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 ;-)

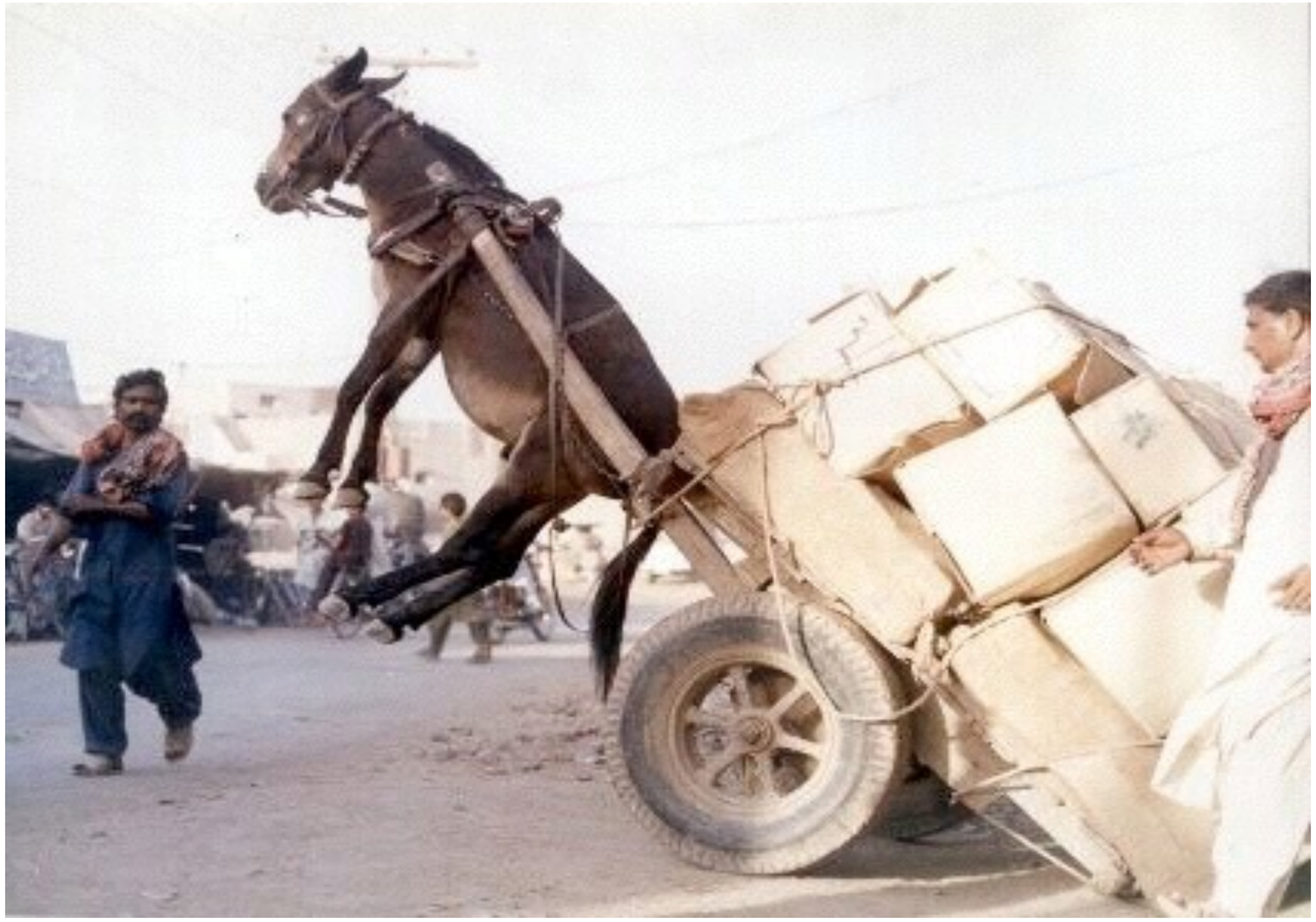
Why MySQL Performance ?...

Why MySQL Performance ?..

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

Why MySQL Performance ?..

- Until it did not reach its limit..



Why MySQL Performance ?..

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

Why MySQL Performance ?..

- And reach a similar limit..

Why MySQL Performance ?..

- Analyzing your workload performance and testing your limits may help you to understand ahead the resistance of your solution to incoming potential problems ;-)

Why MySQL Performance ?..

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

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

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

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

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

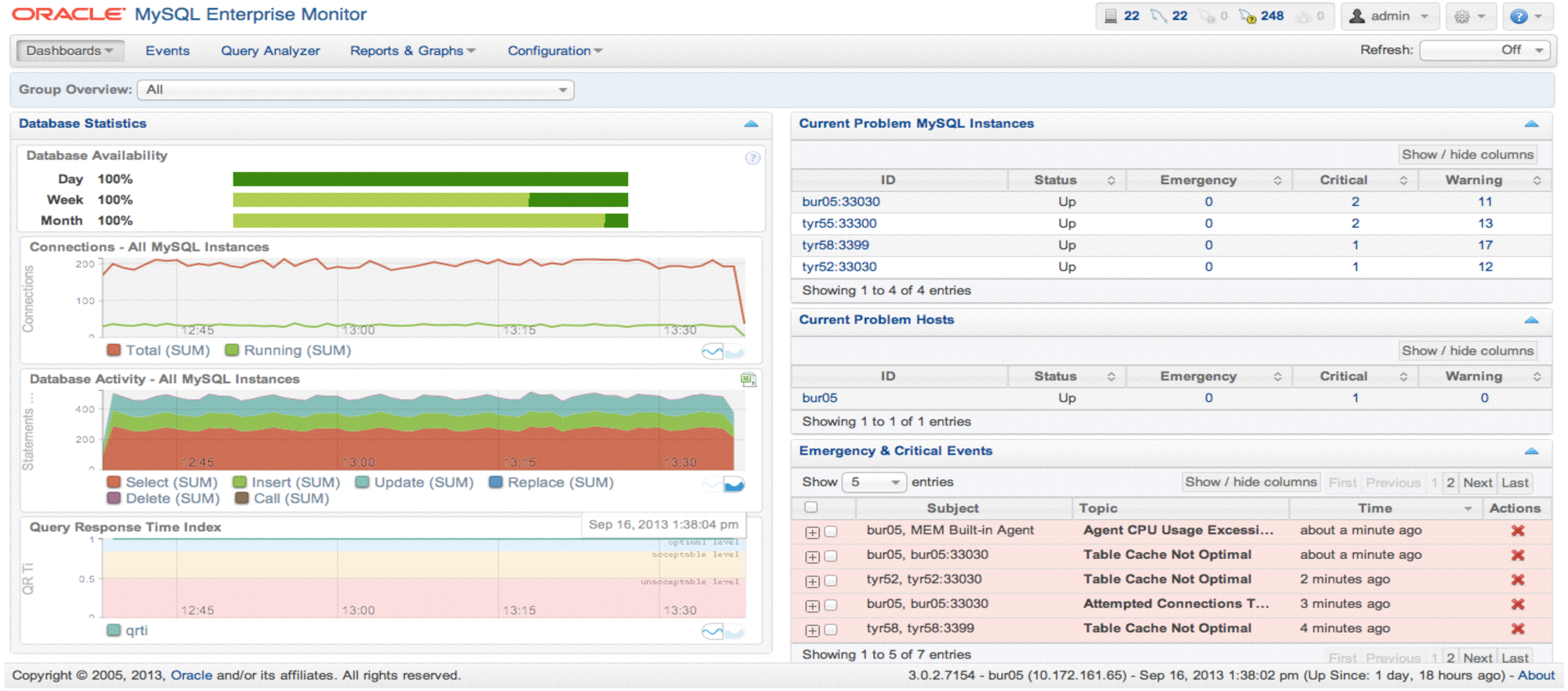
USE YOUR BRAIN !!!... ;-)

#2 - Monitoring is THE MUST !

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

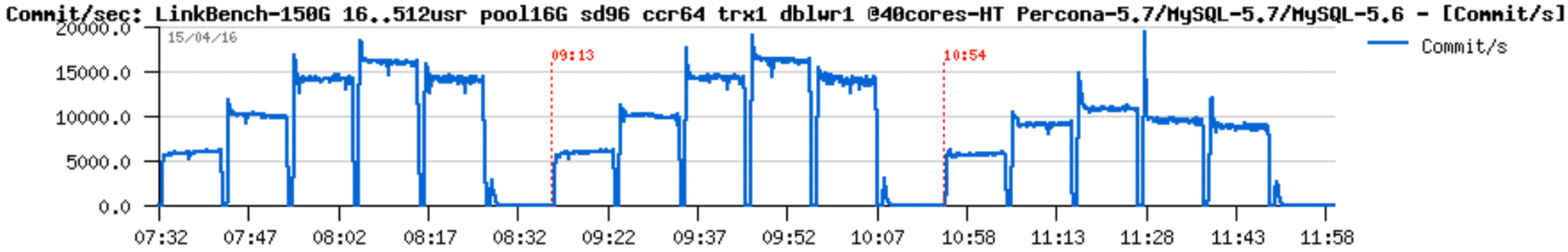
MySQL Enterprise Monitor

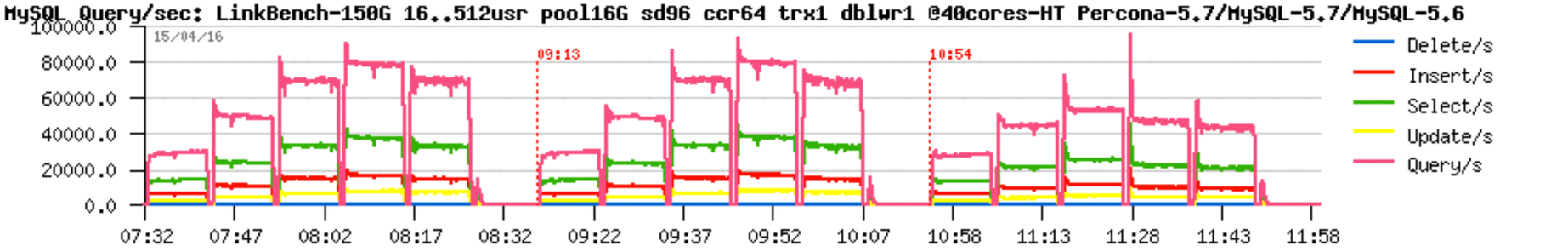
- Fantastic tool!
  - Did you already try it?.. Did you see it live?..



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
    - for ex: run in loop "show processlist", etc..
- well, nothing is coming for free, so **think** about what you're doing !
- (#1 best practice once again ;-))

The following materials are about…

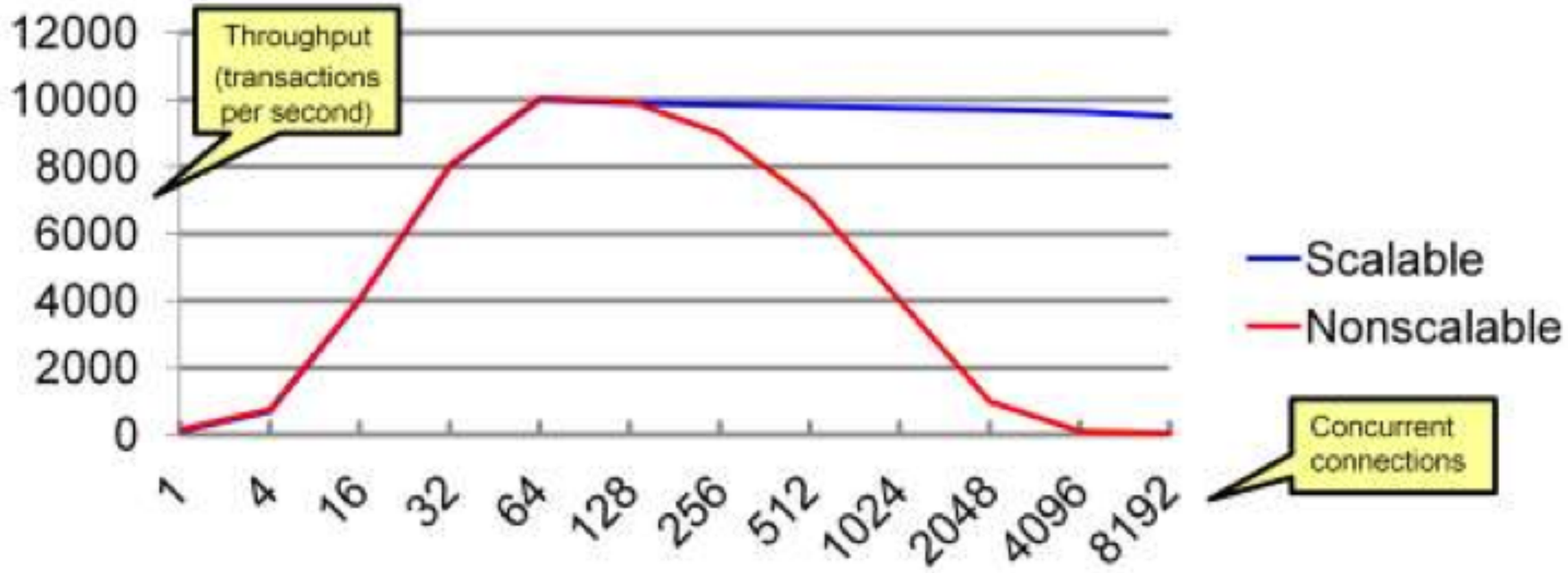
- Single MySQL Instance Performance & Scalability
  - single HW host
  - no replication
  - just to understand how far MySQL Server may scale..
  - what are the limits
  - what to care about ahead
  - which situations are absolutely to avoid.
- NOTE:
  - this talk is mainly focused on **benchmark results** as for Apr.2016
  - all details about related tuning are tomorrow

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)
  - intel 2CPU : 44cores-HT (March 2016)..
  - …
- 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 ;-))

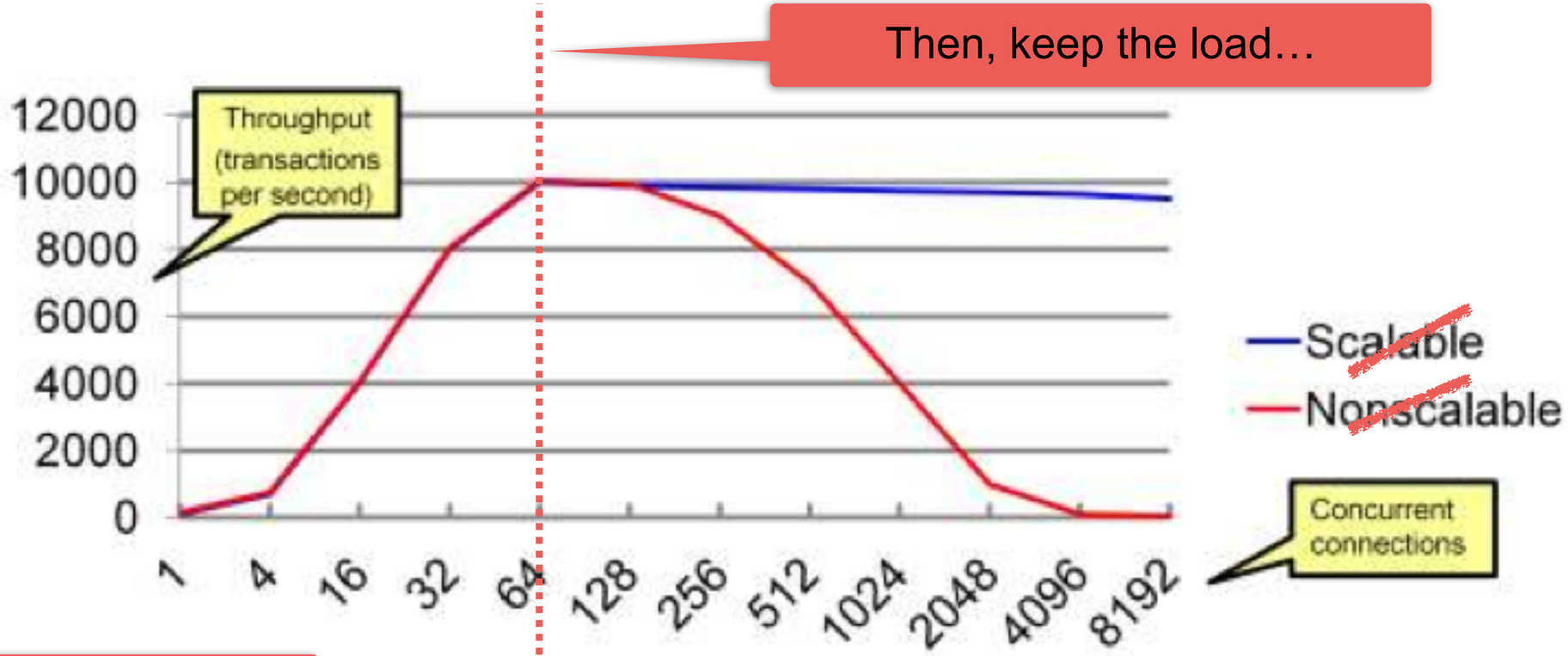
A B-shit Slide…

- Odd interpretation of Scalability…



A B-shit Slide… (2)

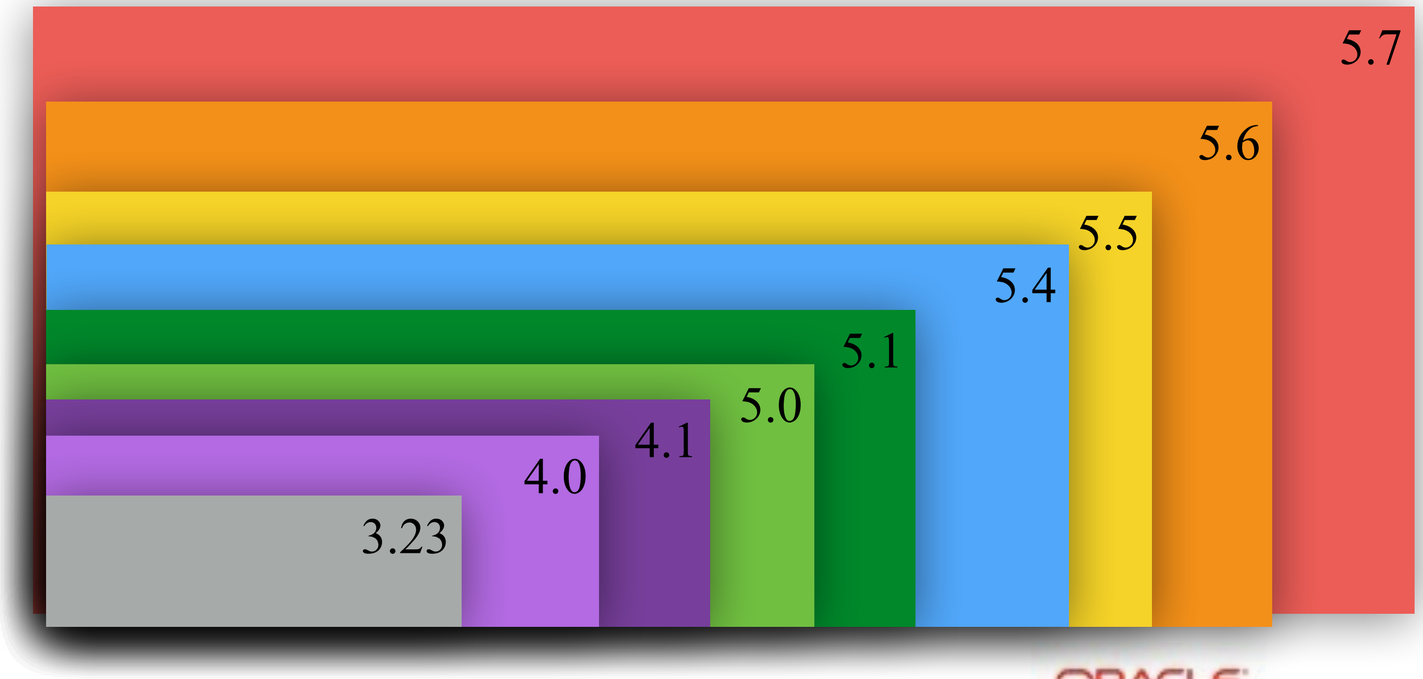
- Odd interpretation of Scalability…



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

MySQL Performance Evolution

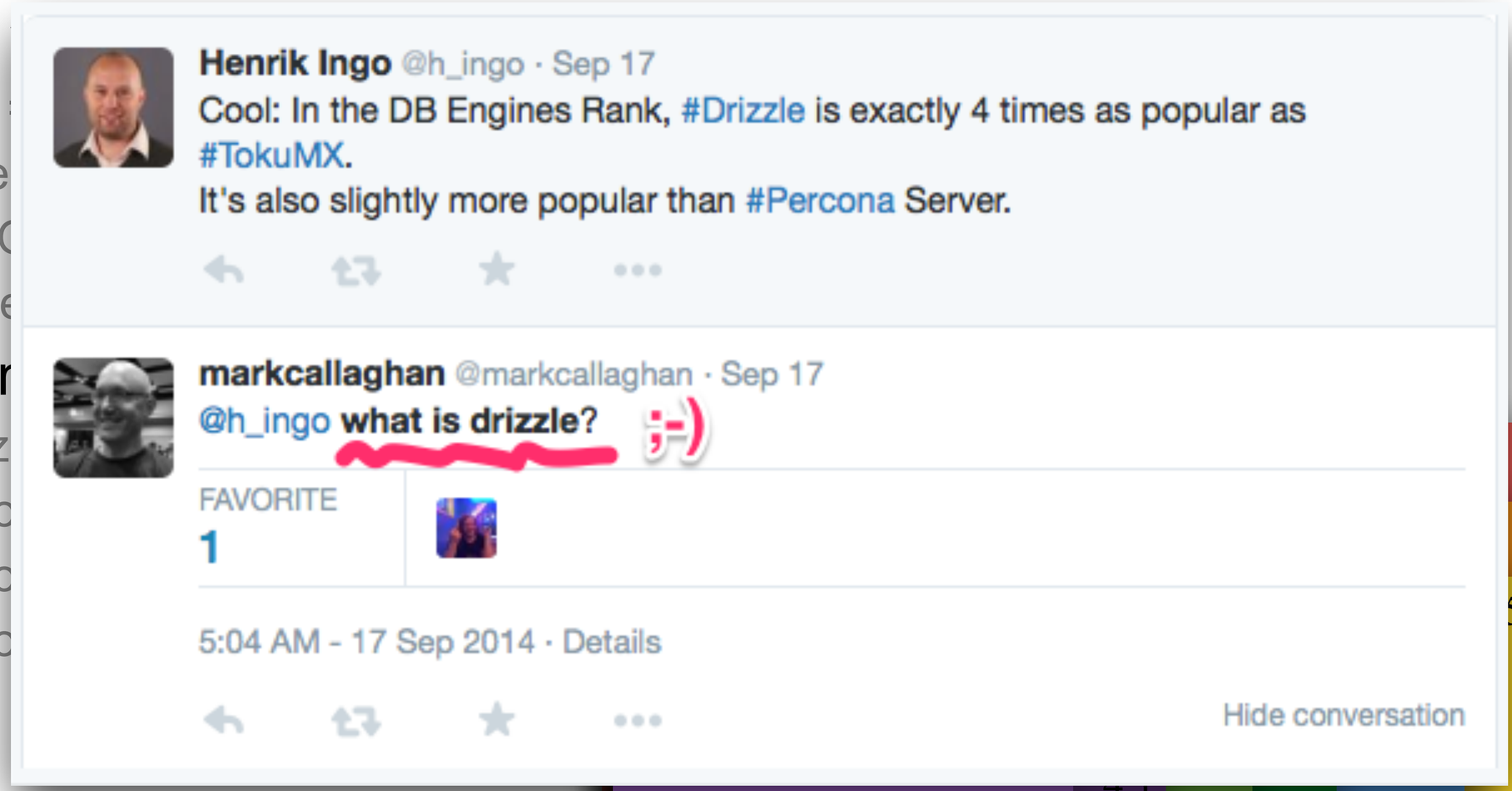
- From version-to-version :
  - 3.23 => 4.0 => 4.1 => 5.0 => 5.1 => 5.4 => 5.5 => 5.6 => 5.7 …
  - more features => longer code path.. (see: “What is new in MySQL 5.7” by Geir, Morgan,..)
  - MySQL/InnoDB code is very sensible to CPU cache(s)..
  - single user / low load => going slower..
- Looking back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



MySQL Performance Evolution

- From
  - 3.23
  - more
  - MyS
  - single
- Lookir
  - Drizz
  - do yc
  - do yc
  - do yc

r, Morgan,..)



Performance Investigation Efforts (relative)

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

“Don’t Trust Benchmarks !!!” ;-)

- You were told this already, right ?
- Well, generally it’s true..
- But my own you can trust ;-))
  - (kidding ;-))
- Seriously :
  - #1 you should not base your conclusions on a single benchmark result
  - #2 to be able to evaluate the result you’re seeing, you first have to understand the benchmark workload and its impact on database engine..
- Usually the obtained numbers itself are less important comparing to what is behind the numbers ;-))
  - some “industrial” benchmarks may not spell at al to a final user..
  - MySQL Performance is not yet there ;-))
  - **most MySQL Benchmarks today are rather focused on problem resolving !**

Benchmarking & Tuning : Why & What ?

- Why Benchmarks ?..
  - the last thing you should do is to validate your tuning tweaks on live production ;-)
  - rather take a time to create a test case to reproduce the problem
  - then test the fix on “dev” (or “benchmark”) server(s)..
- The best Benchmark Workload for you ?
  - the Benchmark reproducing your own production conditions !
  - the collection of your production test cases may quickly become your own Benchmark Suite to validate any tuning or HW changes impact..
- If you don’t have :
  - adopt “standard” / existing benchmark workload
  - test your tuning, your HW, your database Engine
  - try to adapt the workload conditions to be more close to your production
  - etc..

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 <= “Entry Ticket”
  - 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 ;-)

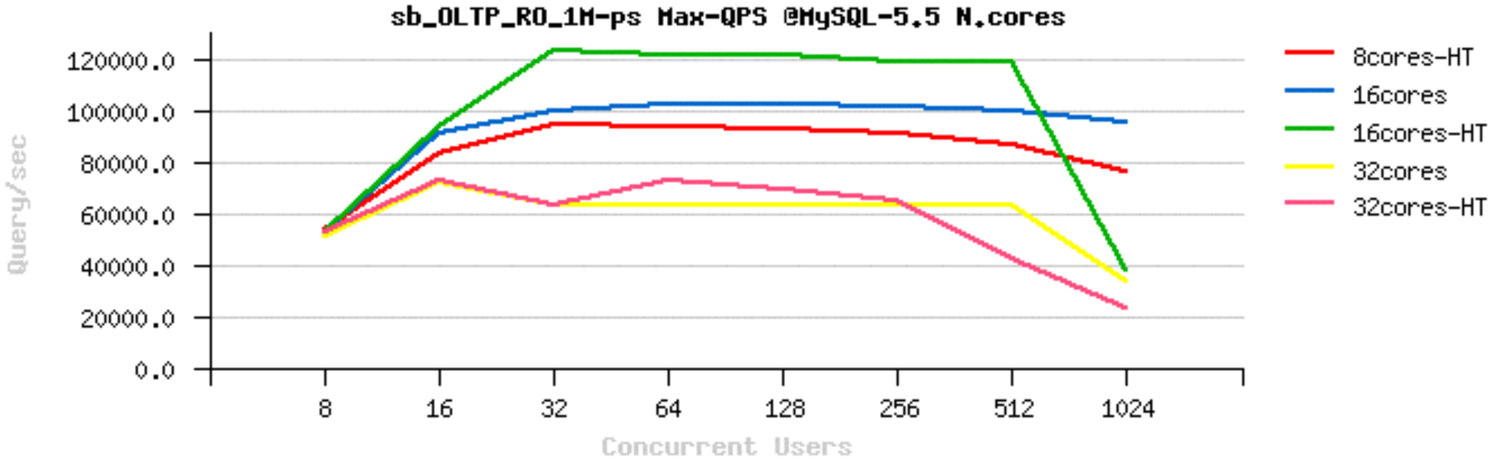
Testing Workloads: from a more Simple to more Complex

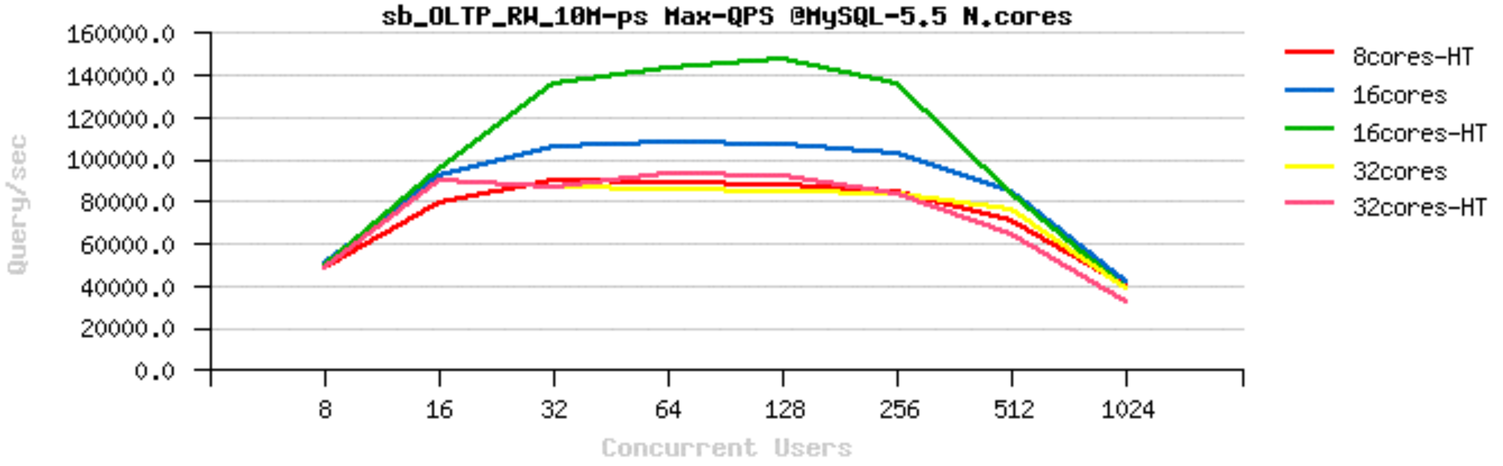
- Read-Only (RO) :
  - Nothing more simple when comparing DB Engines, HW configs, etc..
  - RO In-Memory : data set fit in memory / BP / cache
  - RO IO-bound : data set out-passing a given memory / BP / cache
- Read+Write (RW) :
  - I/O is **ALWAYS** present ! - storage performance matters a lot !
  - 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 !!!

Why so much attention to RO Performance in MySQL 5.7 ?..

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

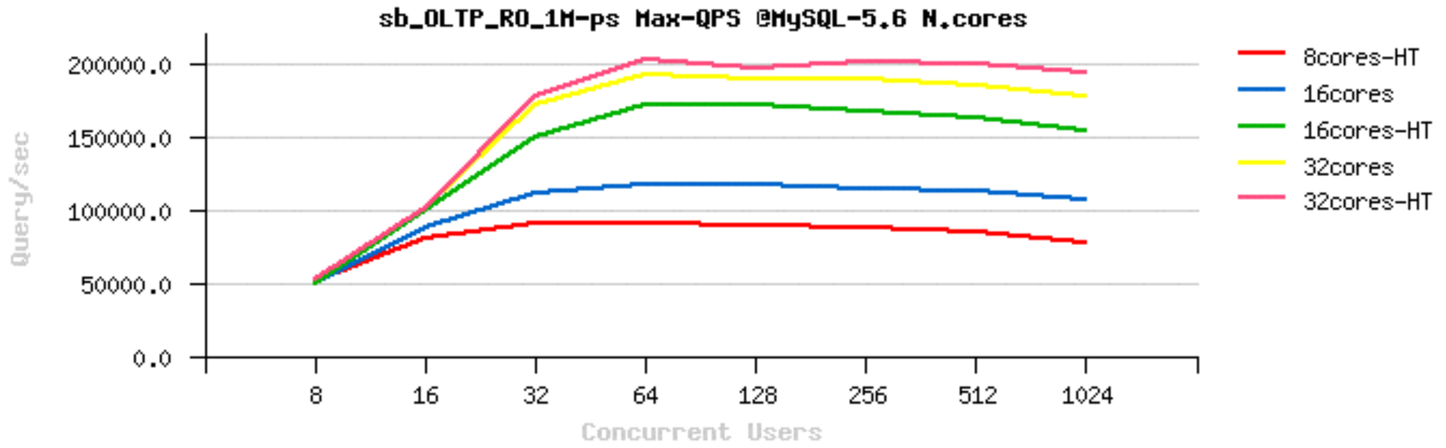
- MySQL 5.5 : RO & RW
  - QPS Max on 16cores
  - worse on 32cores
  - Note: RW out-pass RO!

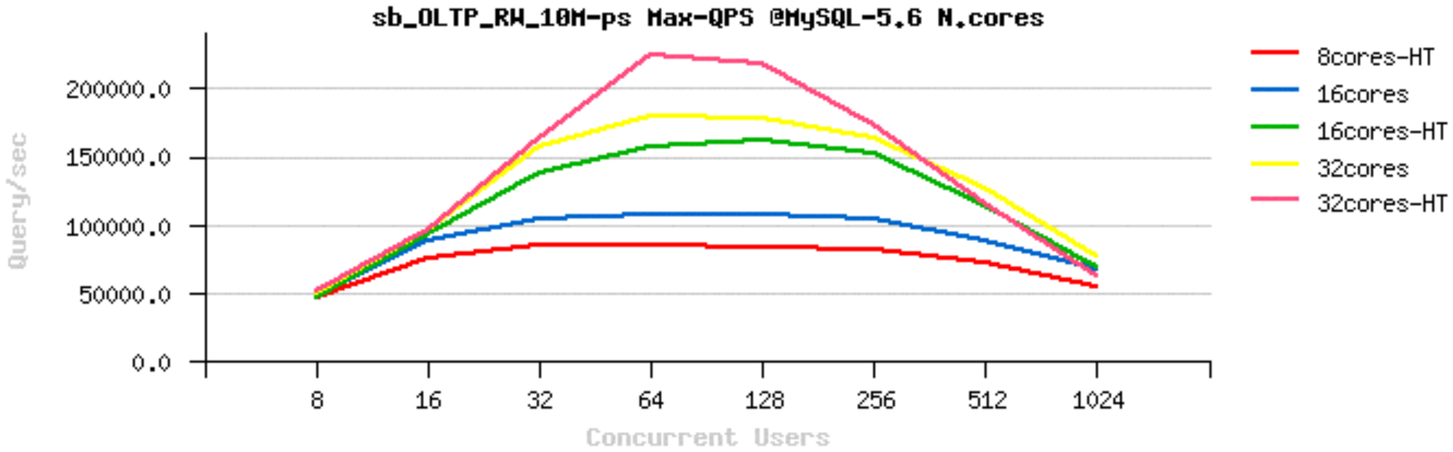




From where we're coming with MySQL 5.7 ?..

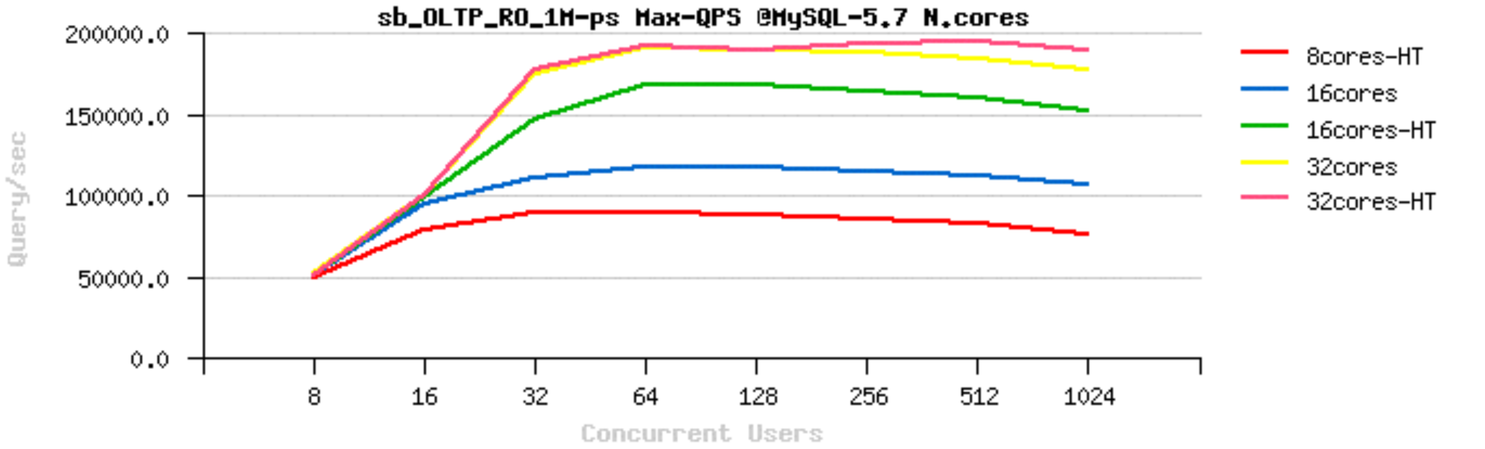
- MySQL 5.6 : RO & RW
  - not lower on 32cores!! ;-)
  - RW out-pass RO !!..??

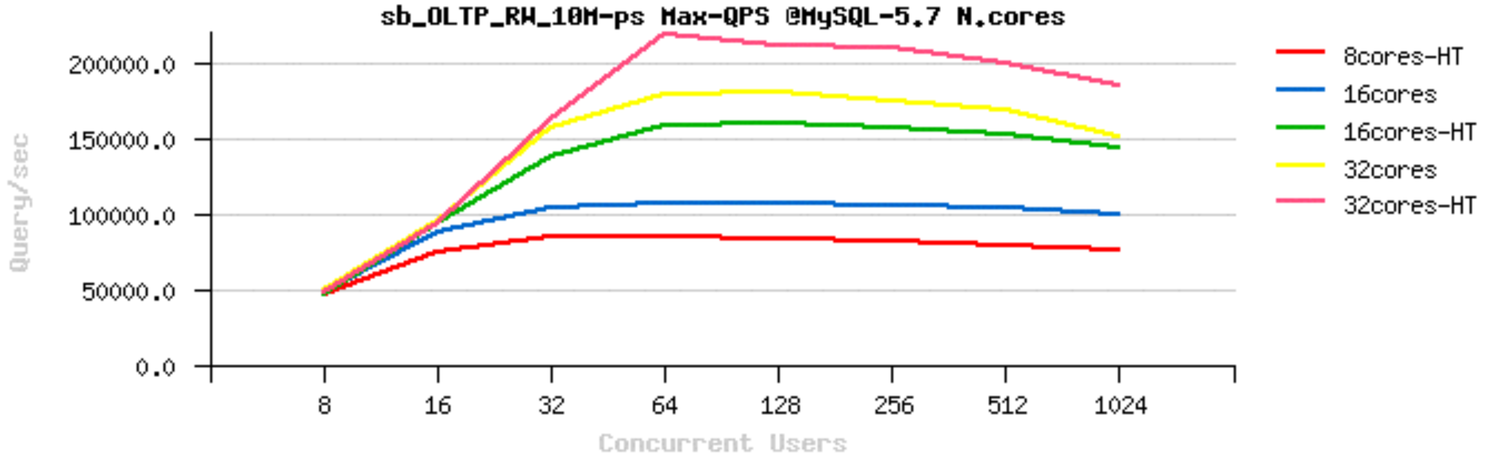




From where we're coming with MySQL 5.7 ?..

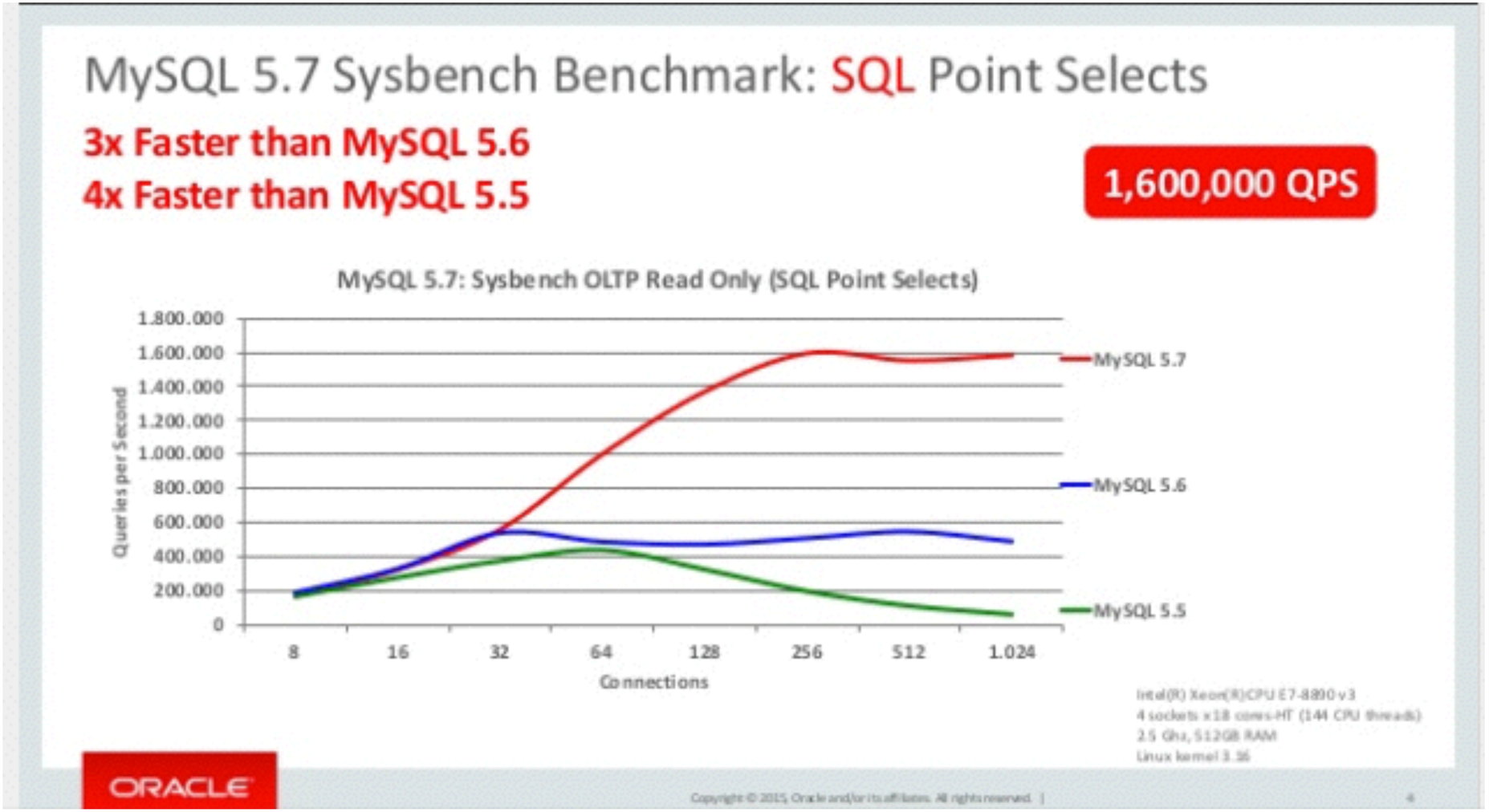
- MySQL 5.7.1 : RO & RW
  - more stable than 5.6
  - **RW** out-pass RO !!..





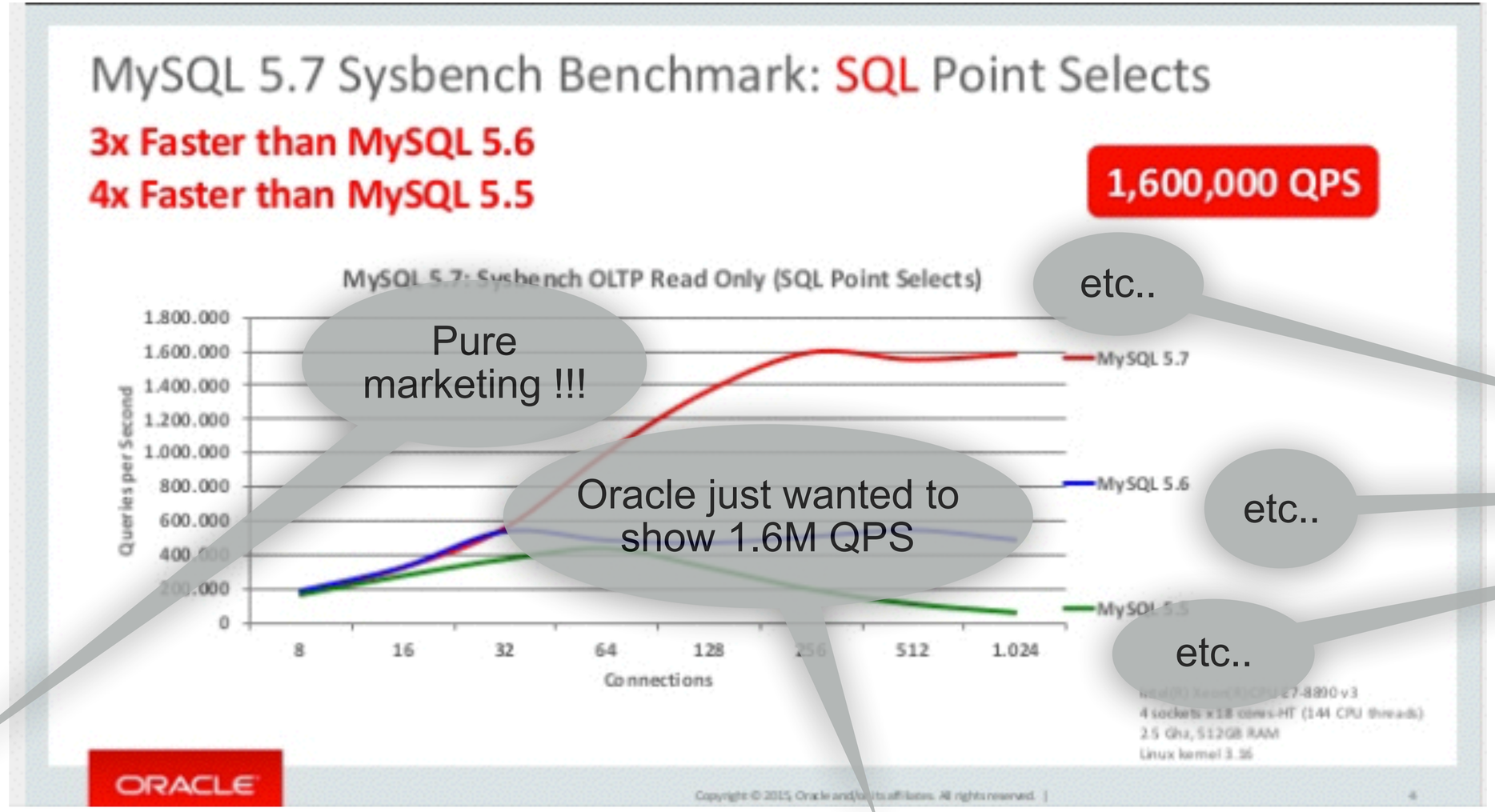
MySQL 5.7 : 1.6M QPS

- What is behind this number ?..



MySQL 5.7 : 1.6M QPS

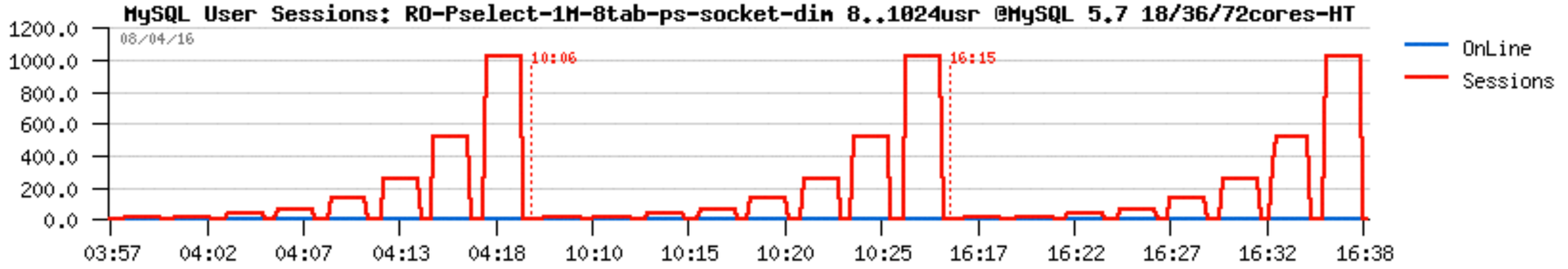
- What is behind this number ?..



Marketing ?..

- Why you did not hear then about **1.8M QPS** ?.. ;-))
  - same 72cores-HT server, same MySQL 5.7
  - we are not running after numbers ;-)
  - numbers are just reflecting what is behind..

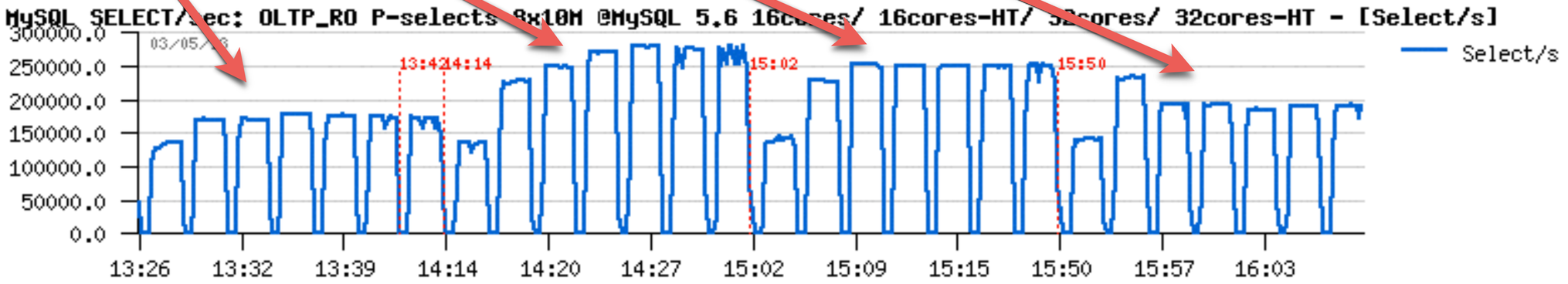
1.8M QPS

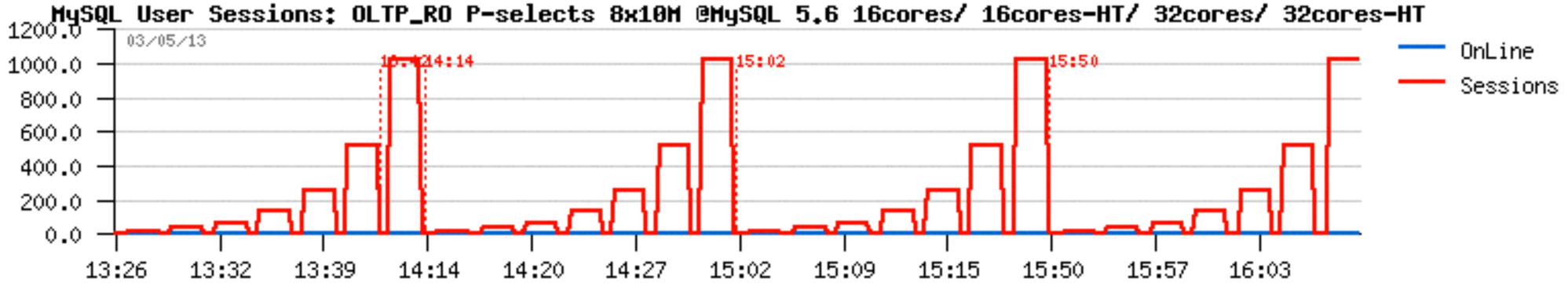


ORACLE

Behind the numbers...

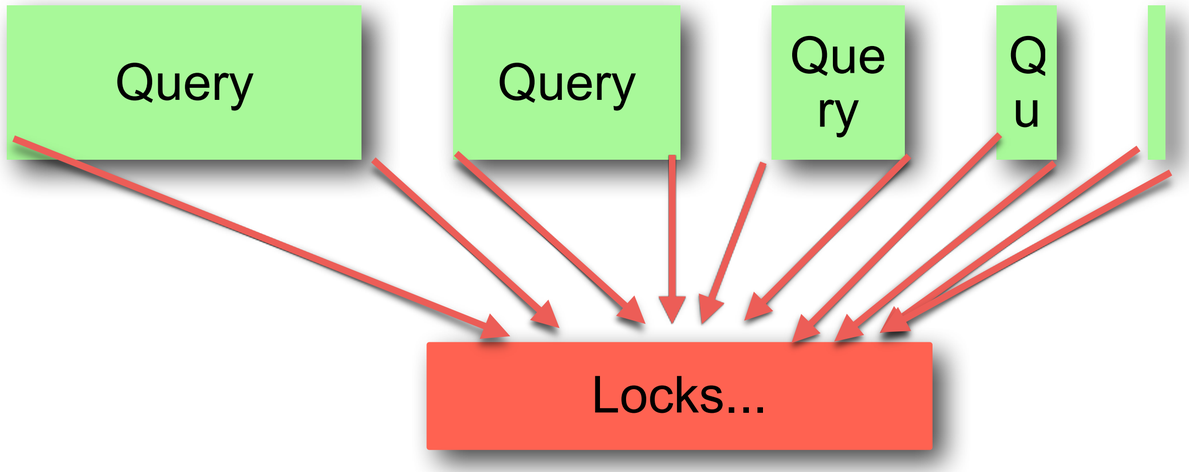
- MySQL 5.6, RO Point-Select Performance
  - 16cores, 16cores-HT, 32cores, 32cores-HT

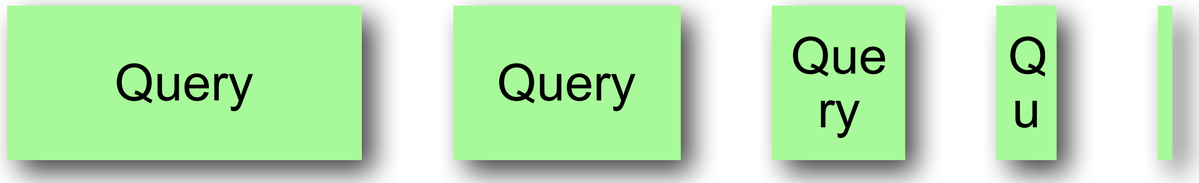




Behind the numbers...

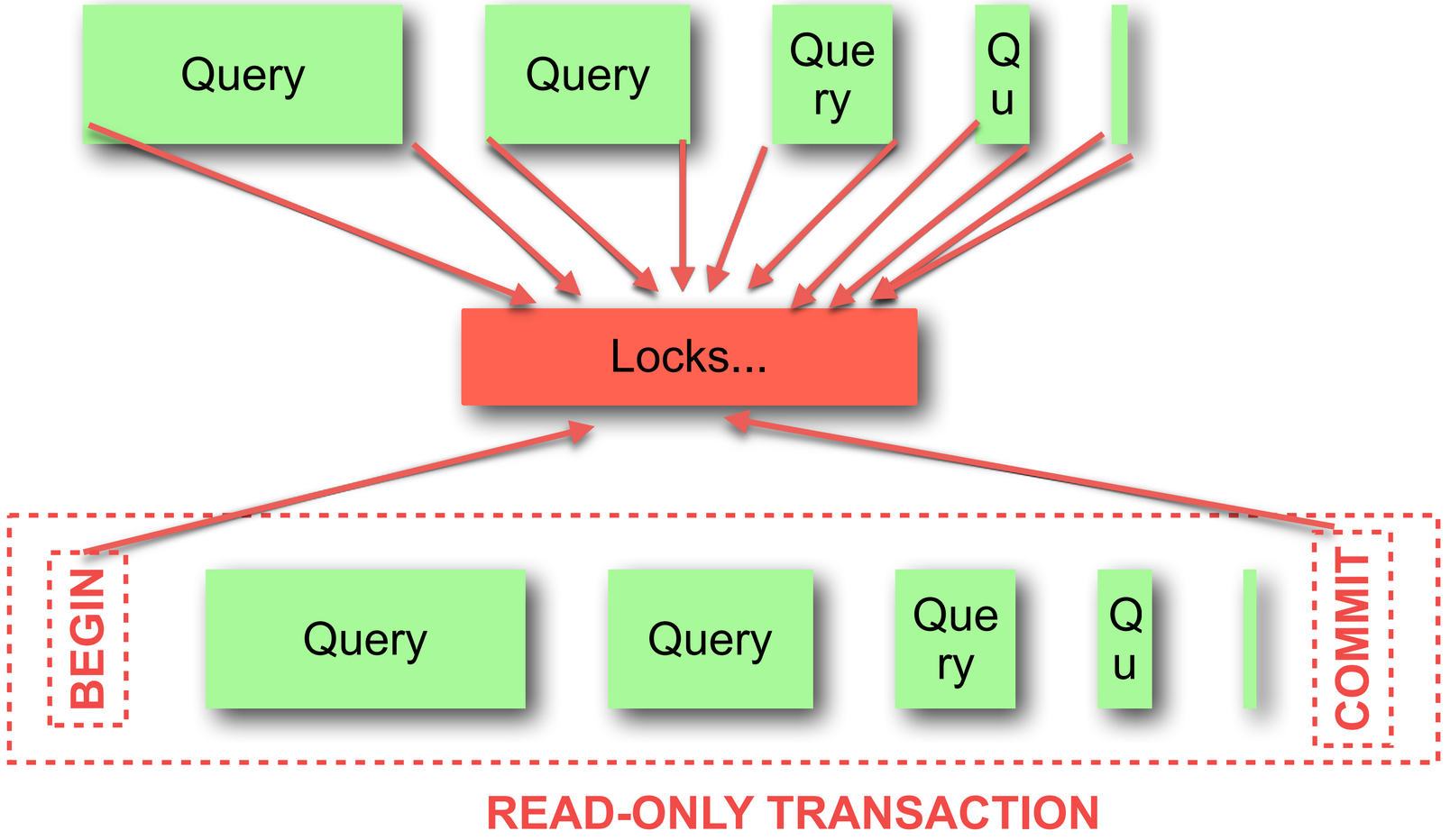
- Why ?..





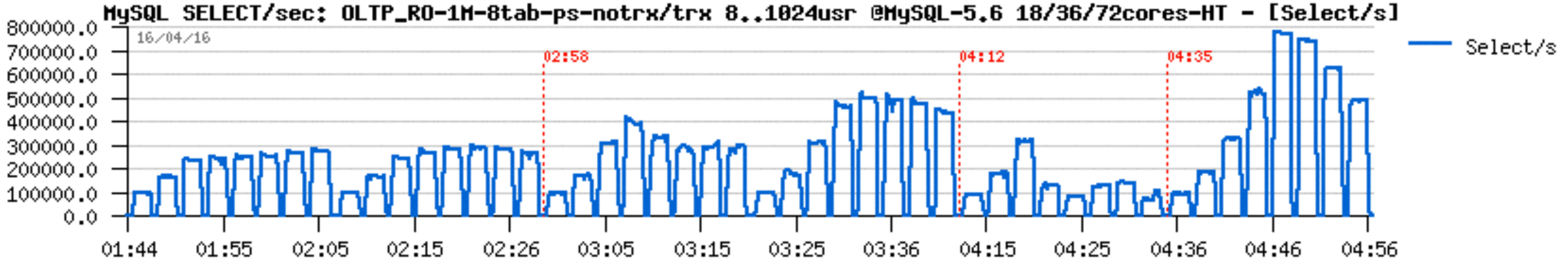
Behind the numbers...

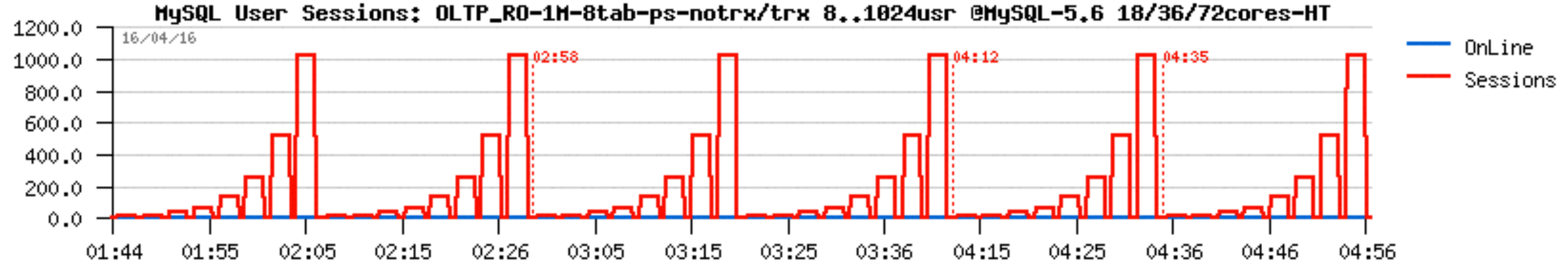
- MySQL 5.6 : Read-Only Transactions "workaround" :



Behind the numbers...

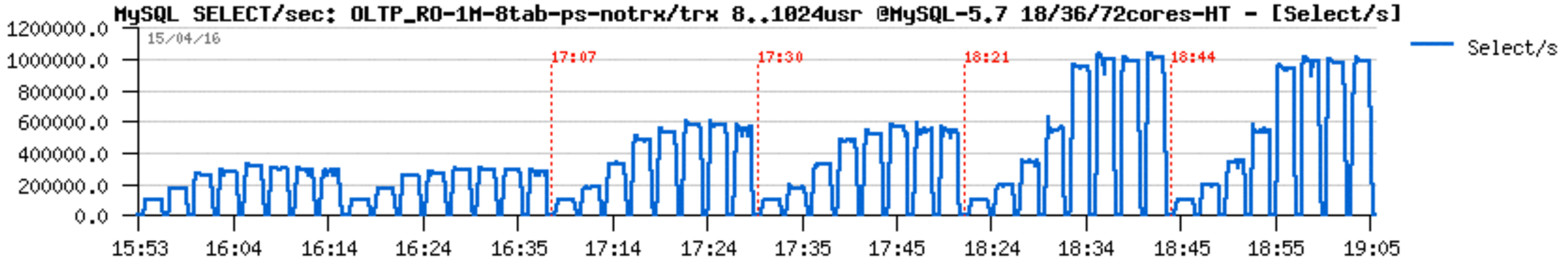
- MySQL 5.6, OLTP_RO-1Mx8-tables, 72cores-HT
  - OLTP_RO : [x14 SELECT Queries]
  - without / with transaction enclosure, 18/36/72cores-HT

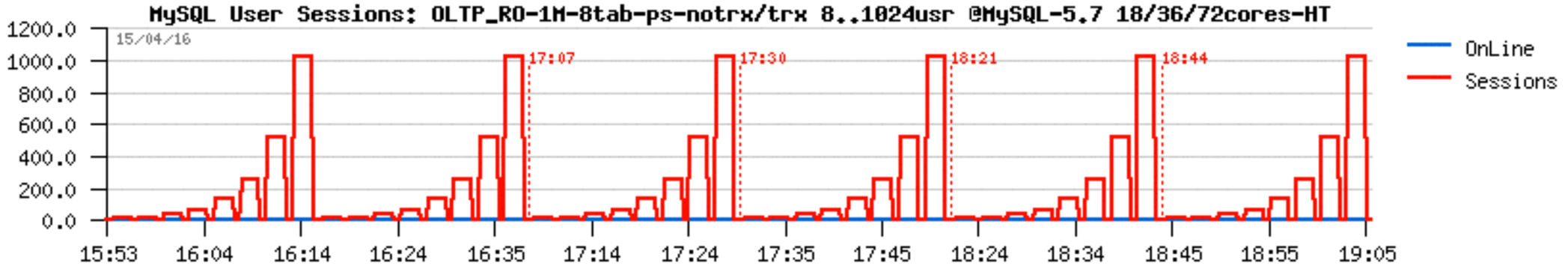




Behind the numbers...

- MySQL 5.7, OLTP_RO-1Mx8-tables, 72cores-HT
  - OLTP_RO : [x14 SELECT Queries]
  - without / with transaction enclosure, 18/36/72cores-HT



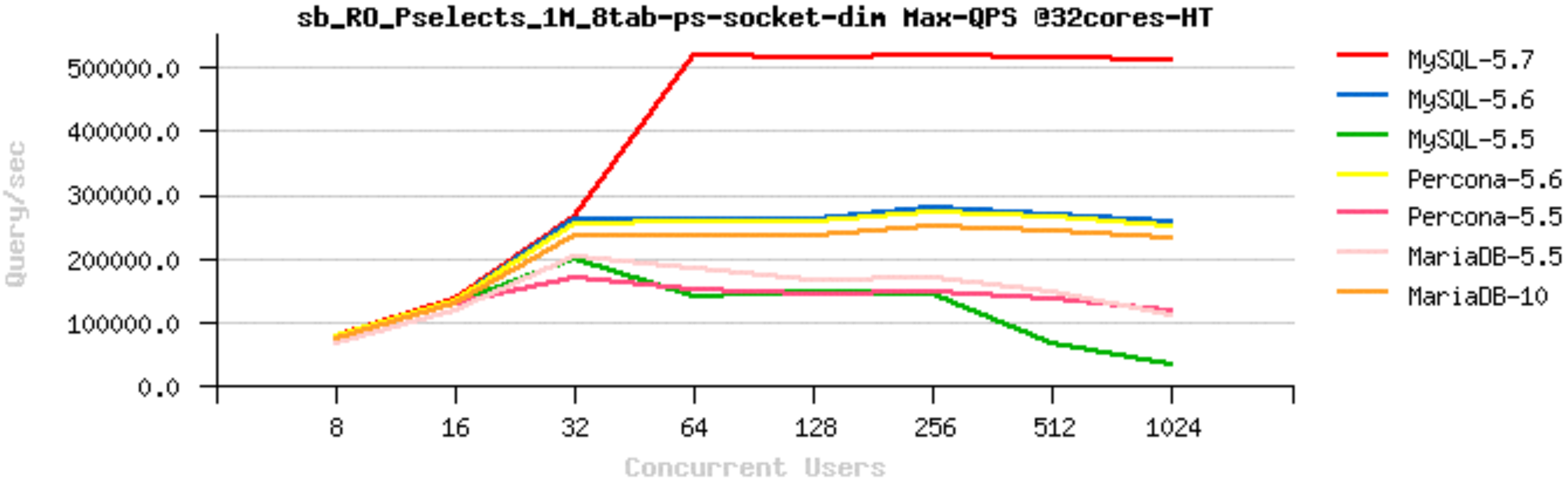


Behind the numbers...

- Up to you to decide what is less or more significant for you..
- If for ex. [x1000(!) Point-Select Queries] in transaction is OK
  - as was done by MariaDB to show their 1M QPS result..
  - hm.. and nobody called this Marketing ?..

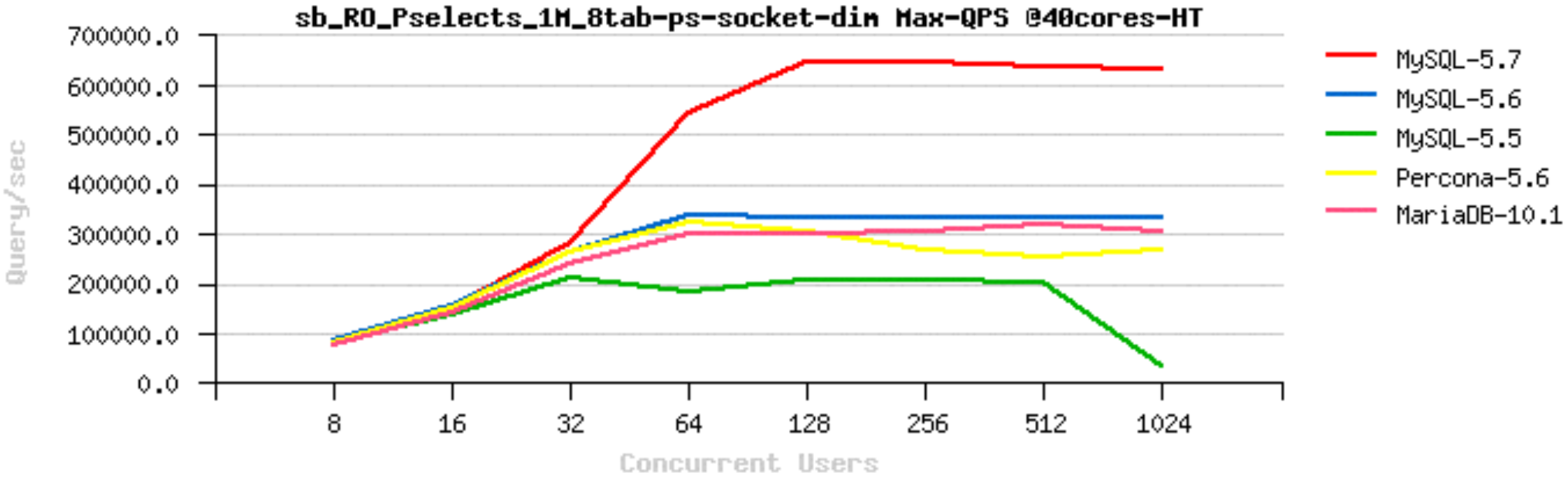
RO In-Memory @MySQL 5.7

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



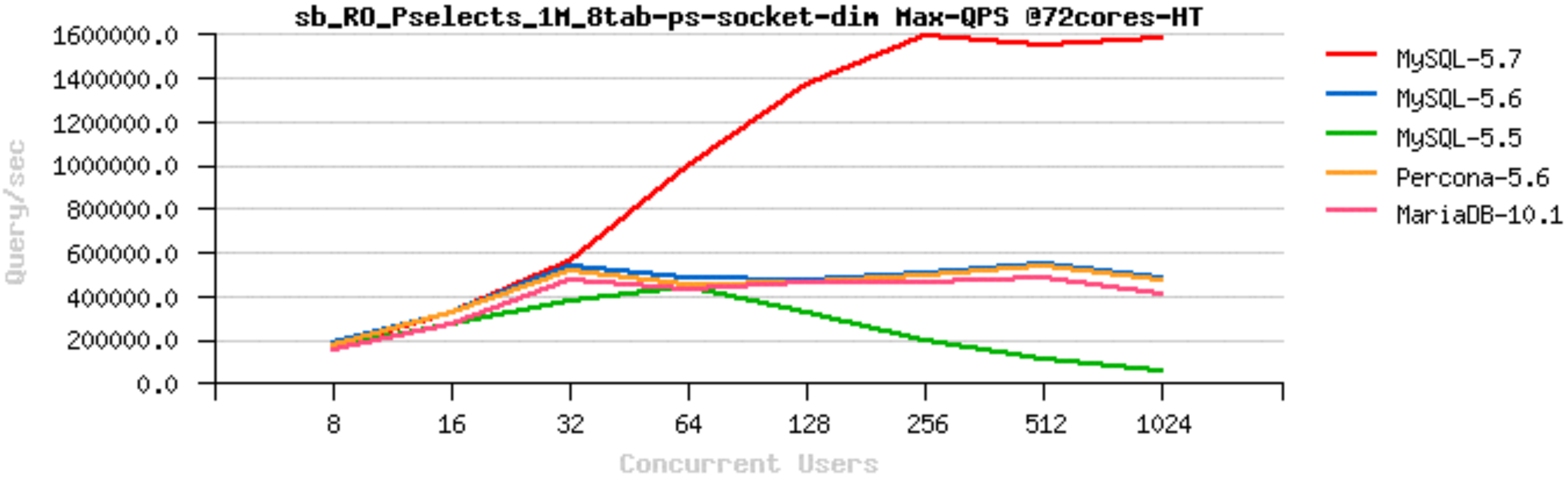
RO In-Memory @MySQL 5.7

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



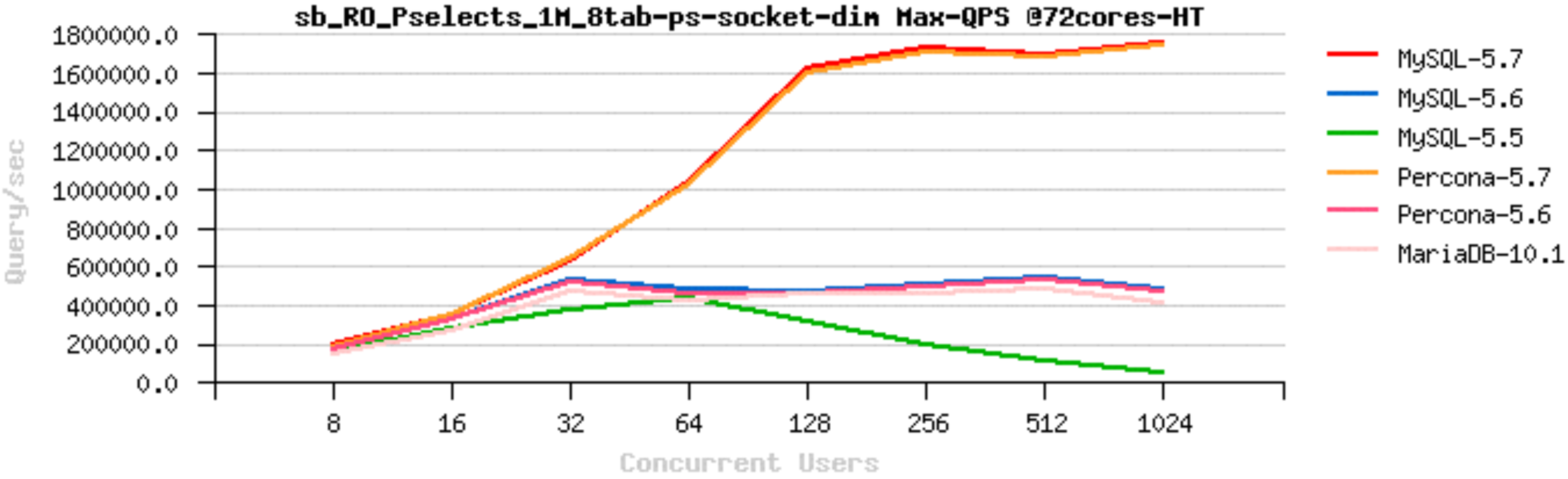
RO Point-Selects @MySQL 5.7 (Oct.2015)

- **1.6M (!!) QPS** Sysbench Point-Selects 8-tab :
  - 72cores-HT



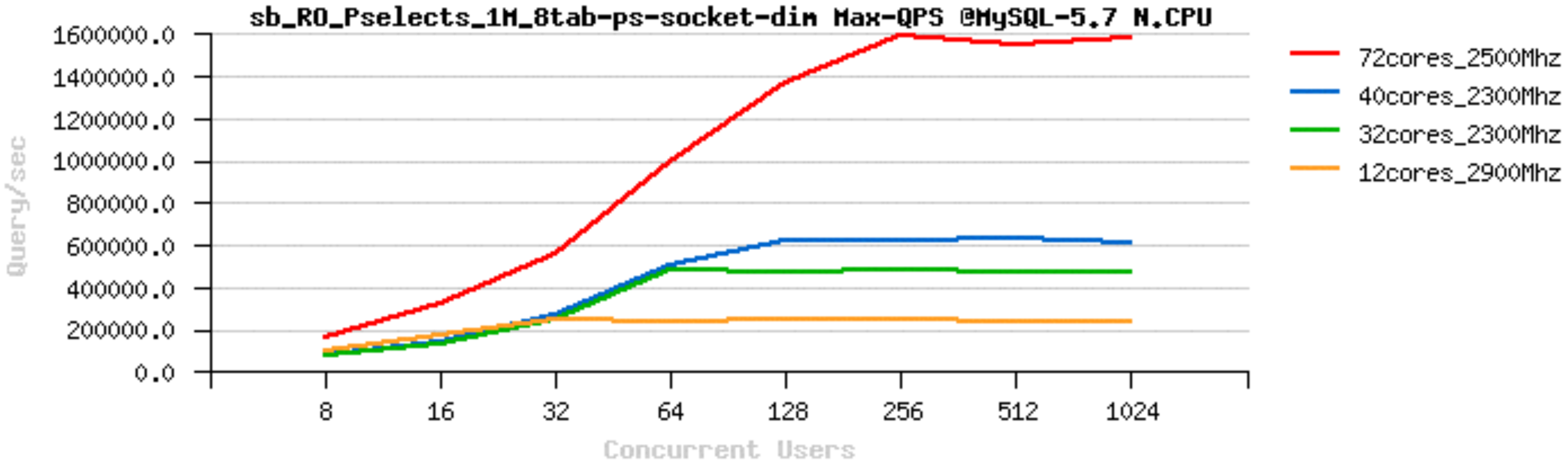
RO Point-Selects @MySQL 5.7 (Apr.2016)

- **1.8M QPS** Sysbench Point-Selects 8-tab, 72cores-HT :
  - or even more, if you really run after numbers.. ;-))



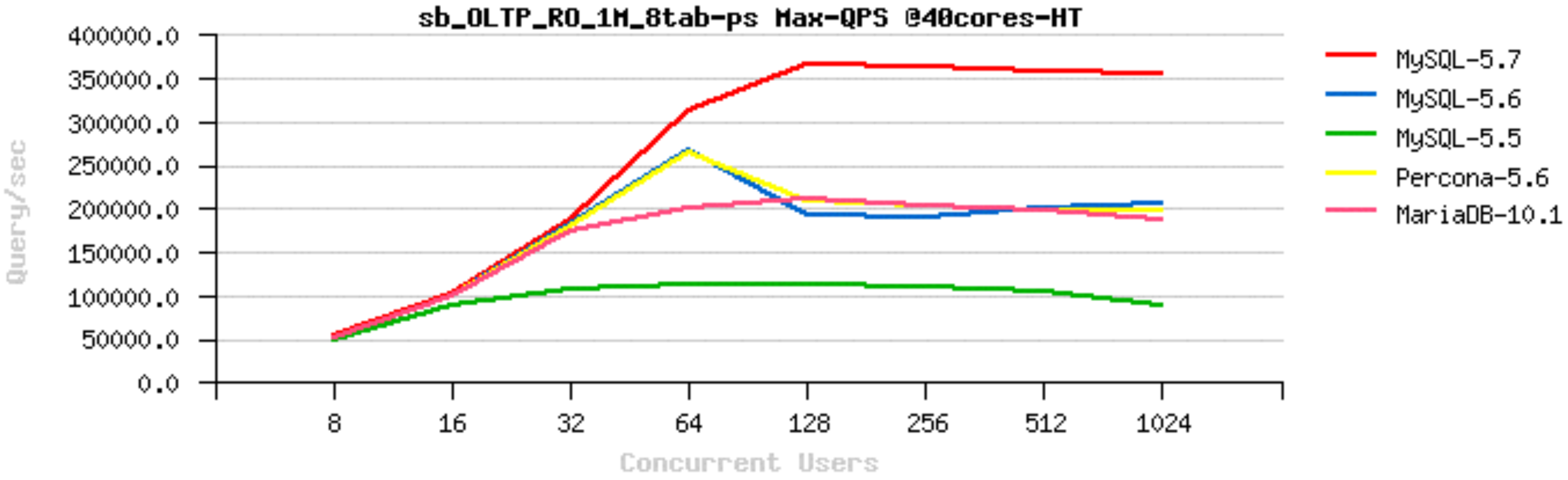
RO Point-Selects @MySQL 5.7

- Sysbench Point-Selects 8-tab => HW Progress :
  - new Intel CPU chips rock! (on 72cores-HT server here)



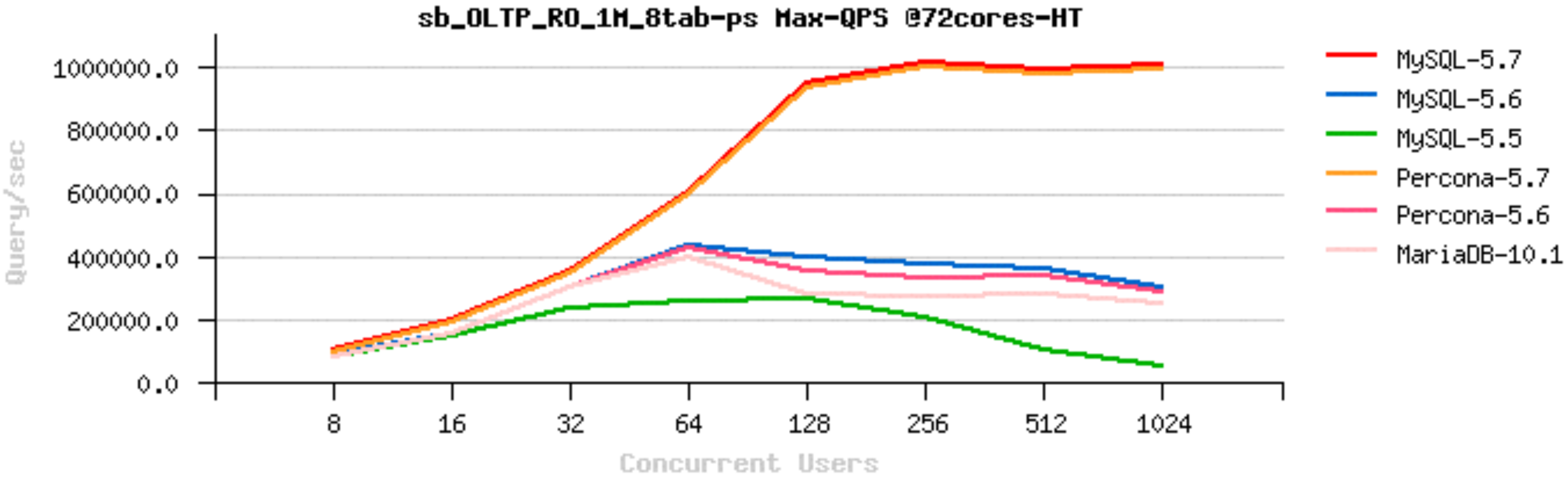
OLTP_RO : 8-tables

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



OLTP_RO : 8-tables

- Sysbench OLTP_RO 1Mx8-tables - **~1M (!!) QPS**
  - 72cores-HT



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 72 CPU cores-HT for sure, reported on more cores too..
  - Note : code path is growing with new features! (small HW may regress)
- IO-bound :
  - could be limited by storage (if you're not using a fast flash)
  - or by internal contentions (InnoDB file_sys mutex)
- Limitations
  - **there are still some limitations "by design"** (AHI, block lock, file_sys, etc..)
  - all in TODO to be fixed, but some are really needing a deep redesign..

Read+Write (RW) Workloads Scalability @MySQL 5.7

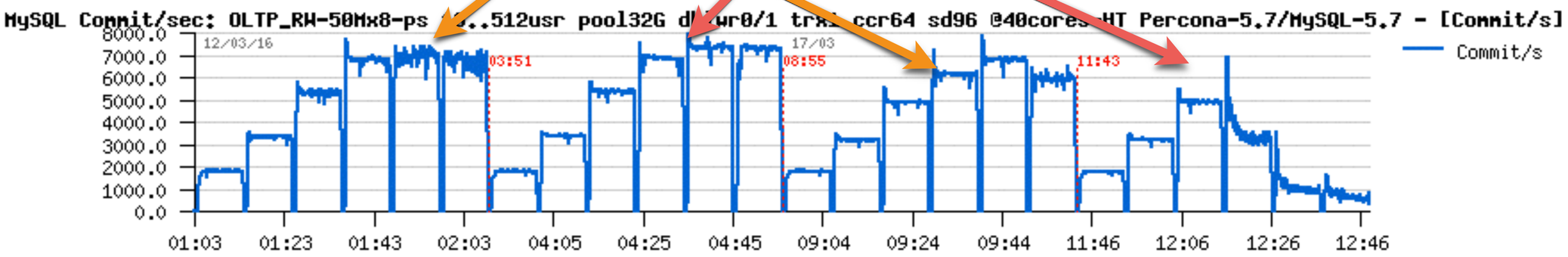
- Huge progress is already here too!
  - improved index locking
  - reduced lock_sys mutex contention
  - parallel flushing + improved flushing design
  - much better observability of internals
  - etc..
- However, not yet as good as Read-Only..
  - Performance continues to increase with more CPU cores
  - But on move from 16 to 32cores-HT you may gain only 50% better
  - Better performance on a faster storage as well
  - **On OLTP_RW can use a full power of fast flash for today!**
  - **More work in progress ;-)**
  - Internal contentions & Design limitations are the main issues here..
  - still many things are in pipe & prototype..

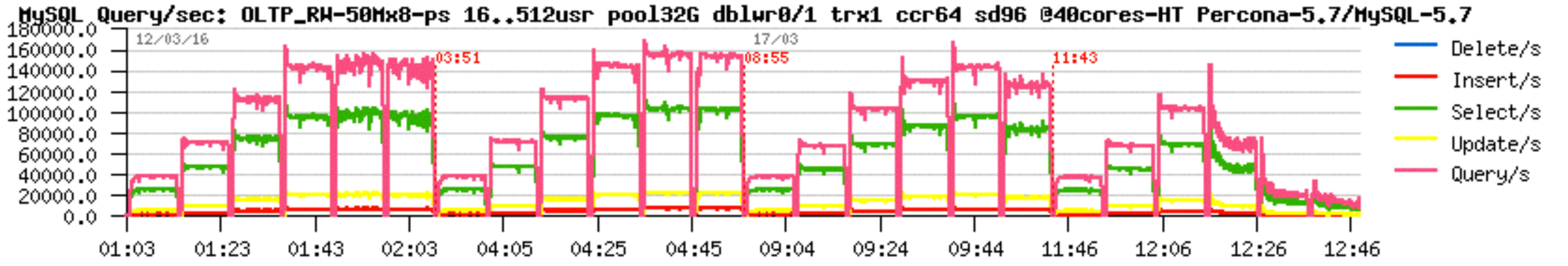
Read+Write Performance @MySQL / InnoDB

- Transactional processing
  - your CPU-bound transactional processing defines your Max possible TPS
  - with a bigger volume / more IO / etc. => Max TPS will not increase ;-)
- Pending issues :
  - same as RO + REDO (log_sys), locks (lock_sys), TRX (trx_sys), AHI=off, etc..
  - Purge lagging, more improved Adaptive Flushing
- 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 since a so long time.. => / re-design / etc. in urgency ;-)
    - Fusion-io atomic writes is one of (**true** support in MySQL 5.7)
    - a true re-design is still preferable ;-)

InnoDB Double-Write (DBLWR)

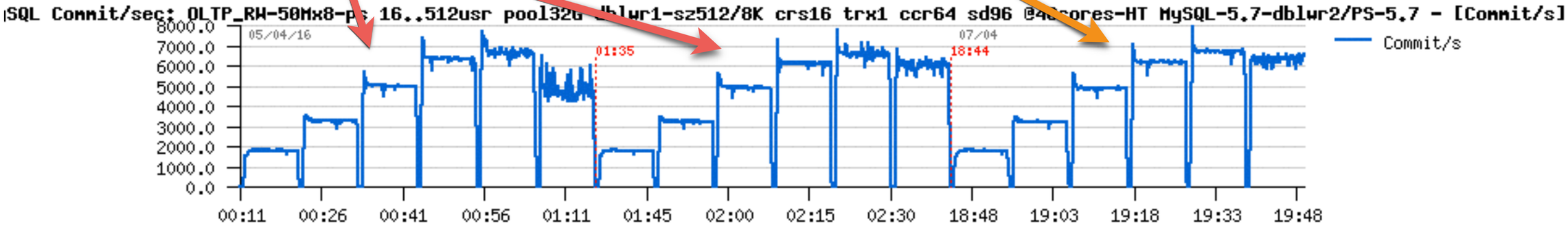
- OLTP_RW 50M x8-tables (120G dataset)
  - BP=32G, trx=1, dblwr=0/1, checksum=crc32, Flash “Nytro” Seagate-XP6500
  - Percona-5.7 / MySQL-5.7 (Jan.2016)

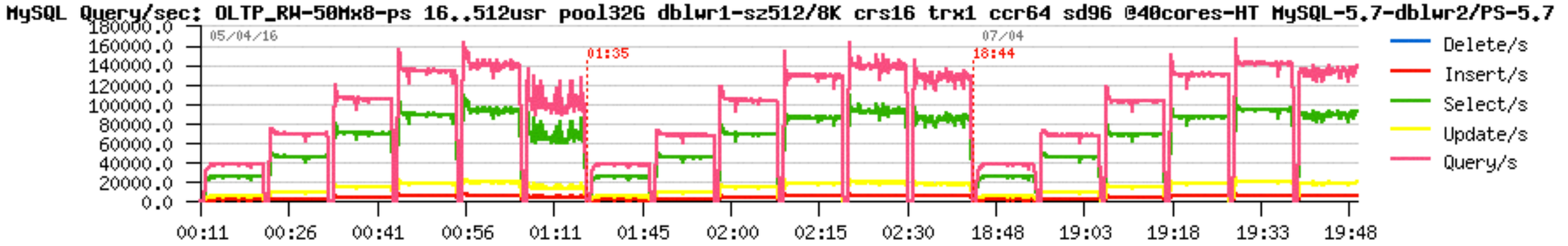




InnoDB Double-Write (DBLWR)

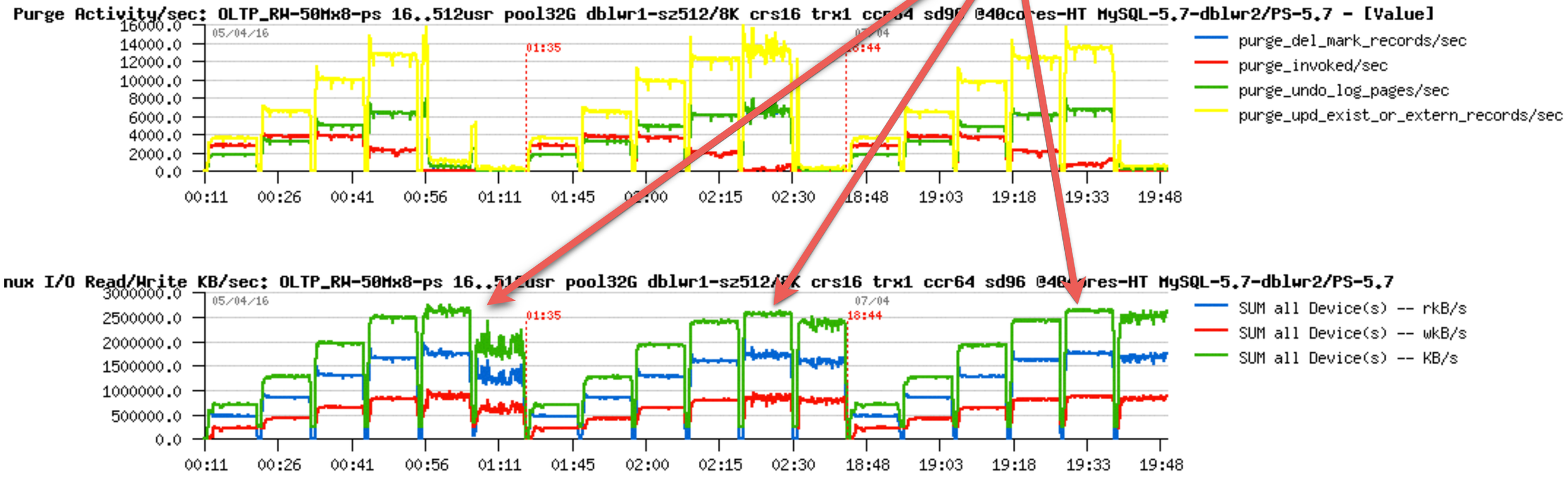
- OLTP_RW 50M x8-tables (120G dataset)
  - BP=32G, trx=1, dblwr=1, checksum=crc32, Flash “Nytro” Seagate-XP6500
  - MySQL-5.7-dblwr (work-in-progress) / Percona-5.7





InnoDB Double-Write (DBLWR) - Side Note..

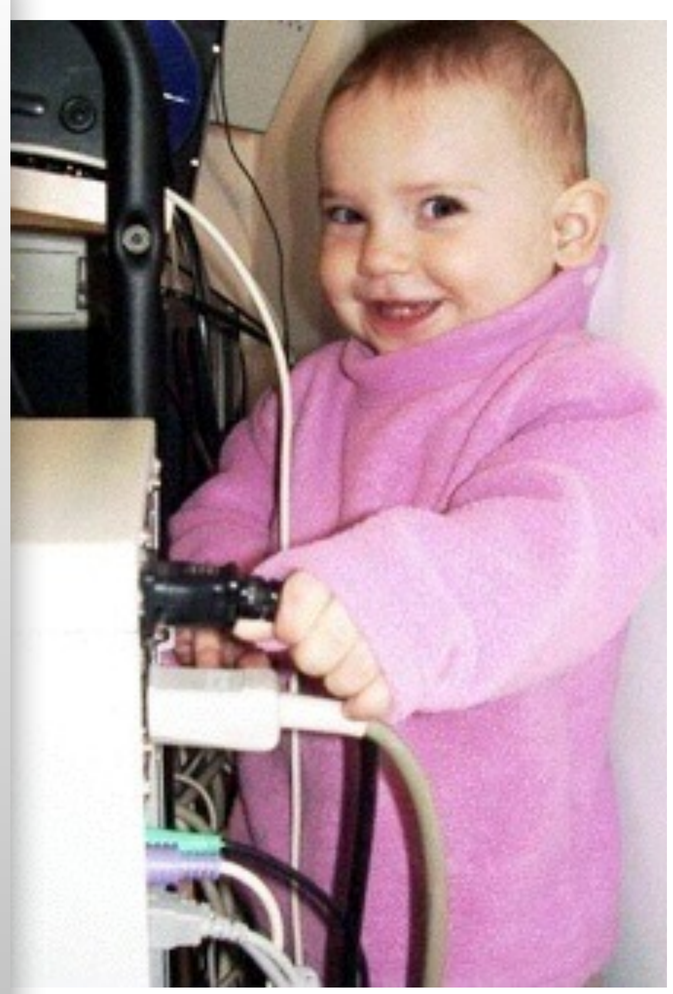
- OLTP_RW 50M x8-tables (120G dataset)
  - Purge lagging can be a very serious issue..
  - 5.7 with Flash “Nytro” Seagate-XP6500 => **over 2500 MB/sec** (16K InnoDB pages)



RW related starter configuration settings

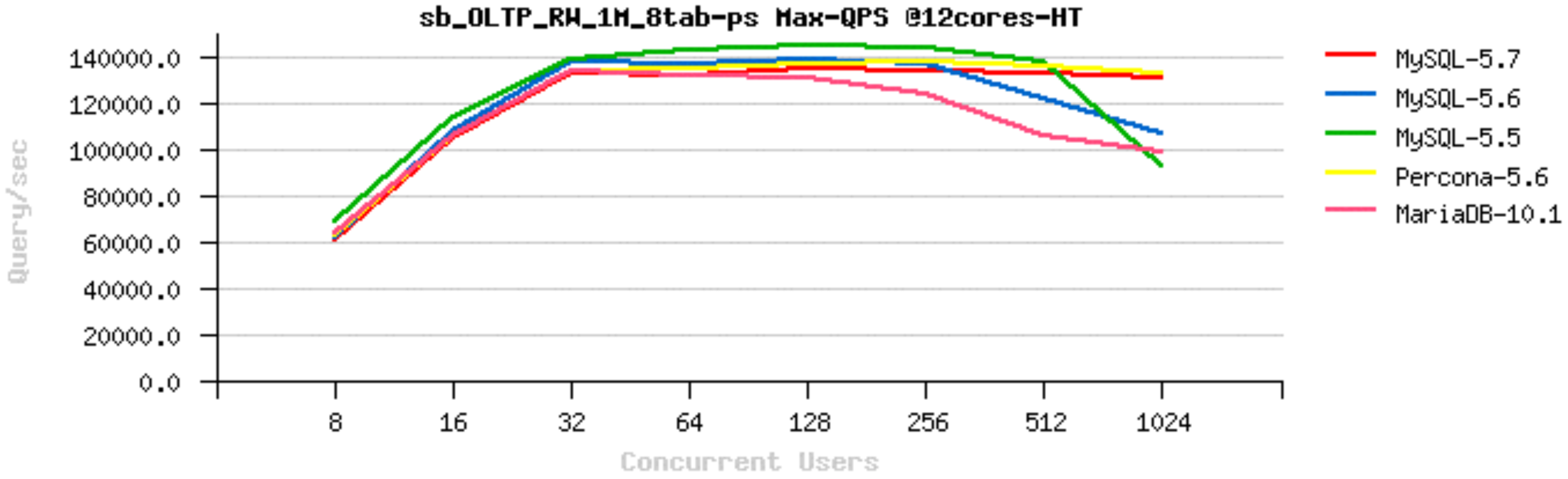
- my.conf :

```
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_NO_FSYNC
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 ??
```



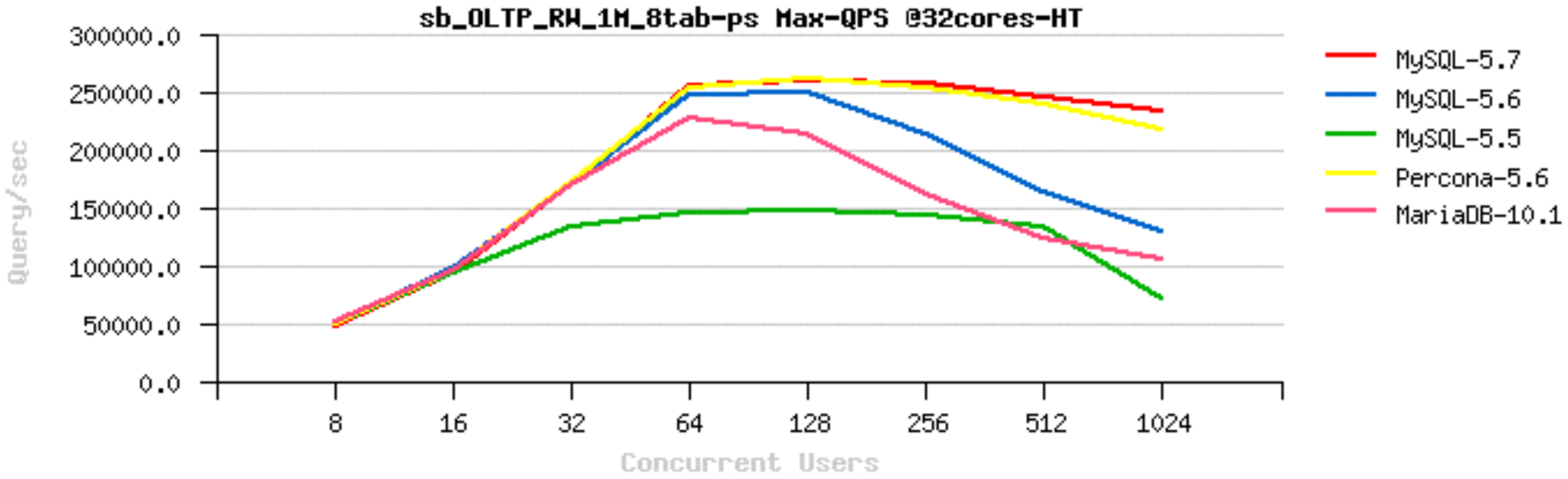
OLTP_RW : 8-tables

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



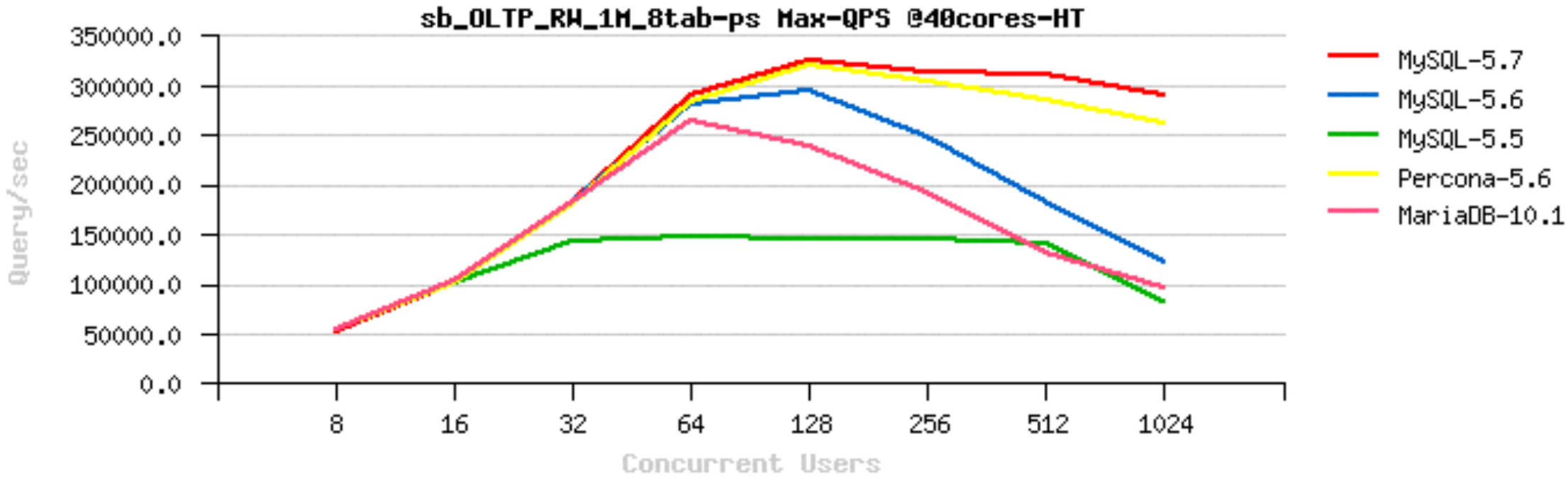
OLTP_RW : 8-tables

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



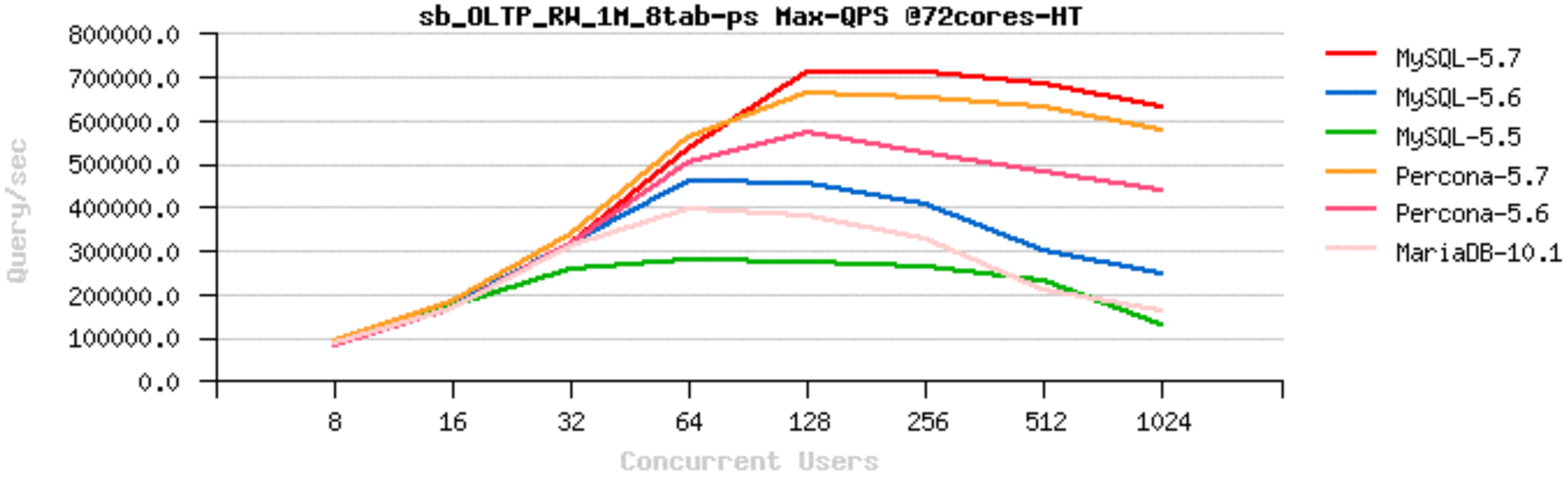
OLTP_RW : 8-tables

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



OLTP_RW : 8-tables

- Sysbench OLTP_RW 1Mx8-tables
  - 72cores-HT
  - and the winner is: MySQL 5.7 !! (or Percona-5.7 + patch ;-))

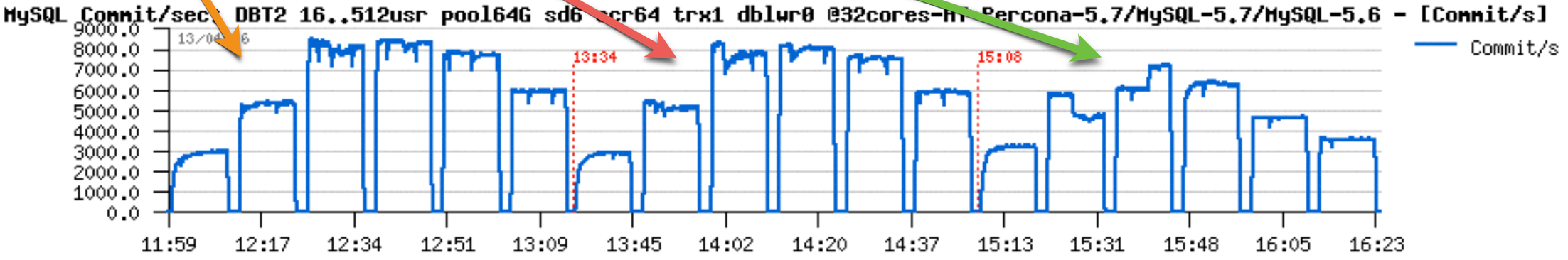


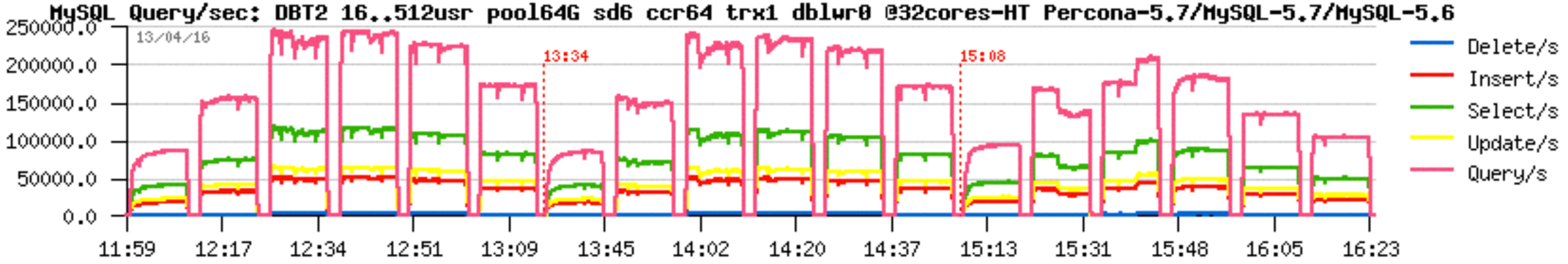
DBT2 (TPC-C) Benchmark

- Tested workload: DBT2 W500 or W512
  - (single database 8 databases of W64 size each)
  - **concurrent users: 16, 32, .. 512**
- Engines: Percona-5.7, MySQL-5.7-dblwr, MySQL-5.6
- Configurations:
  - 64G BP size (REDO-driven flushing), and 16G (LRU-bound flushing)
- Safe options:
  - trx_commit = 1 dblwr=0
  - trx_commit = 1 dblwr=1
- Tuning:
  - spin_delay = 6 / 96
  - innodb thread concurrency = 64
  - redo = 32G

DBT2 W500, 32cores-HT

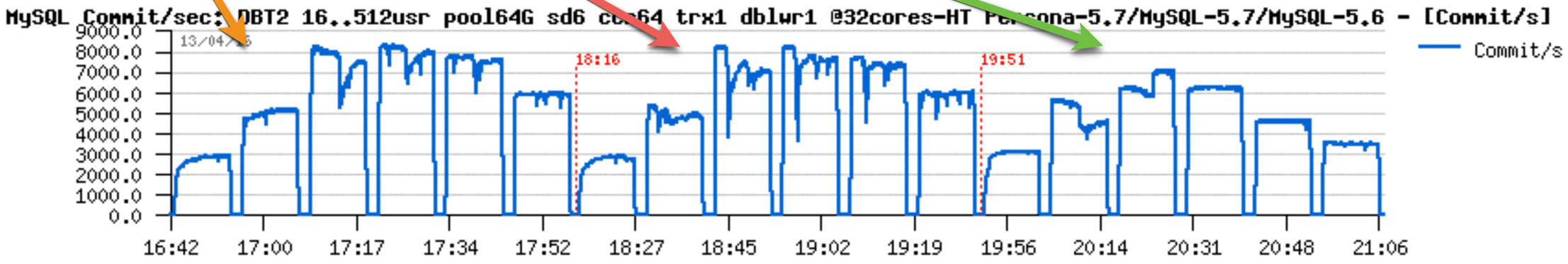
- 64G pool, trx=1, dblwr=0
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

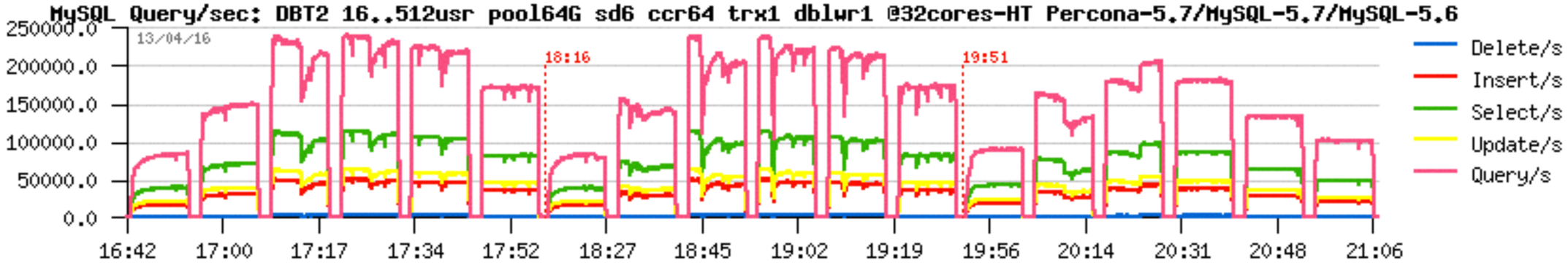




DBT2 W500, 32cores-HT

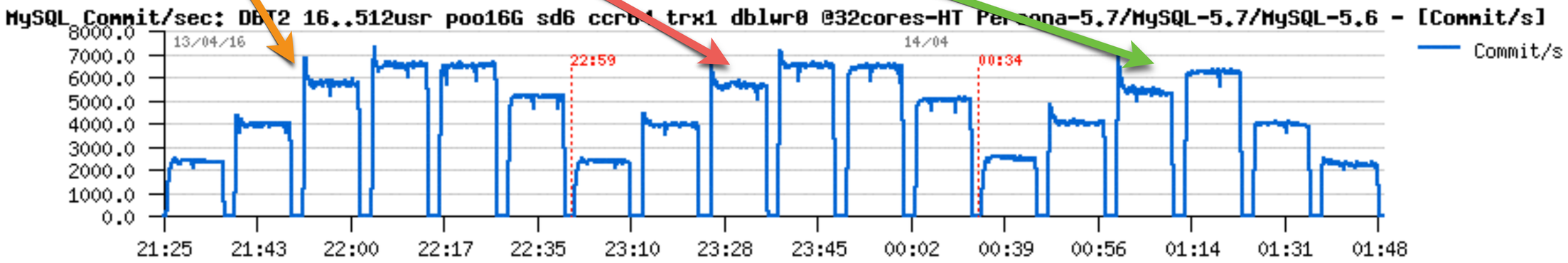
- 64G pool, trx=1, dblwr=1
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

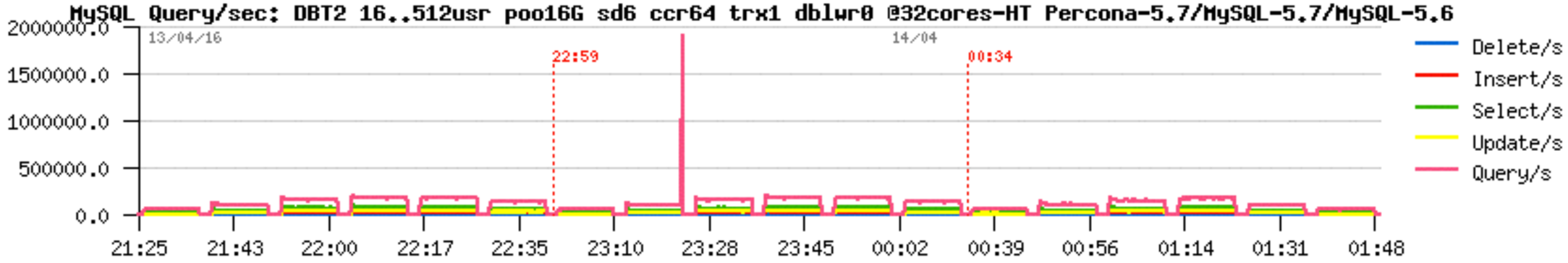




DBT2 W500, 32cores-HT

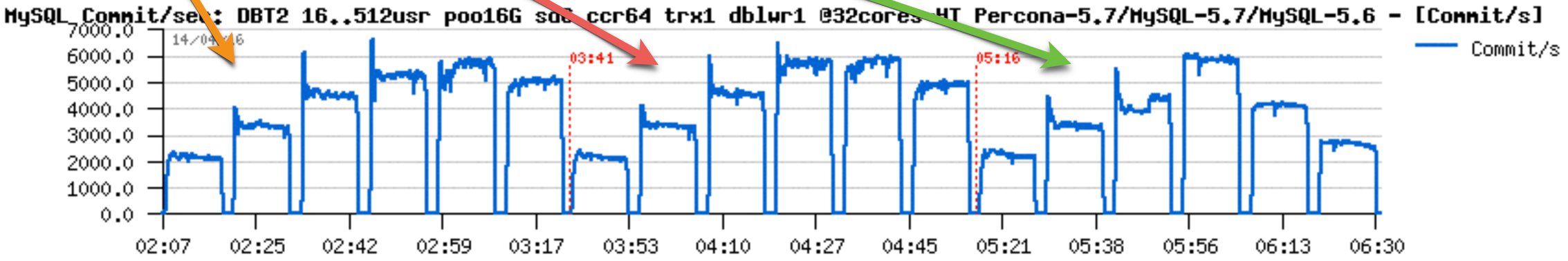
- 16G pool, trx=1, dblwr=0
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

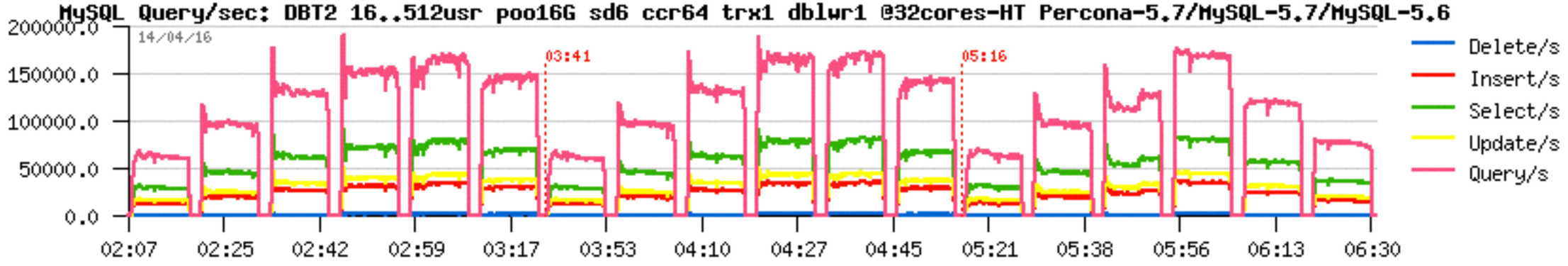




DBT2 W500, 32cores-HT

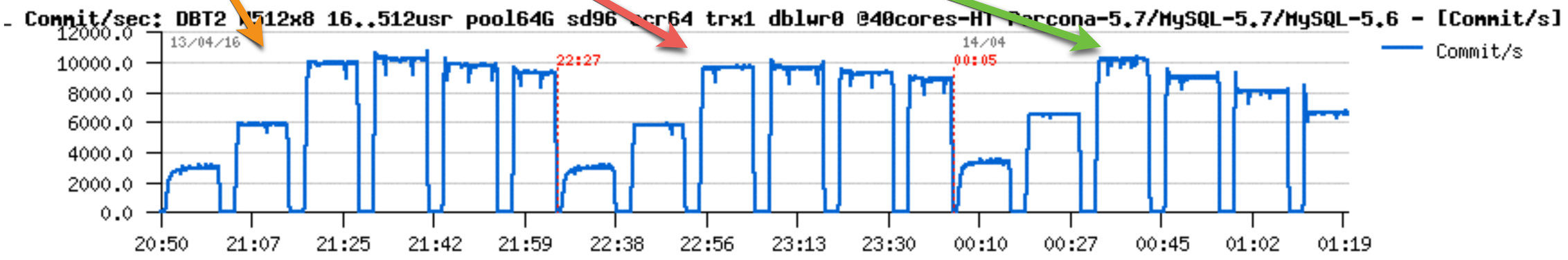
- 16G pool, trx=1, dblwr=1
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

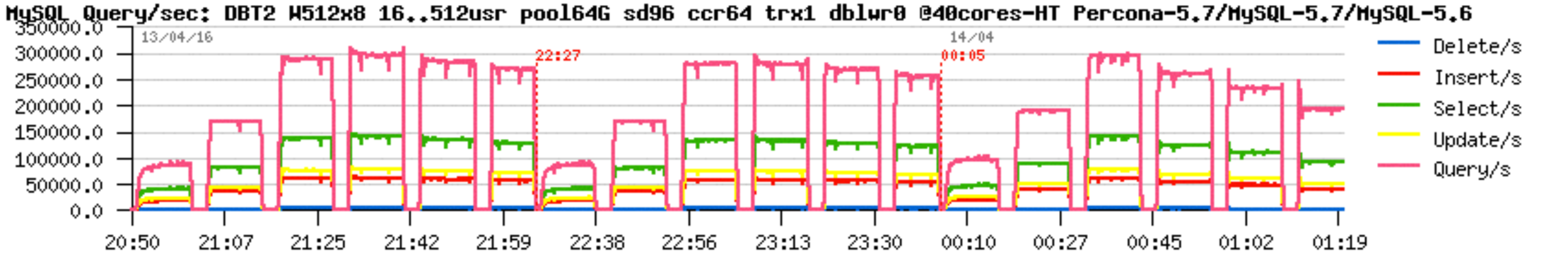




DBT2 W500, 40cores-HT

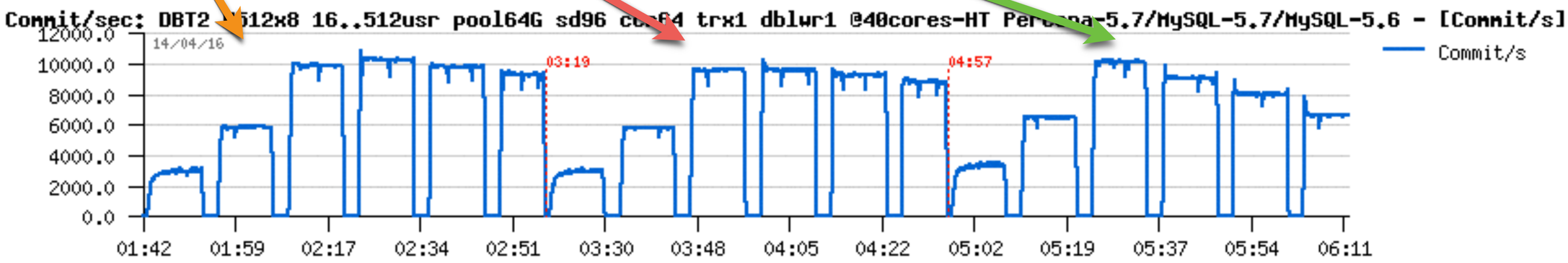
- 64G pool, trx=1, dblwr=0
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

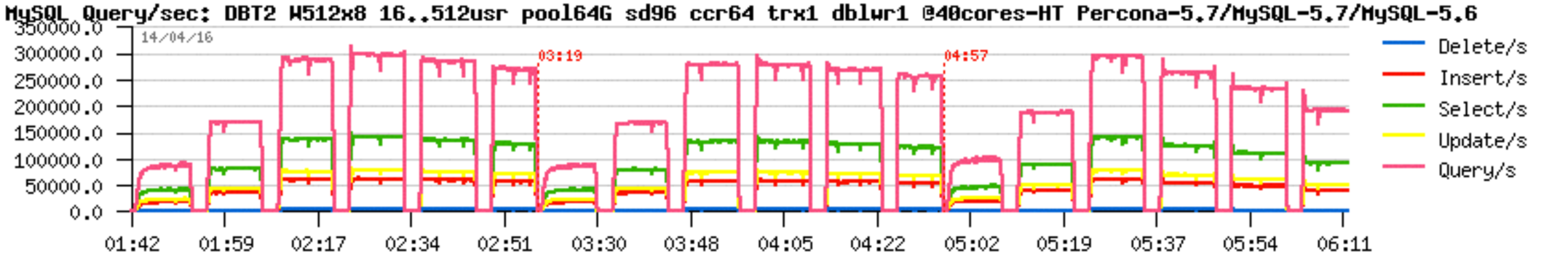




DBT2 W500, 40cores-HT

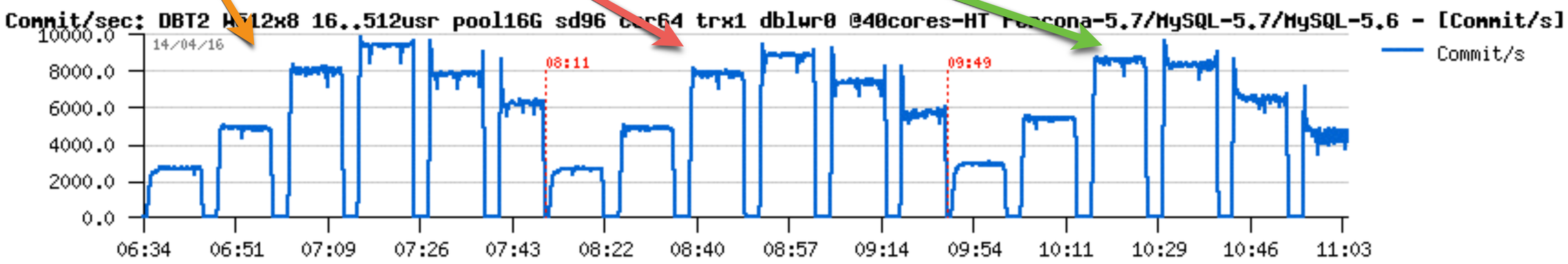
- 64G pool, trx=1, dblwr=1
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

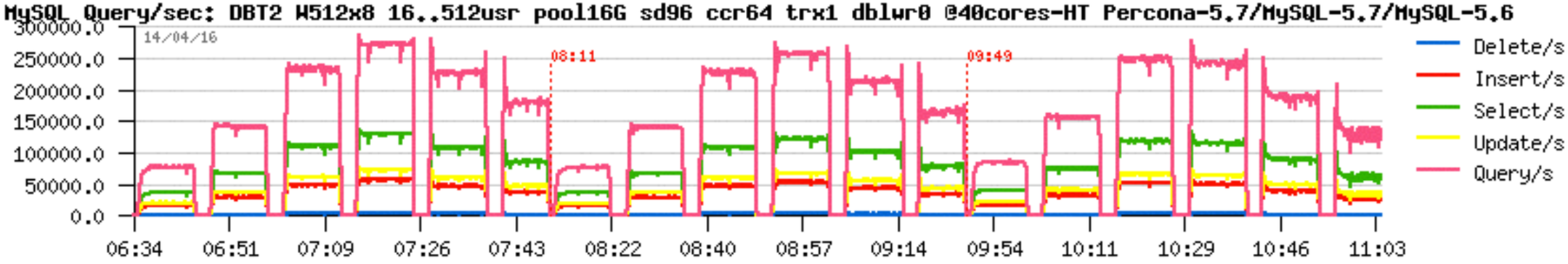




DBT2 W500, 40cores-HT

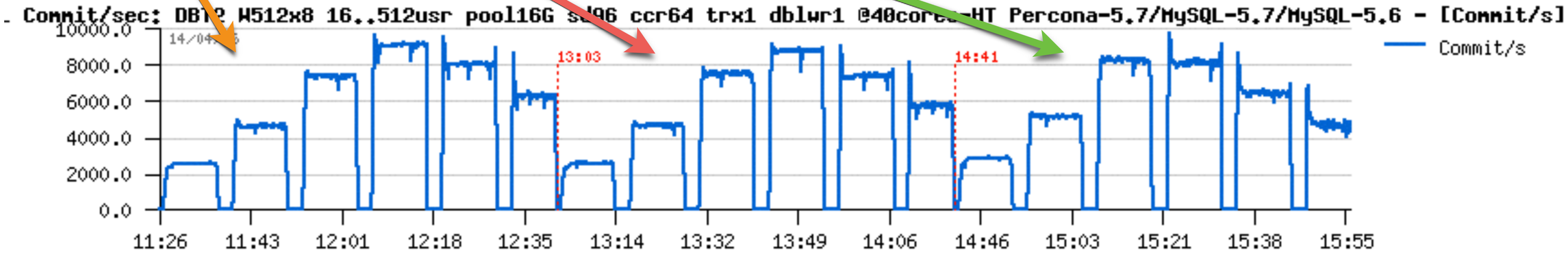
- 16G pool, trx=1, dblwr=0
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

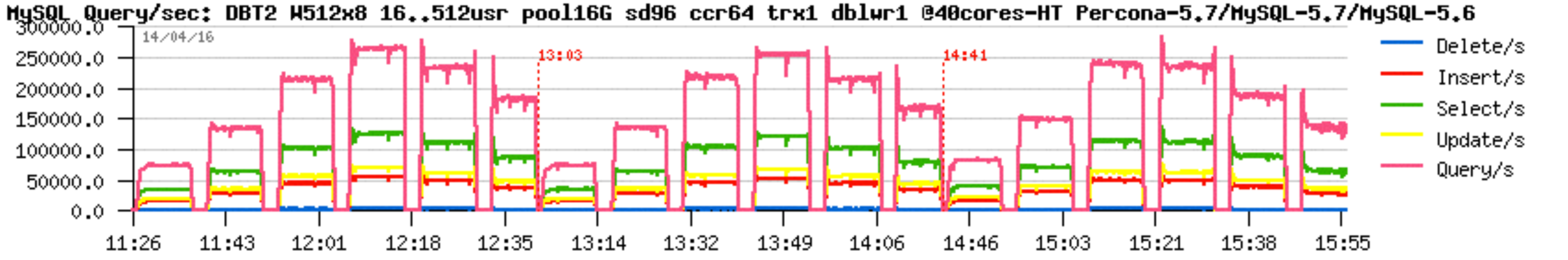




DBT2 W500, 40cores-HT

- 16G pool, trx=1, dblwr=1
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6



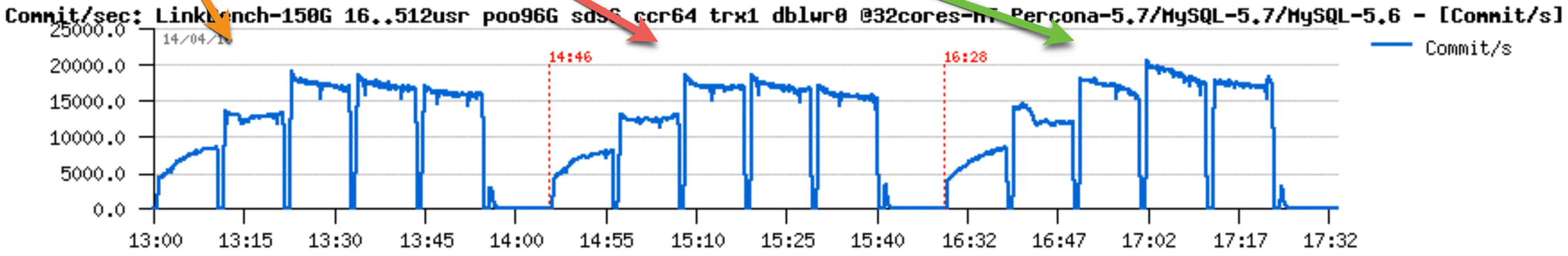


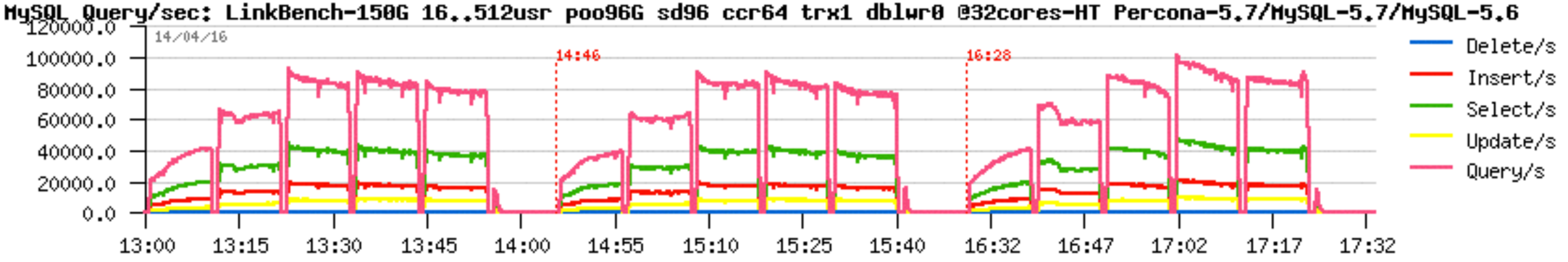
LinkBench Benchmark

- Tested workload:
  - 150M data (~200G data)
  - **concurrent users: 16, 32, .. 512**
- Engines: Percona-5.7, MySQL-5.7-dblwr, MySQL-5.6
- Configurations:
  - 96/128G BP size (REDO-driven flushing), and 16G (LRU-bound flushing)
- Safe options:
  - trx_commit = 1 dblwr=0
  - trx_commit = 1 dblwr=1
- Tuning:
  - spin_delay = 6 / 96
  - innodb thread concurrency = 64
  - redo = 32G

LinkBench-150M, 32cores-HT

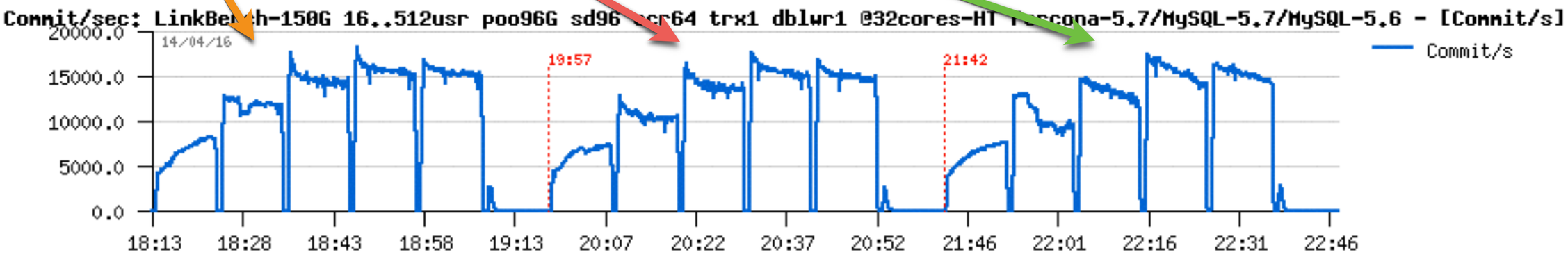
- 96G pool, trx=1, dblwr=0
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

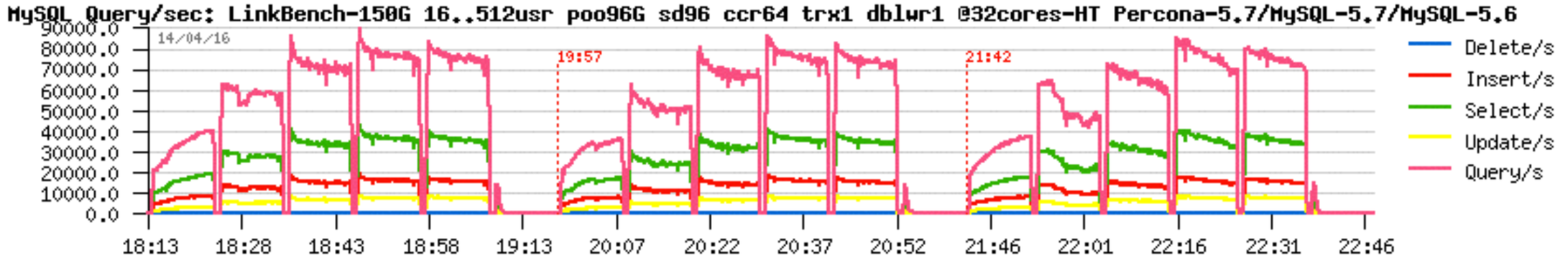




LinkBench-150M, 32cores-HT

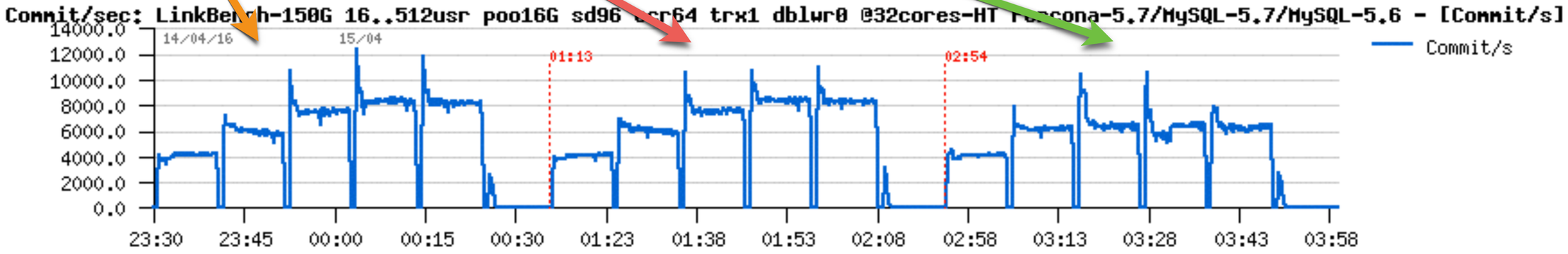
- 96G pool, trx=1, dblwr=1
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

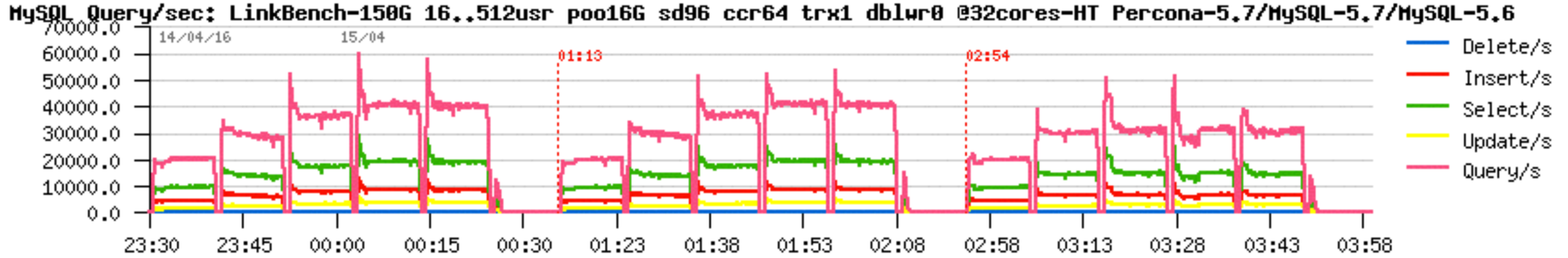




LinkBench-150M, 32cores-HT

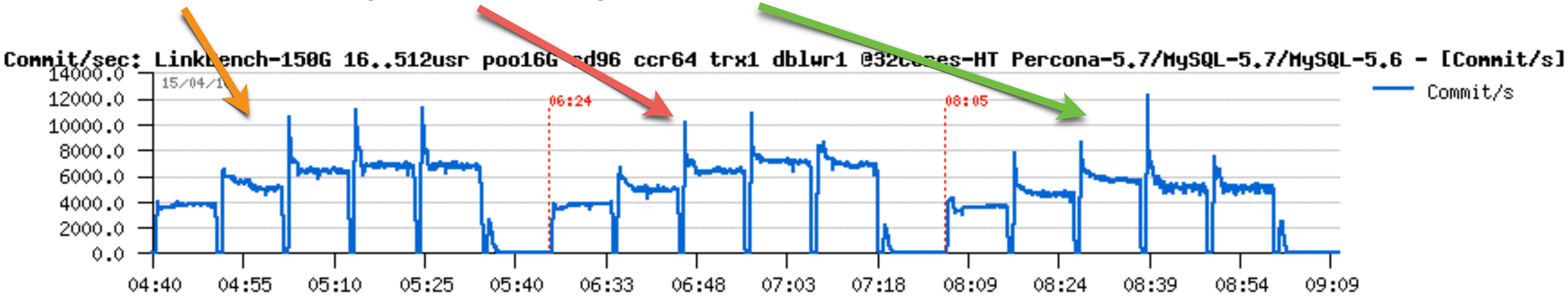
- 16G pool, trx=1, dblwr=0
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

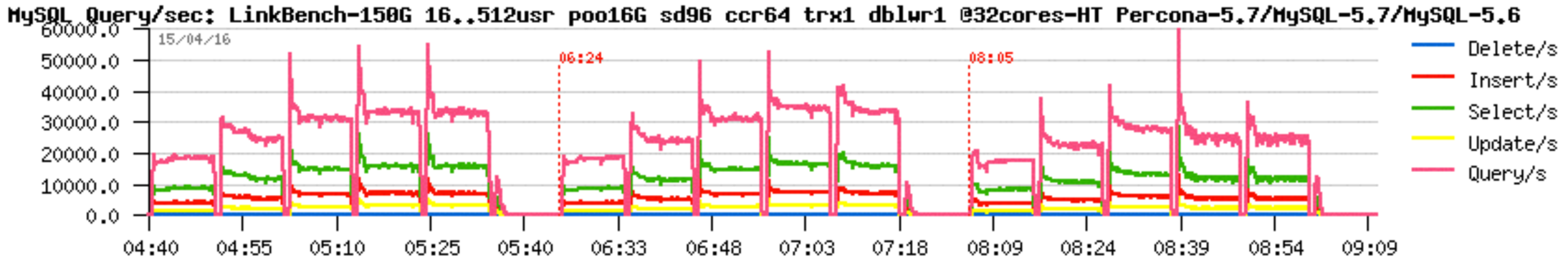




LinkBench-150M, 32cores-HT

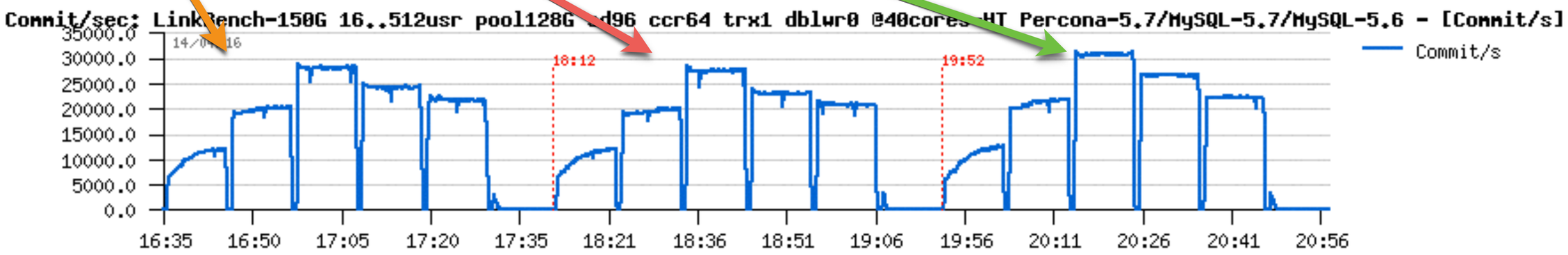
- 16G pool, trx=1, dblwr=1
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

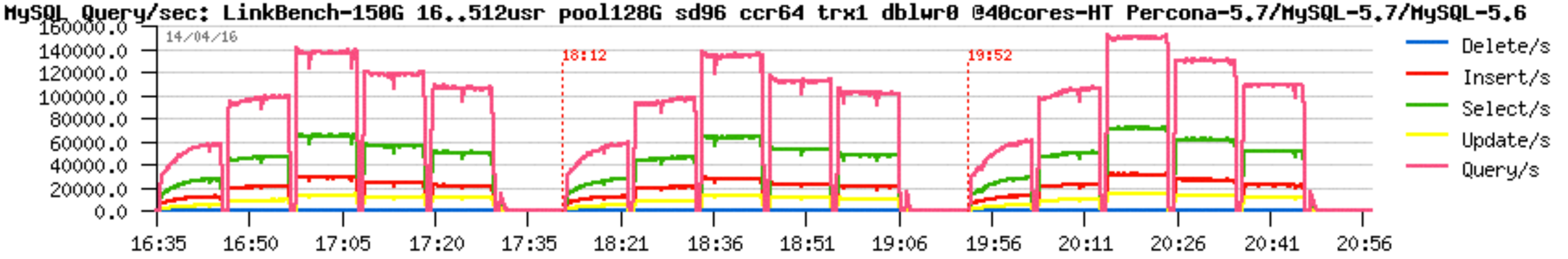




LinkBench-150M, 40cores-HT

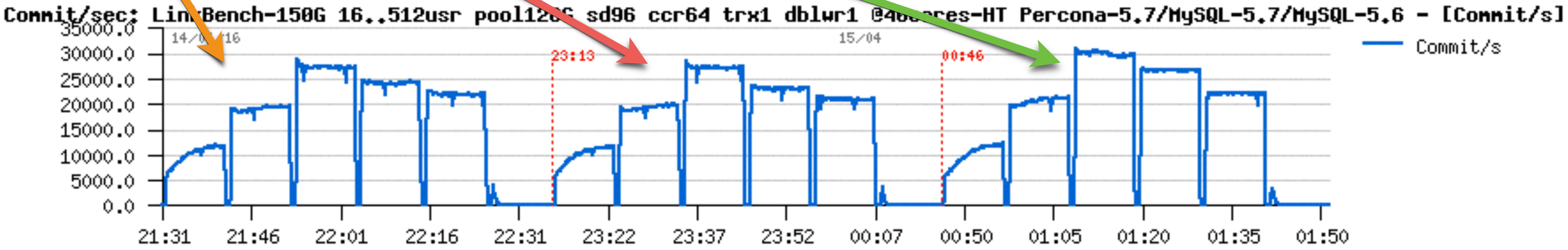
- 128G pool, trx=1, dblwr=0
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

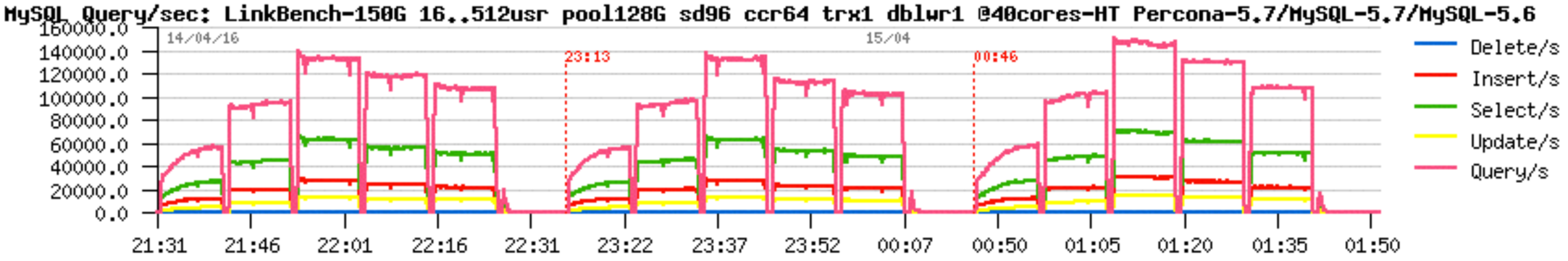




LinkBench-150M, 40cores-HT

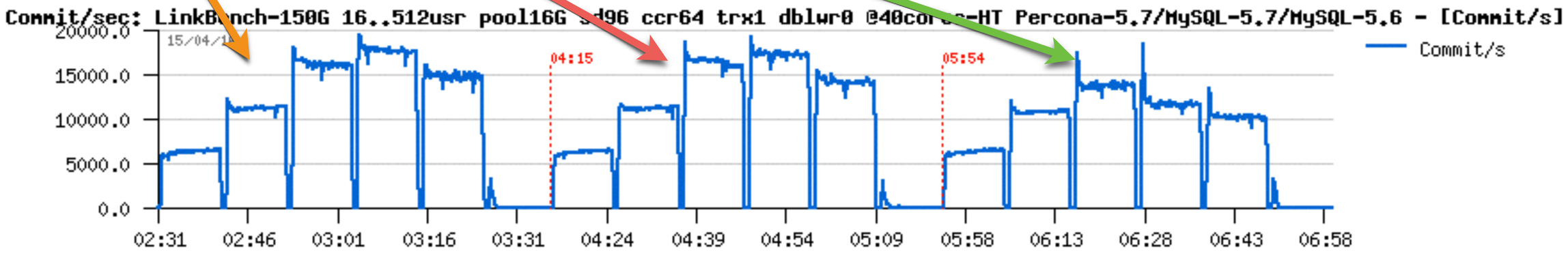
- 128G pool, trx=1, dblwr=1
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

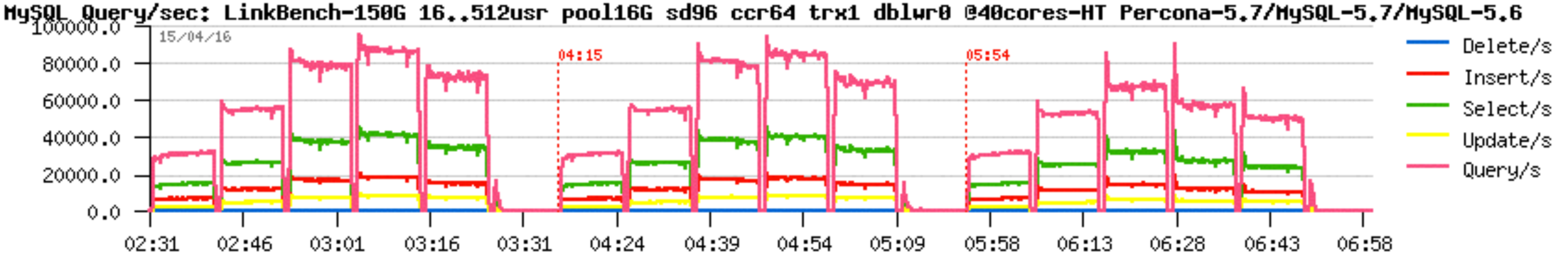




LinkBench-150M, 40cores-HT

- 16G pool, trx=1, dblwr=0
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

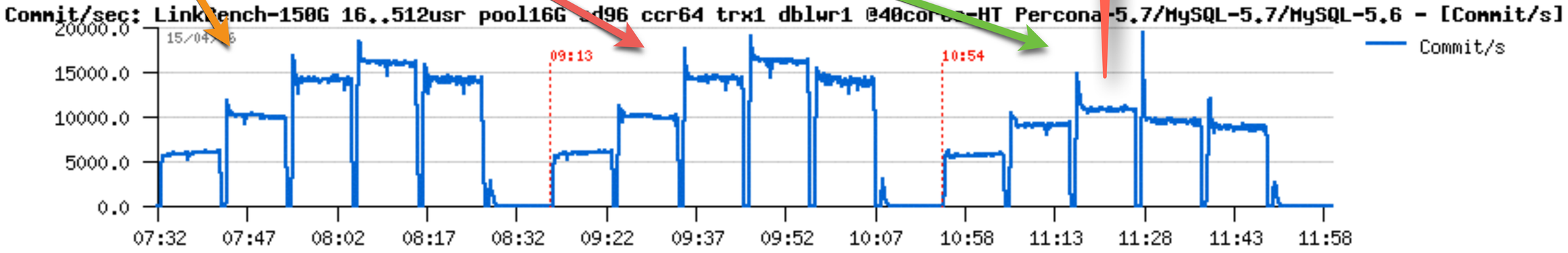


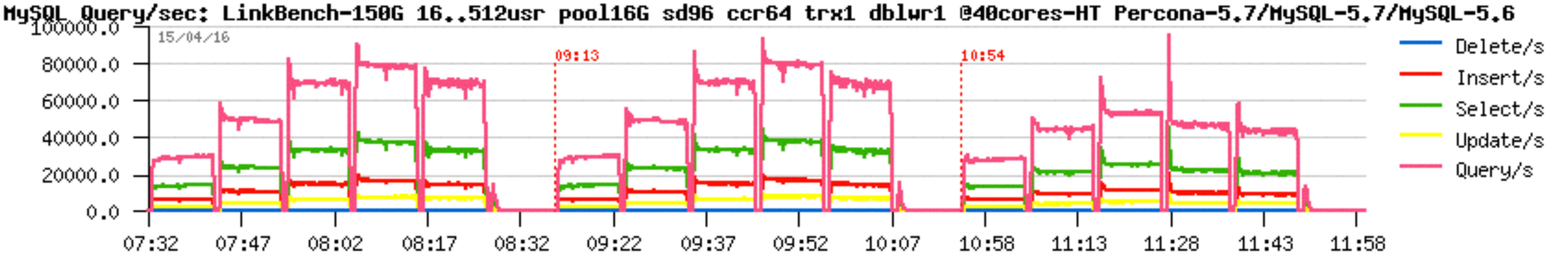


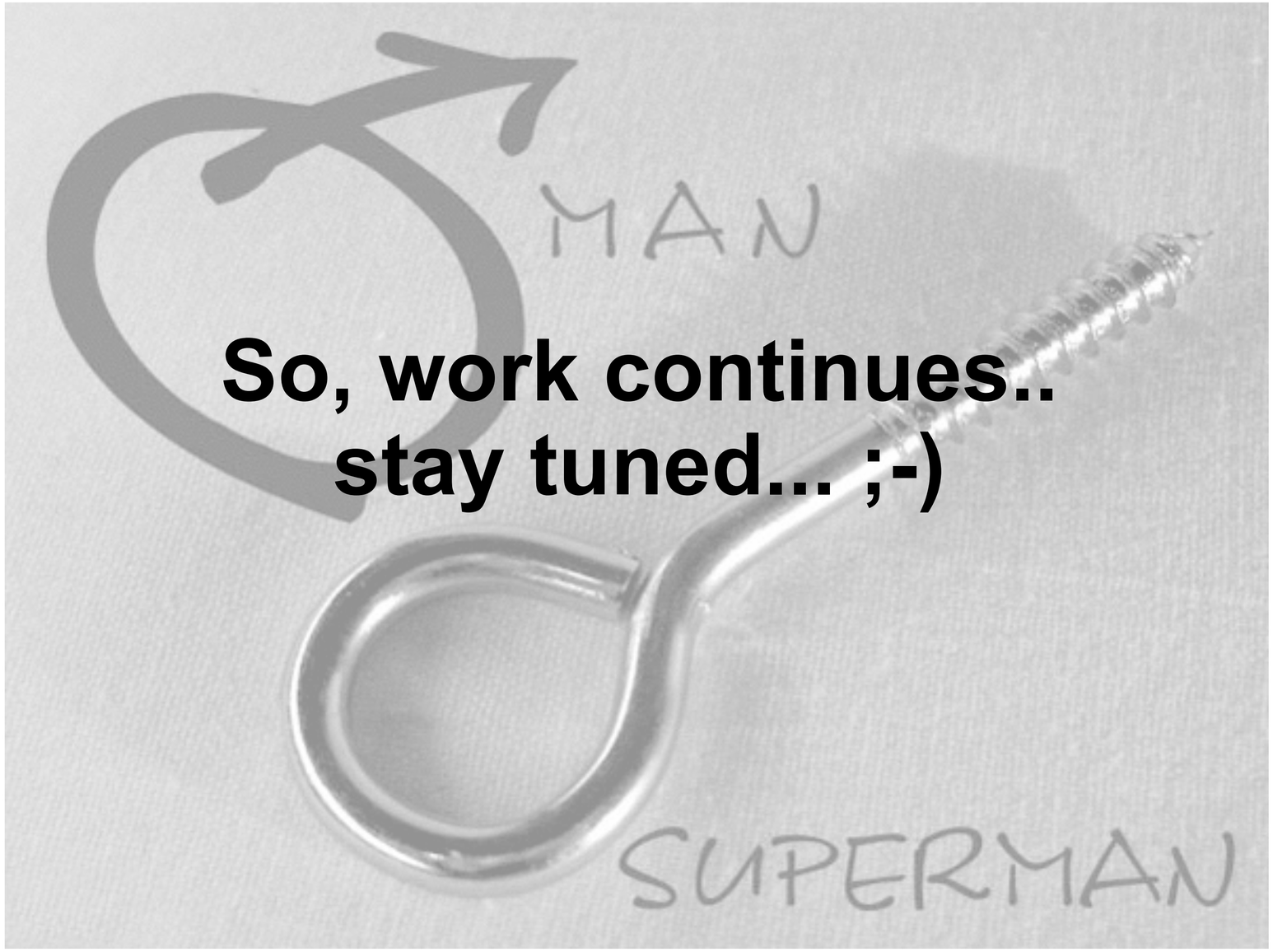
LinkBench-150M, 40cores-HT

- 16G pool, trx=1, dblwr=1
  - Percona-5.7 / MySQL-5.7 / MySQL-5.6

Please, upgrade me to 5.7 !!! ;-)






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

One more thing ;-)

- All graphs are built with dim_STAT (http://dimitrik.free.fr)
  - All System load stats (CPU, I/O, Network, RAM, Processes,...)
  - Manly for Linux, Solaris, OSX (and any other UNIX too :-)
  - Add-Ons for MySQL, Oracle RDBMS, 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! :-)
- Links
  - http://dimitrik.free.fr - dim_STAT, dbSTRESS, Benchmark Reports, etc.
  - http://dimitrik.free.fr/blog - Articles about MySQL Performance, etc.