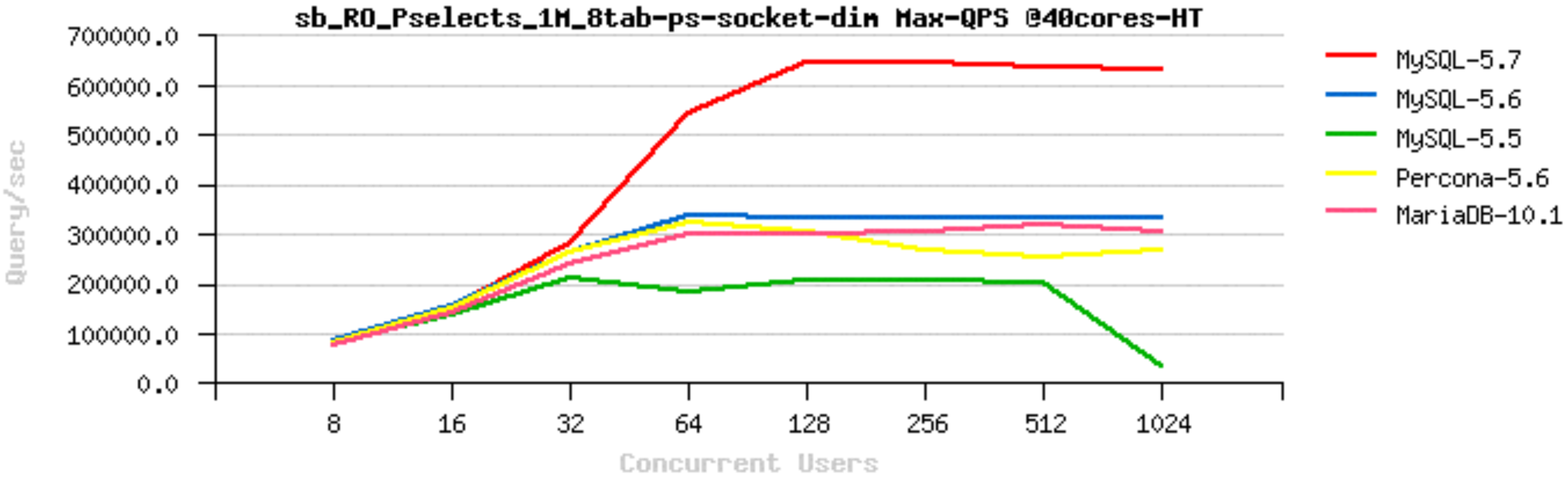
RO In-Memory @MySQL 5.7

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



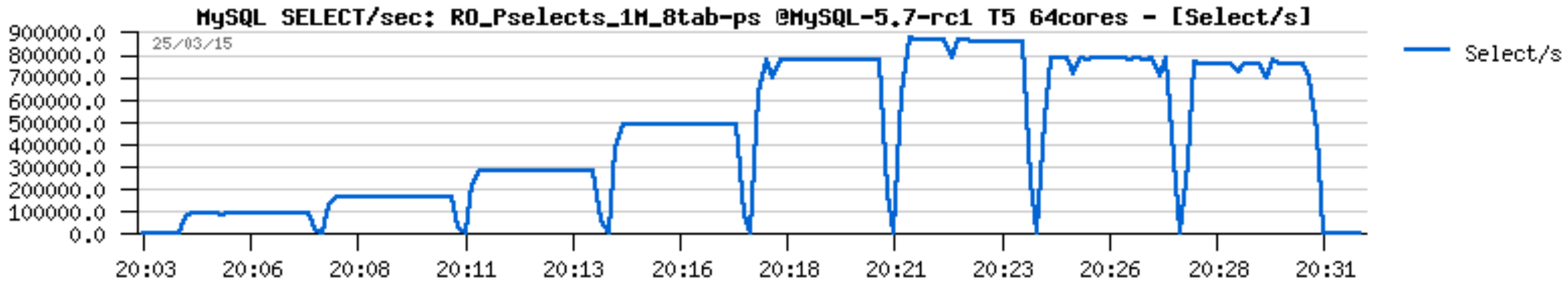
RO In-Memory @MySQL 5.7

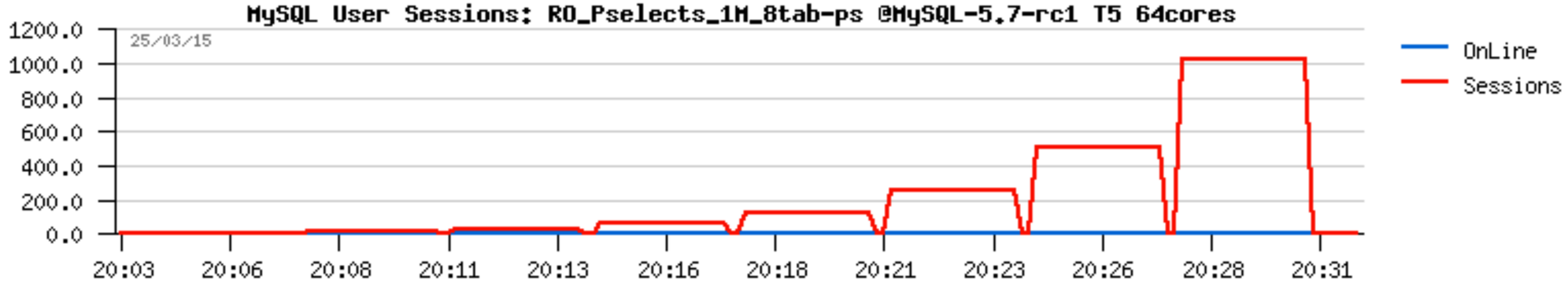
- **645K QPS** Sysbench Point-Selects 8-tab, 40cores-HT :



RO In-Memory @MySQL 5.7 - Apr.2015

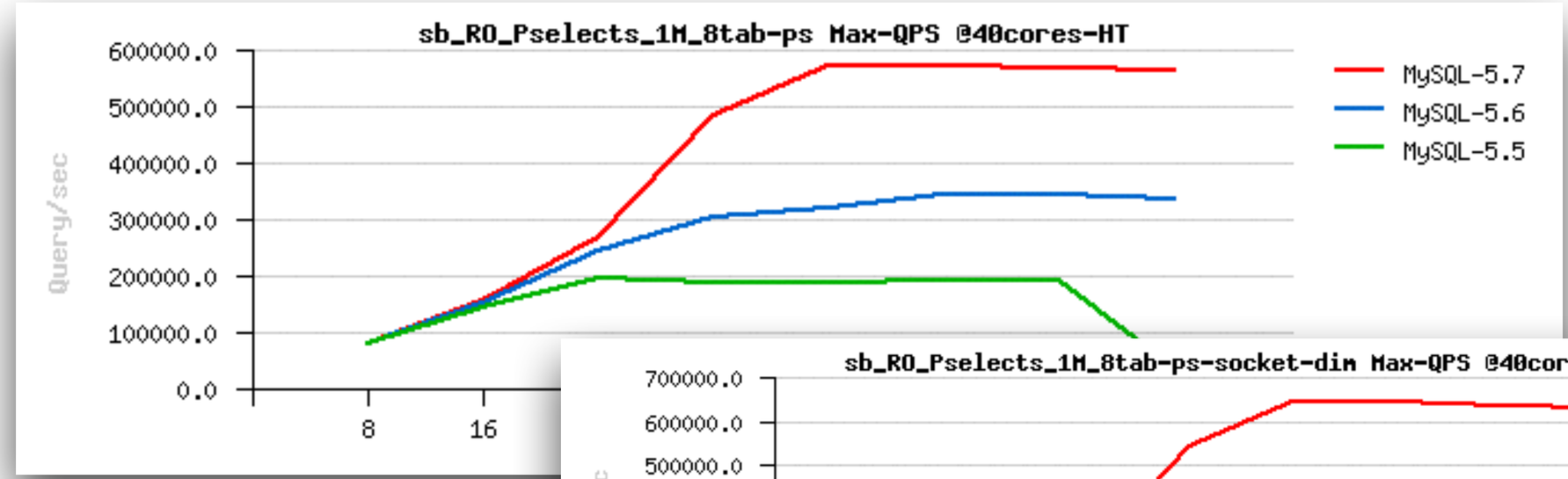
- **Around 900K QPS** Sysbench Point-Selects 8-tab, 64cores SPARC T5 :
  - Just a probe test with zero efforts ;-)

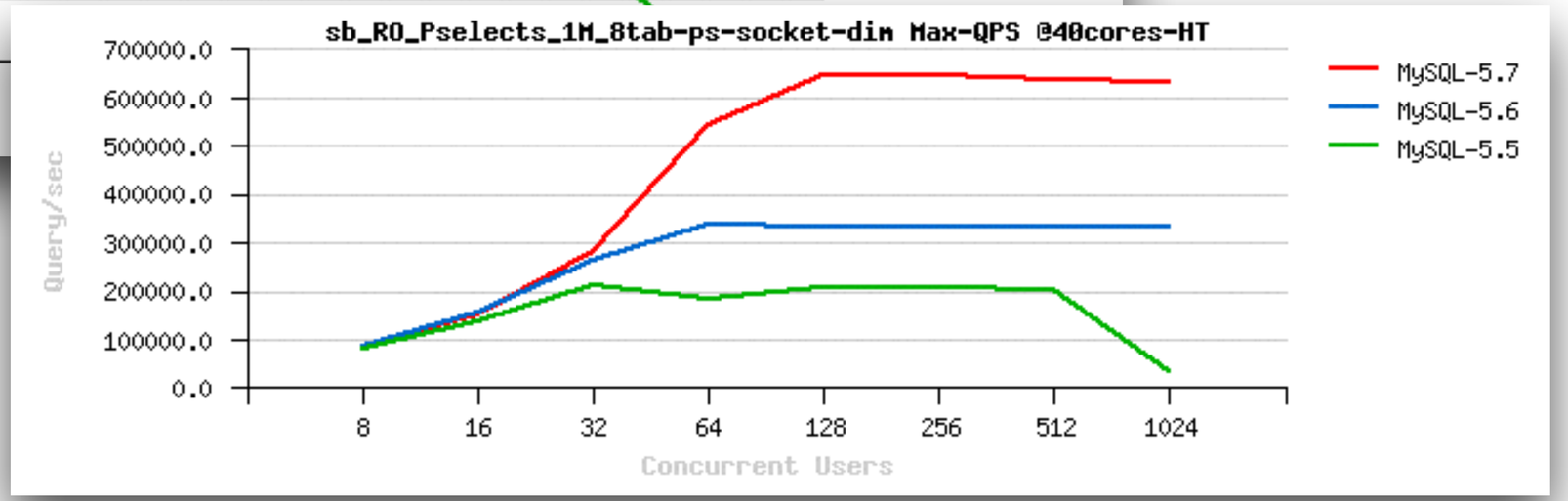




Few words about RO scalability

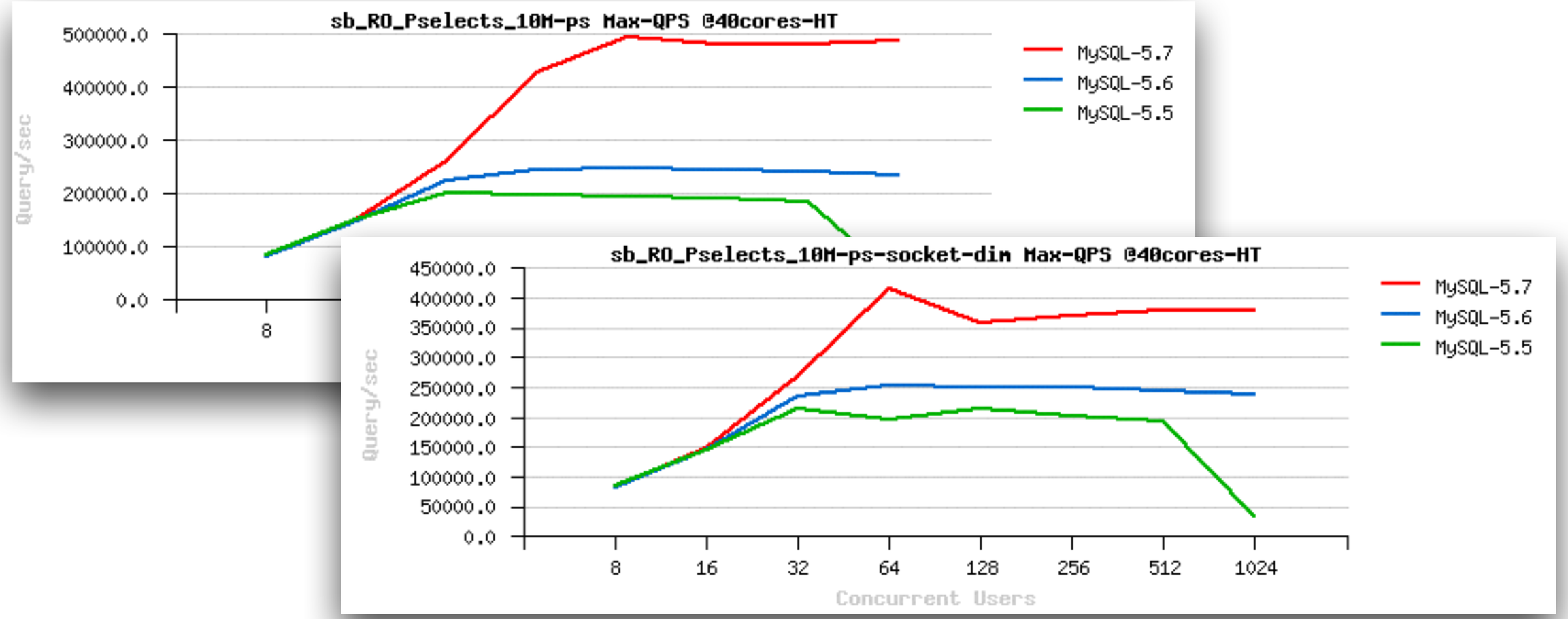
- OLTP_RO Point-selects **8**-tables, the same 40cores host
  - IP socket & sysbench 0.4.13 -vs- UNIX socket & sysbench 0.4.8 :





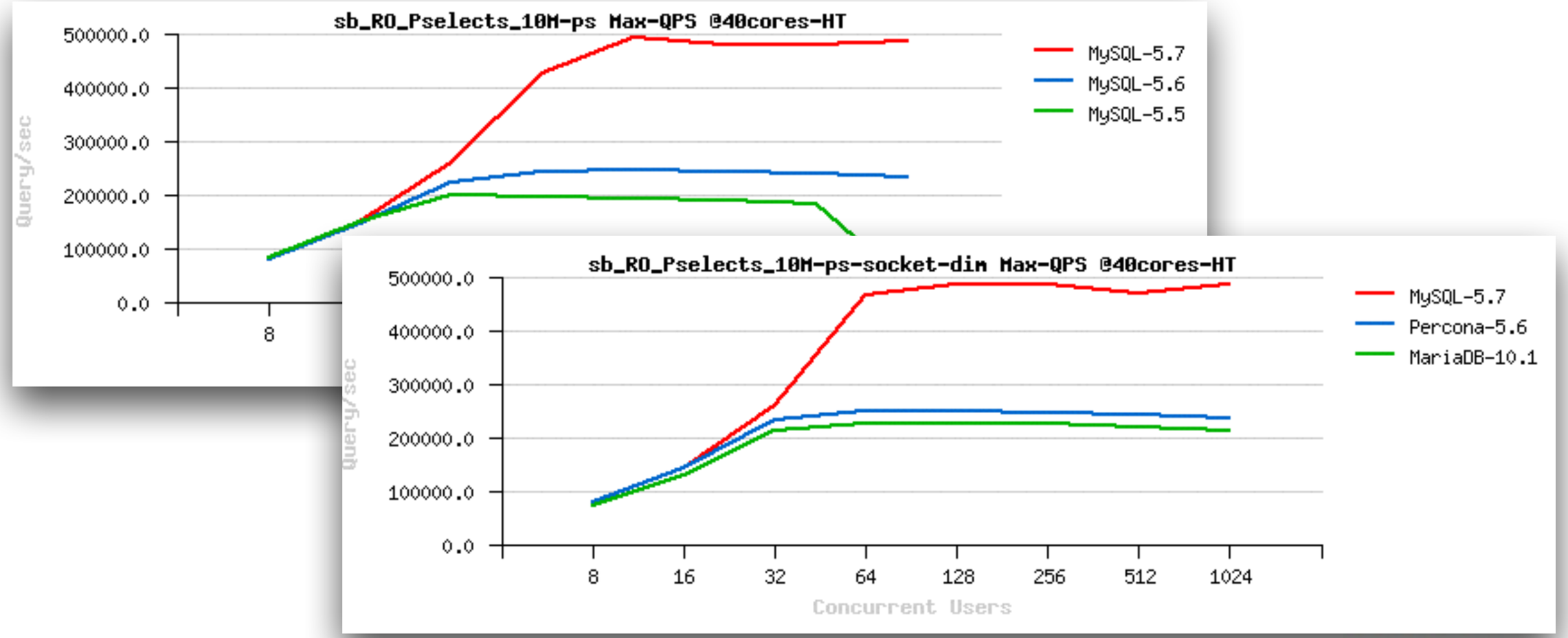
Few words about RO scalability (bis) - Oct.2014

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



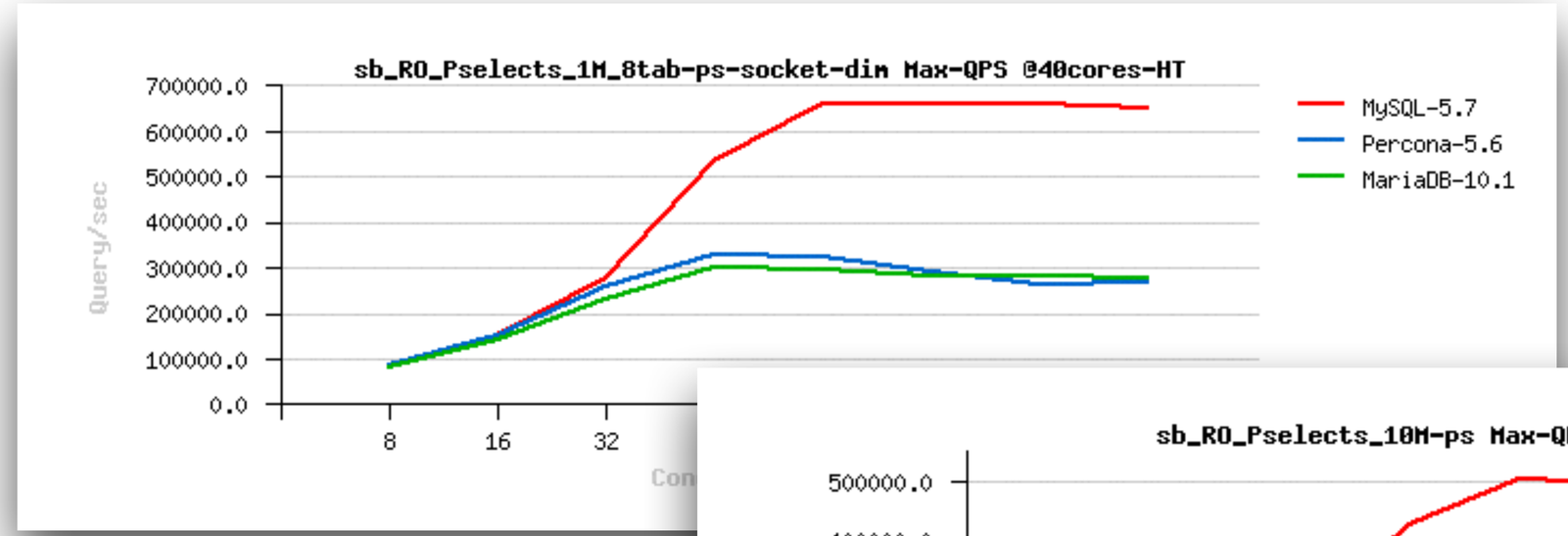
Few words about RO scalability (bis) - Apr.2015

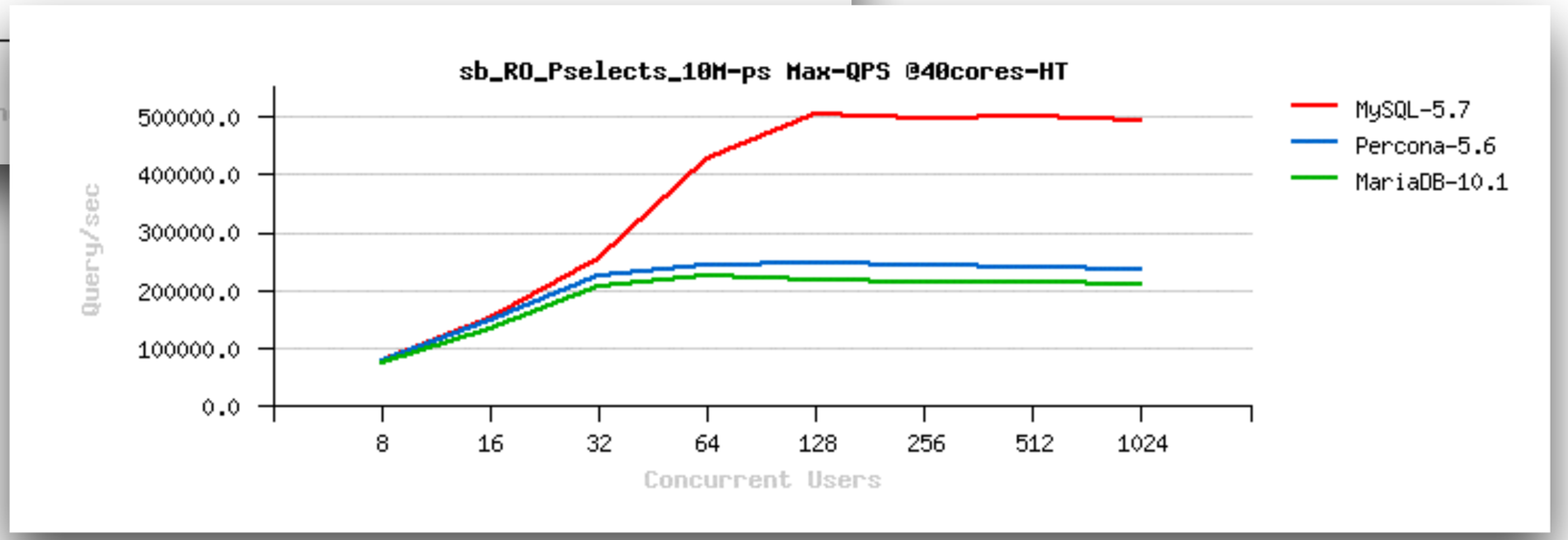
- OLTP_RO Point-selects **1**-table, the same 40cores host
  - IP socket & sysbench 0.4.13 -vs- UNIX socket & sysbench 0.4.8 :



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

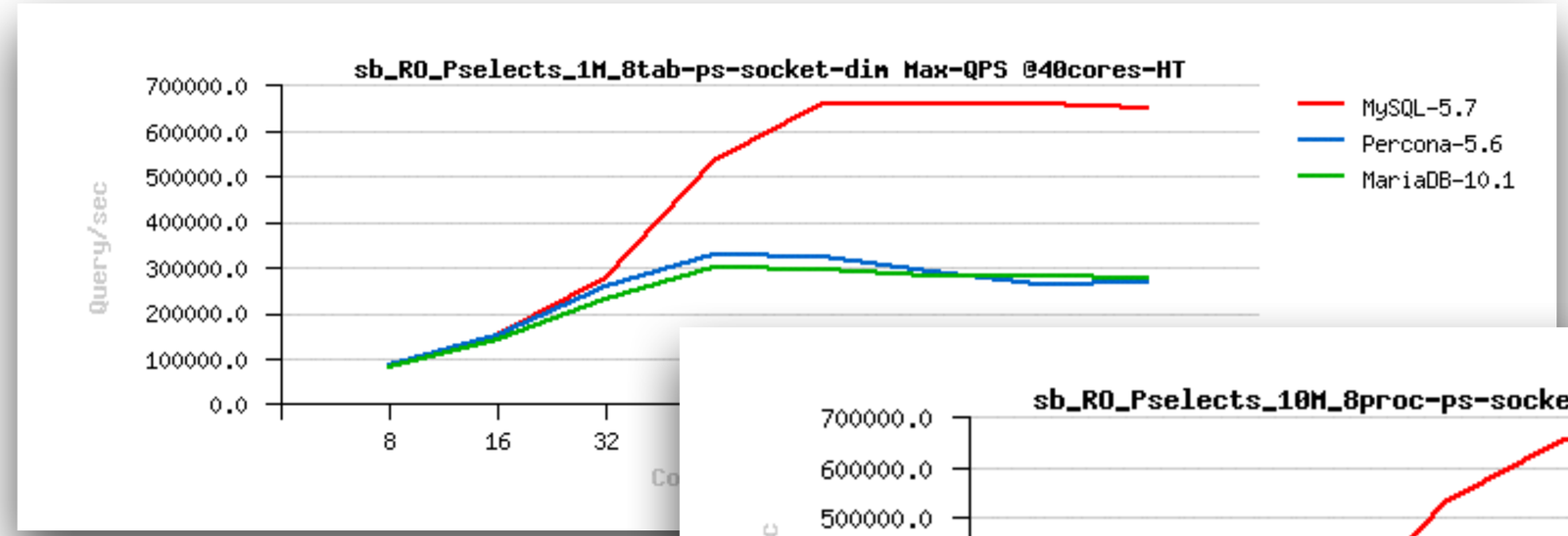
- OLTP_RO Point-selects **8**-tables -vs- **1**-table, the same 40cores host
  - Note : running 8 sysbench processes on 8-tables



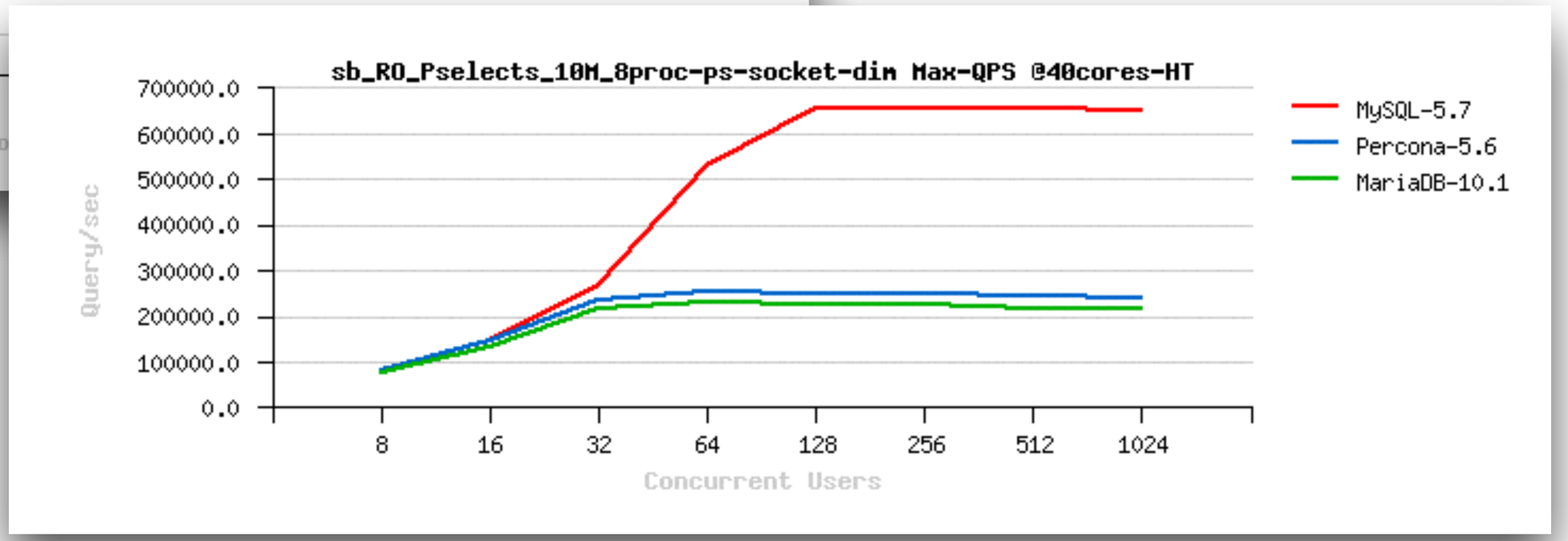


Few words about RO scalability (bis2) - Apr.2015

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

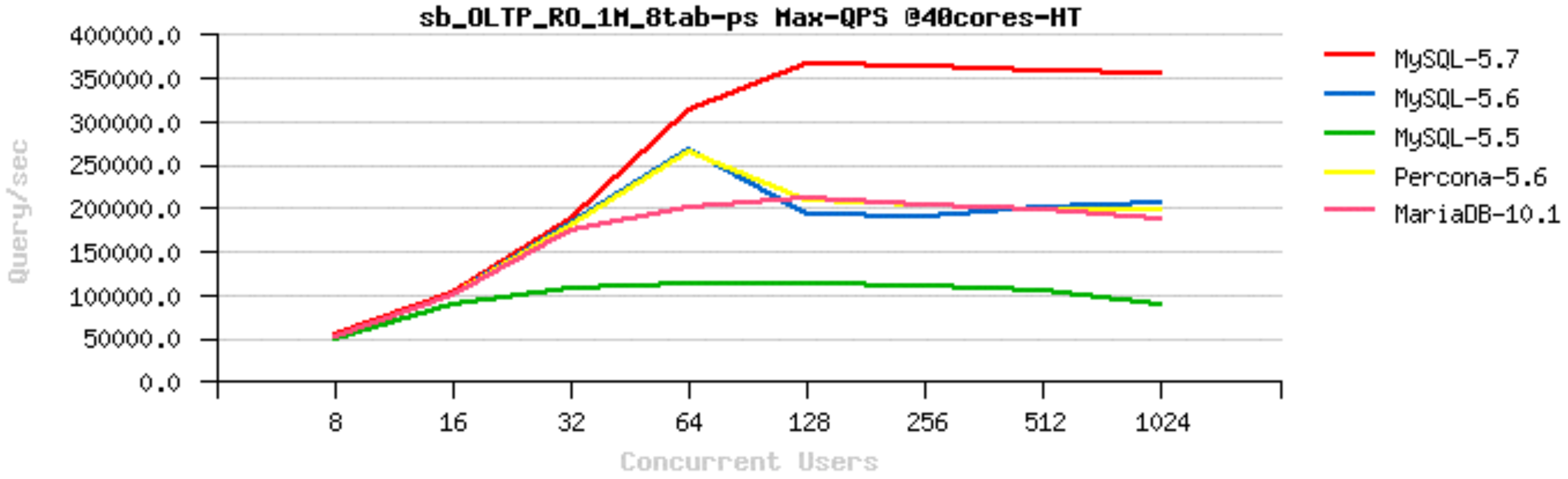


TODO: fix Sysbench ;-)



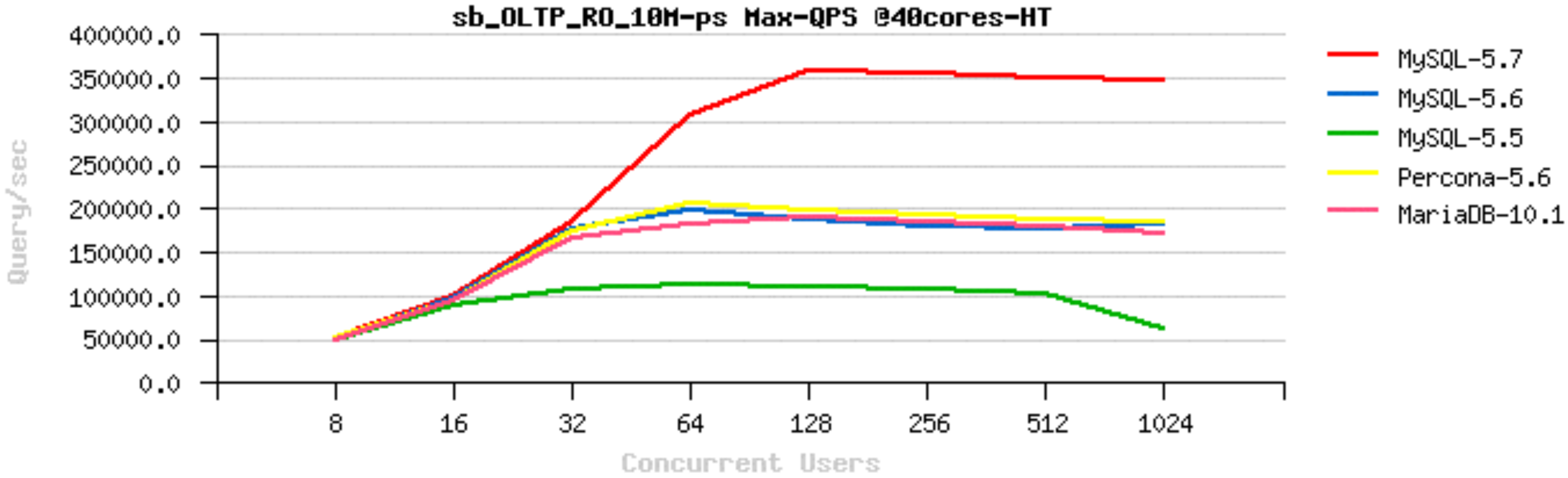
OLTP_RO : 8-tables

- Sysbench OLTP_RO 1Mx8-tables
  - 40cores-HT



OLTP_RO : 1-table

- Sysbench OLTP_RO 10M
  - 40cores-HT

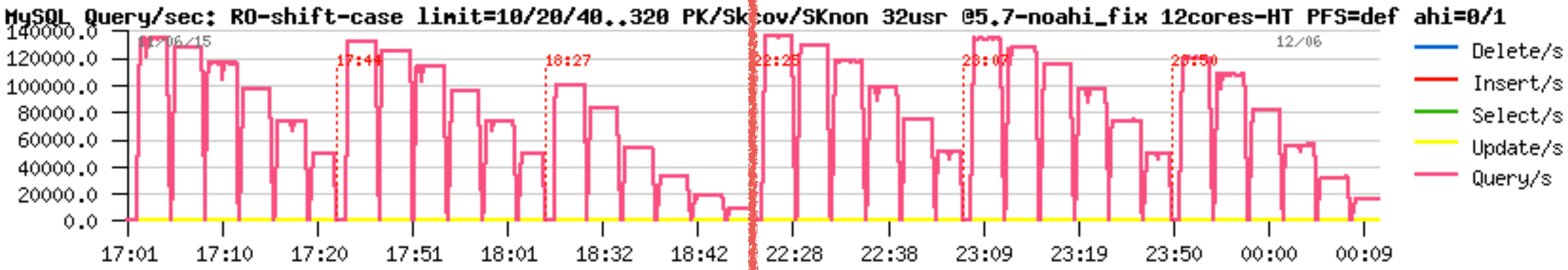


RO Pending Issues…

- InnoDB Adaptive Hash Index (AHI)
  - implemented with a global RW-lock
  - InnoDB RW-locks are not scaling by design (CPU cache syncs)
  - using table partitions helps to split indexes
  - using AHI partitions (5.7) helps to split RW-locks (coop. with Percona)
  - yet far from fixed..
  - 5.8 : AHI re-write / re-design

RO Pending Issues…

- PK vs Sec.IDX lookups
  - AHI helps
  - using covering indexes helps
  - reading less rows per query helps too.. (in ex: 10/20/40.. 320 rows)
  - PK Cov.IDX Sec.IDX PK Cov.IDX Sec.IDX



AHI=off AHI=on

RO Pending Issues…

- InnoDB Block Lock
  - seen when the same pages are accessed concurrently..
  - how to see : “show mutex” is back ;-)
  - workarounds :
    - avoid such an access pattern, don’t do this ;-)
    - use a smart query cache (like ProxySQL), or row cache (memcached, etc.)..
  - expected to be fixed in 5.8 : page re-design
    - but nothing yet promised.. ;-)

Read+Write (RW) Workloads Scalability @MySQL 5.7

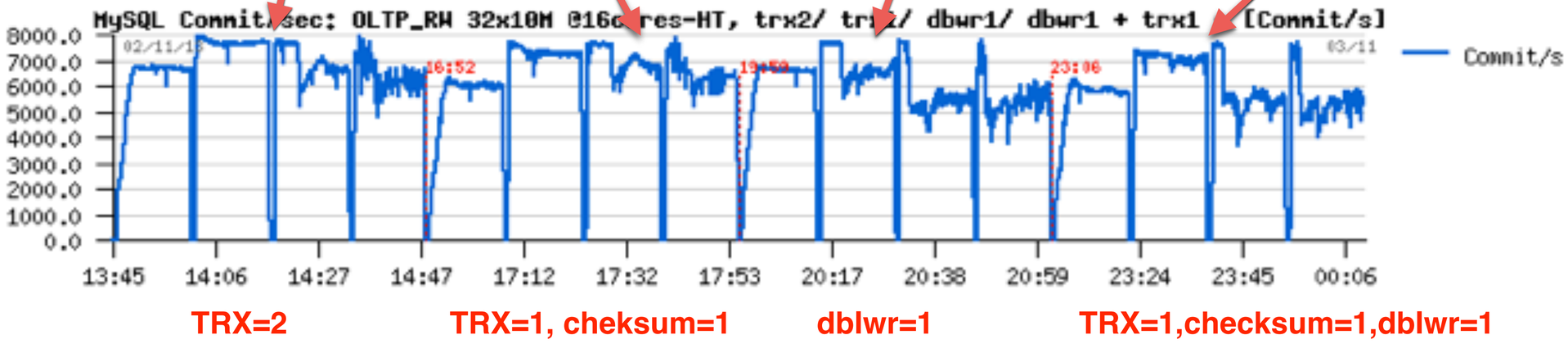
- Huge progress is already here too!
  - improved index locking
  - reduced lock_sys mutex contention
  - parallel flushing + improved flushing design
  - much better observability of internals
  - etc..
- However, not yet as good as Read-Only..
  - Performance continues to increase with more CPU cores
  - 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

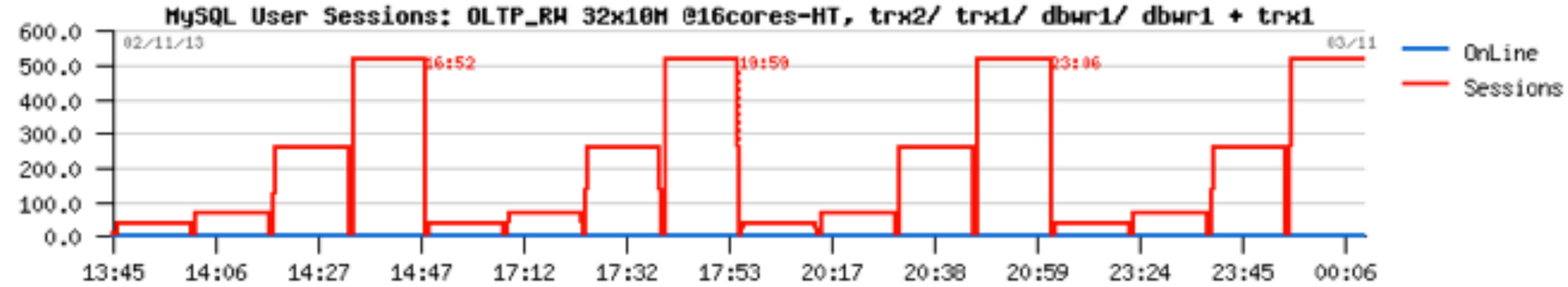
- Transactional processing
  - your CPU-bound transactional processing defines your Max possible TPS
  - with a bigger volume / more IO / etc. => Max TPS will not increase ;-)
- Data Safety
  - binlog : overhead + bottleneck (be sure you have binlog group commit)
  - InnoDB checksums : overhead (reasonable 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 + chksum=1 | dblwr=1 | trx=1 + chksum=1 + dblwr=1



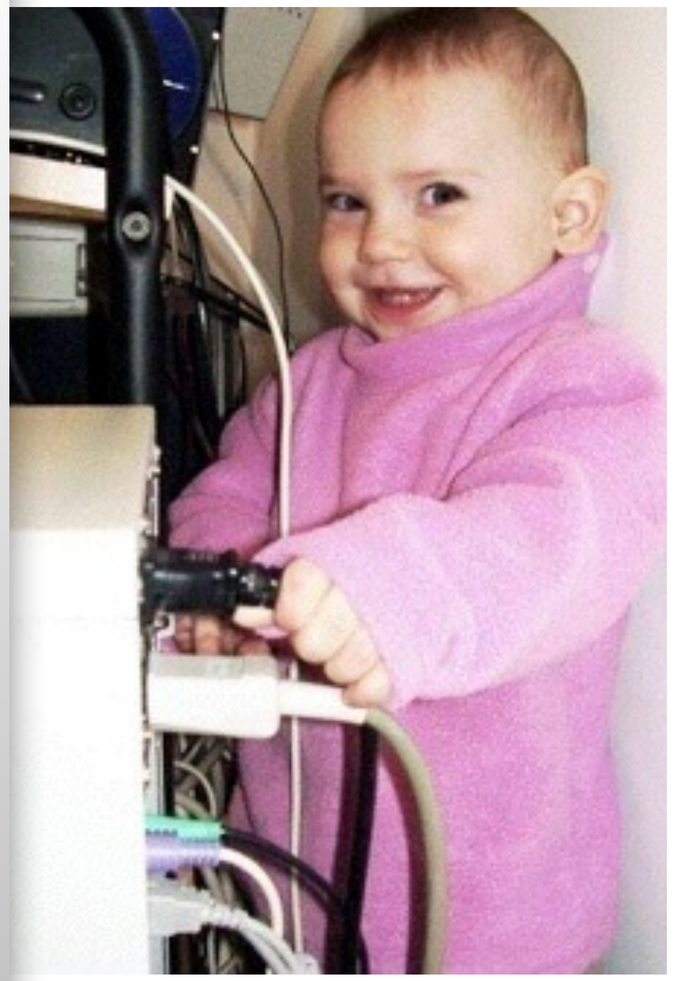
TRX=2 TRX=1, cheksum=1 dblwr=1 TRX=1,checksum=1,dblwr=1



RW related starter configuration settings

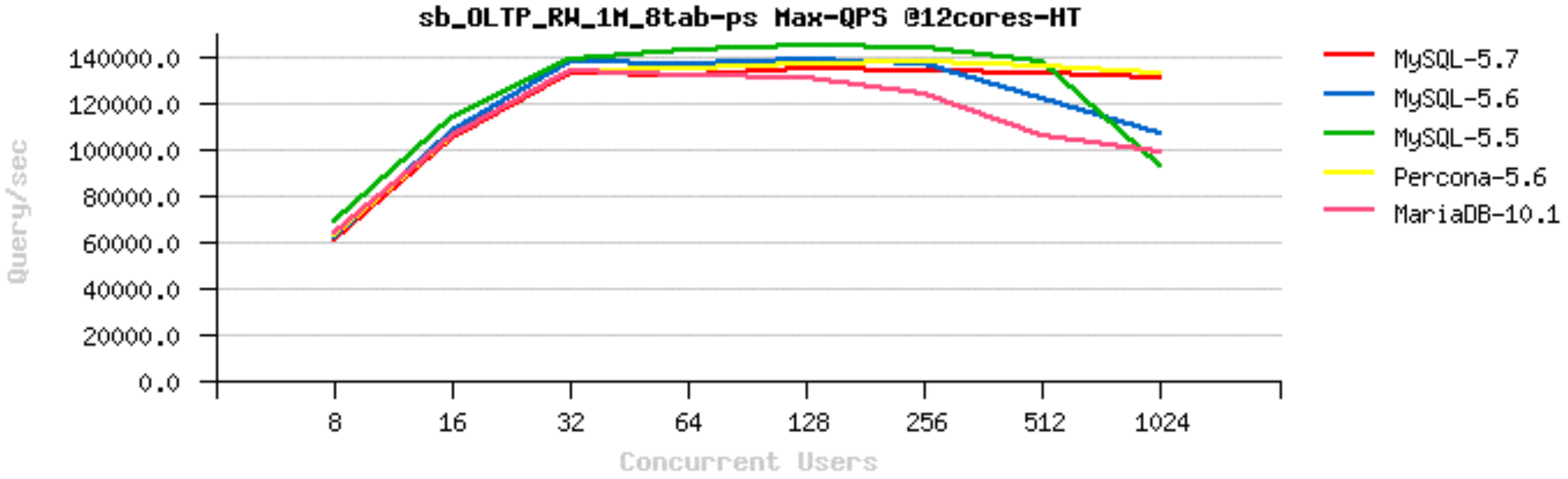
- my.conf :

```
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=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= 0 / 1000000

binlog ??
```



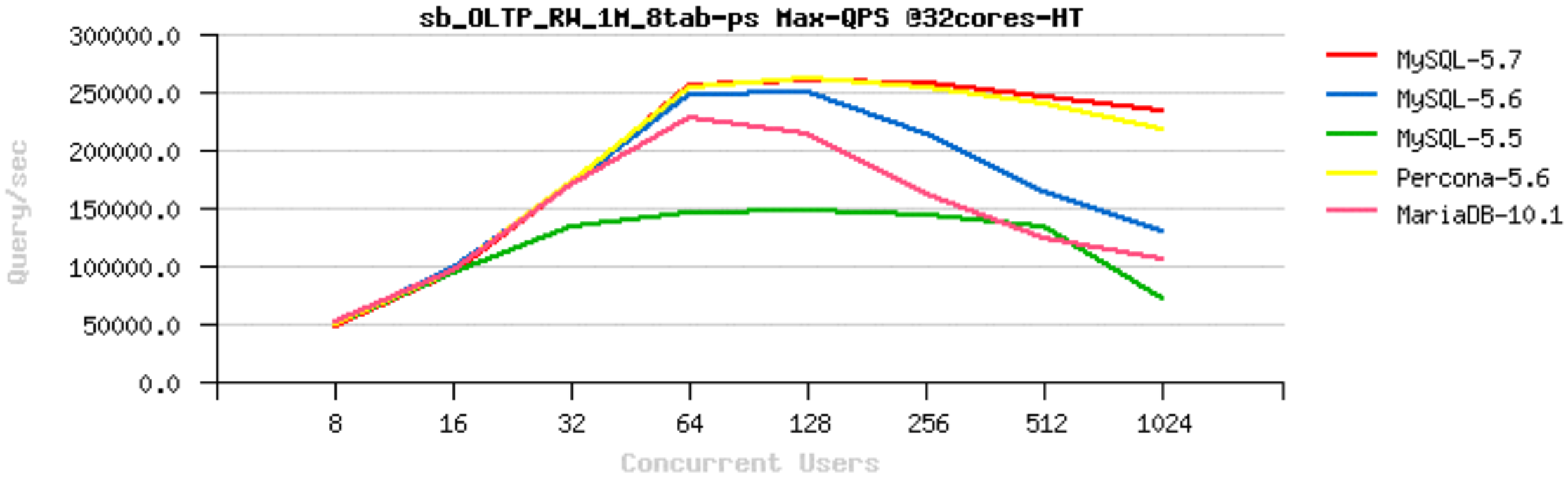
OLTP_RW : 8-tables

- Sysbench OLTP_RW 1Mx8-tables
  - 12cores-HT
  - and the winner is: MySQL 5.5 !! ;-))



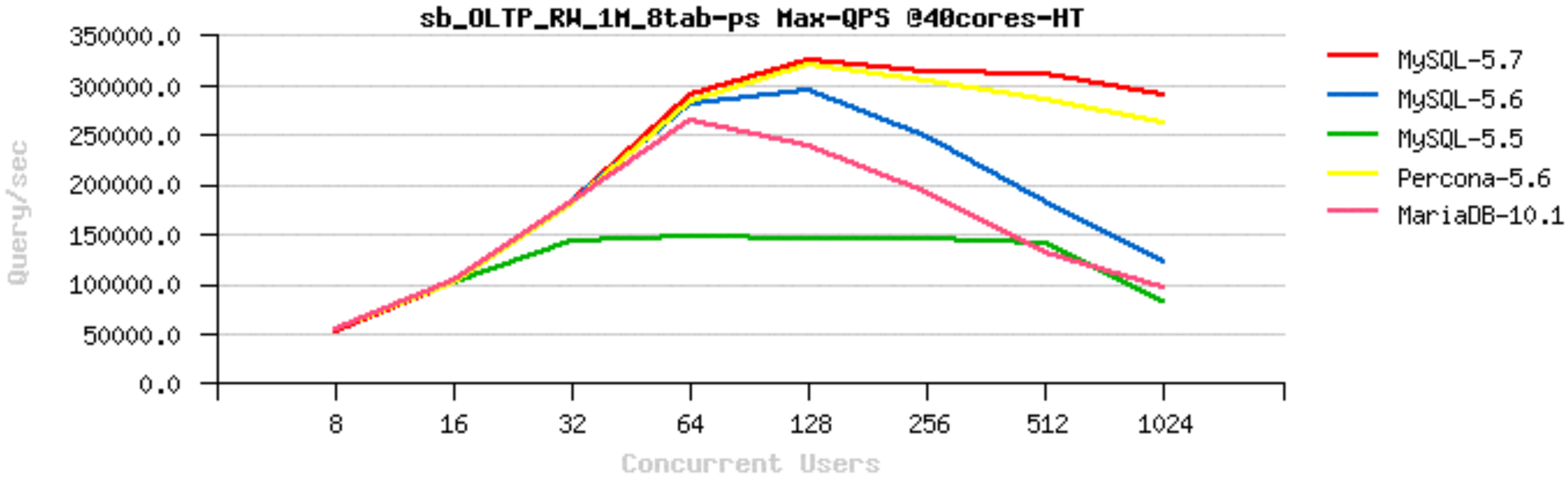
OLTP_RW : 8-tables

- Sysbench OLTP_RW 1Mx8-tables
  - 32cores-HT
  - and the winner is: rather MySQL 5.7 !! ;-))



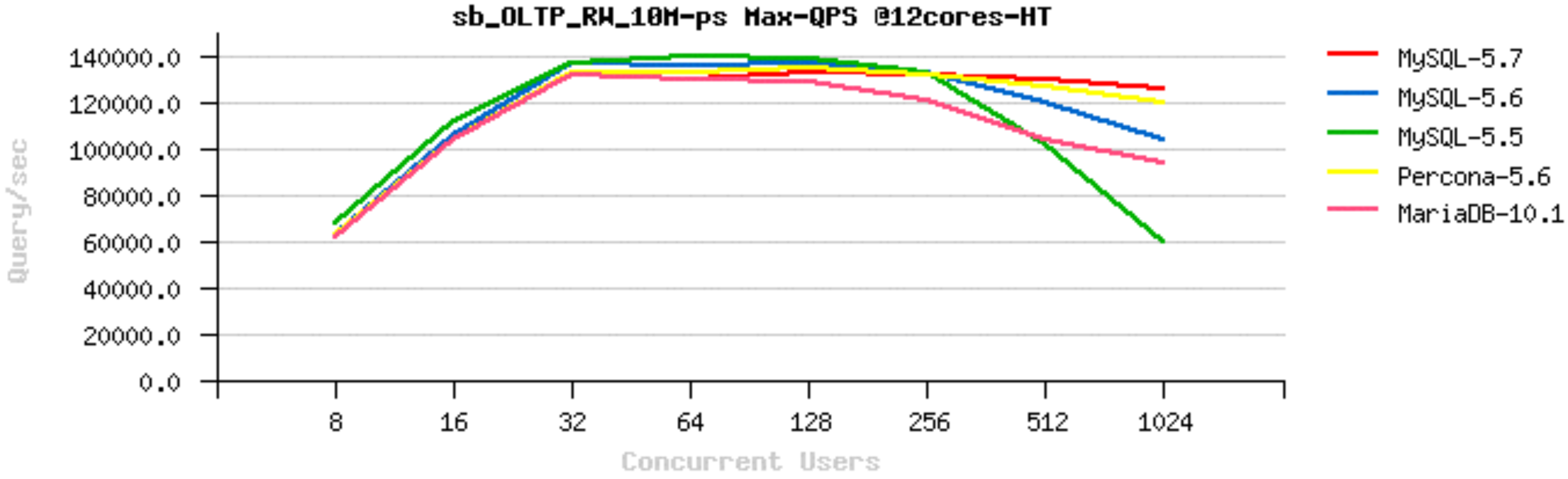
OLTP_RW : 8-tables

- Sysbench OLTP_RW 1Mx8-tables
  - 40cores-HT
  - and the winner is: rather MySQL 5.7 !! ;-))



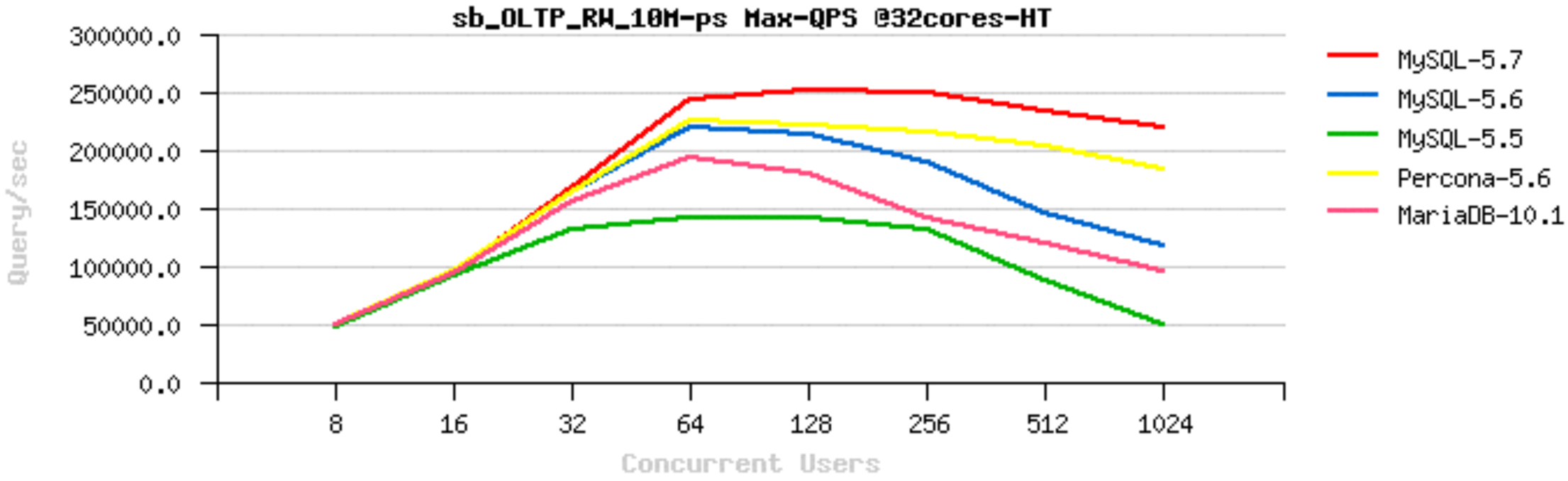
OLTP_RW : 1-table

- Sysbench OLTP_RW 10M
  - 12cores-HT
  - and the winner is: again MySQL 5.5 !! ;-))



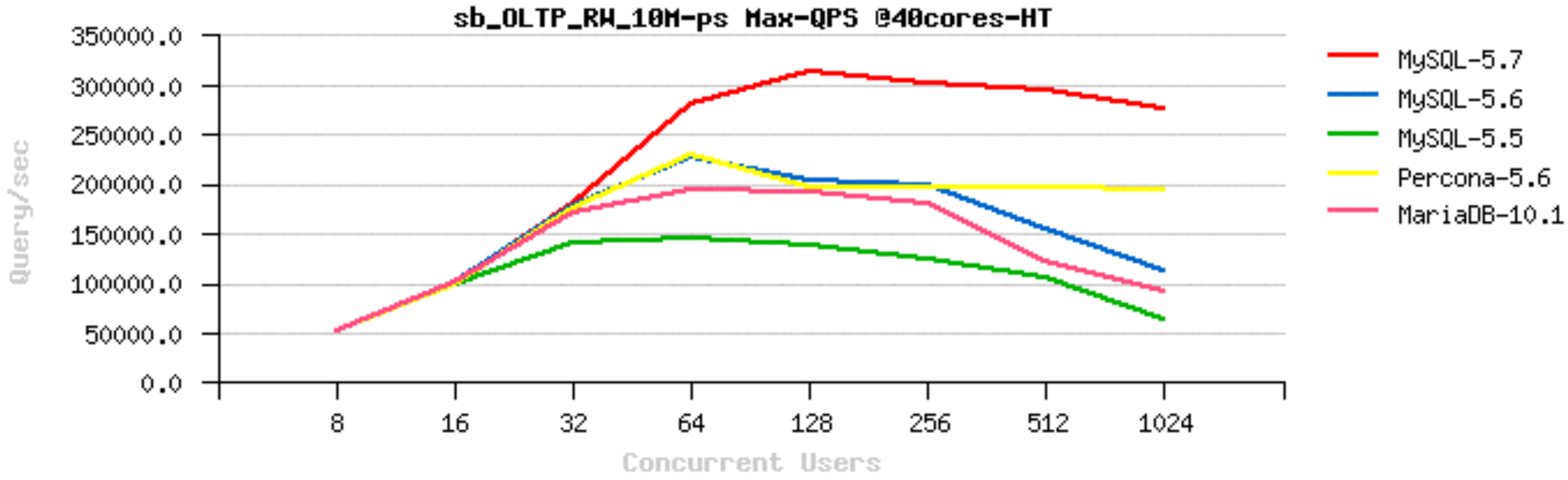
OLTP_RW : 1-table

- Sysbench OLTP_RW 10M
  - 32cores-HT
  - and the winner is: far MySQL 5.7 !! ;-))



OLTP_RW : 1-table

- Sysbench OLTP_RW 10M
  - 40cores-HT
  - and the winner is: far MySQL 5.7 !! ;-))



RW Scalability Limits and Problems

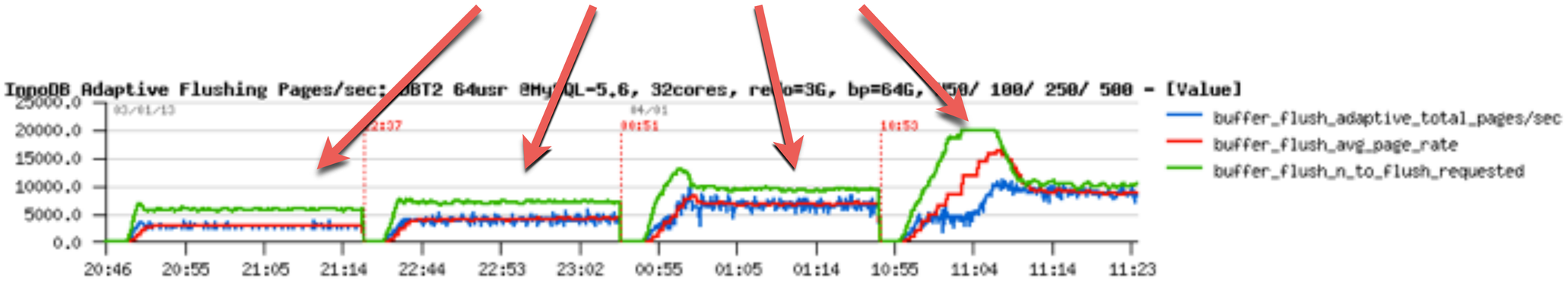
- Show-stoppers :
  - REDO log (log_sys contention) : need a re-design..
  - DBLWR Buffer (not IO, but its internal locking) : need a full re-write..
  - fil_sys mutex is limiting I/O operations rate..
- Pending problems :
  - InnoDB Purge may be lagging : need UNDO & co. re-design..
    - workaround : tune max lag to not let History Length growing by write throttling
    - 5.7 : allocated UNDO space can be truncated !! (free your disk space)
  - huge impact of writes on reads
  - IO layers are needing yet more instrumentation / observability
  - AIO needs more control / tunable(s)
  - AHI re-design
  - go yet more far with Adaptive Flushing
  - etc. etc. etc…

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:
  - on Linux : using **AIO + O_DIRECT** seems to be the most optimal for RW IO-bound
  - but always check yourself ;-)

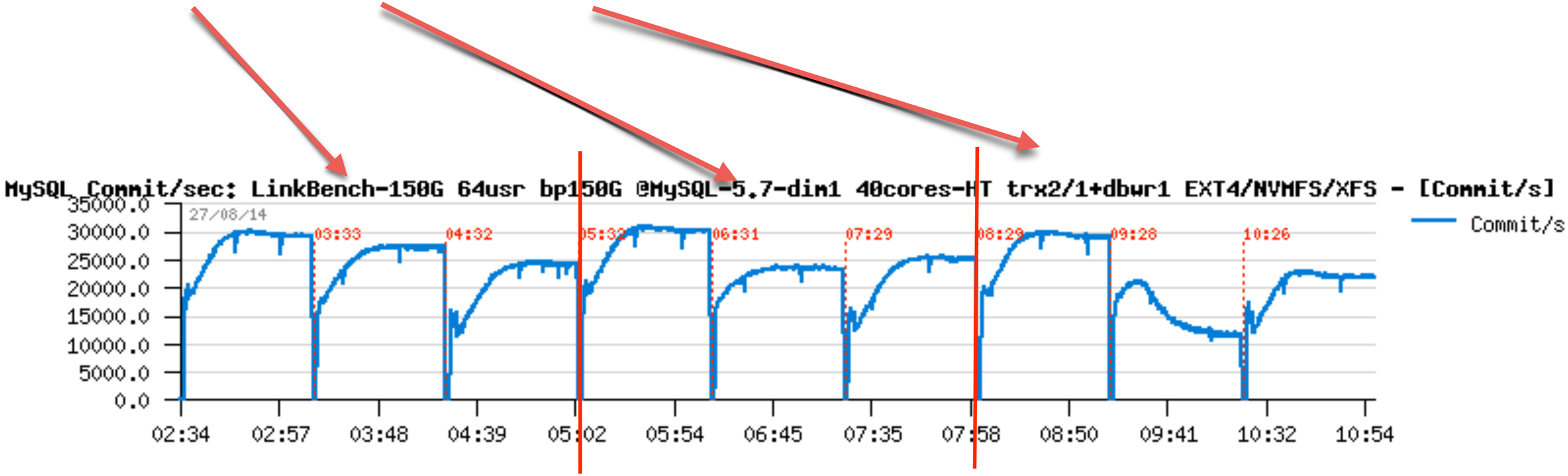
RW IO-bound “In-Memory”

- 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

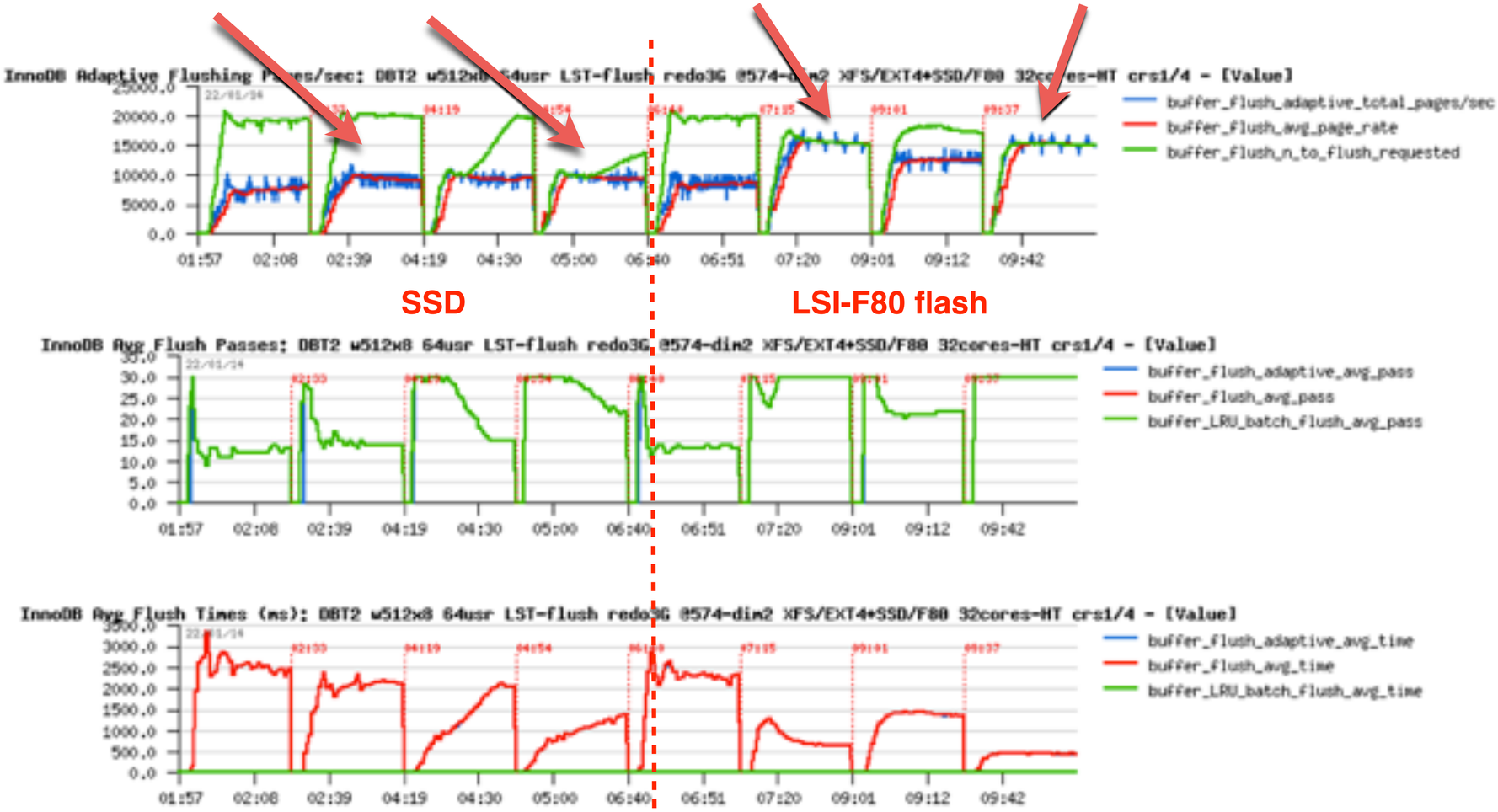
- LinkBench 150G workload
  - test cases : “safety” options on 64usr, Fusion-io
  - EXT4 -vs- NVMFS -vs- XFS



EXT4 NVMFS 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



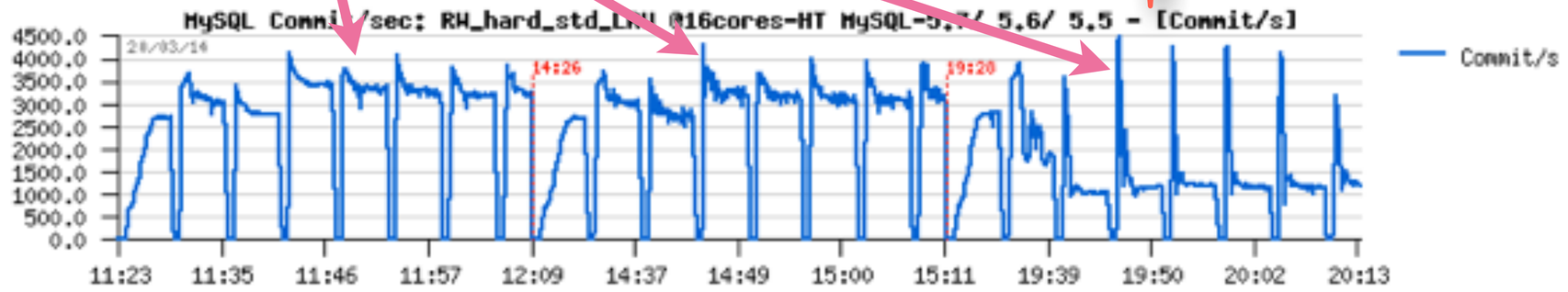
RW IO-bound “Out-of-Memory”

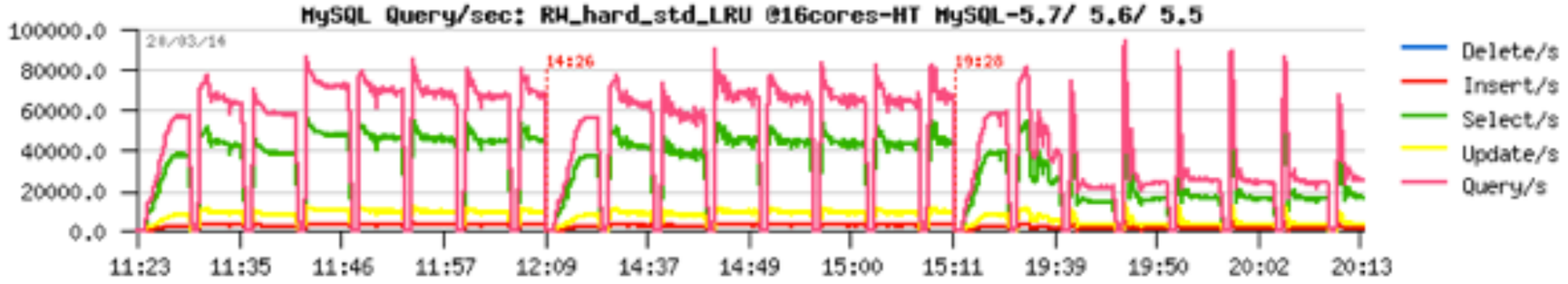
- The “entry” limit here is storage performance
  - as you’ll have a lot of IO reads..
- Once storage is no more an issue :
  - you may hit internal contentions (ex. InnoDB file_sys mutex)
  - or other engine design limitations..
  - sometimes a more optimal config settings may help..
  - but sometimes 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 !!!



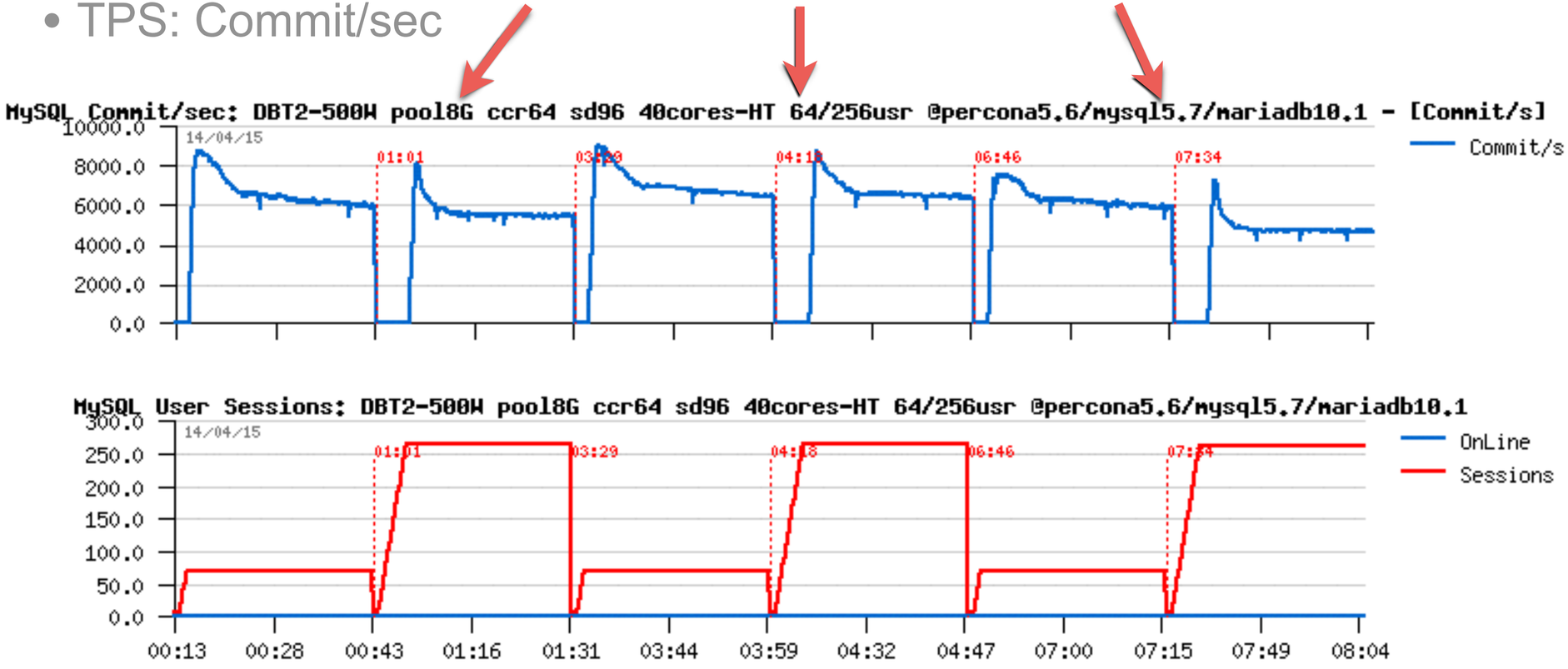


Analyzing DBT2-500W Workload @40cores-HT

- Mostly IO-bound (~100G database)
  - so, storage layer: Fusion-io flash, EXT4
- Test cases :
  - engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
  - concurrent user sessions: 64, 256
  - Buffer Pool size: 8G (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
  - …

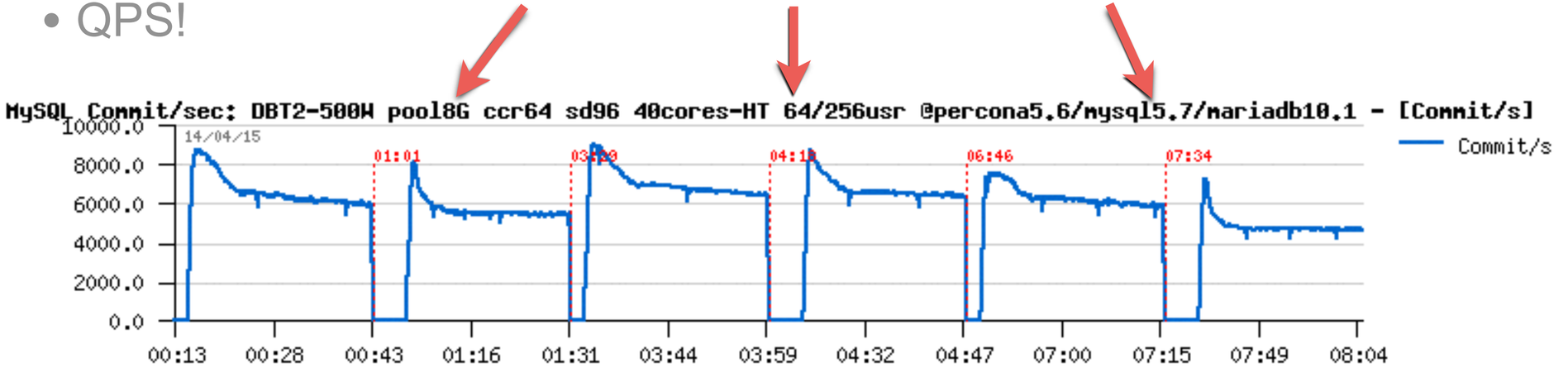
DBT2-500W Workload @40cores-HT

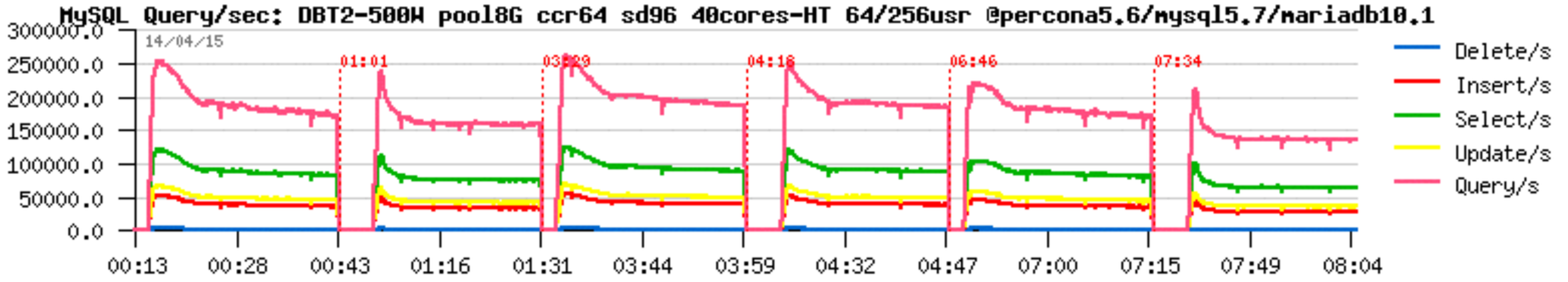
- LRU-bound (BP=8G): Final Results
  - engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
  - TPS: Commit/sec



DBT2-500W Workload @40cores-HT

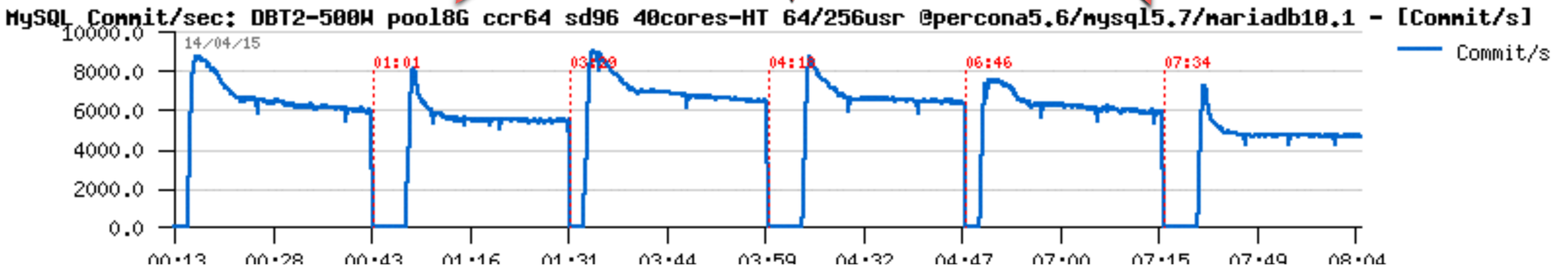
- LRU-bound (BP=8G): Final Results
  - engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
  - QPS!

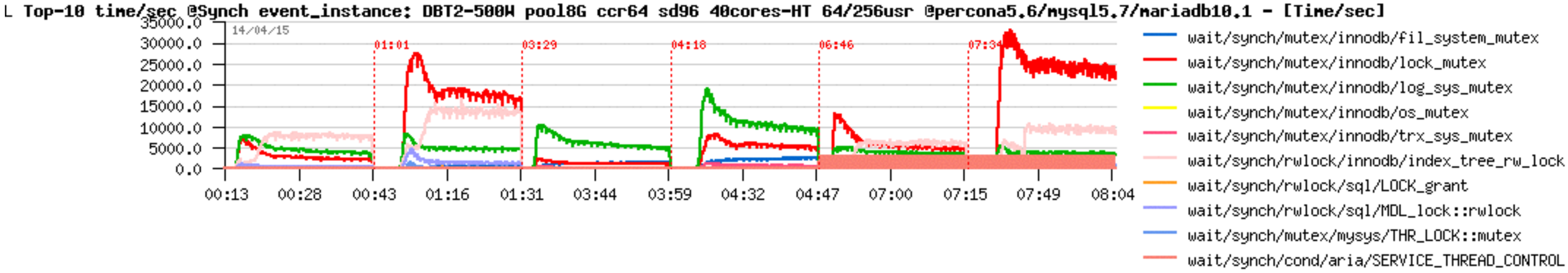




DBT2-500W Workload @40cores-HT

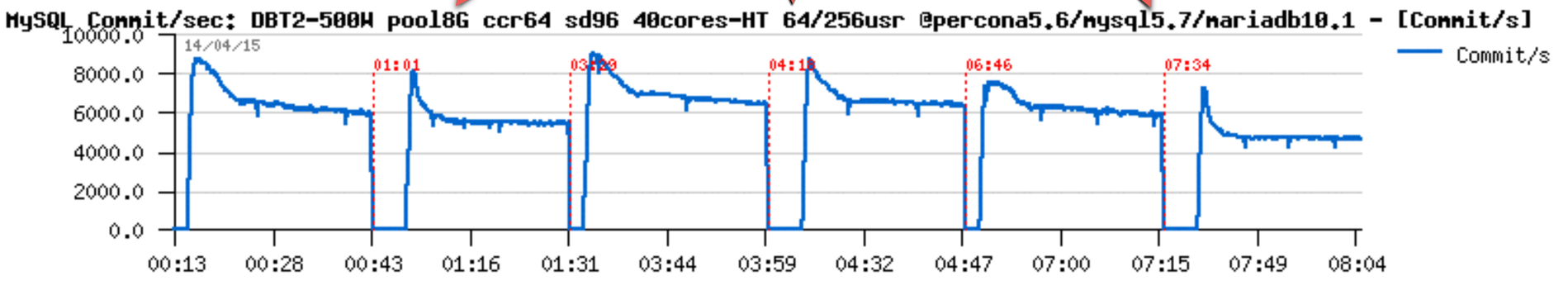
- LRU-bound (BP=8G): Final Results
  - engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
  - Lock waits: index lock impact…

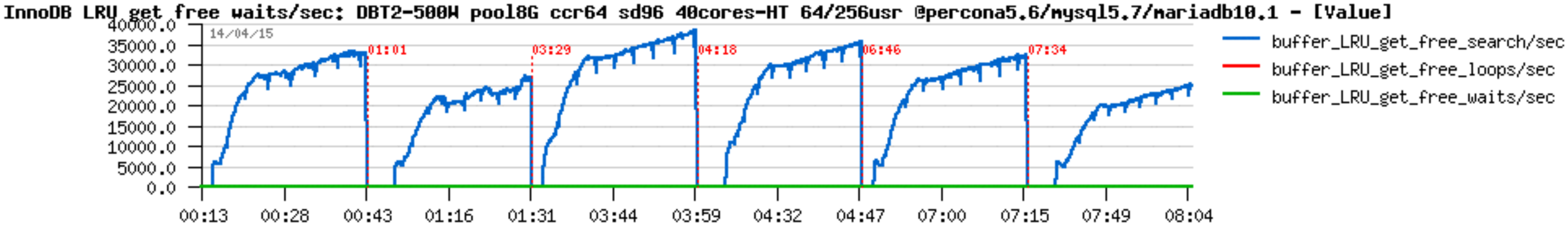




DBT2-500W Workload @40cores-HT

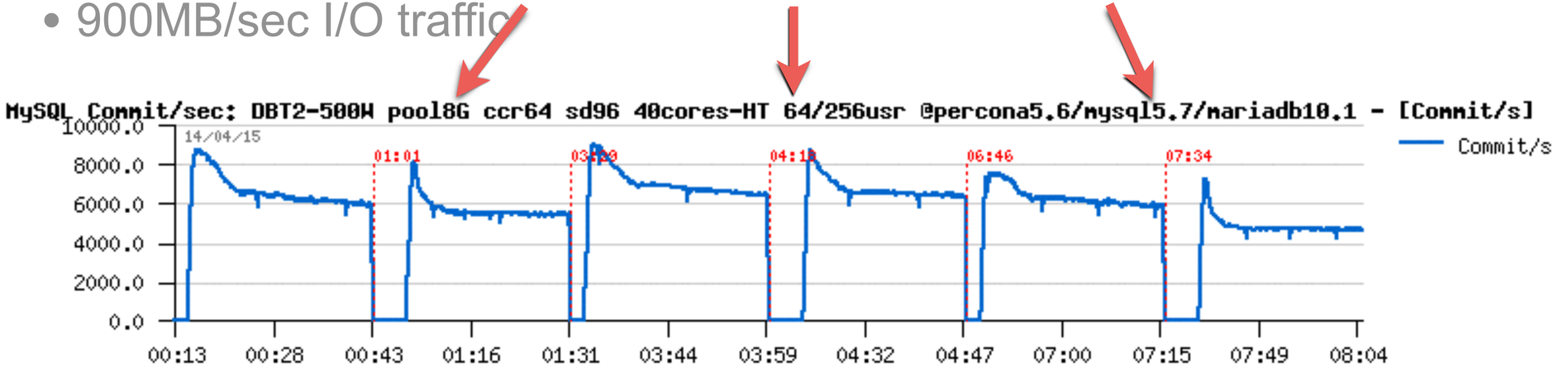
- LRU-bound (BP=8G): Final Results
  - engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
  - up to 40K page reads/sec rate

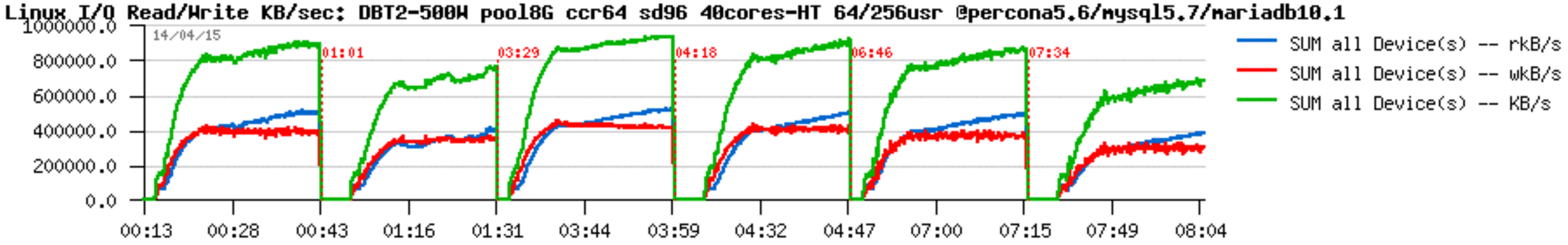




DBT2-500W Workload @40cores-HT

- LRU-bound (BP=8G): Final Results
  - engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
  - 900MB/sec I/O traffic

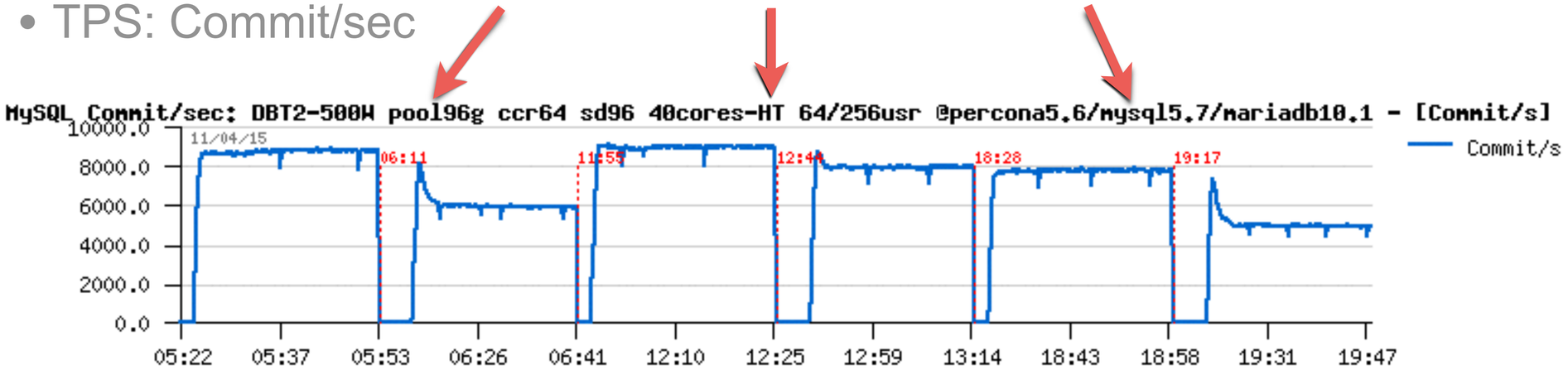


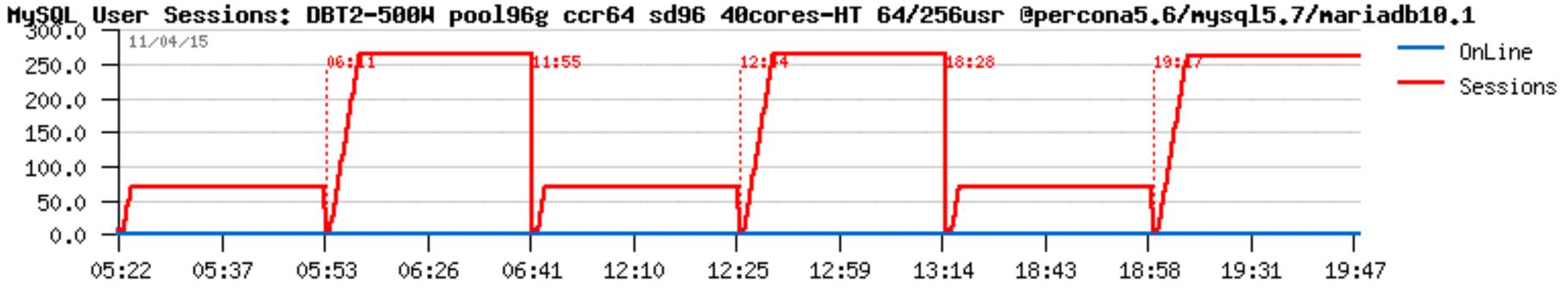


ORACLE

DBT2-500W Workload @40cores-HT

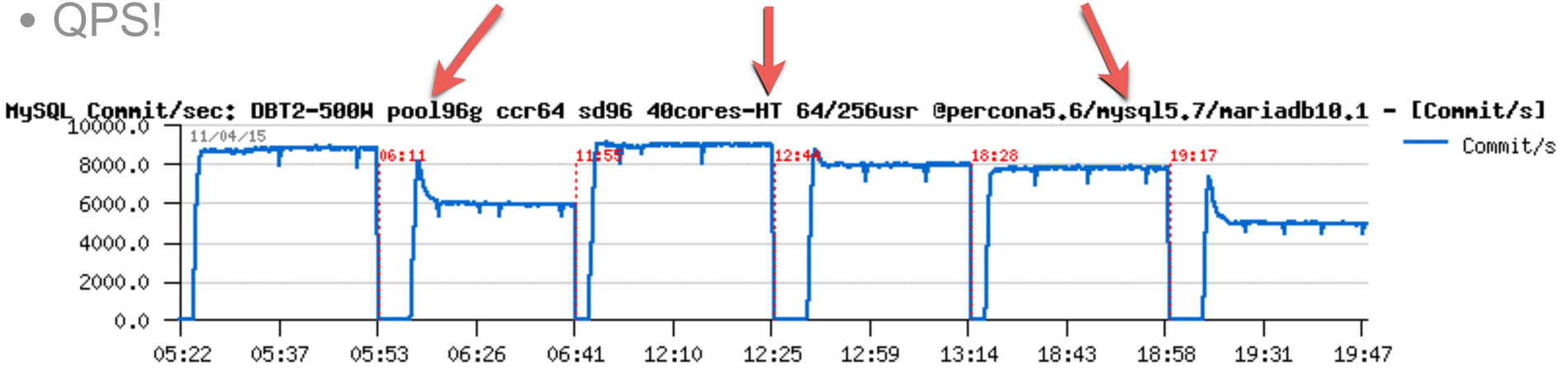
- Flushing-bound (BP=96G): Final Results
  - engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
  - TPS: Commit/sec

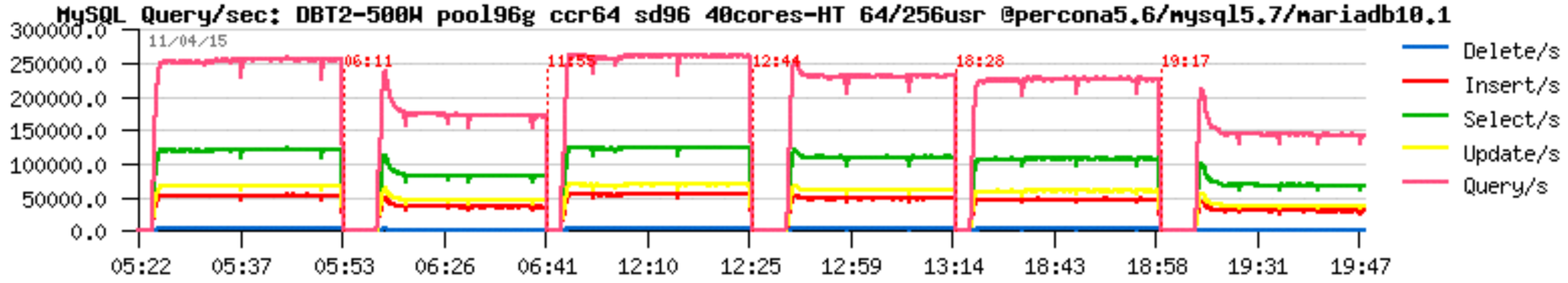




DBT2-500W Workload @40cores-HT

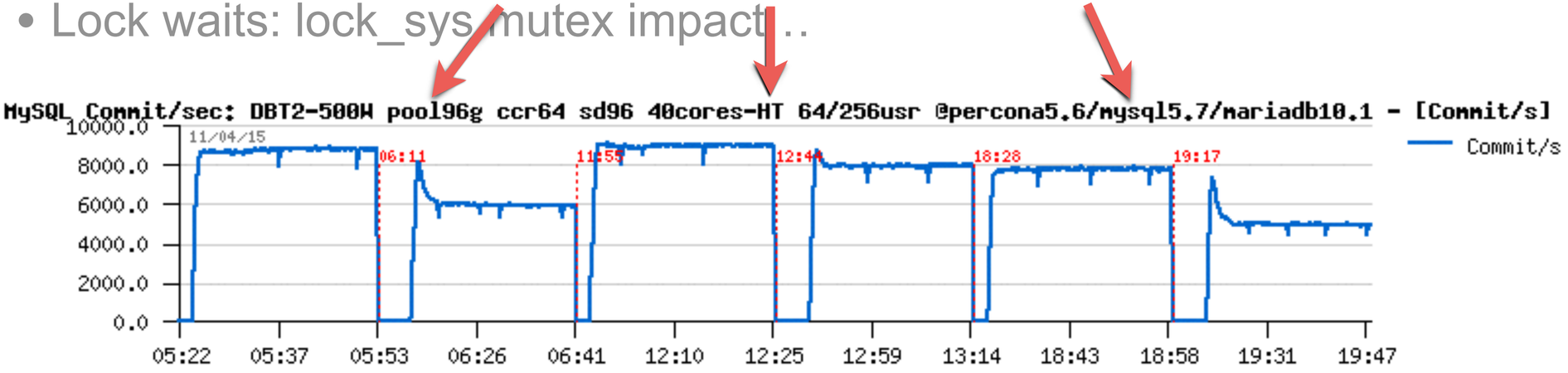
- Flushing-bound (BP=96G): Final Results
  - engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
  - QPS!

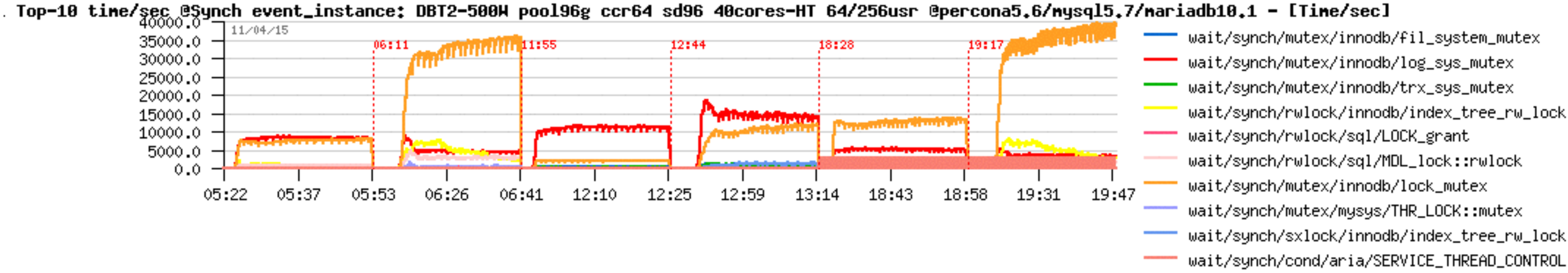




DBT2-500W Workload @40cores-HT

- Flushing-bound (BP=96G): Final Results
  - engines: Percona Server 5.6 / MySQL 5.7 / MariaDB 10.1
  - Lock waits: lock_sys mutex impact ..





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

Come to see us at OpenWorld-2015 in SF, 25-30/Oct. !!!

ORACLE

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.