MySQL

ORACLE

MySQL 8.0-dev 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..
- Progress in MySQL 8.0-dev & Benchmark results..
- Q & A

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
  - where we're in MySQL Dev as of today..

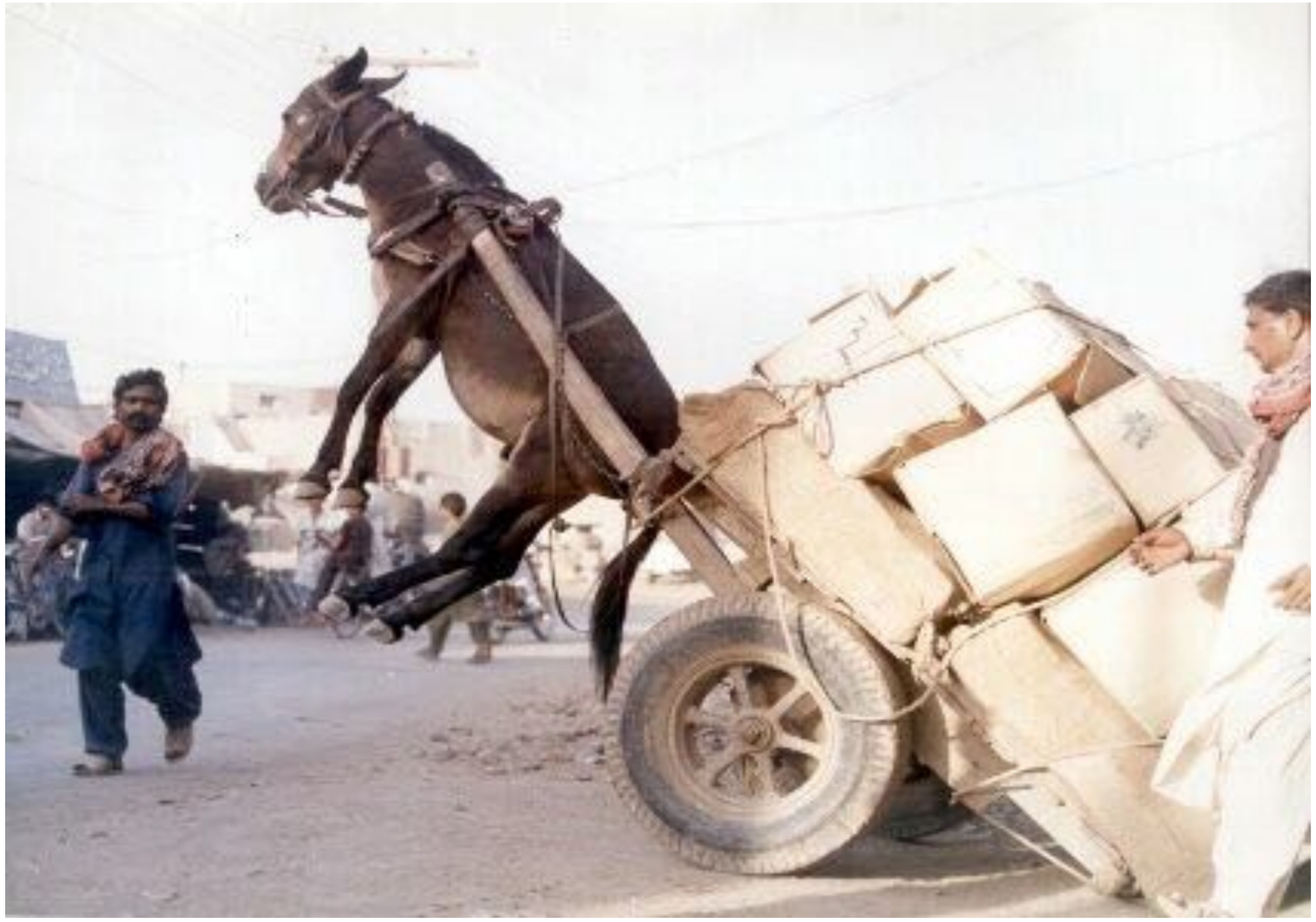
Why MySQL Performance ?...

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... ???..

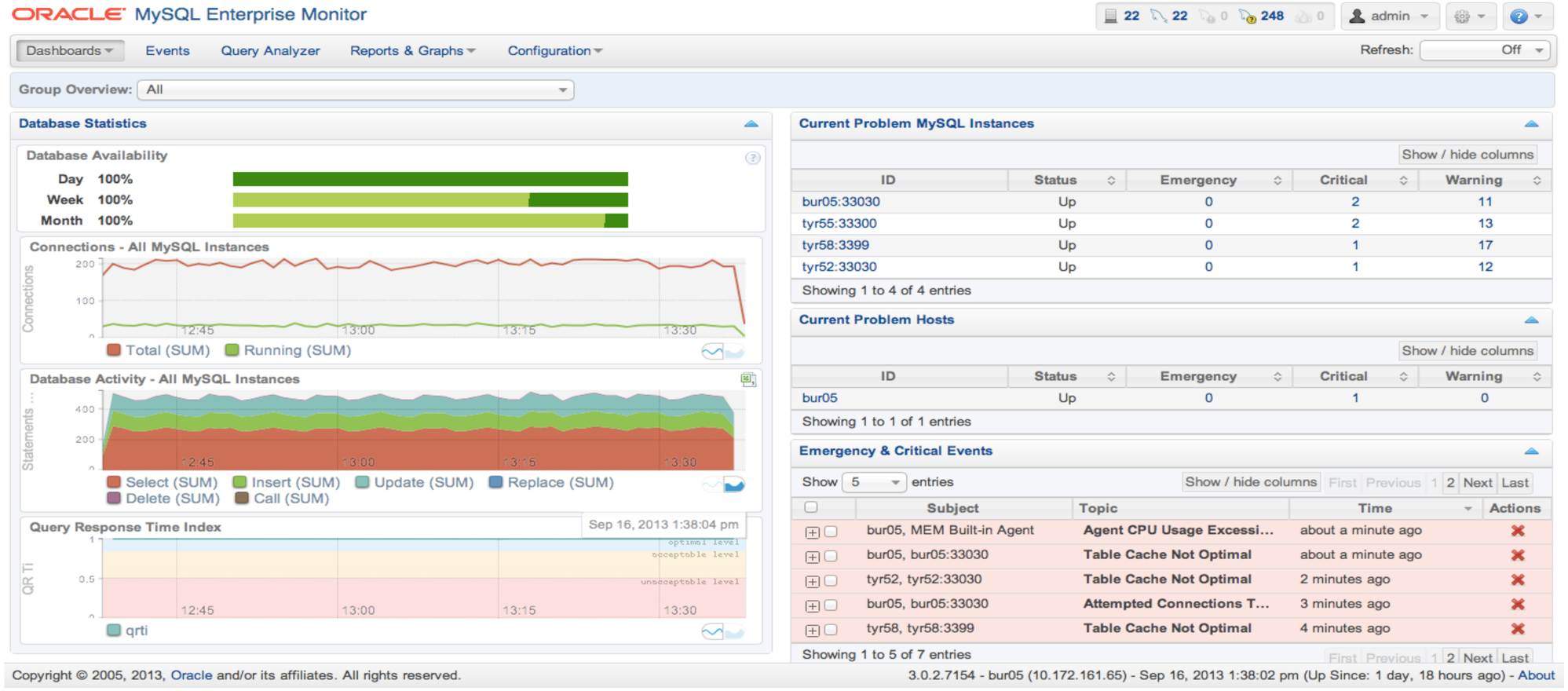
USE YOUR BRAIN !!!... ;-)

#2 - Monitoring is THE MUST !

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

MySQL Enterprise Monitor

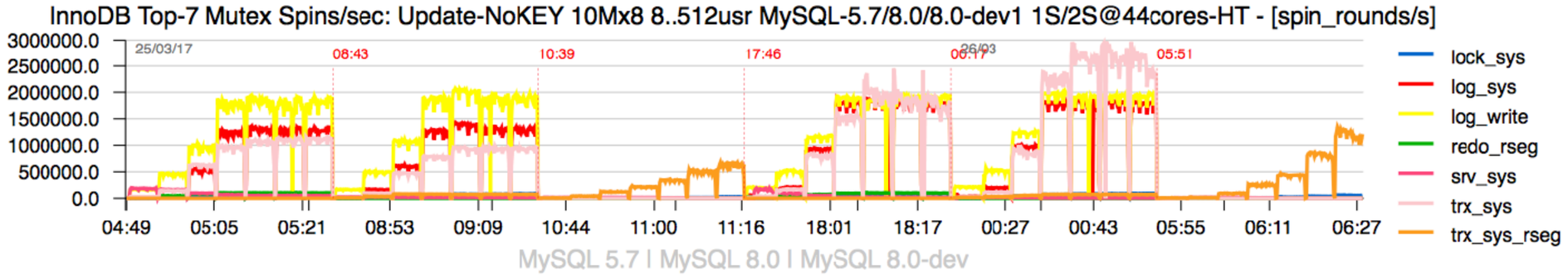
- Fantastic tool!
  - Did you already try it?.. Did you see it live?..

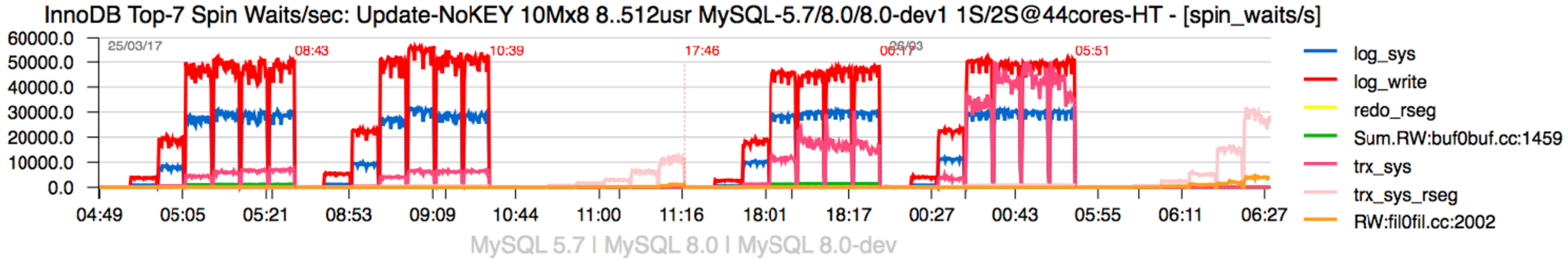


ORACLE

Other Monitoring Tools

- Cacti, Zabbix, Nagios, Solarwinds, VividCortex, PMM, etc…..
- ***dim_STAT***
  - yes, 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…

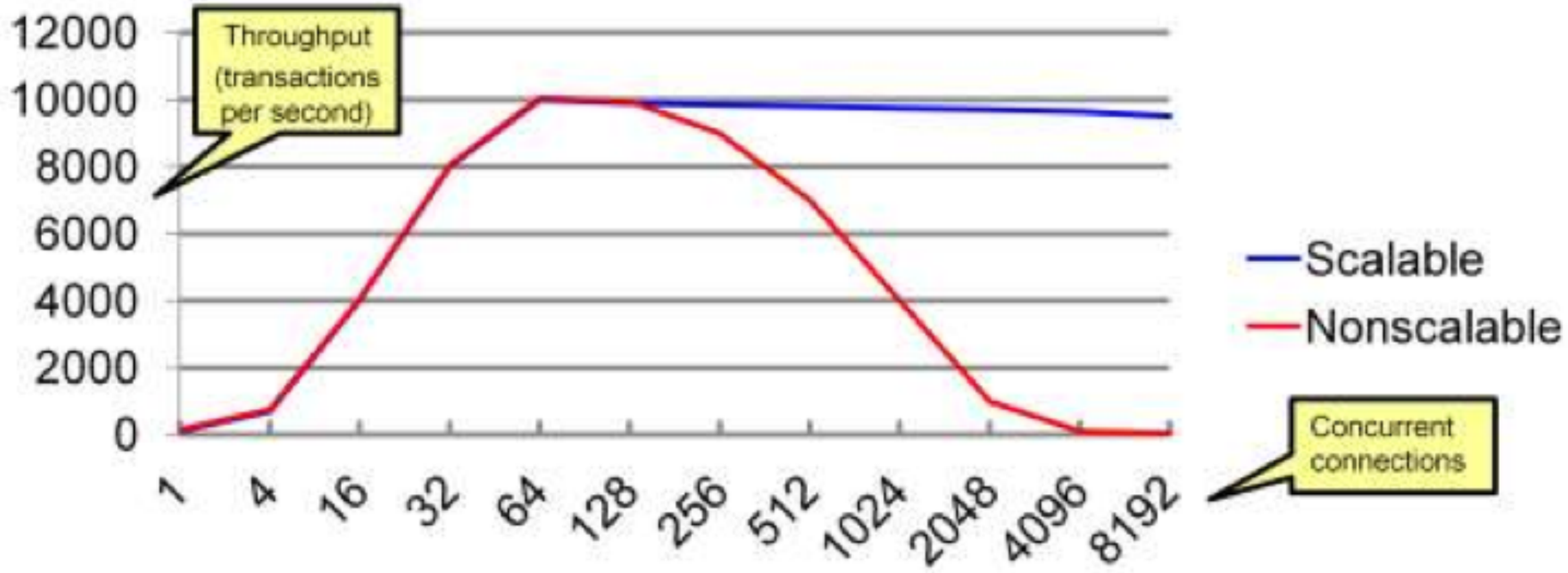
- **always** validate the impact of your Monitoring on your Production ;-)
- taking 1sec measurements is fine, except :
  - if it's eating 100% CPU time on one or more CPU cores..
  - reducing your network traffic / latency..
  - eats your RAM, etc.
- avoid to be too much intrusive on MySQL/InnoDB internals..
  - 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 ;-))

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)..
  - Intel 2CPU : 56cores-HT (March 2017)..
  - …
- Scalability In Few Words :
  - your software is able to deliver a **higher** throughput if more HW resources are available..
  - (then, scaling well or not is another story ;-))

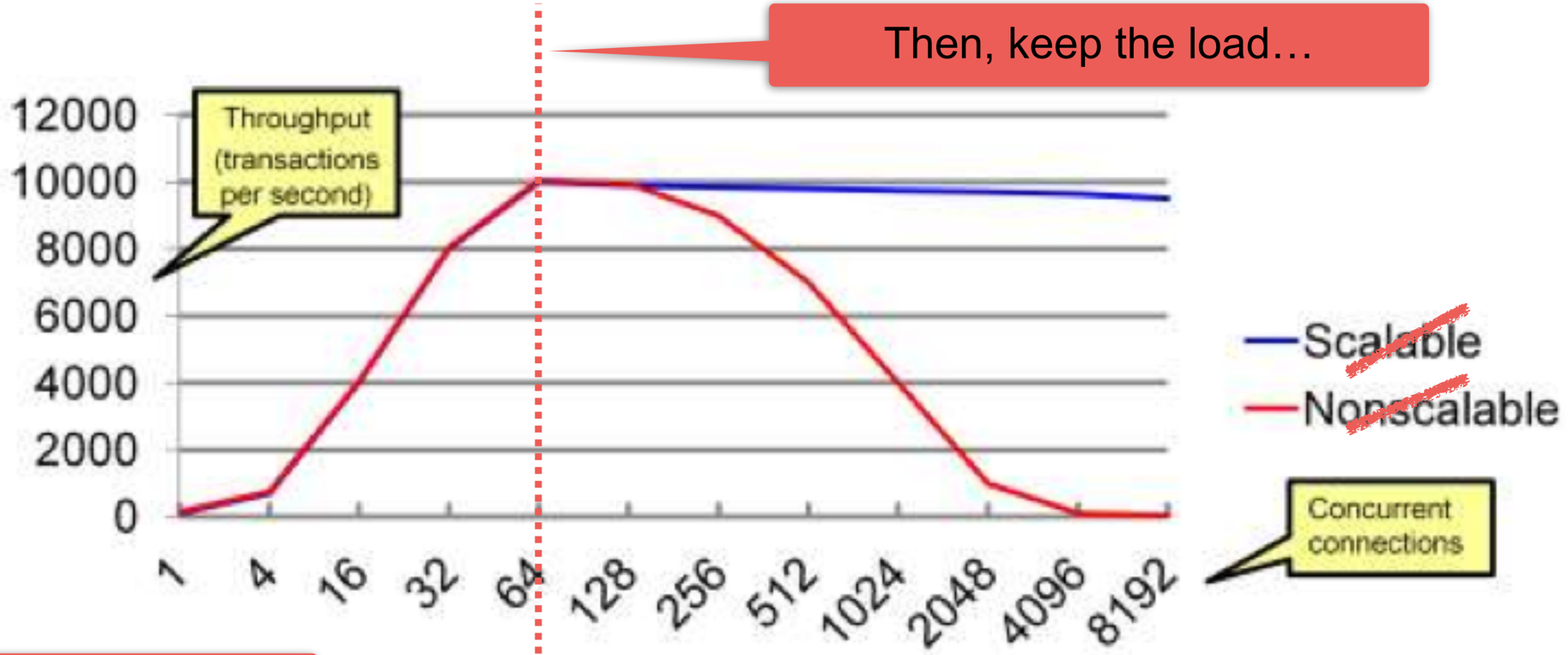
A B-shit Slide…

- 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..

Why Benchmarks ?..

- Production Performance issues ?..
  - 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..
- Don’t arrange feet to boots (BenchMarketing)..

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..” ;-))
  - And even if you were doing something wrong, try to understand its impact..
  - (Best Practice #1 once again ;-))

“Generic” Test Workloads @MySQL

- Sysbench
  - “Entry Ticket” Workloads, looks simple, but still the most complete test kit !
  - OLTP, RO/RW, N-tables, lots test workload load options, deadlocks
- DBT2 / TPCC-like
  - OLTP, RW, pretty complex, growing db, no options, deadlocks
  - in reality using mostly 2 tables only! (thanks Performance Schema ;-))
- dbSTRESS
  - OLTP, RO/RW, several tables, one most hot, configurable, no deadlocks
- iiBench
  - pure INSERT bombarding (like time series) + optionally SELECTs
- LinkBench (Facebook)
  - OLTP, RW, looks intensive and IO-hungry, needs more investigations..
- DBT3
  - DWH, RO, complex heavy queries, loved by Optimizer Team ;-)

Test Plan (by Dimitri) / *if you really want to understand..*

- Step #1 - Read-Only, In-Memory
  - dataset : not too small, not too big - for ex. 10M x 8-tables
  - tuning : just default
  - goals :
    - check the max QPS you can obtain..
    - do you scale with growing load ?..
    - what is blocking ?..
    - anyone doing better than you on the same HW/OS ?..
    - if yes : why ?.. => until this remains unresolved, no reason to go more far ;-))
- Note :
  - be sure your queries are having a “good” execution plan, no missed indexes, etc..
  - get this all fixed first, otherwise no need to continue ;-)

Test Plan (by Dimitri) / *if you really want to understand..*

- Step #2 - Read-Only, In-Memory
  - dataset : much bigger than in Step #1, but still in-memory.. - for ex. 50M x 8-tables
  - tuning : just default
  - goals :
    - check the max QPS you can obtain.. - and now also vs Step #1
    - do you still scale with growing load ?..
    - what is blocking ?..
    - anyone doing better than you on the same HW/OS ?..
    - if yes : why ?.. => until this remains unresolved, no reason to go more far ;-))

Test Plan (by Dimitri) / *if you really want to understand..*

- Step #3 - Read-Only, 1/2 In-Memory (going IO-bound)
  - dataset : same as in Step #2, just BP size = 1/2 of dataset size
  - tuning :
    - BP size = 1/2 dataset
    - O_DIRECT_NO_FSYNC (to be sure we don’t read from FS cache)
  - goals :
    - your IO subsystem will be much involved now..
    - check the max QPS you can obtain.. - and now also vs Step #2
    - do you still scale with growing load ?..
    - are you already blocked by IO ?..
    - what else is blocking ?..
    - anyone doing better than you on the same HW/OS ?..
    - if yes : why ?.. => until this remains unresolved, no reason to go more far ;-))

Test Plan (by Dimitri) / *if you really want to understand..*

- Step #4 - Read-Only, 1/4 In-Memory (going very IO-bound)
  - dataset : same as in Step #2, just BP size = 1/4 of dataset size now
  - tuning :
    - BP size = 1/4 dataset
    - O_DIRECT_NO_FSYNC (to be sure we don’t read from FS cache)
  - goals :
    - your IO subsystem will be yet much more involved now..
    - check the max QPS you can obtain.. - and now also vs Step #2 & #3..
    - do you still scale with growing load ?..
    - are you already blocked by IO ?..
    - what else is blocking ?..
    - anyone doing better than you on the same HW/OS ?..
    - if yes : why ?.. => until this remains unresolved, no reason to go more far ;-))

Test Plan (by Dimitri) / *if you really want to understand..*

- Step #1-bis - Read+Write, In-Memory
  - dataset : not too small, not too big - for ex. 10M x 8-tables
  - tuning :
    - use big REDO (ex. 32GB), enough BP instances (ex.16), O_DIRECT+AIO, IO capacity..
    - trx_commit=1
  - goals :
    - follow your Checkpoint Age, tune IO capacity / IO capacity max, see your flushing rate/time
    - check the max QPS you can obtain..
    - do you scale with growing load ?.. do you need to limit concurrency ?..
    - what is blocking ?..
    - anyone doing better than you on the same HW/OS ?..
    - if yes : why ?.. => until this remains unresolved, no reason to go more far ;-))
- Note :
  - be sure your workload is free of “artificial” deadlocks, etc.
  - get this all fixed first, otherwise no need to continue ;-)

Test Plan (by Dimitri) / *if you really want to understand..*

- Step #2-bis - Read+Write, In-Memory
  - dataset : much bigger, but still kept in memory - for ex. 50M x 8-tables
  - tuning :
    - use big REDO (ex. 32GB), enough BP instances (ex.16), O_DIRECT+AIO, IO capacity..
    - trx_commit=1
  - goals :
    - follow your Checkpoint Age, tune IO capacity / IO capacity max, do you need to adjust ?
    - check the max QPS you can obtain..
    - do you still scale with growing load ?.. do you need to limit concurrency ?..
    - what is blocking ?..
    - anyone doing better than you on the same HW/OS ?..
    - if yes : why ?.. => until this remains unresolved, no reason to go more far ;-))

Test Plan (by Dimitri) / *if you really want to understand..*

- Step #3-bis - Read+Write, 1/2 In-Memory
  - dataset : same as in #2-bis, BP size= 1/2 dataset
  - tuning :
    - use big REDO (ex. 32GB), enough BP instances (ex.16), O_DIRECT+AIO, IO capacity..
    - trx_commit=1
  - goals :
    - follow your Checkpoint Age, tune IO capacity / IO capacity max, do you need to adjust ?
    - follow your free pages demand rate => you'll probably need to adjust LRU depth..
    - follow your LRU flushing/evict rate/times => need adjust cleaners/ BP instances ?..
    - check the max QPS you can obtain..
    - do you still scale with growing load ?.. do you need to limit concurrency ?..
    - what is blocking ?.. are you already limited by IO ?..
    - anyone doing better than you on the same HW/OS ?..
    - if yes : why ?.. => until this remains unresolved, no reason to go more far ;-))

Test Plan (by Dimitri) / *if you really want to understand..*

- Step #4-bis - Read+Write, 1/4 In-Memory
  - dataset : same as in #2-bis, BP size= 1/4 dataset
  - tuning :
    - use big REDO (ex. 32GB), enough BP instances (ex.16), O_DIRECT+AIO, IO capacity..
    - trx_commit=1
  - goals :
    - follow your Checkpoint Age, tune IO capacity / IO capacity max, do you need to re-adjust ?
    - follow again your free pages demand rate => you'll probably need to Re-adjust LRU depth..
    - follow your LRU flushing/evict rate/times => need Re-adjust cleaners/ BP instances ?..
    - check the max QPS you can obtain..
    - do you still scale with growing load ?.. do you need to limit concurrency ?..
    - what is blocking ?.. are you already limited by IO ?..
    - anyone doing better than you on the same HW/OS ?..
    - if yes : why ?.. => until this remains unresolved, no reason to go more far ;-))

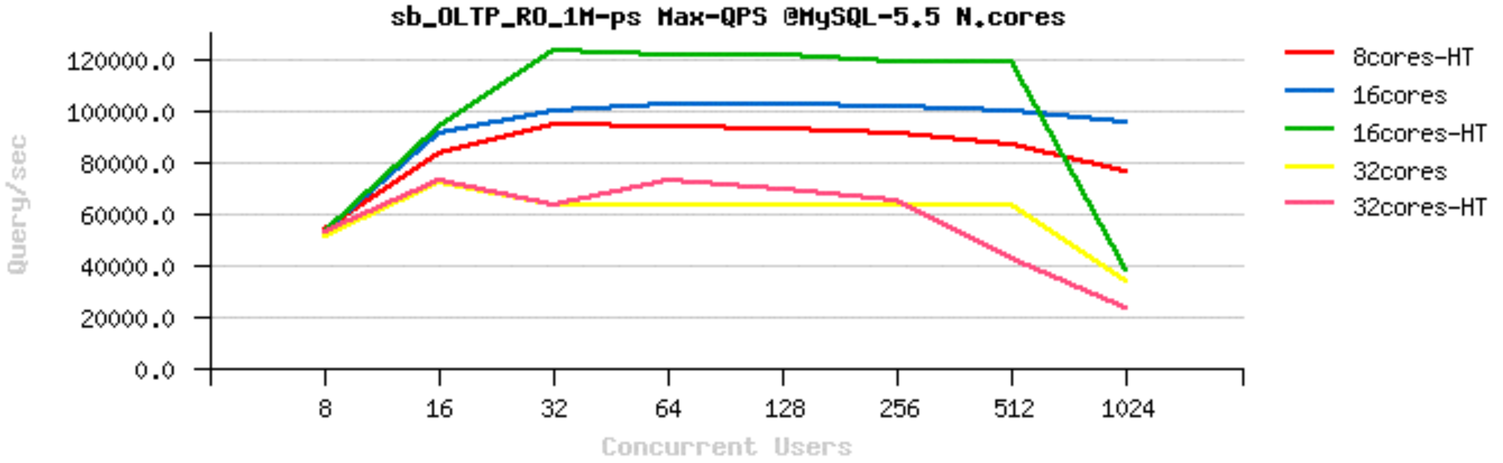
MySQL Performance milestones

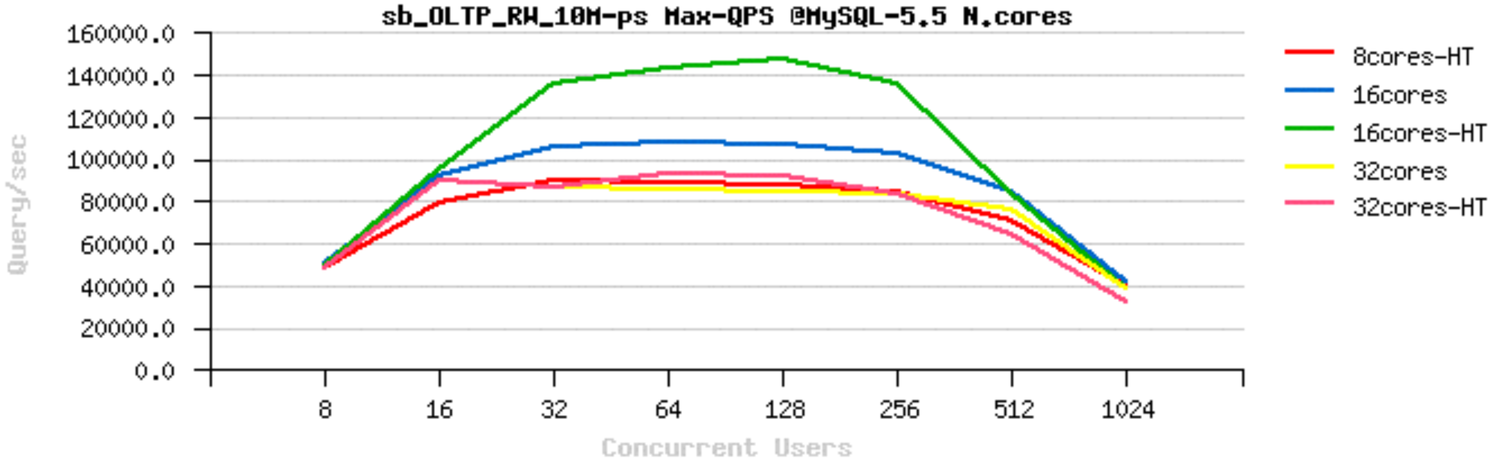
- MySQL 5.5
  - delivered “already known” solutions (except BP instances and few other)..
- MySQL 5.6
  - first fundamental changes (kernel_mutex split, G5 patch, RO transactions, etc..)
- MySQL 5.7
  - finally fully unlocked READs, no more contentions on the “Server” layer, etc..
- MySQL 8.0-dev
  - main focus is on efficiency : do more on the same HW ;-))
  - work in progress..

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

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

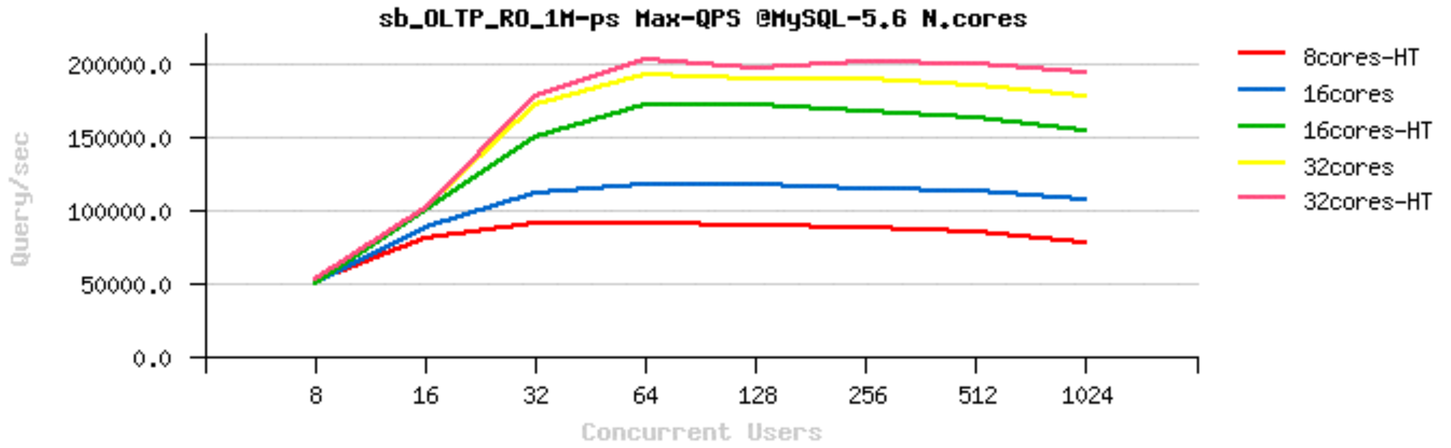
- MySQL 5.5 : RO & RW
  - QPS Max on 16cores
  - worse on 32cores
  - Note: RW out-pass RO!

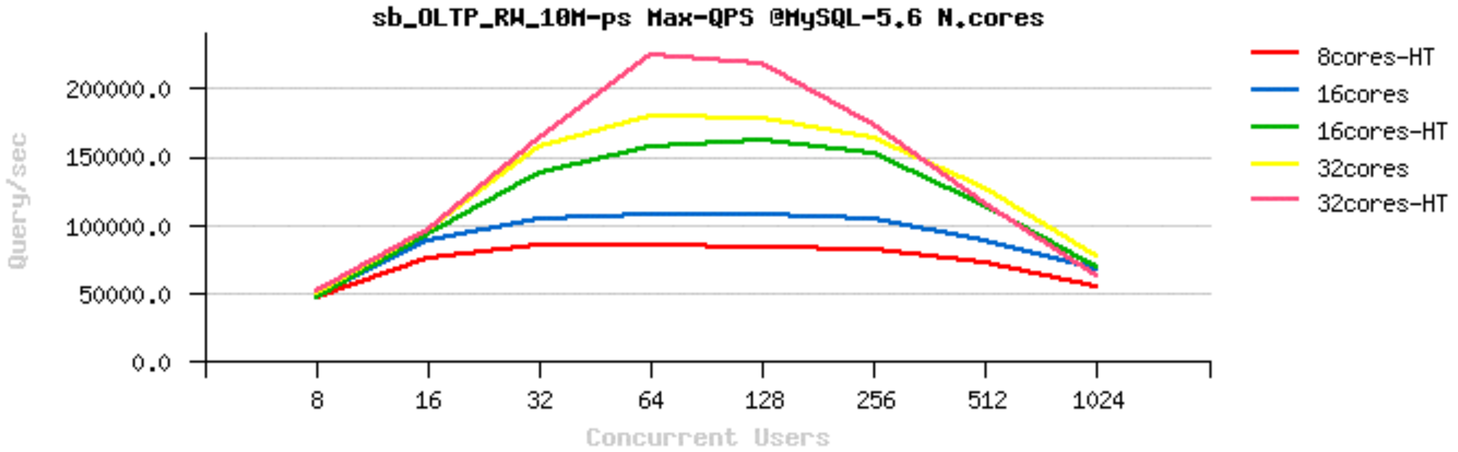




From where we're coming with MySQL 5.7 ?..

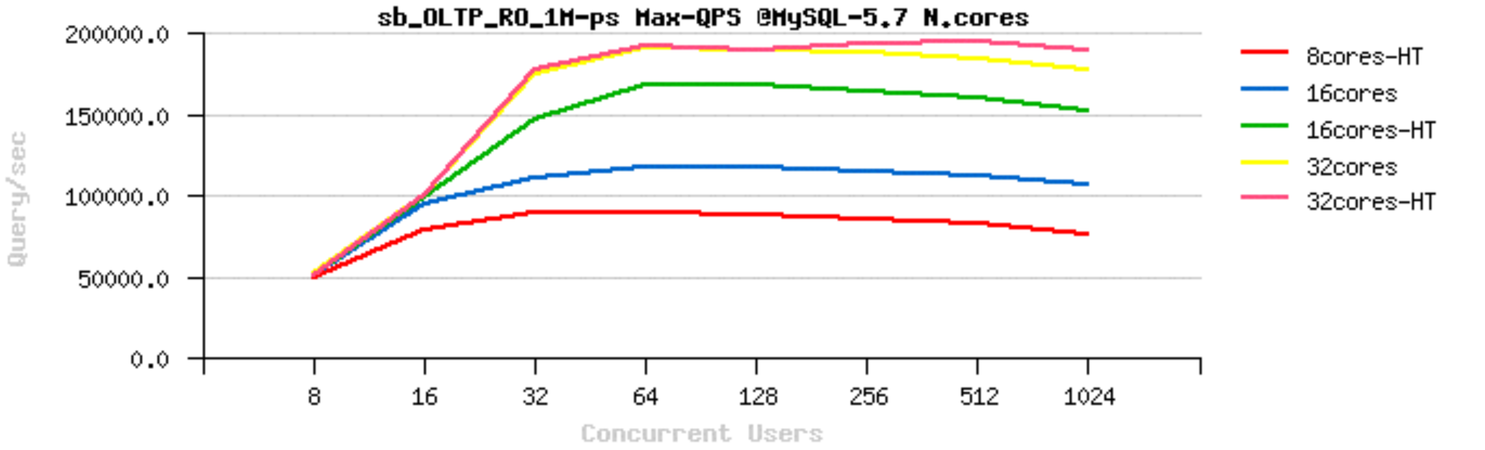
- MySQL 5.6 : RO & RW
  - not lower on 32cores!! ;-)
  - RW out-pass RO !!..??

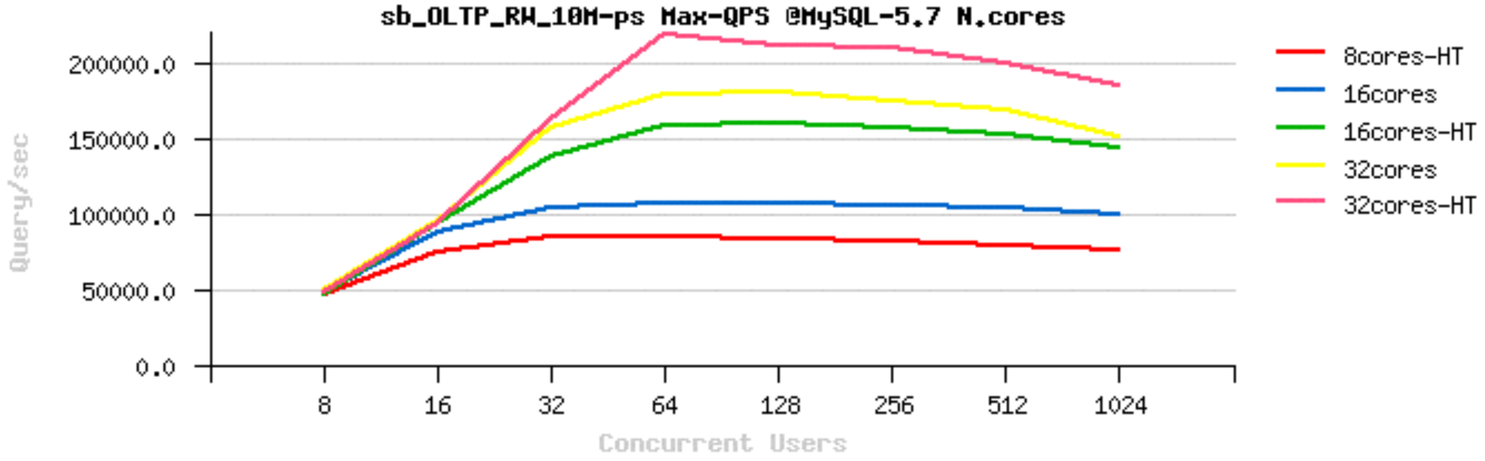




From where we're coming with MySQL 5.7 ?..

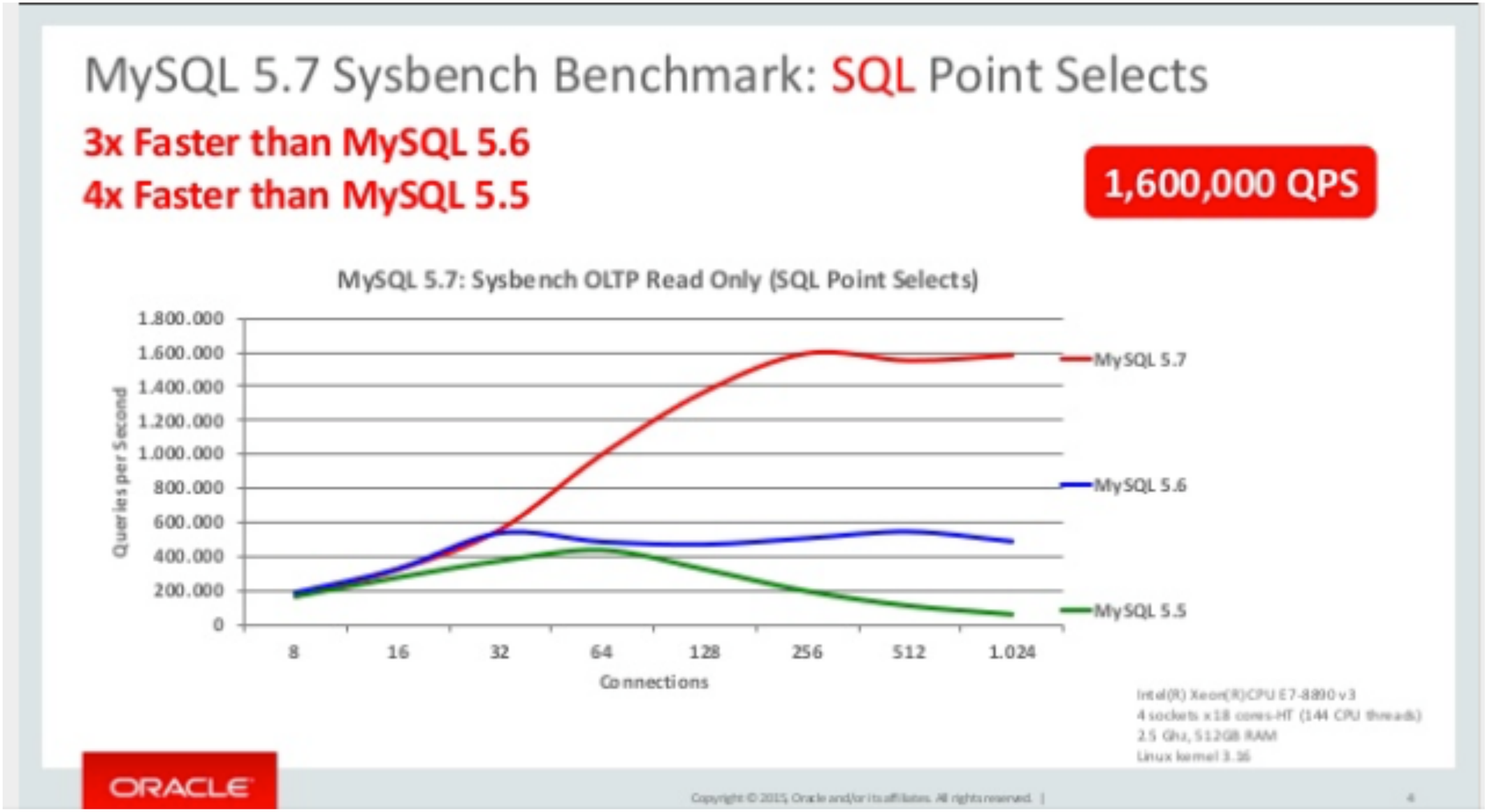
- MySQL 5.7.1 : RO & RW
  - more stable than 5.6
  - **RW** out-pass RO !!..





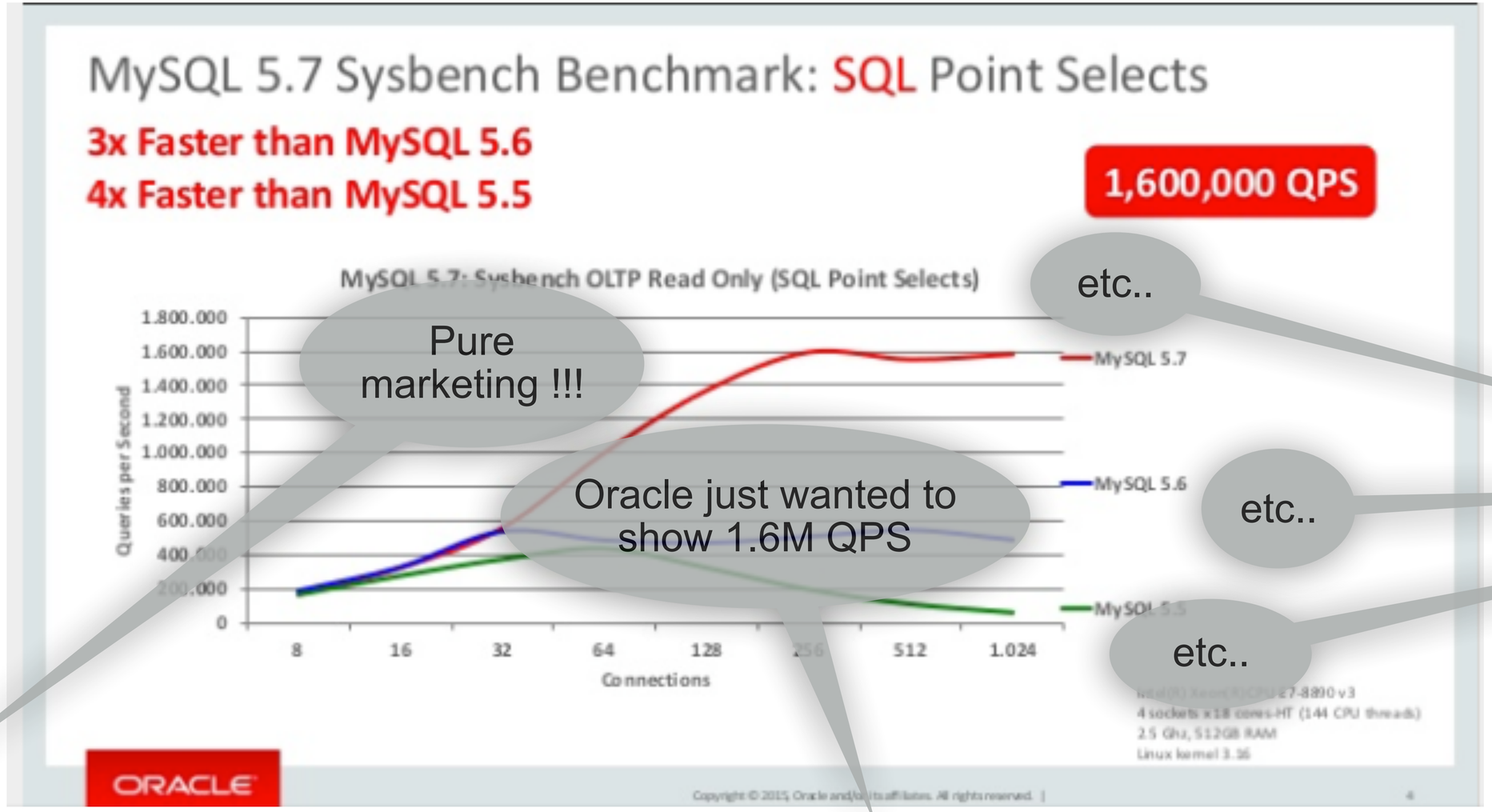
MySQL 5.7 : 1.6M QPS

- What is behind this number ?..



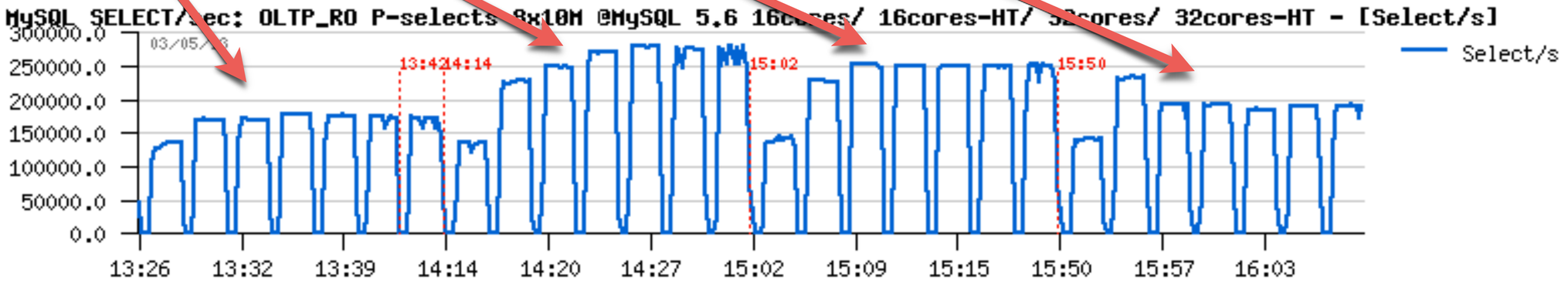
MySQL 5.7 : 1.6M QPS

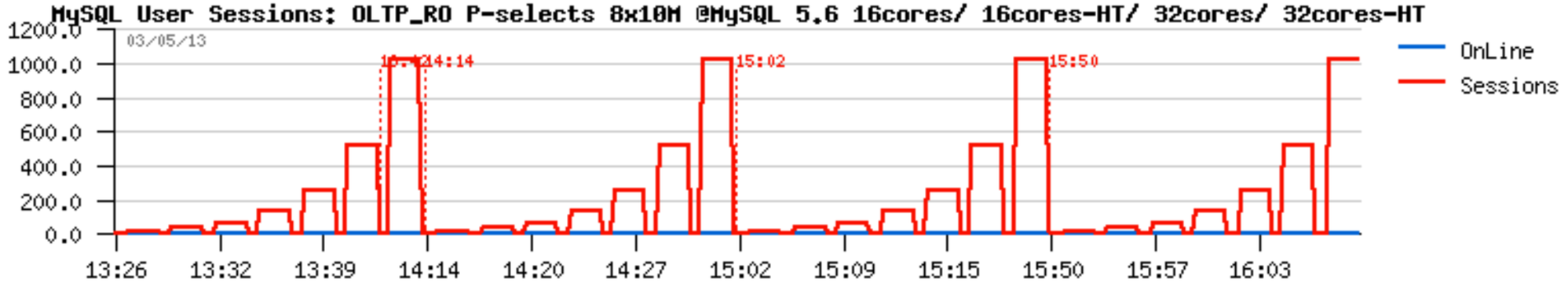
- What is behind this number ?..



Behind the numbers...

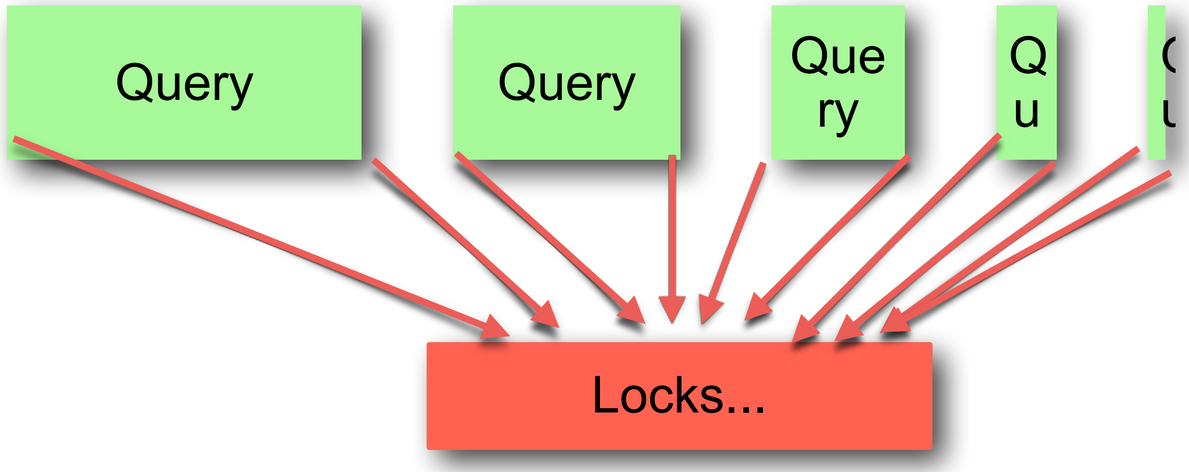
- MySQL 5.6, RO Point-Select Performance
  - 16cores, 16cores-HT, 32cores, 32cores-HT

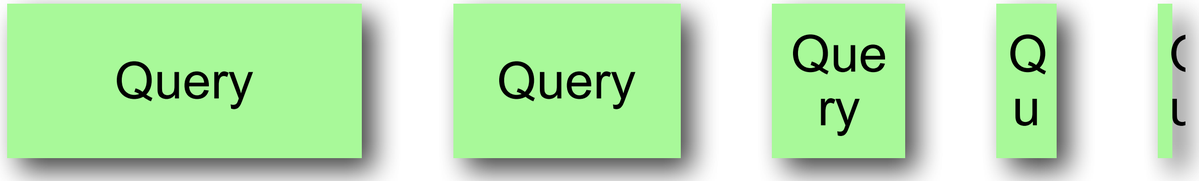




Behind the numbers...

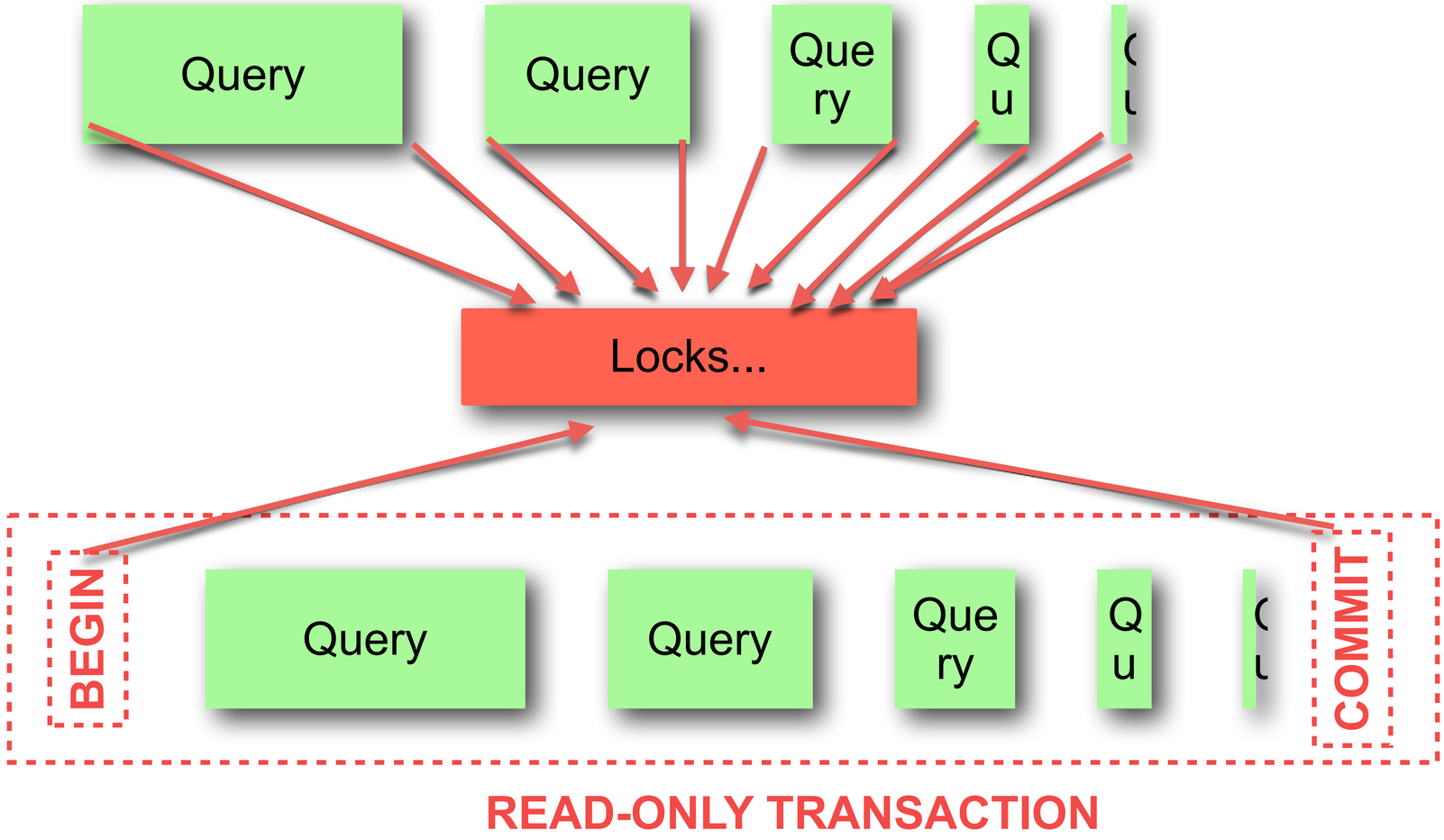
- Why ?..





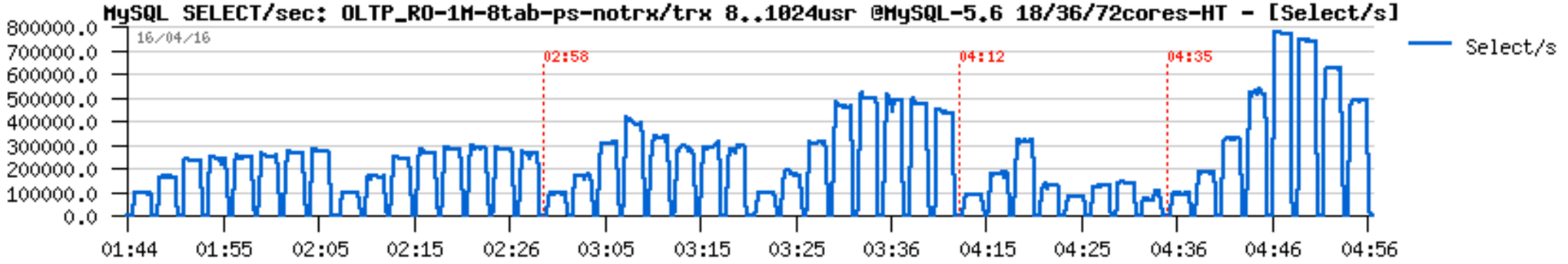
Behind the numbers...

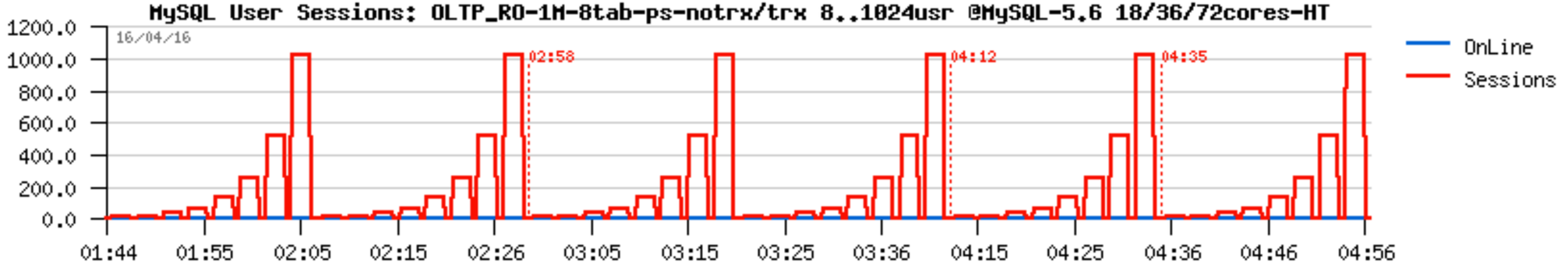
- MySQL 5.6 : Read-Only Transactions "workaround" :



Behind the numbers...

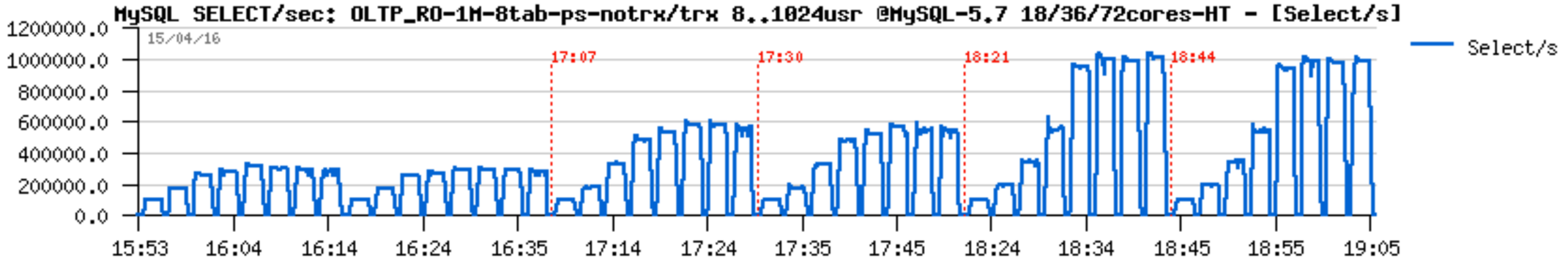
- MySQL 5.6, OLTP_RO-1Mx8-tables, 72cores-HT
  - OLTP_RO : [x14 SELECT Queries]
  - without / with transaction enclosure, 18/36/72cores-HT

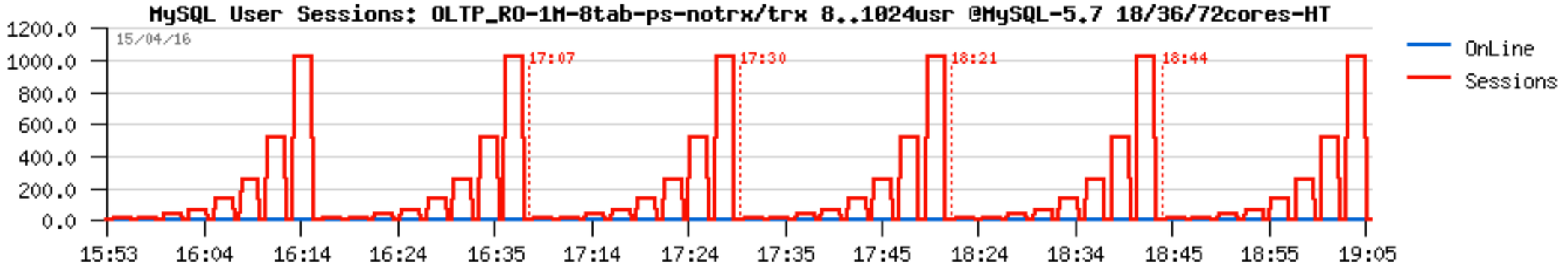




Behind the numbers...

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



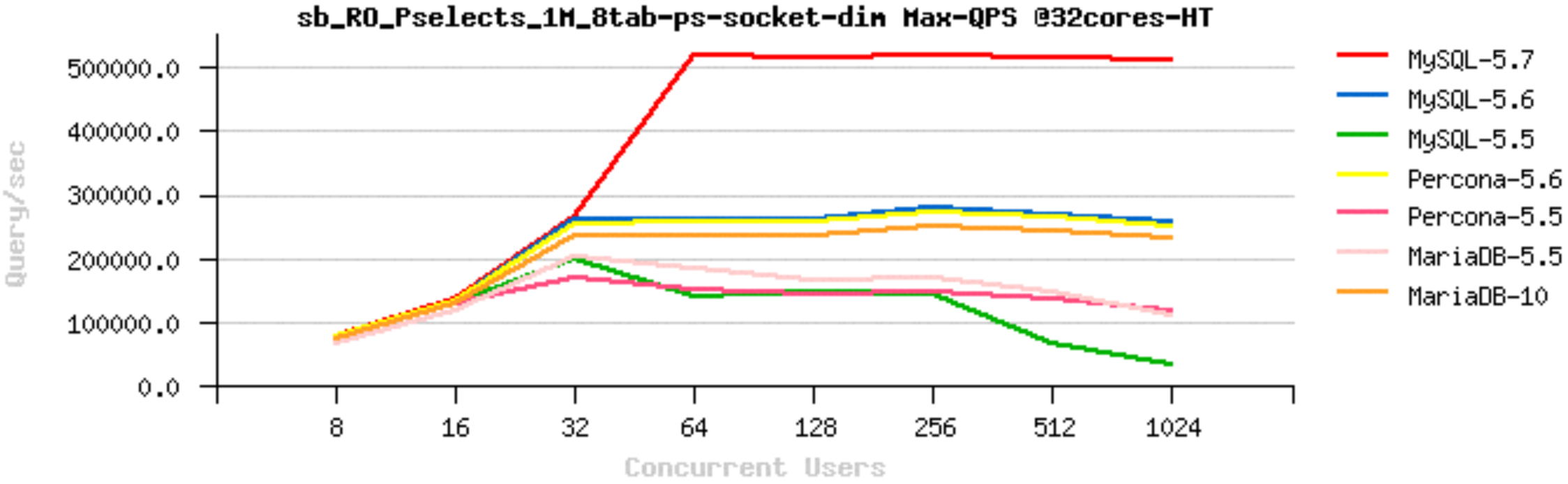


Behind the numbers...

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

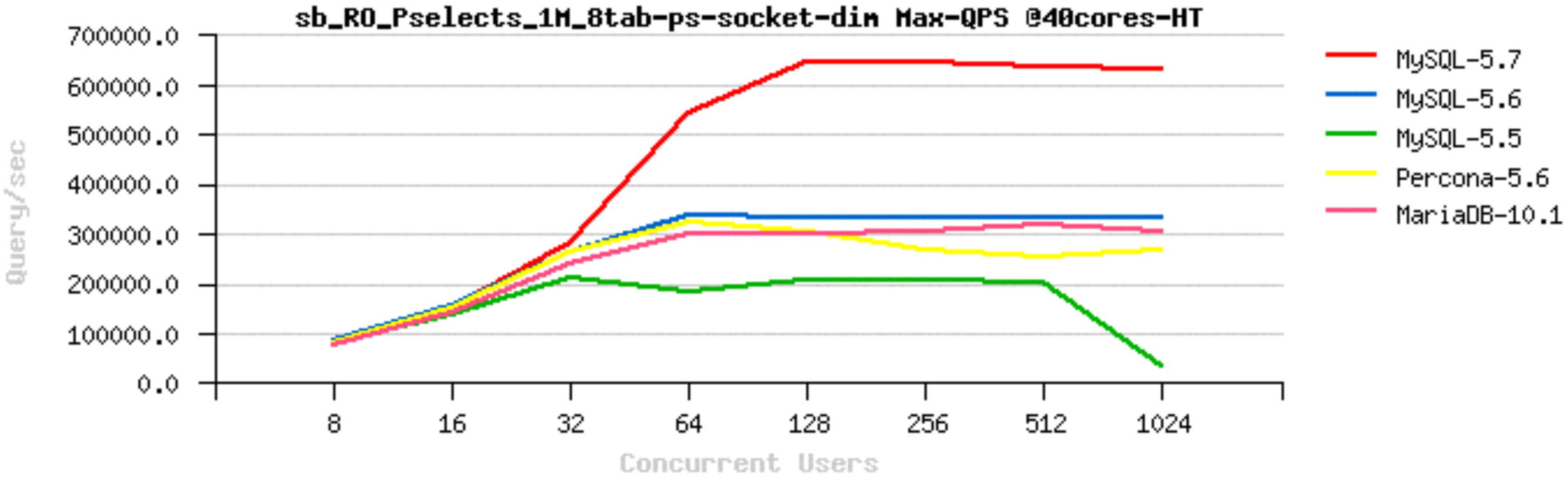
RO In-Memory @MySQL 5.7

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



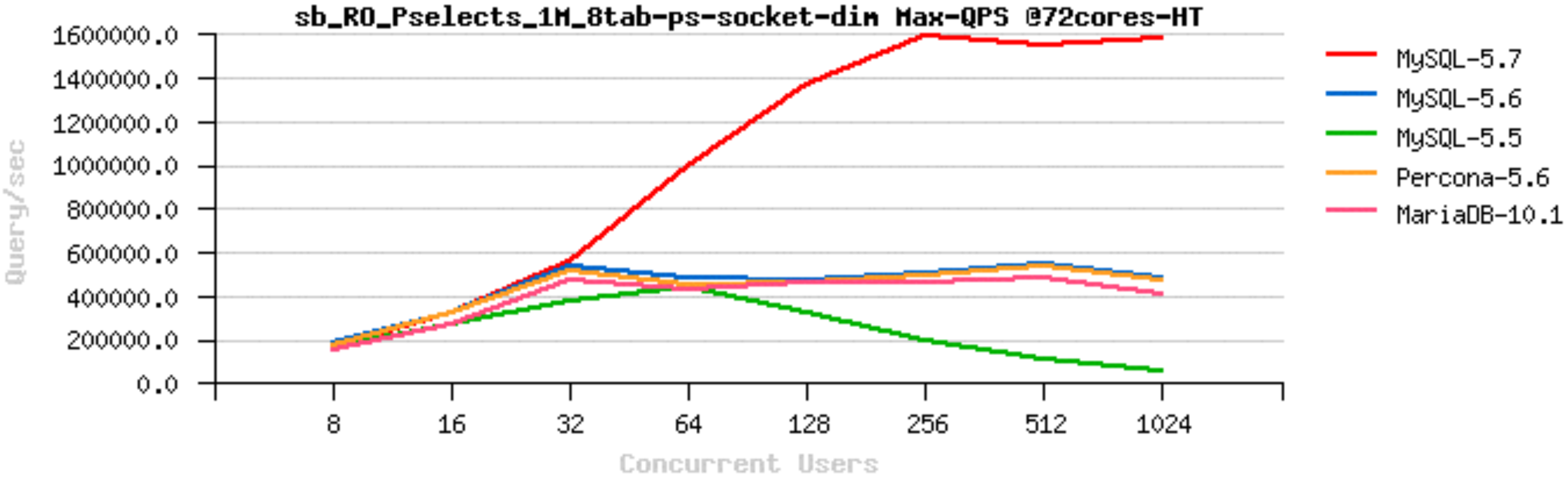
RO In-Memory @MySQL 5.7

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



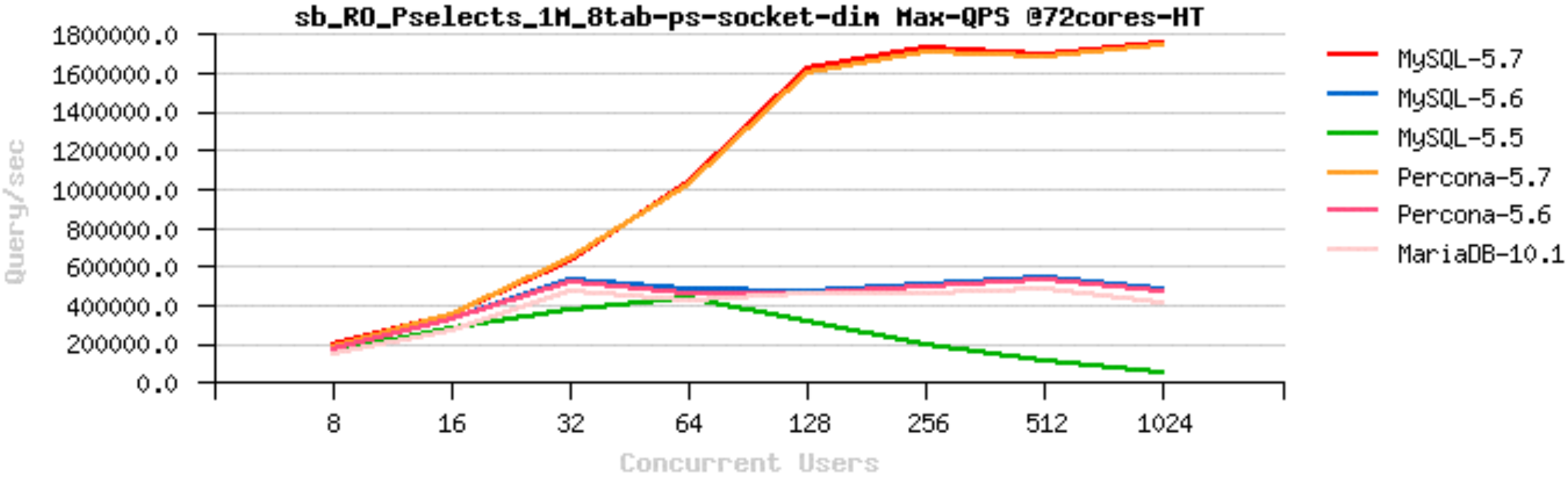
RO Point-Selects @MySQL 5.7 (Oct.2015)

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



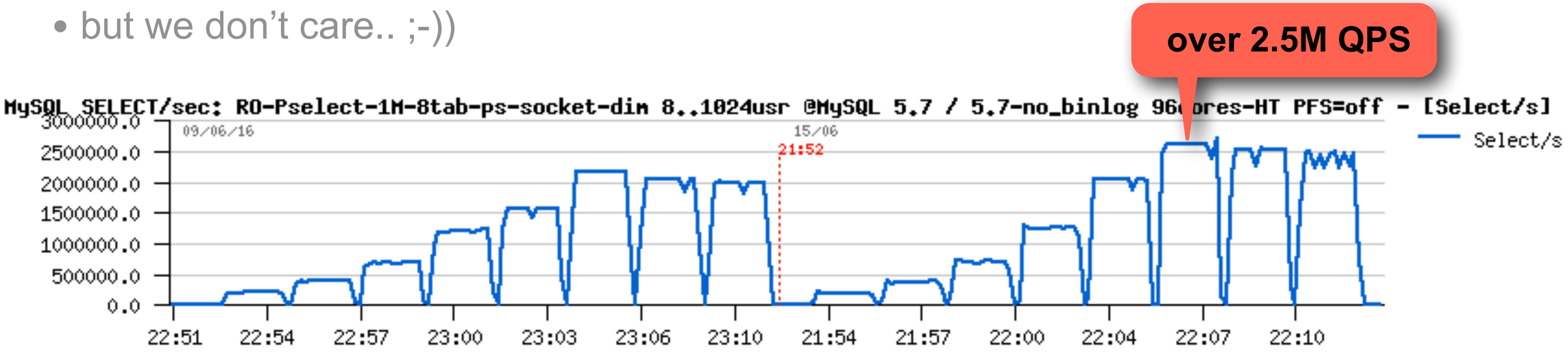
RO Point-Selects @MySQL 5.7 (Apr.2016)

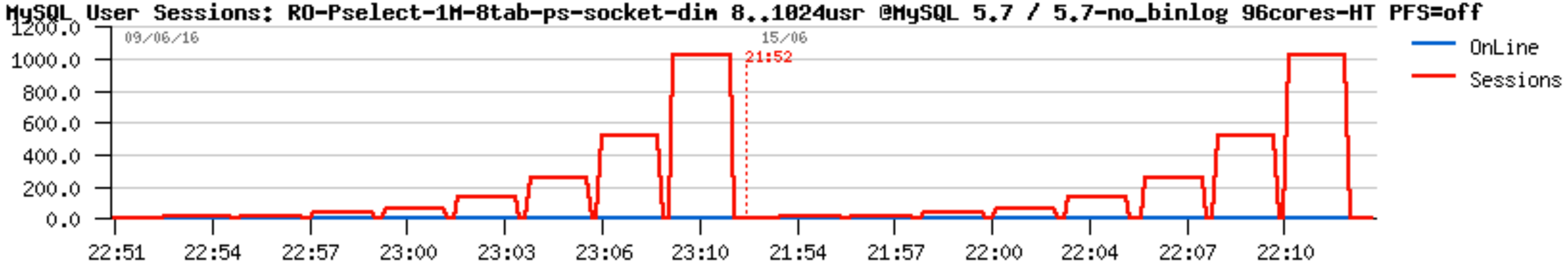
- **1.8M QPS** Sysbench Point-Selects 8-tab, 72cores-HT :
  - or even more, if you really run after numbers.. ;-))



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

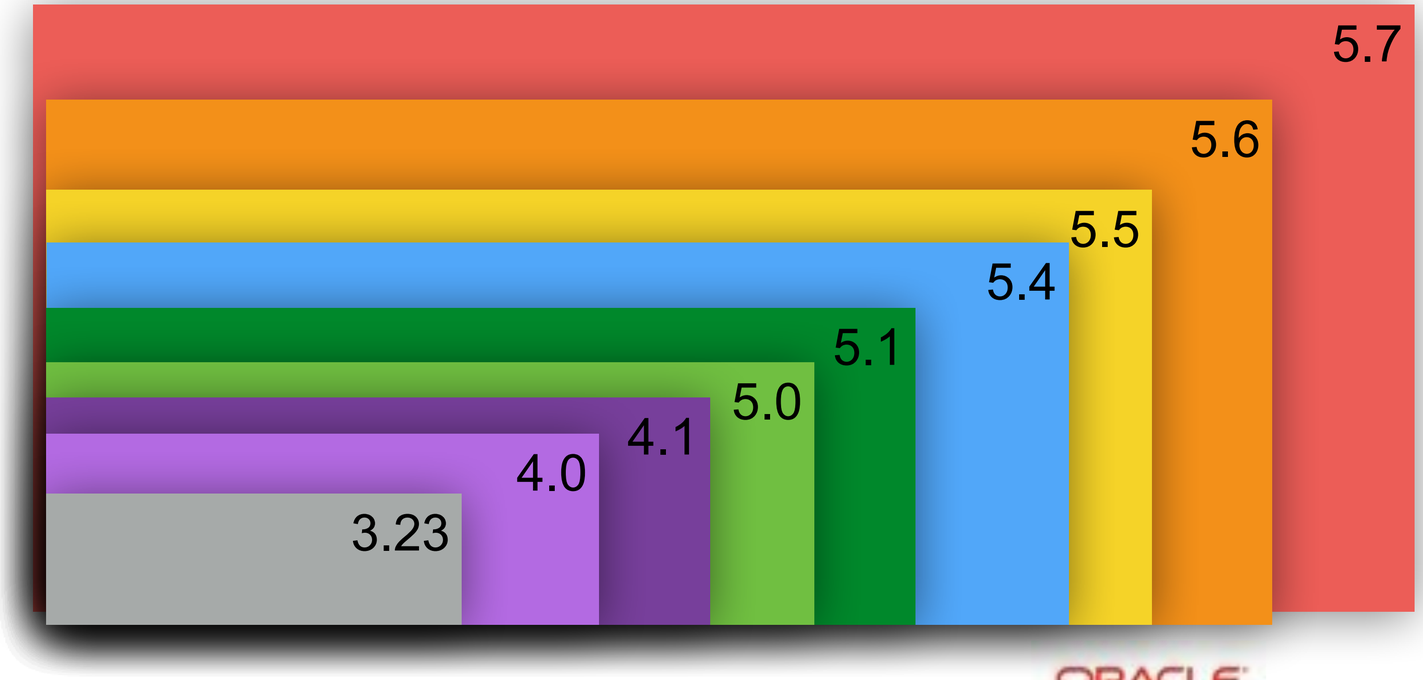
- **2.5M (!!) QPS** Sysbench Point-Selects 8-tab, 96cores-HT :
  - but we don't care.. ;-))





MySQL Performance Evolution

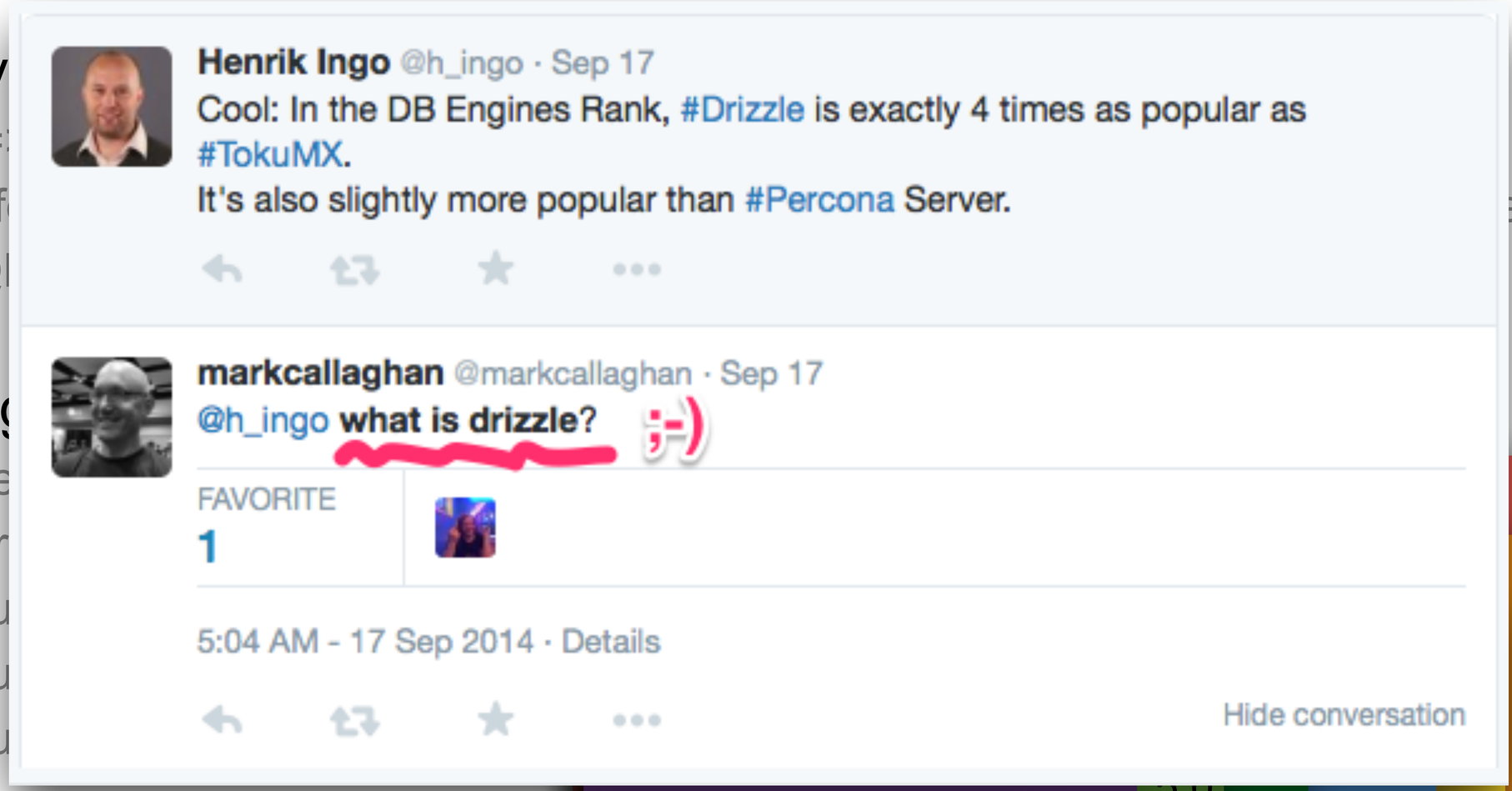
- From version-to-version :
  - 3.23 => 4.0 => 4.1 => 5.0 => 5.1 => 5.4 => 5.5 => 5.6 => 5.7 …
  - more features => longer code path.. (see every: “What is new in MySQL” by Geir)
  - MySQL/InnoDB code is very sensible to CPU cache(s)..
  - single user / low load => going slower..
- Looking back :
  -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

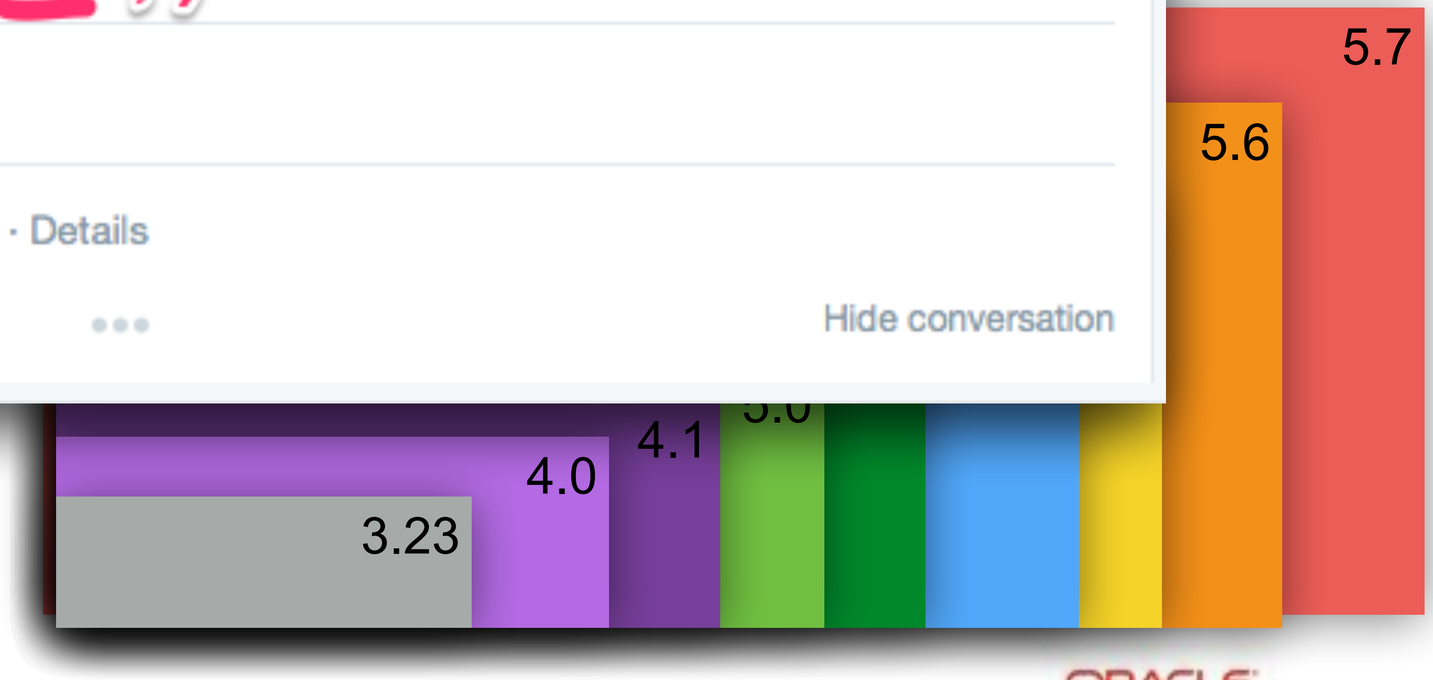


MySQL Performance Evolution

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

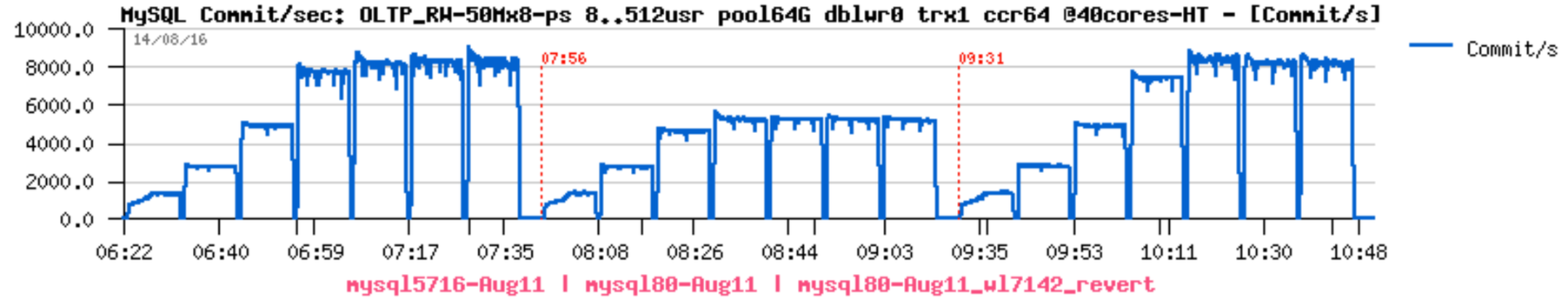
eir)

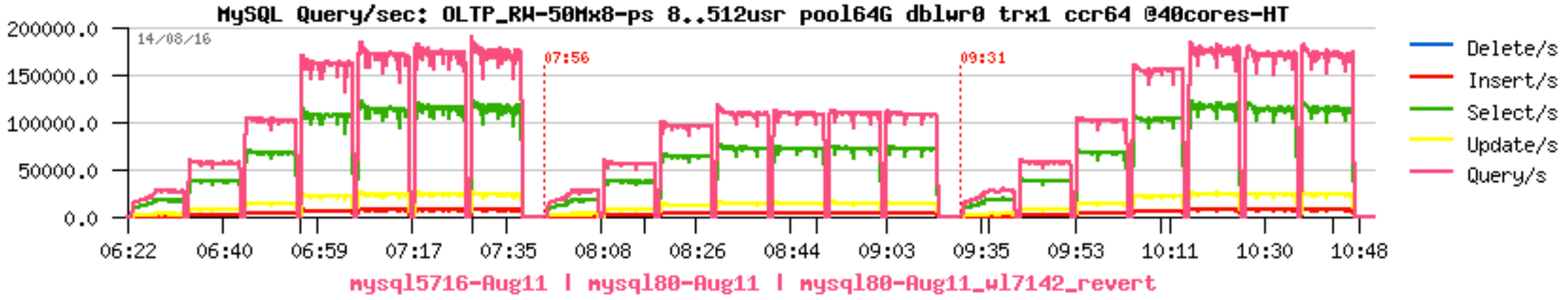




MySQL 8.0 : Why no Performance news until now ?..

- One picture say more than many words ;-)
  - Aug.2016 : revert the changes and re-do the whole work…
  - Team Force : face & fix the problems when they come! (you cannot predict everything)..





Pending issues after MySQL 5.7 GA..

- RO :
  - Block Locks
  - Lookups via Sec.IDX
  - fil_system mutex contention (on every IO)
- RW :
  - Double Write..
  - REDO log related bottlenecks
  - TRX management contentions
  - LOCK management..
  - IO / fil_system
  - RR / RC isolation..
  - UPDATE Performance..
  - INSERT Performance..
  - Purge lagging..
  - …

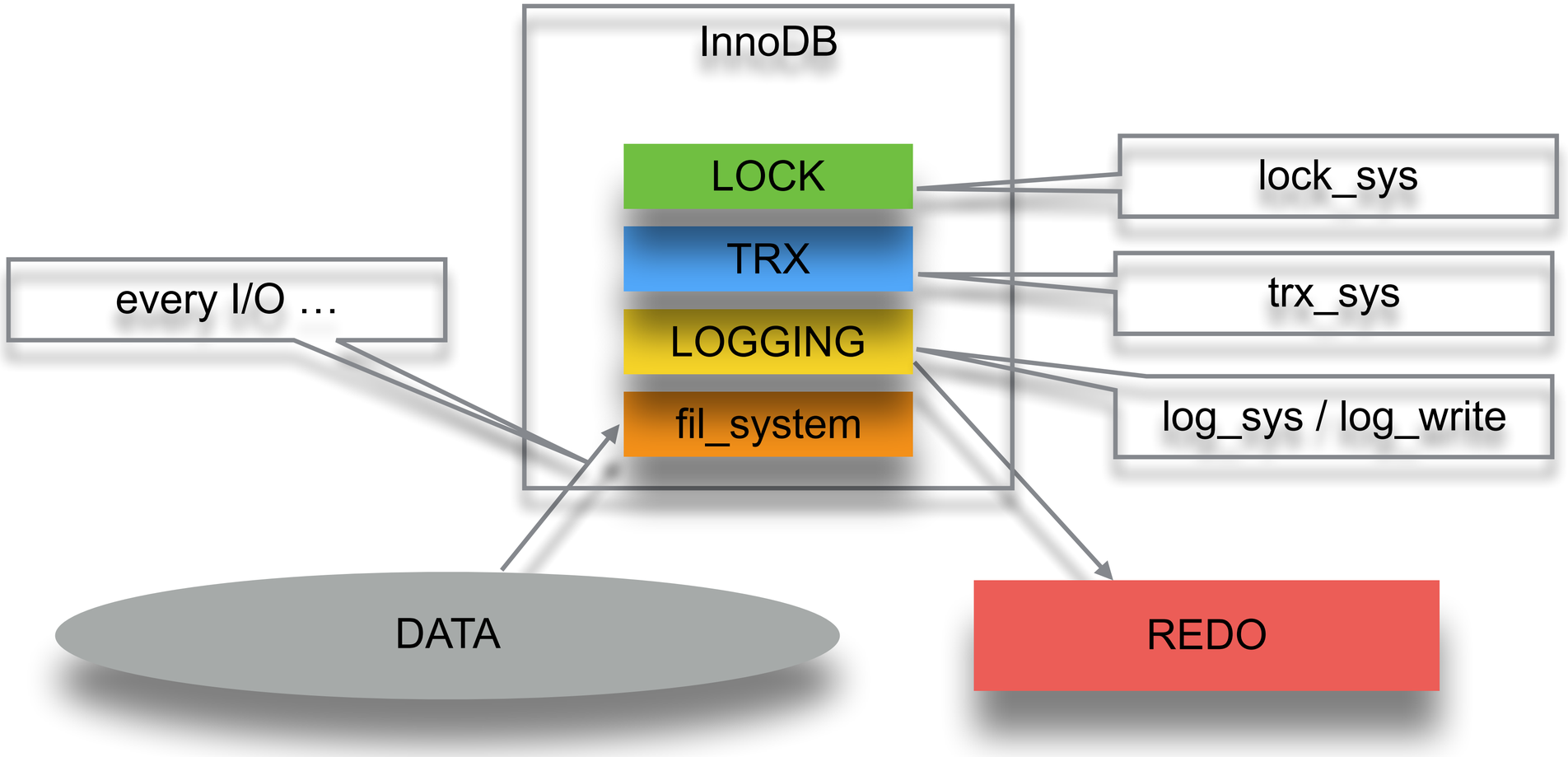
Pending issues after MySQL 5.7 GA..

- RO :
  - Block Locks
  - Lookups via Sec.IDX
  - fil_system mutex contention (on every IO) **<= 8.0-dev**
- RW :
  - Double Write.. **<= 8.0-dev**
  - REDO log related bottlenecks **<= 8.0-dev**
  - TRX management contentions **<= 8.0-dev**
  - LOCK management.. **<= 8.0-dev**
  - IO / fil_system **<= 8.0-dev**
  - RR / RC isolation.. **<= 8.0-dev**
  - UPDATE Performance.. **<= 8.0-dev**
  - INSERT Performance..
  - Purge lagging..
  - …

Few words about Double Write

- Protection from partially written pages..
- Old code is a huge bottleneck itself..
- New code :
  - was ready during winter 2016
  - but not ready on 5.7 GA deadline..
  - means must be validated on 8.0 dev code first..
  - and only then “back-ported” to 5.7 ..
  - but on 8.0 some pre-required changes should be approved before..
  - so, 5.7 still has no new code ;-)
  - expected performance impact : if storage is able to keep x2 higher writes, **then no diff (!)**
  - (double write : writing x2 times more)
- But real fix : get a rid of it ;-)
  - not for soon yet..

MySQL 8.0-dev : New Design for InnoDB Fundamentals..

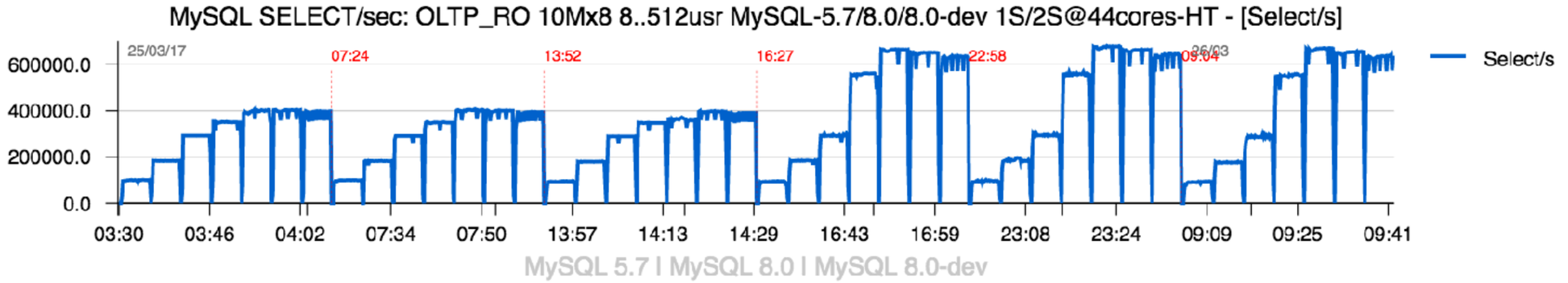


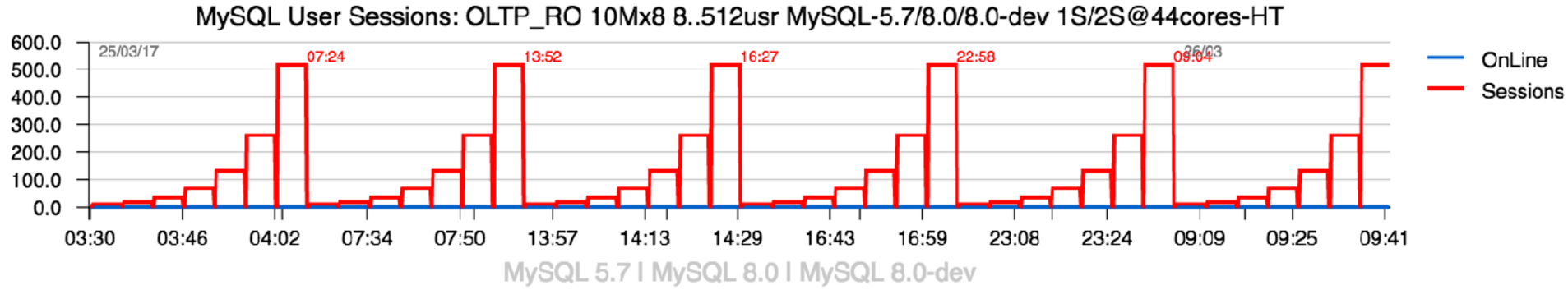
MySQL 8.0-dev Performance : Following “The Test Plan”

- HW :
  - 4S 40cores-HT / 2S 44cores-HT / 4S 96cores-HT (RAM: >= 256G)
    - bug on Linux kernel not allowing to use 96cores fully, so presenting mainly 1S/2S 44cores result
- Workloads :
  - Sysbench RO/ RW/ Update-NoKEY, 10Mx8 / 50Mx8, uniform / pareto
  - DBT2 W1000/ LinkBench-150G/ dbSTRESS-20M (400M)/ iiBench-x16-100M
- Config :
  - BP= 128G/ 64G/ 32G
  - trx_commit=1, concurrency=0, spin delay=6
  - O_DIRECT_NO_FSYNC + AIO, EXT4, flash storage (Seagate, Intel, Samsung)
  - redo 32GB, io capacity max up to 40K
  - RR / RC isolation, VATS
- Load levels:
  - 8, 16, 32, .. 512 concurrent users

Sysbench OLTP_RO 10Mx8-tables

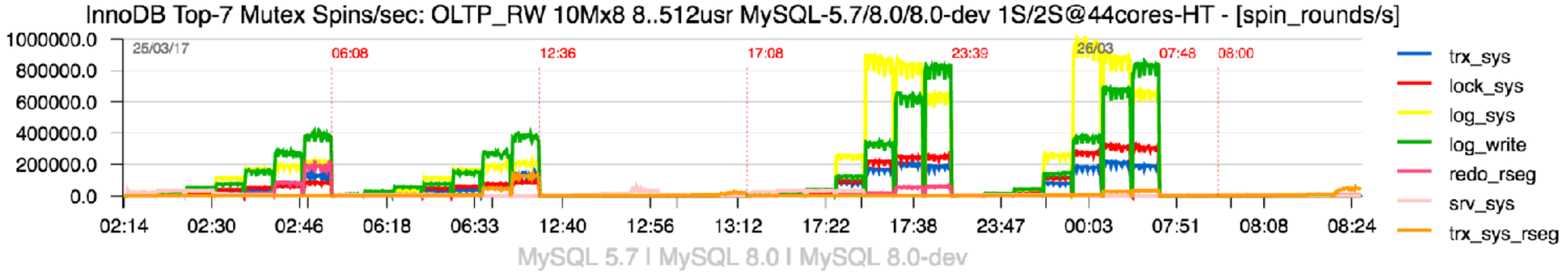
- Observations :
  - same QPS on 5.7 / 8.0-dev, no impact, fine..

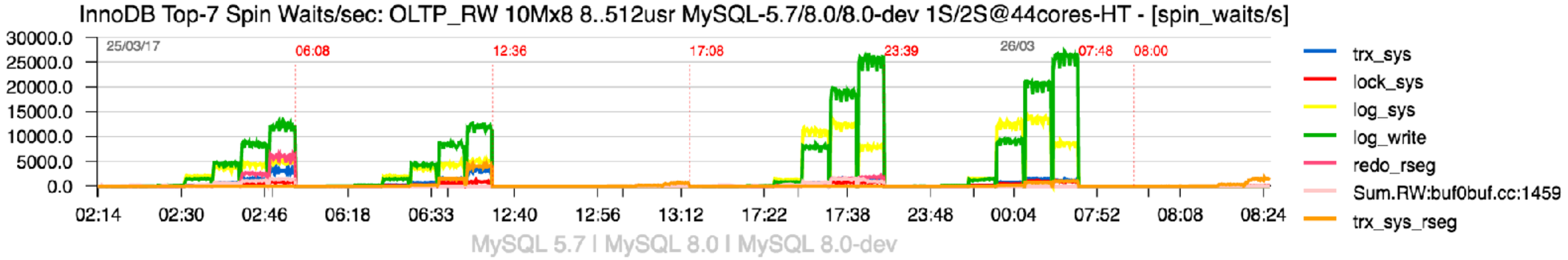




Sysbench OLTP_RW 10Mx8-tables

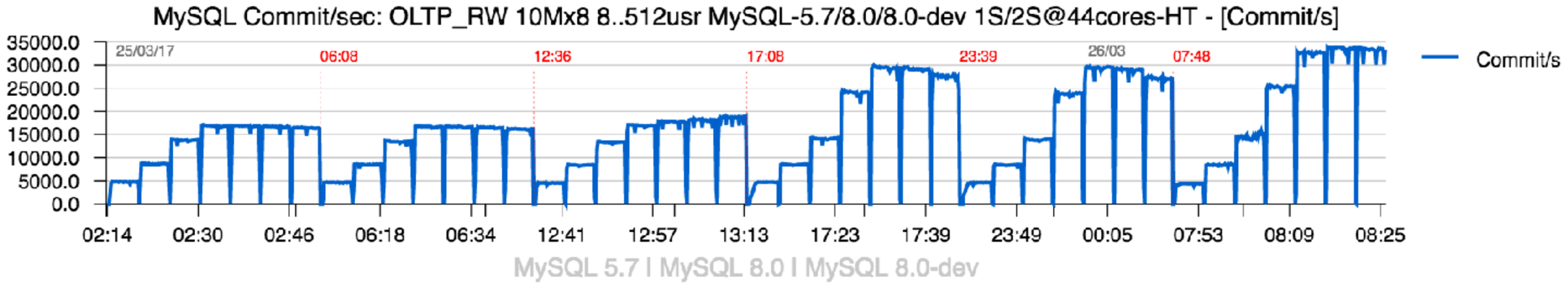
- Observations :
  - 8.0-dev : all hot spin waits are gone..
  - so, what about performance ?..

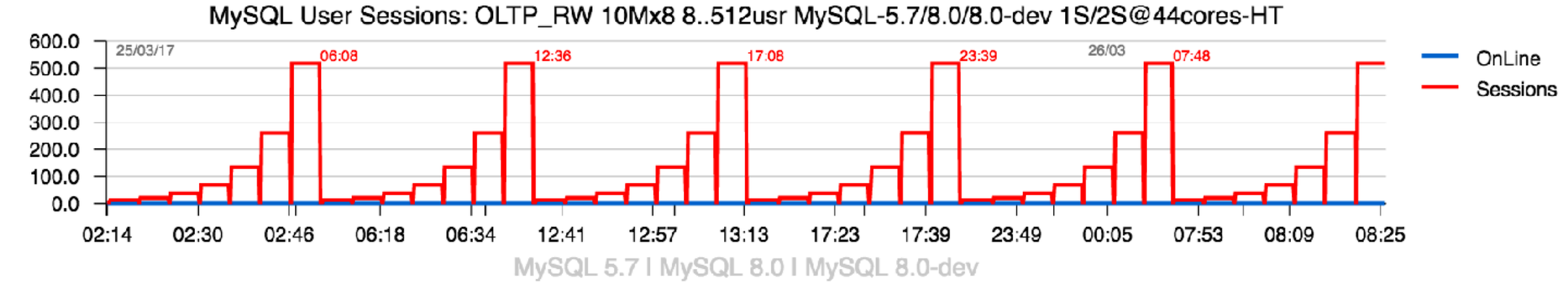




Sysbench OLTP_RW 10Mx8-tables

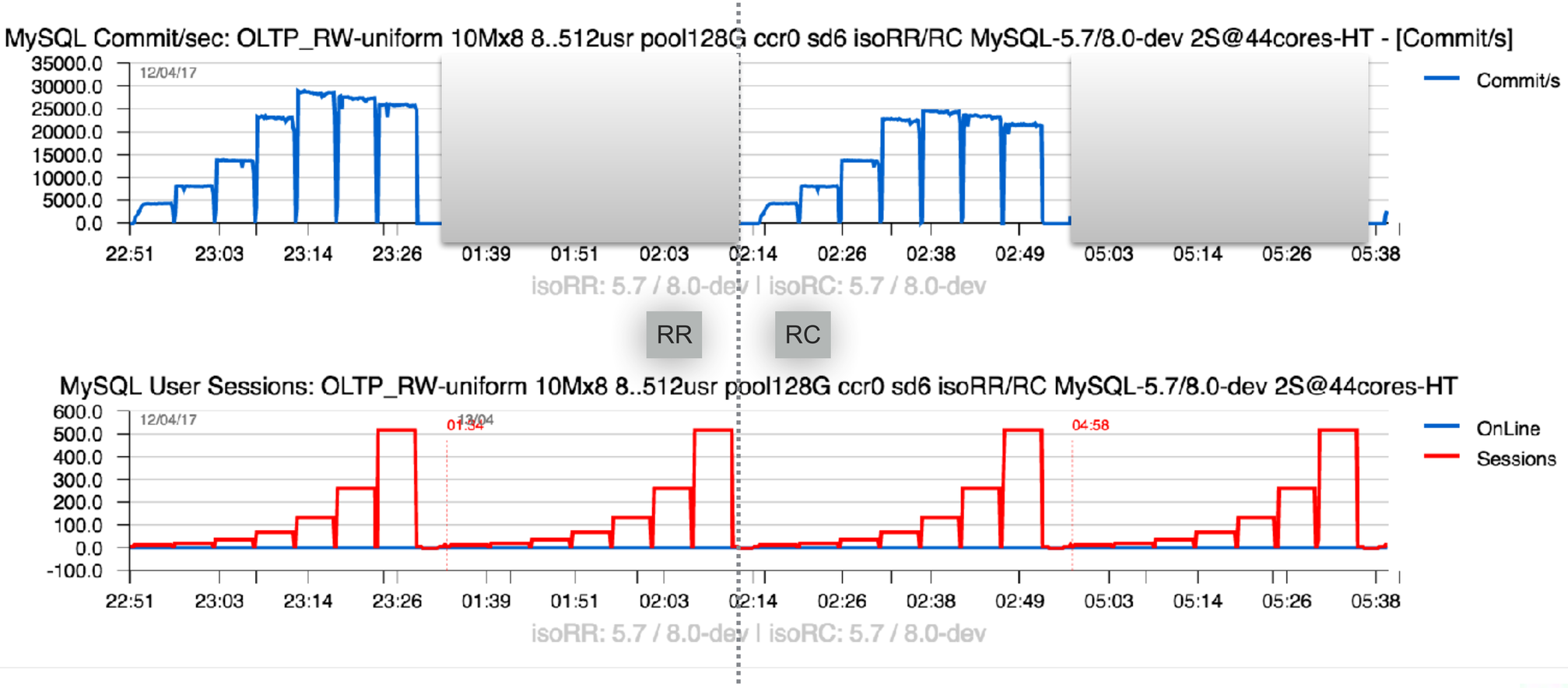
- Observations :
  - 8.0-dev : clear gain, but not more than 10%-15%, why ?..
  - yes, WRITE waits are gone, but mind that READs are dominating in this workload..





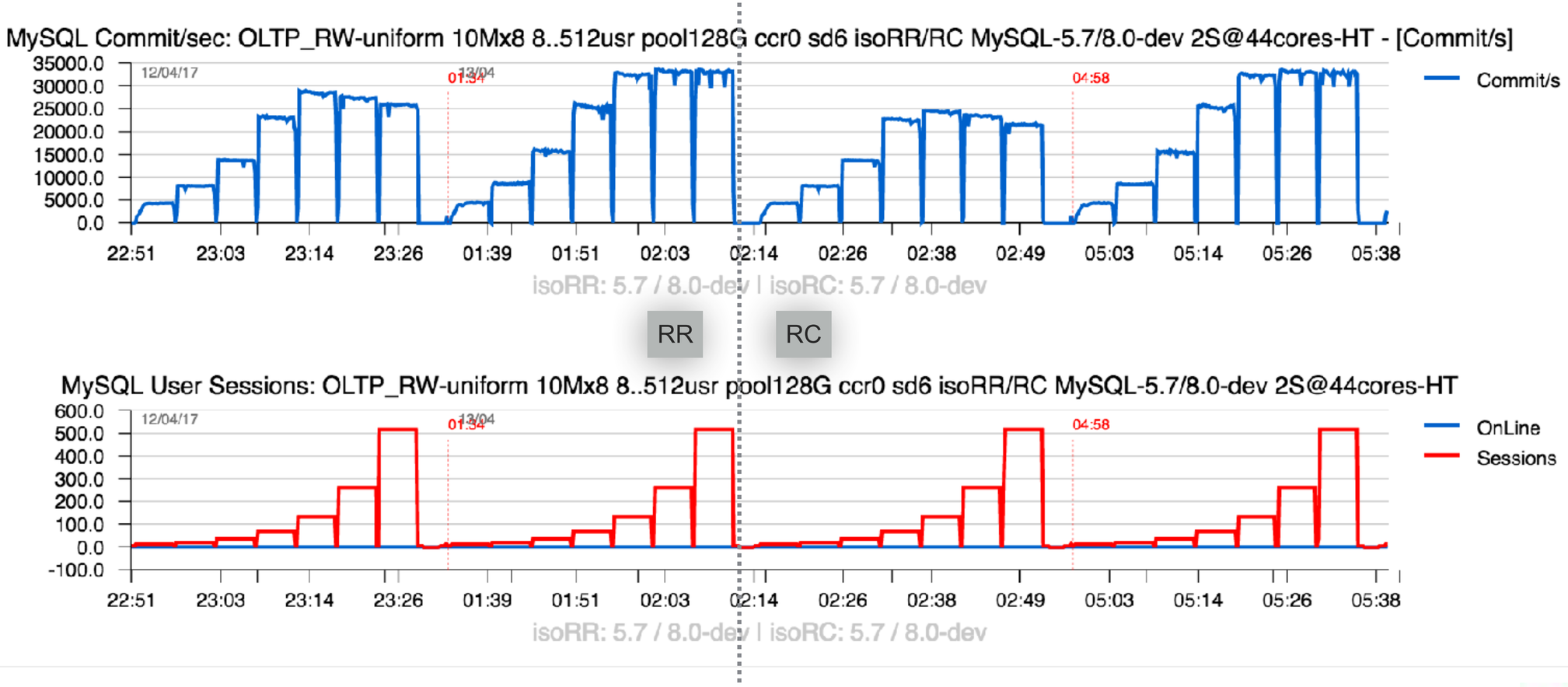
Sysbench OLTP_RW 10Mx8-tables - RR/RC Isolation

- Observations :
  - 5.7 : drop on RR => RC isolation.. (known issue from a very long time, trx_sys cost)..
  - 8.0-dev : … ?



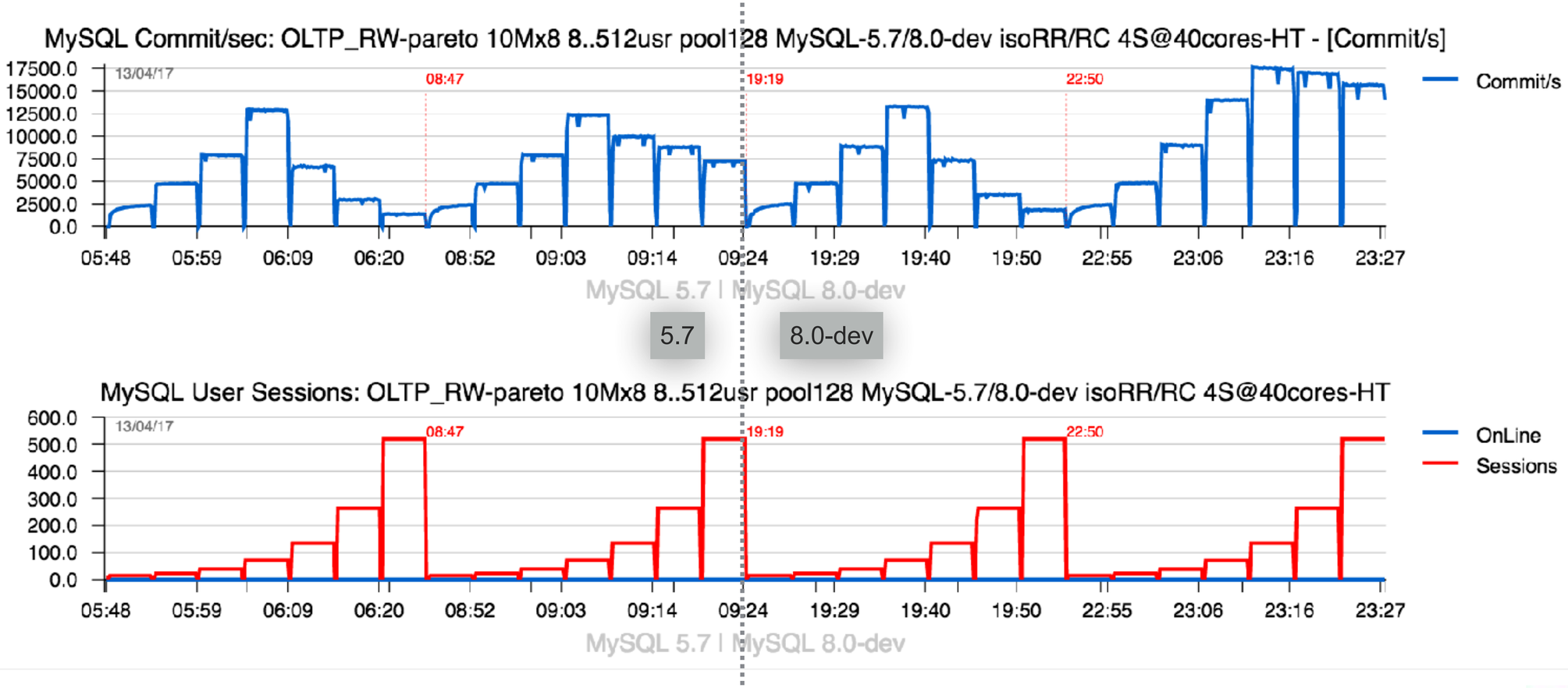
Sysbench OLTP_RW 10Mx8-tables - RR/RC Isolation

- Observations :
  - 5.7 : drop on RR => RC isolation.. (known issue from a very long time, trx_sys cost)..
  - 8.0-dev : same TPS on RR & RC isolation (!!)



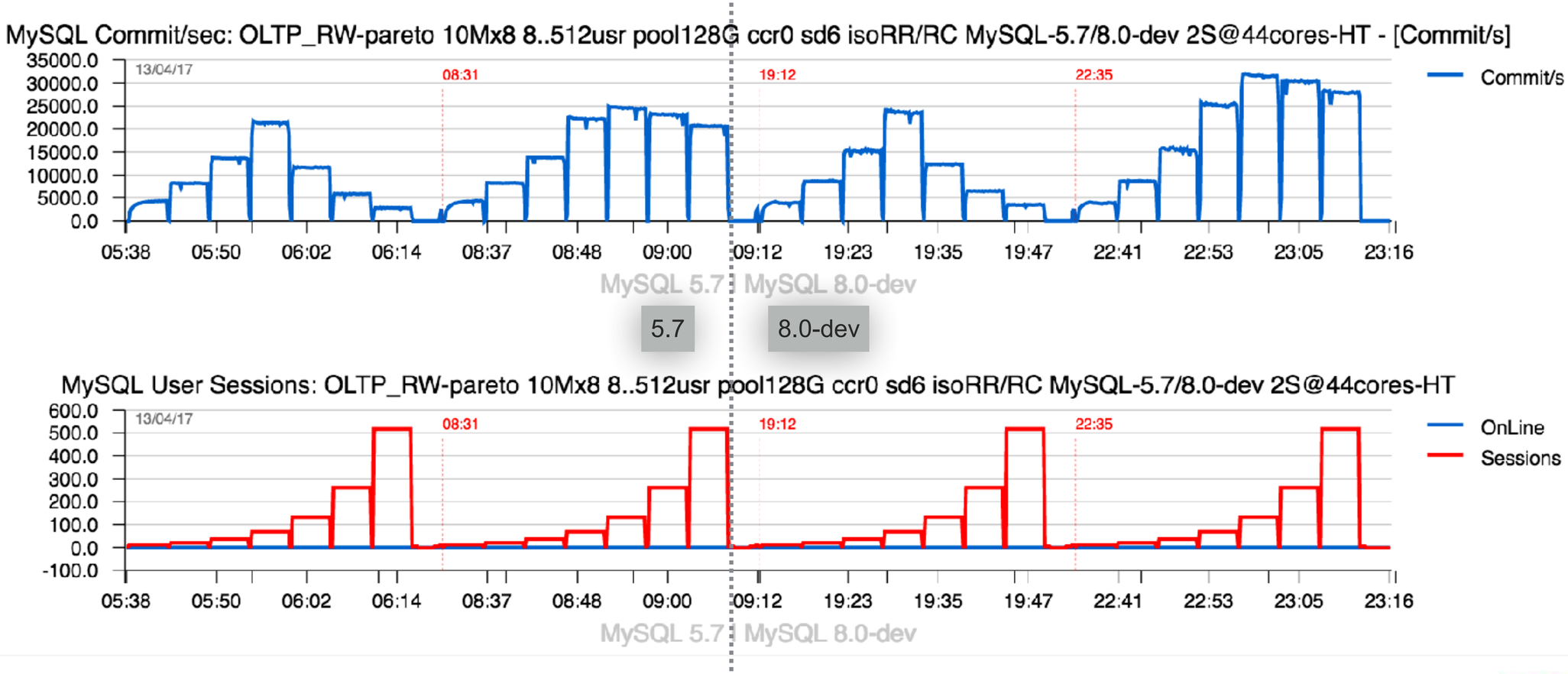
Sysbench OLTP_RW 10Mx8-tables - “pareto” 4S@40cores

- Observations :
  - RR : both Engines loosing TPS on high load due concurrent row access (“pareto")
  - RC : 8.0-dev is doing way better !



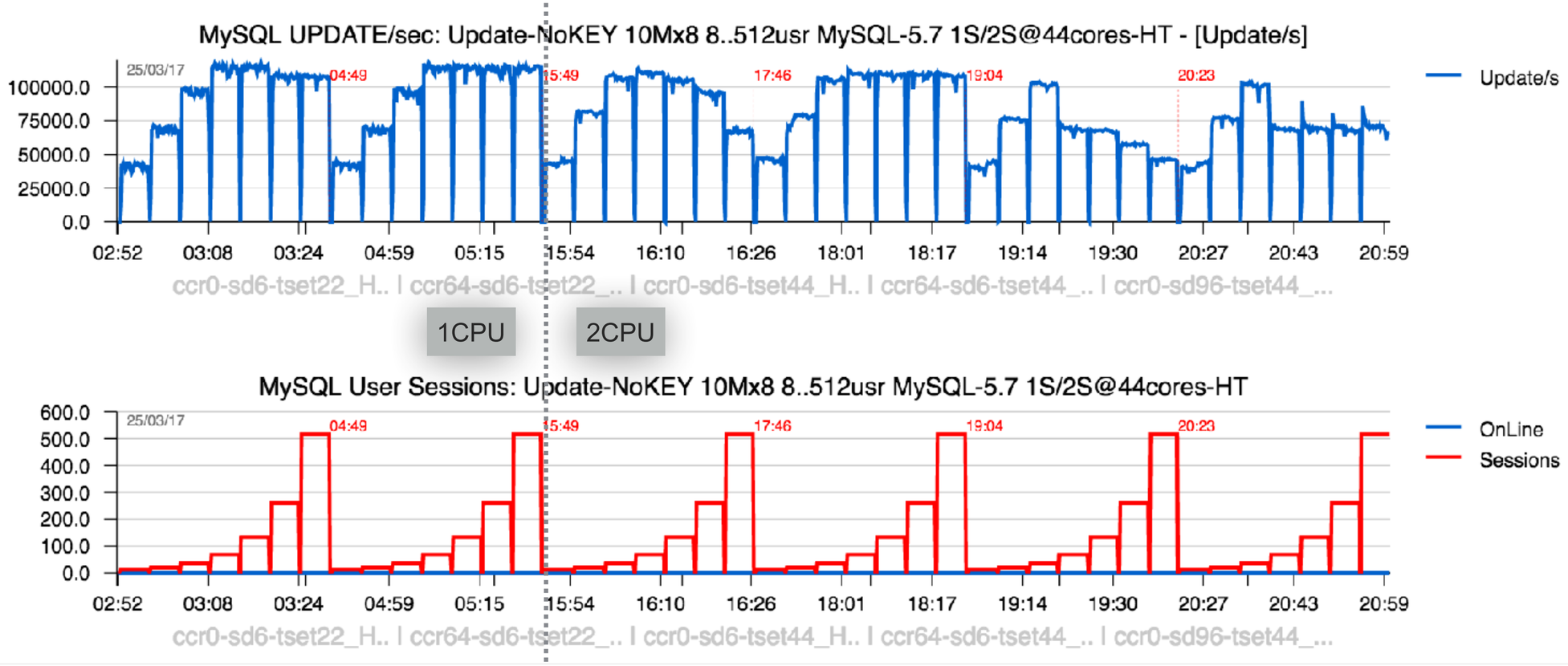
Sysbench OLTP_RW 10Mx8-tables - “pareto” 2S@44cores

- Observations :
  - RR : both Engines loosing TPS on high load due concurrent row access (“pareto")
  - RC : 8.0-dev is doing way better !



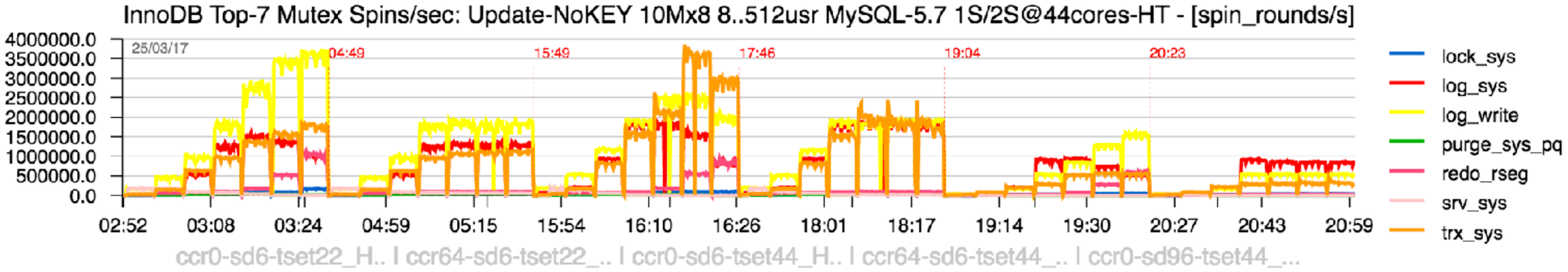
Sysbench Update-NoKEY 10Mx8-tables @MySQL 5.7

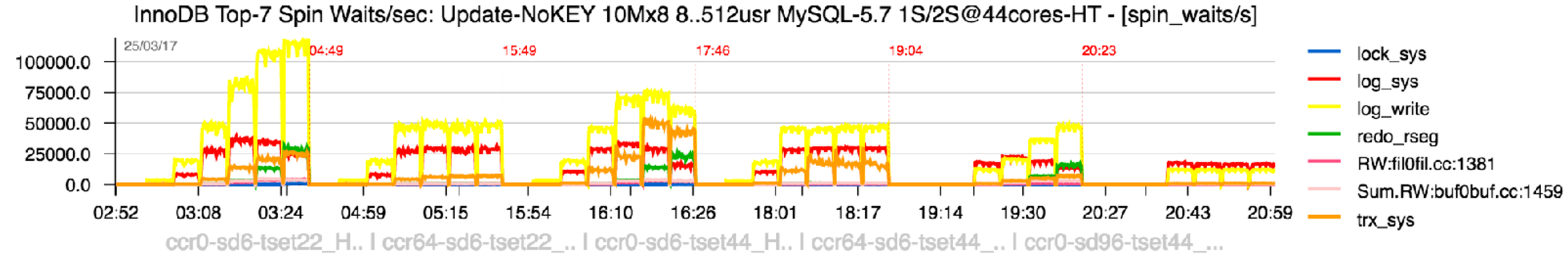
- Observations :
  - MySQL 5.7 : no any gain from 1 CPU socket => 2 CPU sockets.. - why ?..
  - (tuning concurrency=64 helps to avoid drops on high load)



Sysbench Update-NoKEY 10Mx8-tables @MySQL 5.7

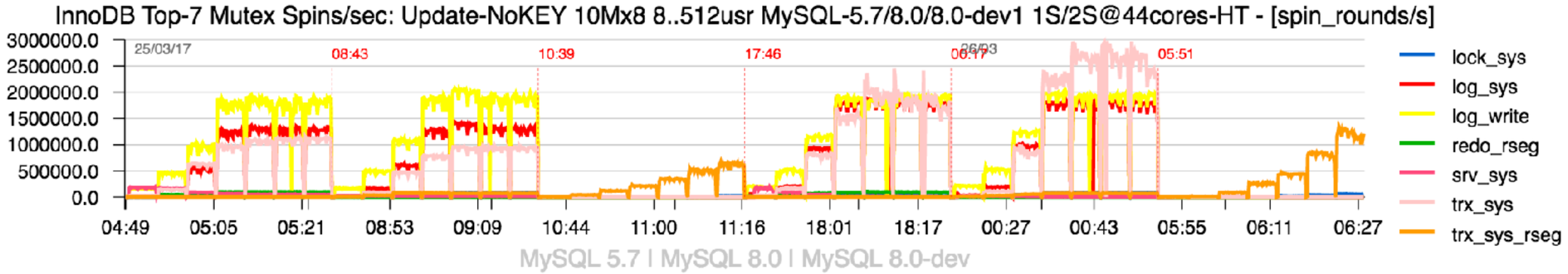
- Observations :
  - MySQL 5.7 : 1 CPU socket => 2 CPU sockets increasing **trx_sys** waits..
  - delaying waits is not a fix (and not helping anymore)..

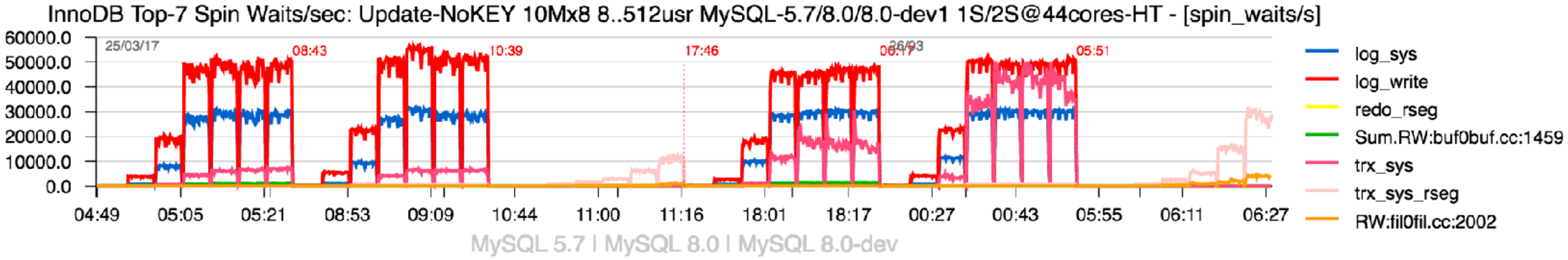




Sysbench Update-NoKEY 10Mx8-tables

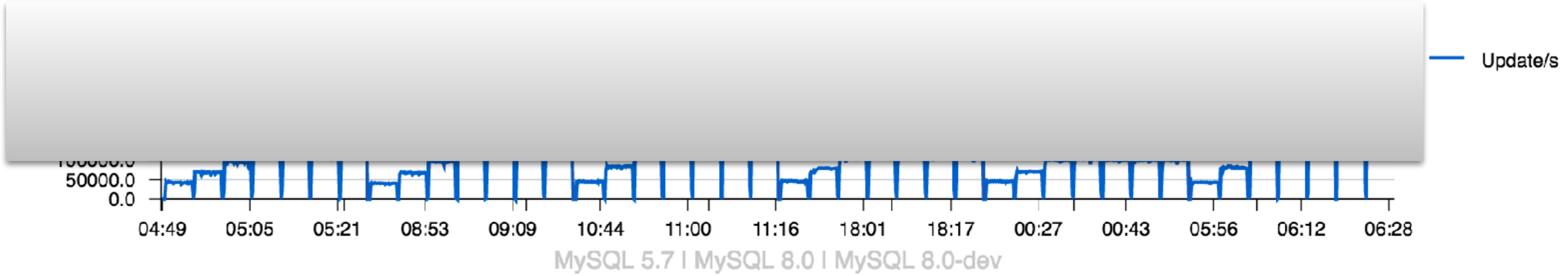
- Observations :
  - all hot contentions are gone on 8.0-dev..
  - what about results ?.. ;-))

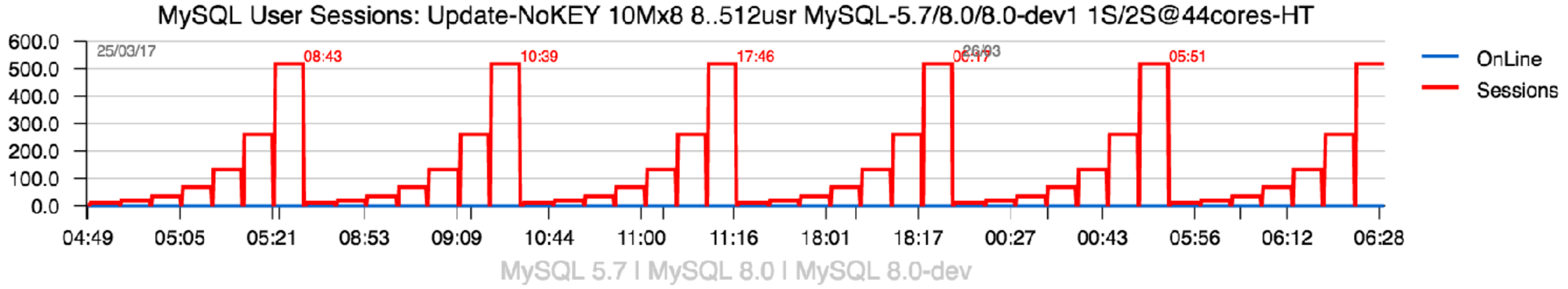




Sysbench Update-NoKEY 10Mx8-tables

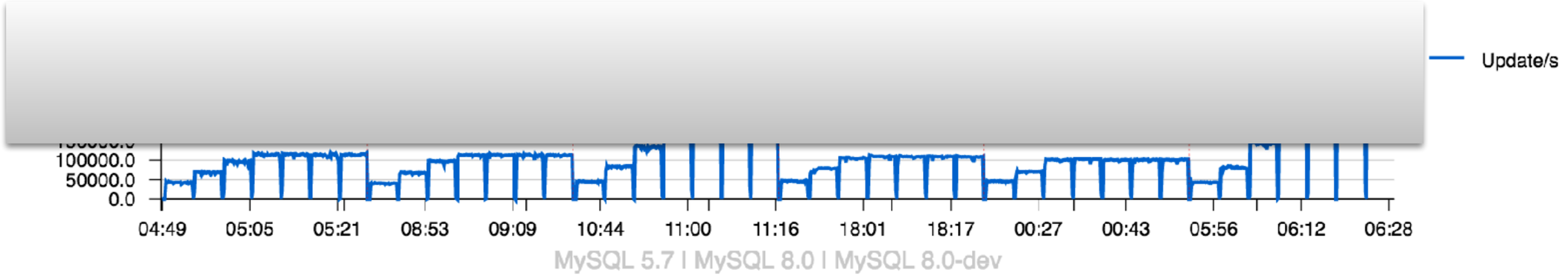
- Observations :
  - all hot contentions are gone on 8.0-dev..
  - what about results ?.. ;-))

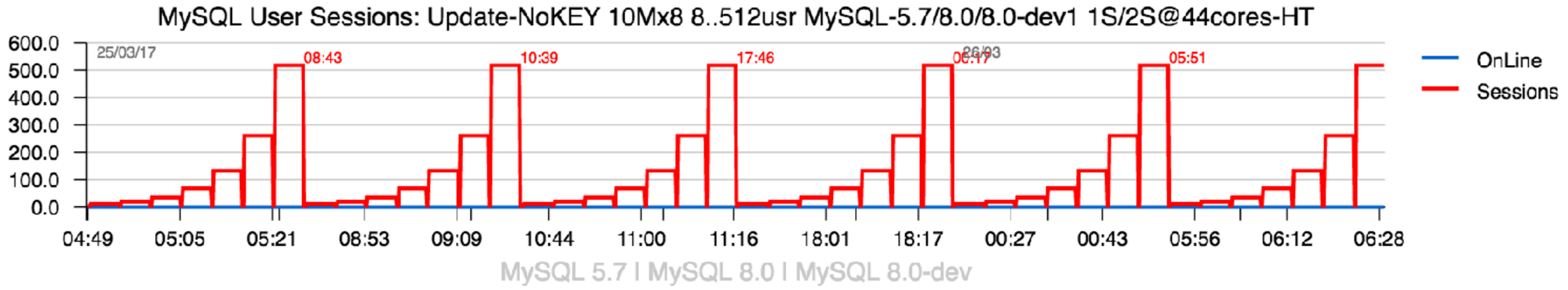




Sysbench Update-NoKEY 10Mx8-tables

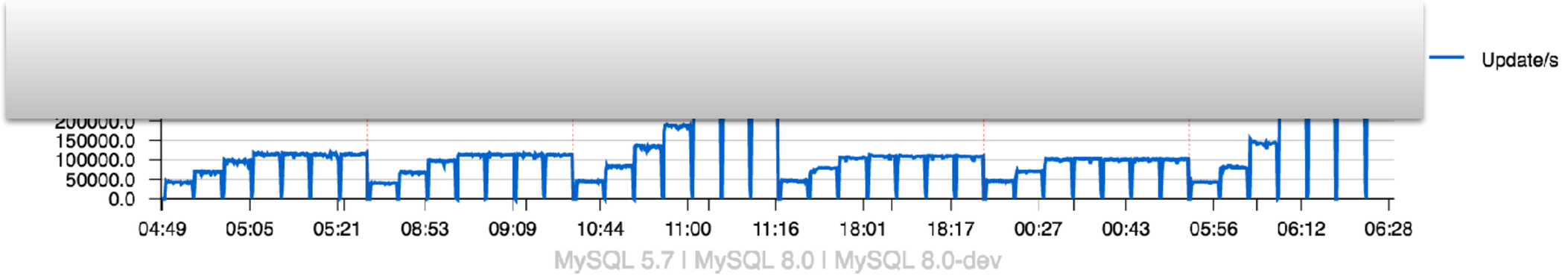
- Observations :
  - all hot contentions are gone on 8.0-dev..
  - what about results ?.. ;-))

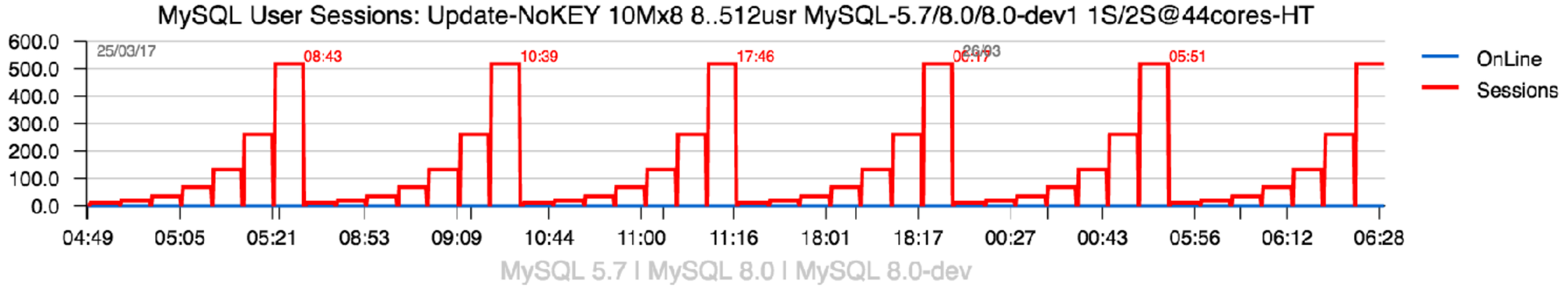




Sysbench Update-NoKEY 10Mx8-tables

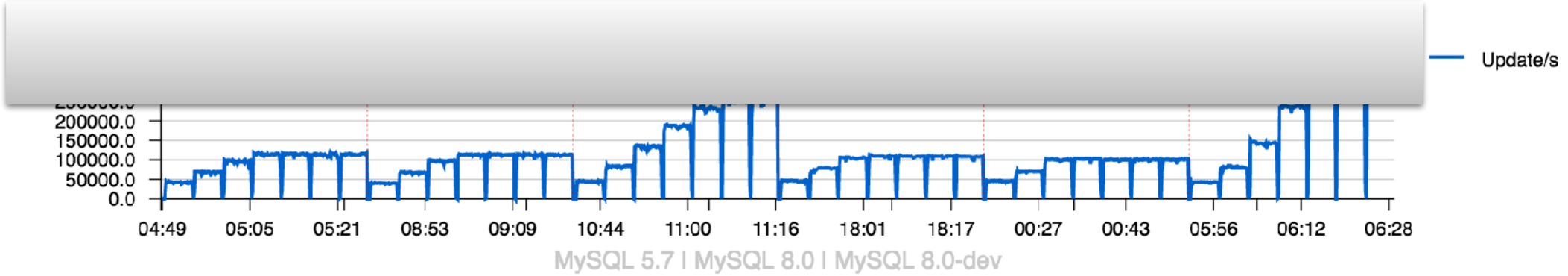
- Observations :
  - all hot contentions are gone on 8.0-dev..
  - what about results ?.. ;-))

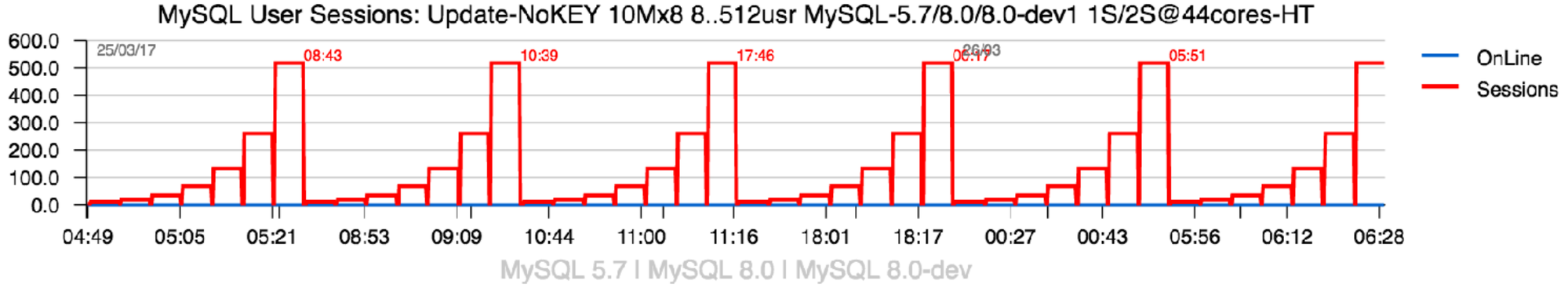




Sysbench Update-NoKEY 10Mx8-tables

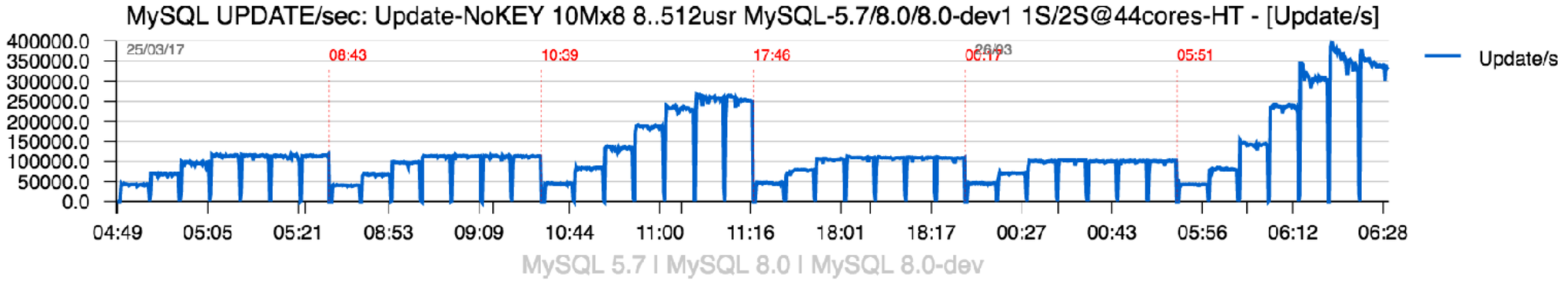
- Observations :
  - all hot contentions are gone on 8.0-dev..
  - what about results ?.. ;-))

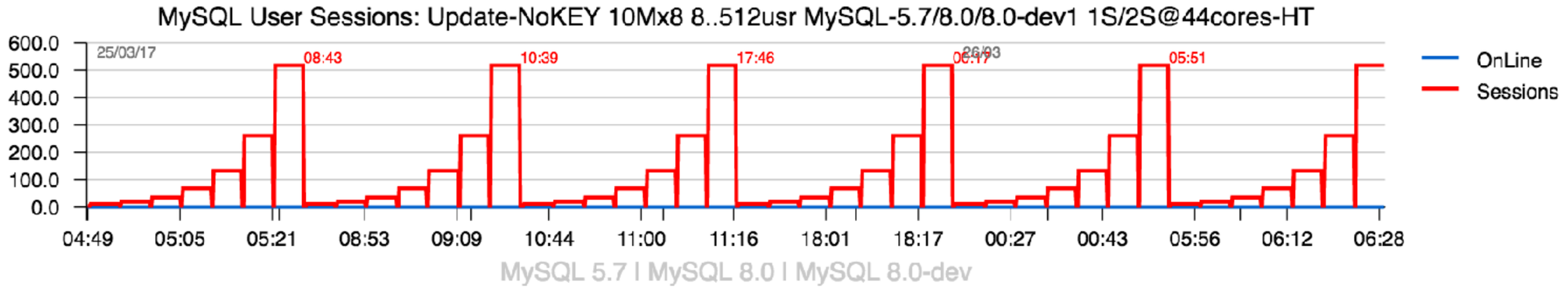




Sysbench Update-NoKEY 10Mx8-tables

- Observations :
  - MySQL 8.0-dev : **x2** times better on 1 CPU socket, **x3** times on 2 CPU !!!
  - NOTE : and even on **8usr** load too !!!



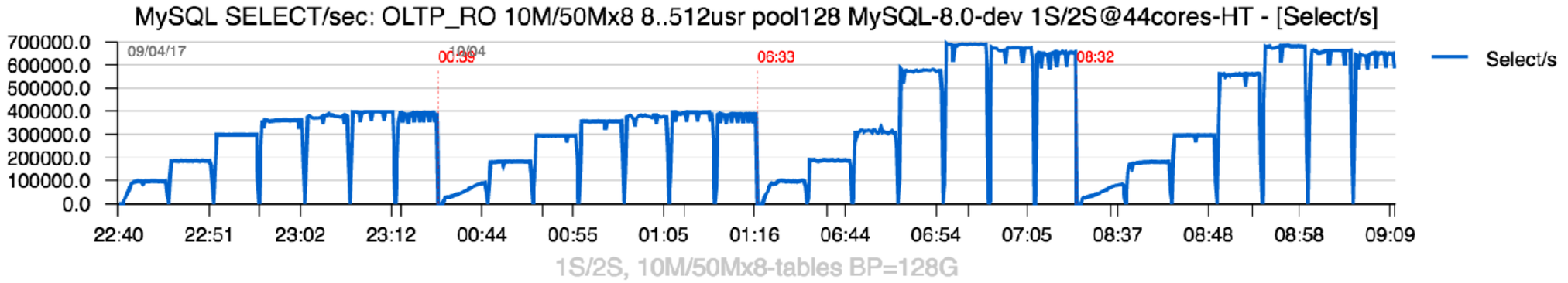


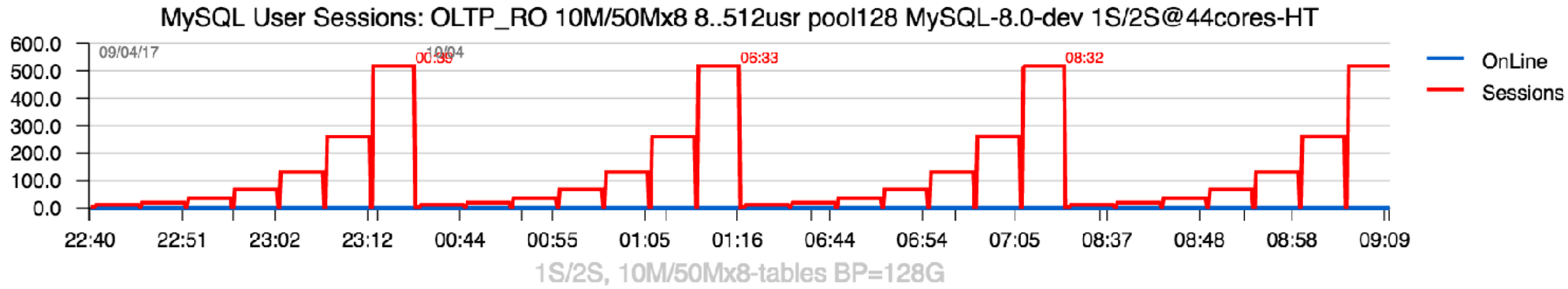
Sysbench Workloads : Data Volume Impact

- Volume :
  - **10Mx8 vs 50Mx8**
- BP :
  - 128GB (all in-memory)
- CPU :
  - 1S (1 CPU socket (22cores-HT))
  - 2S (2 CPU sockets (44cores-HT))
- Access Pattern :
  - UNIFORM (default) vs PARETO
- Workloads :
  - OLTP_RO
  - OLTP_RW
  - Update-NoKEY

Sysbench OLTP_RO : 10Mx8 vs 50Mx8 (BP=128G)

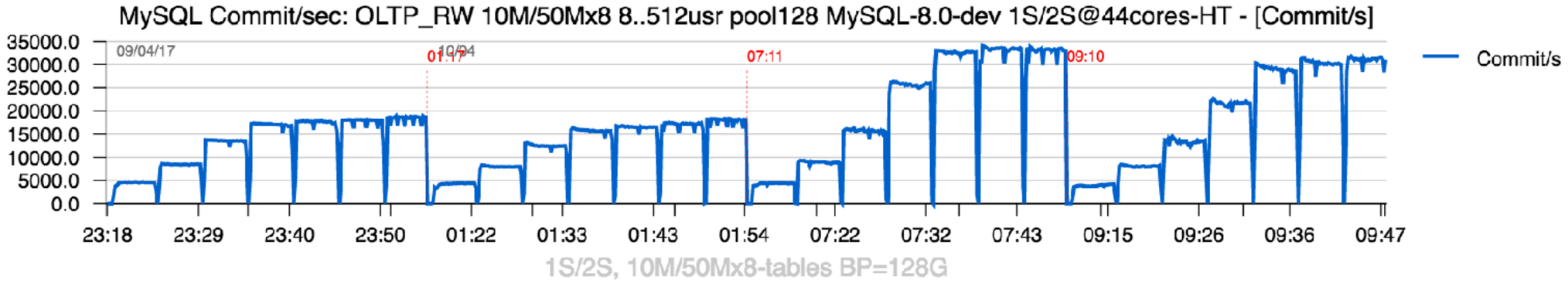
- Observations :
  - Same QPS on 10Mx8 and 50Mx8 data volumes

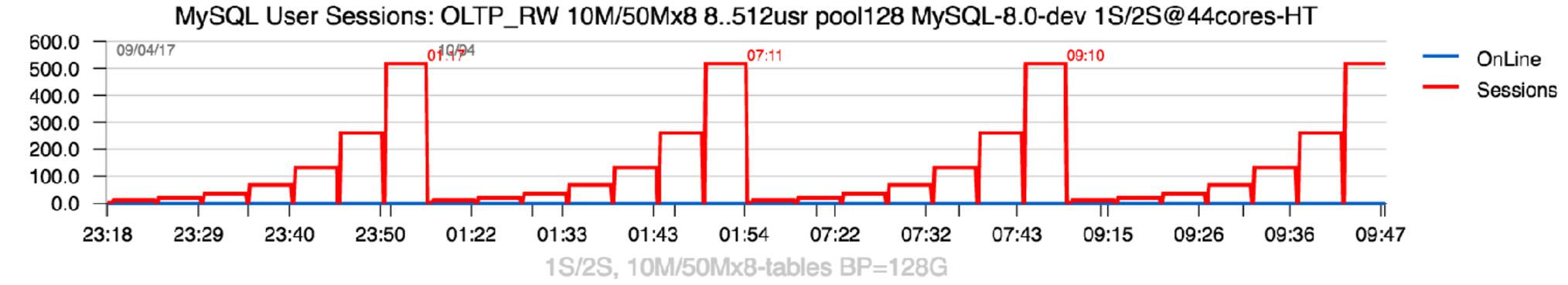




Sysbench OLTP_RW : 10Mx8 vs 50Mx8 (BP=128G)

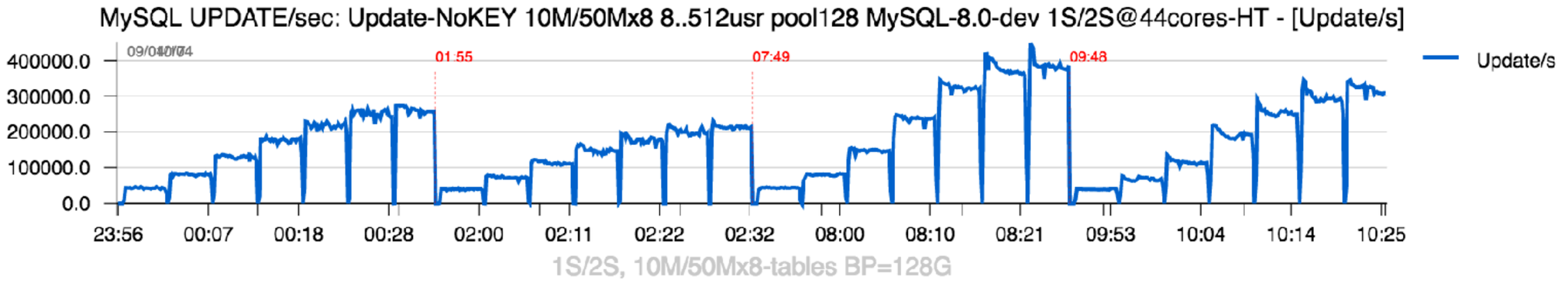
- Observations :
  - similar TPS on 1S
  - but on 2S with 50Mx8 volume TPS level is decreasing..

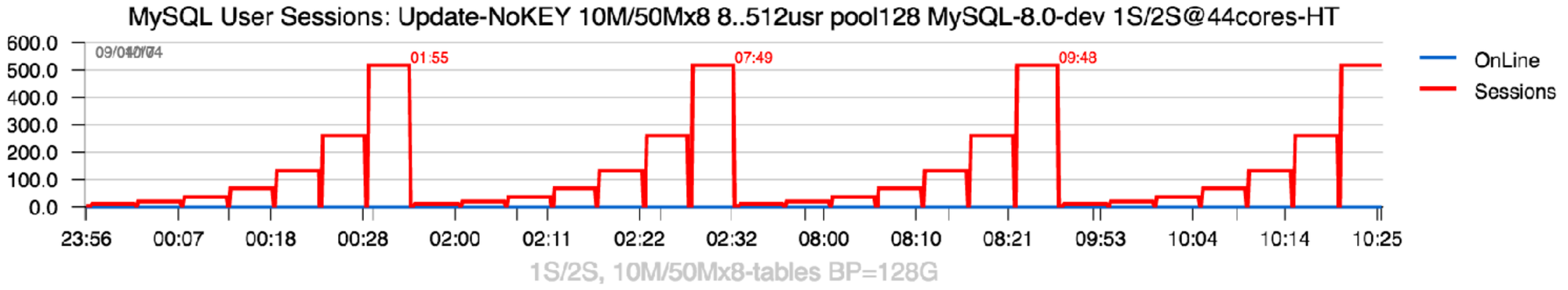




Sysbench Update-NoKEY : 10Mx8 vs 50Mx8 (BP=128G)

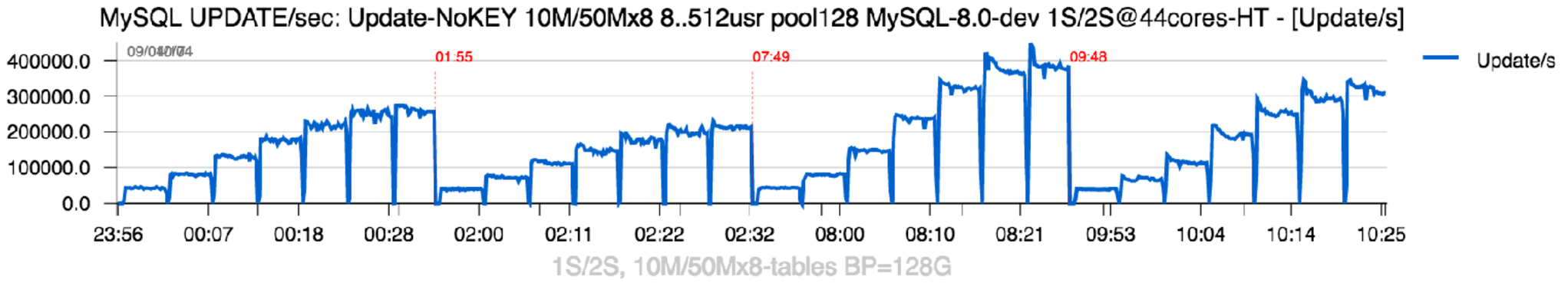
- Observations :
  - all data in-memory, but TPS rate is going lower on 50Mx8 data volume on both 1S & 2S..
  - why ?..

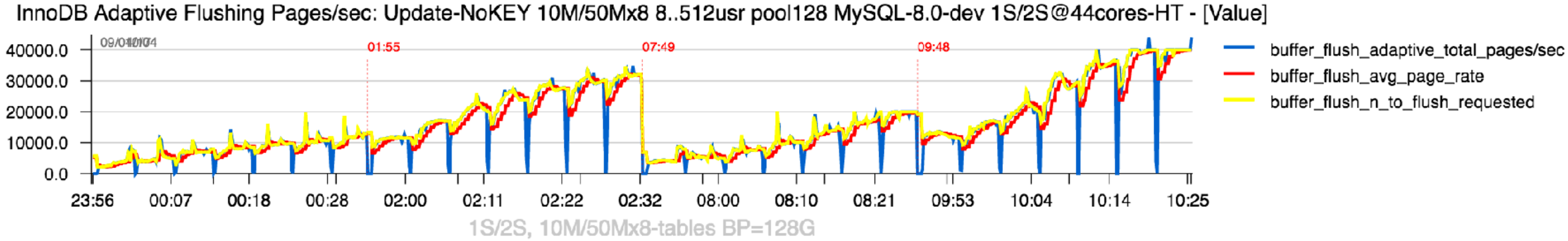




Sysbench Update-NoKEY : 10Mx8 vs 50Mx8 (BP=128G)

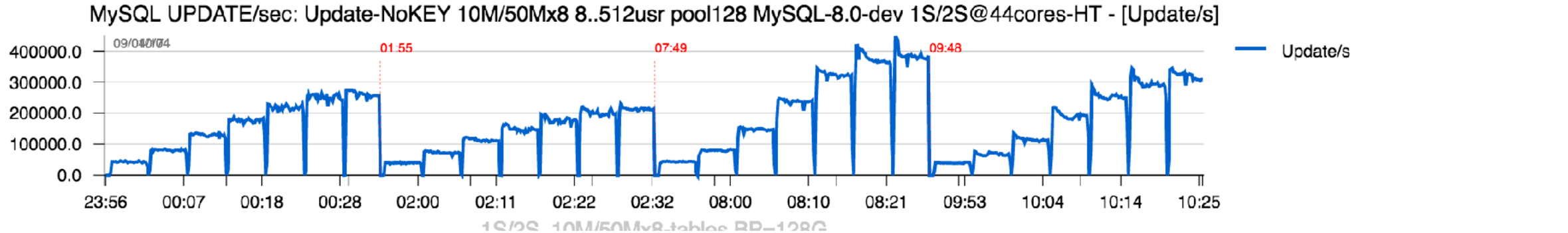
- Observations :
  - all data in-memory, but TPS rate is going lower on 50Mx8 data volume on both 1S & 2S..
  - why ?.. => IO impact..

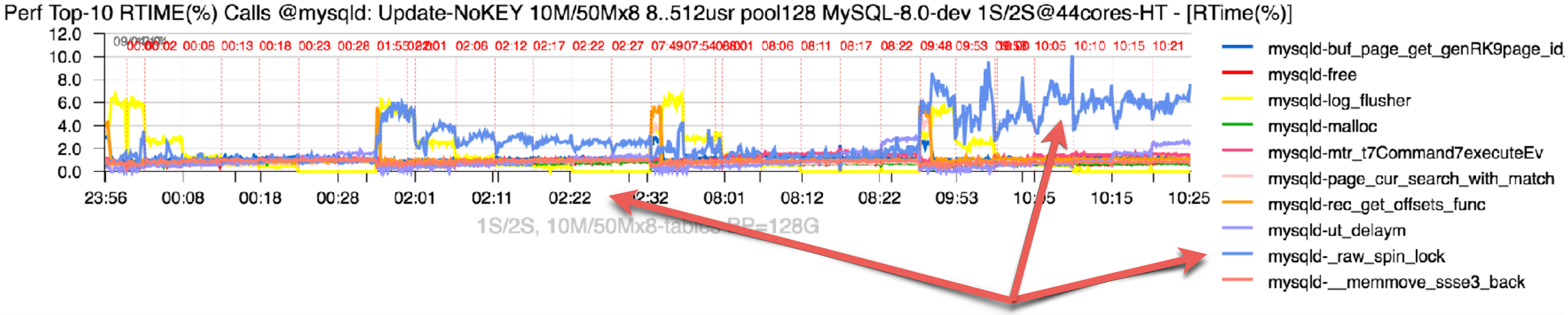




Sysbench Update-NoKEY : 10Mx8 vs 50Mx8 (BP=128G)

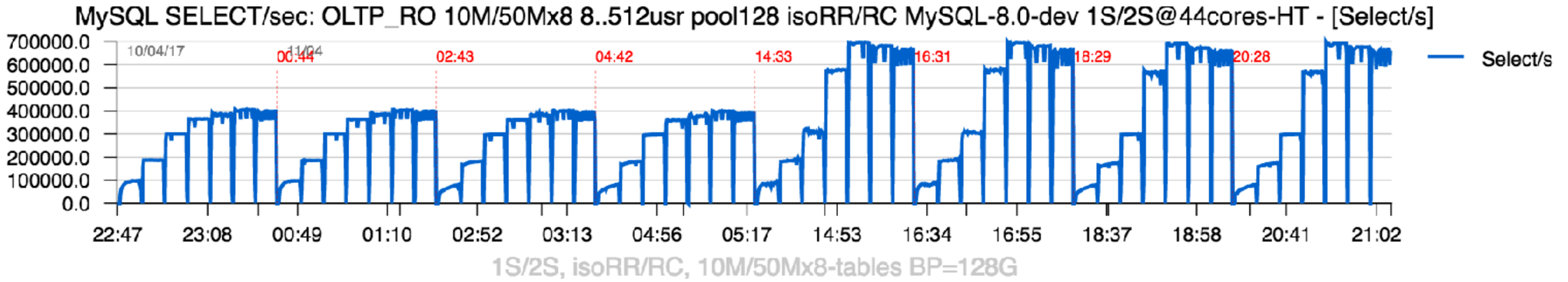
- Observations :
  - all data in-memory, but TPS rate is going lower on 50Mx8 data volume on both 1S & 2S..
  - why impact from background IO writes ???.. => AIO also needs CPU..

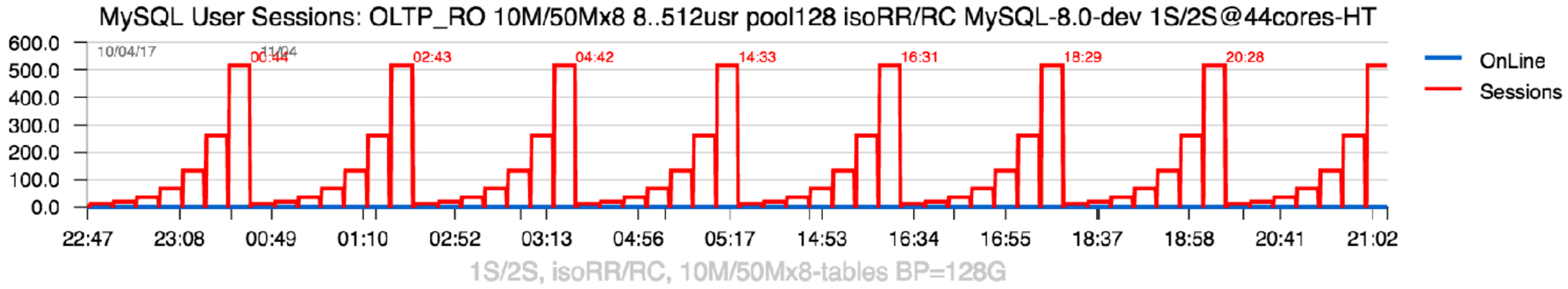




Sysbench OLTP_RO-pareto : 10Mx8 vs 50Mx8 (BP=128G)

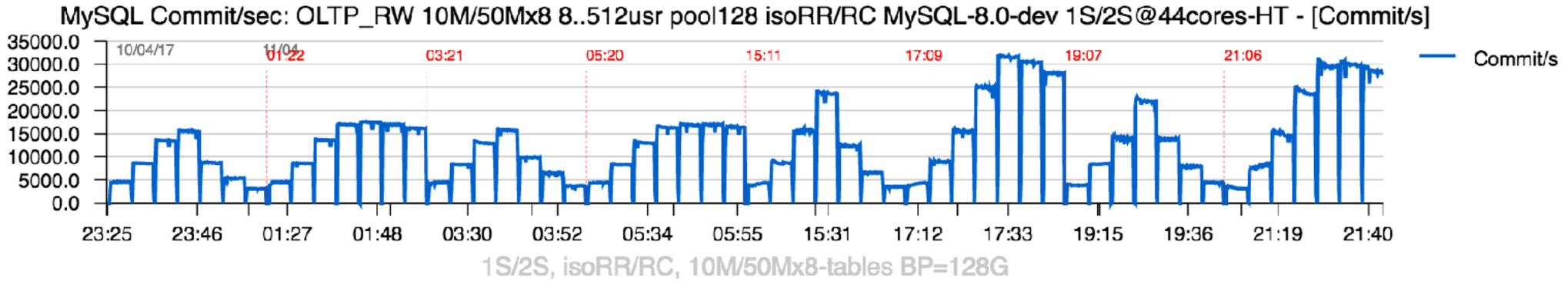
- Observations :
  - Same QPS on 10Mx8 and 50Mx8 data volumes
  - no difference between RR / RC isolation..

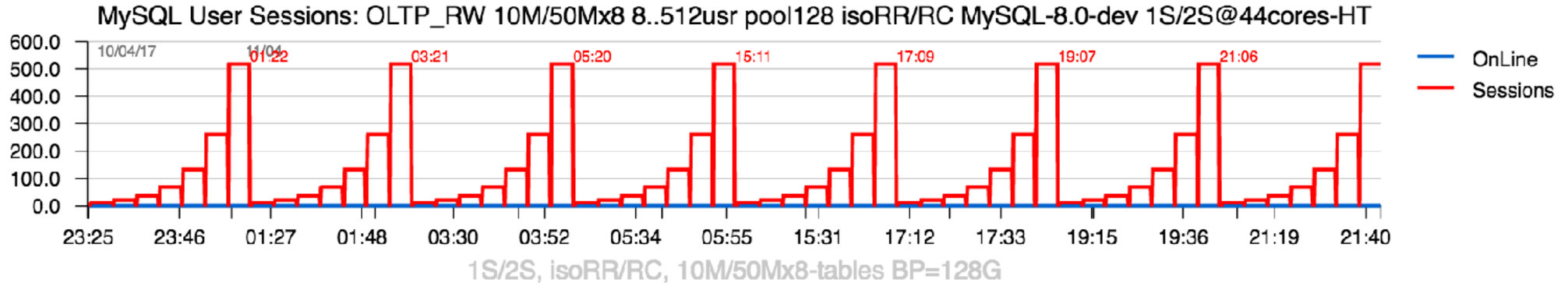




Sysbench OLTP_RW-pareto : 10Mx8 vs 50Mx8 (BP=128G)

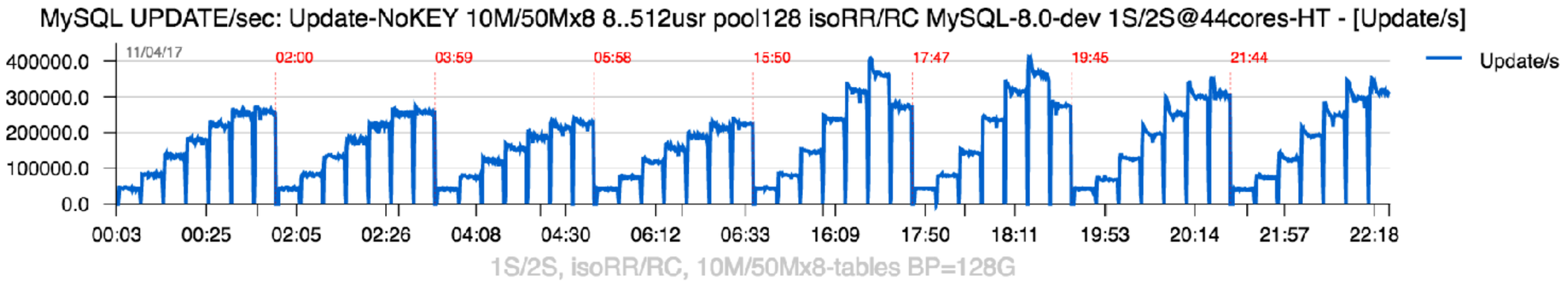
- Observations :
  - indeed, using RC makes a huge difference
  - TPS on 50Mx8 volume is slightly lower than on 10Mx8..

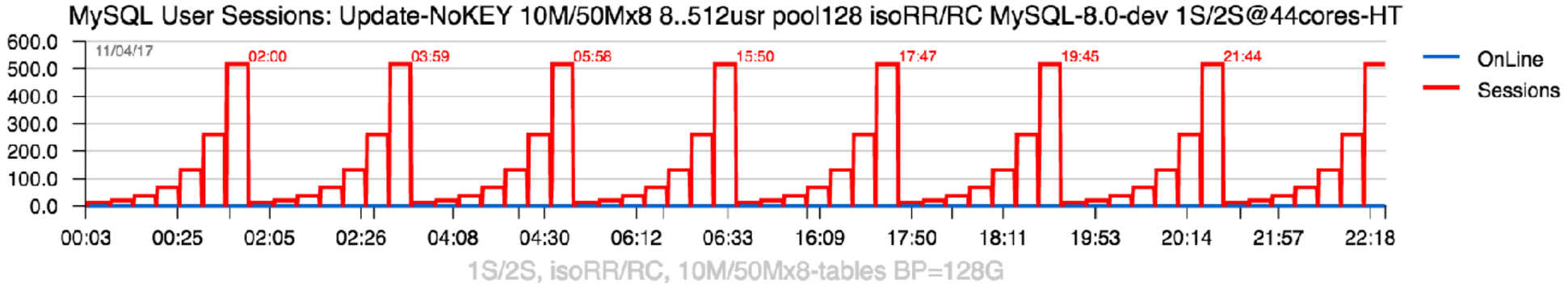




Sysbench Update-NoKEY-pareto : 10Mx8 vs 50Mx8 (BP=128G)

- Observations :
  - similar to UNIFORM, no difference between RR / RC
  - however, on 10Mx8 volume 512usr load hitting row locking contention..



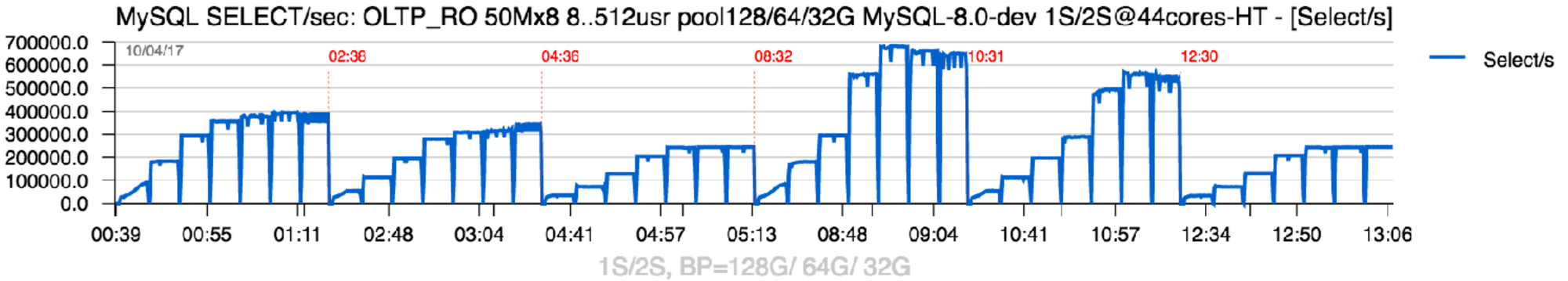


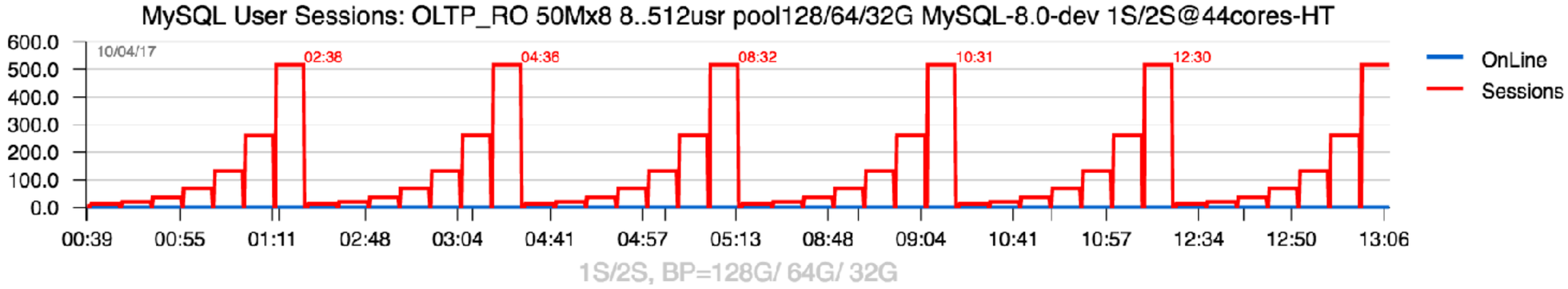
Sysbench Workloads : I/O Impact

- Volume :
  - 50Mx8
- BP :
  - **128GB** (all in-memory)/ **64GB** (1/2 in-memory)/ **32GB** (1/4 in-memory)
- CPU :
  - 1S (1 CPU socket (22cores-HT))
  - 2S (2 CPU sockets (44cores-HT))
- Access Pattern :
  - UNIFORM (default) vs PARETO
- Workloads :
  - OLTP_RO
  - OLTP_RW
  - Update-NoKEY

Sysbench OLTP_RO 50Mx8 : BP=128G/ 64G/ 32G

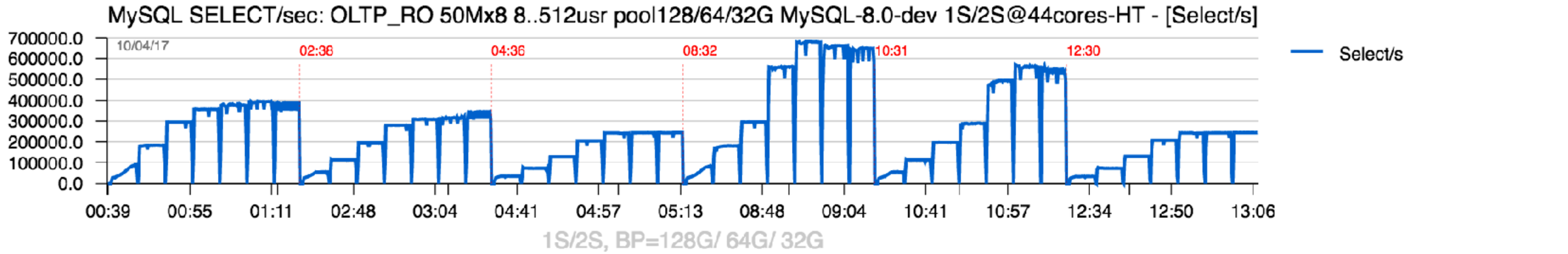
- Observations :
  - Clear I/O impact.. => with 32GB BP size we're testing Storage, and no more MySQL ;-)
  - IO : **3500** MB/sec (!!!) - the max possible for a given NVMe..

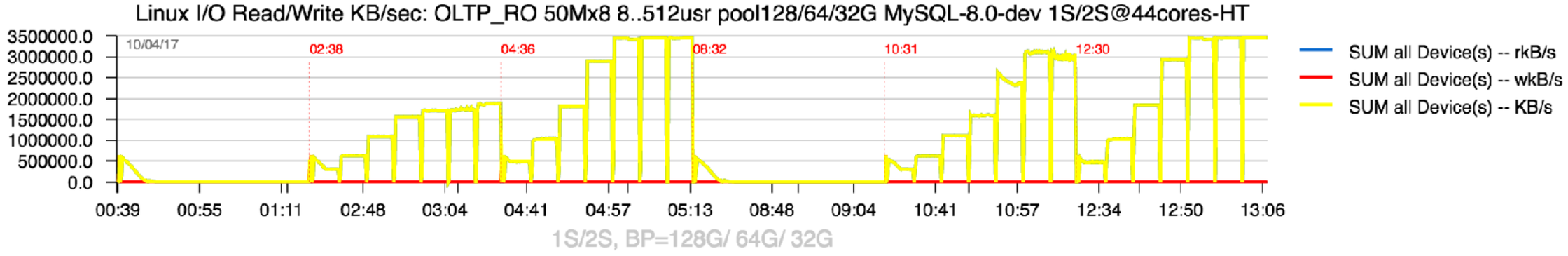




Sysbench OLTP_RO 50Mx8 : BP=128G/ 64G/ 32G

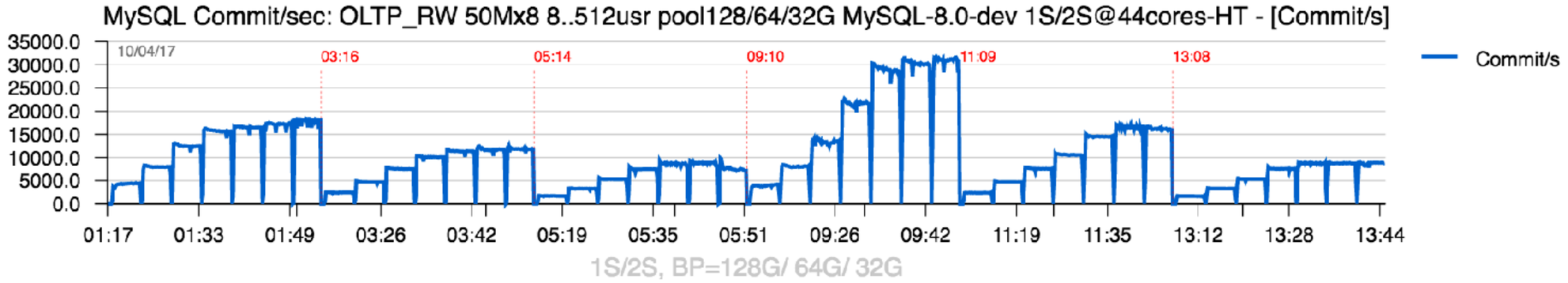
- Observations :
  - Clear I/O impact.. => with 32GB BP size we're testing Storage, and no more MySQL ;-)
  - IO : **3500** MB/sec (!!!) - the max possible for a given NVMe..

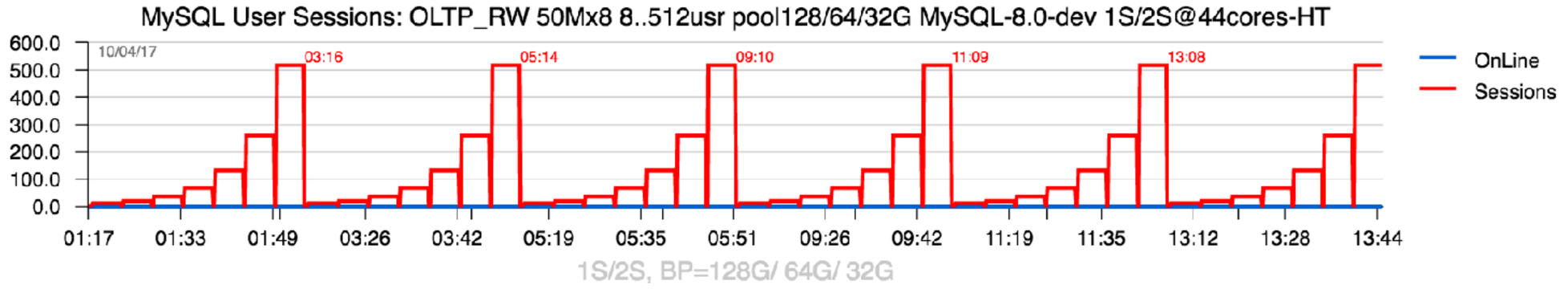




Sysbench OLTP_RW 50Mx8 : BP=128G/ 64G/ 32G

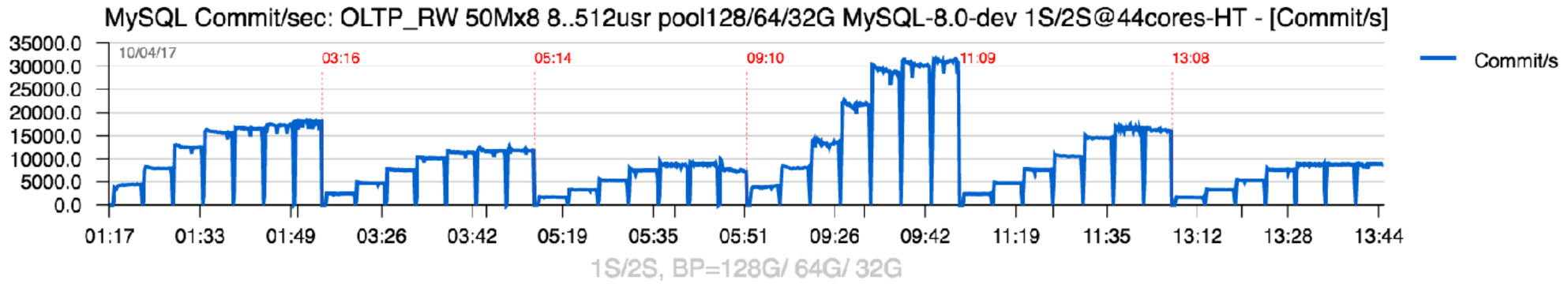
- Observations :
  - Clear I/O impact.. => significant TPS drop from 128G => 64G => 32G BP..
  - IO : **3000** MB/sec - the max possible “mixed” for a given NVMe ?..

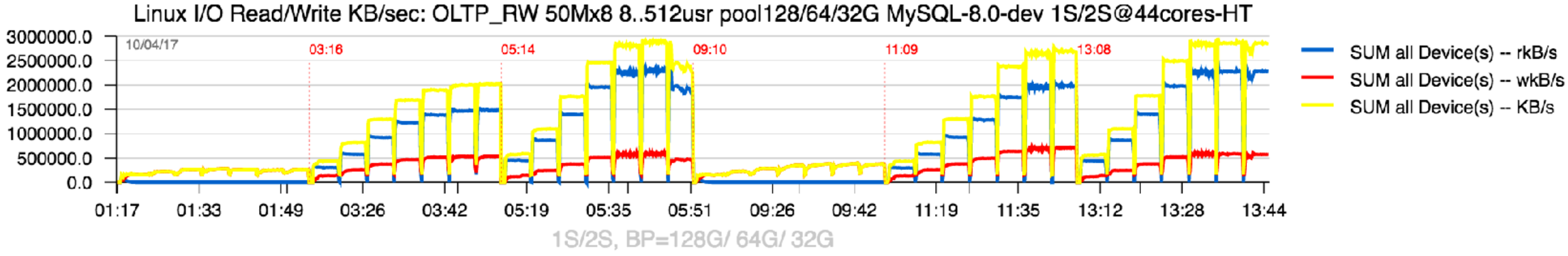




Sysbench OLTP_RW 50Mx8 : BP=128G/ 64G/ 32G

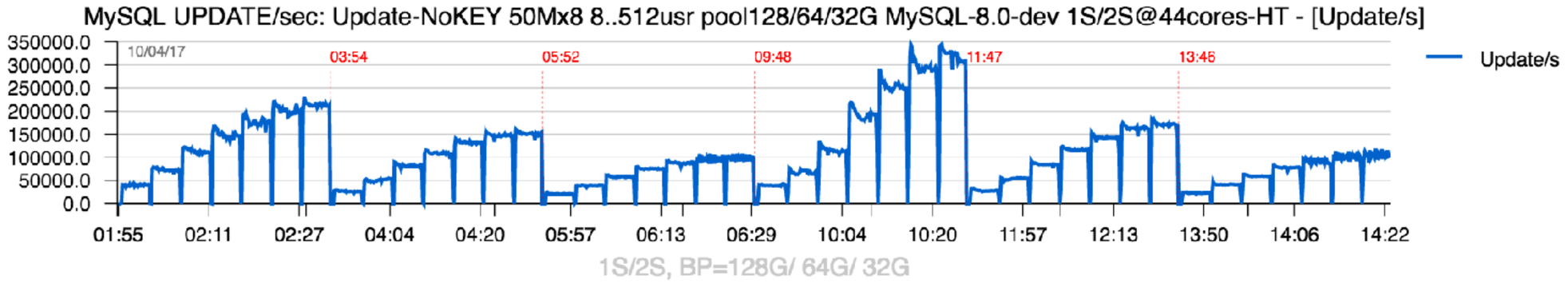
- Observations :
  - Clear I/O impact.. => significant TPS drop from 128G => 64G => 32G BP..
  - IO : **3000** MB/sec - the max possible “mixed” for a given NVMe ?..

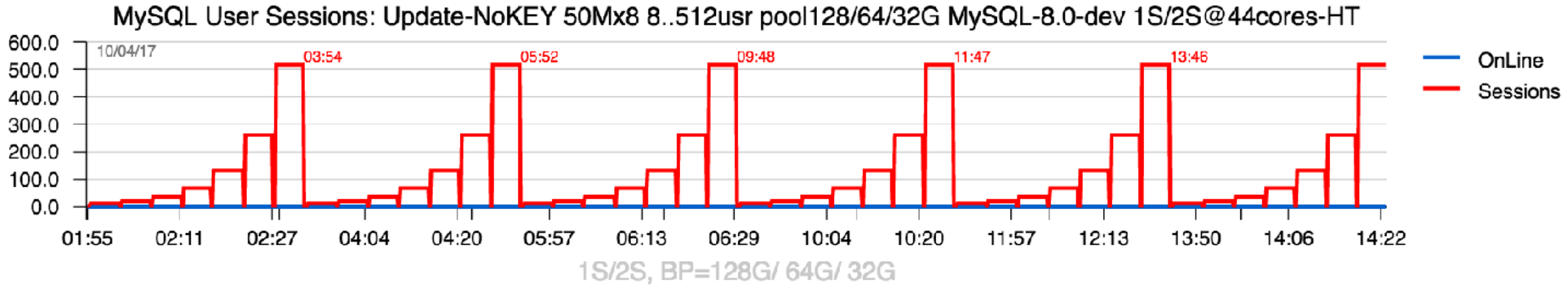




Sysbench Update-NoKEY 50Mx8 : BP=128G/ 64G/ 32G

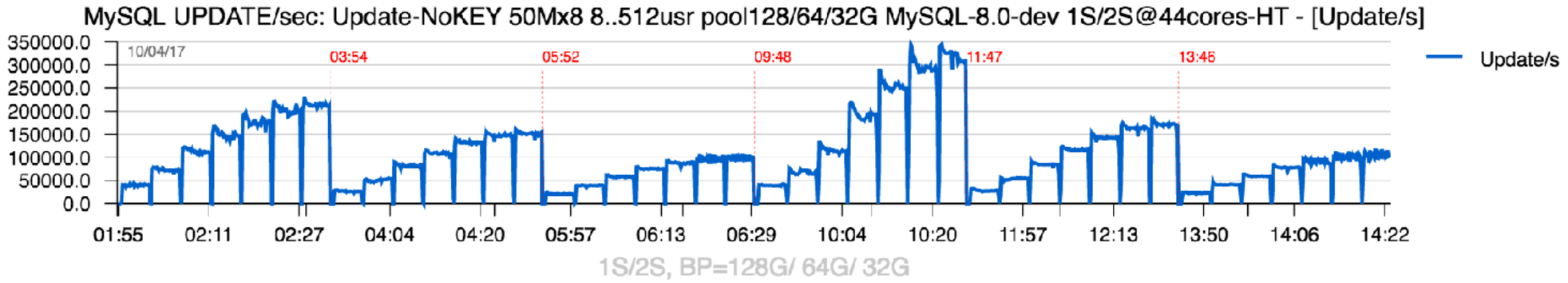
- Observations :
  - Clear I/O impact.. => significant TPS drop from 128G => 64G => 32G BP..
  - 64GB BP : why near no difference between 1S / 2S ?..

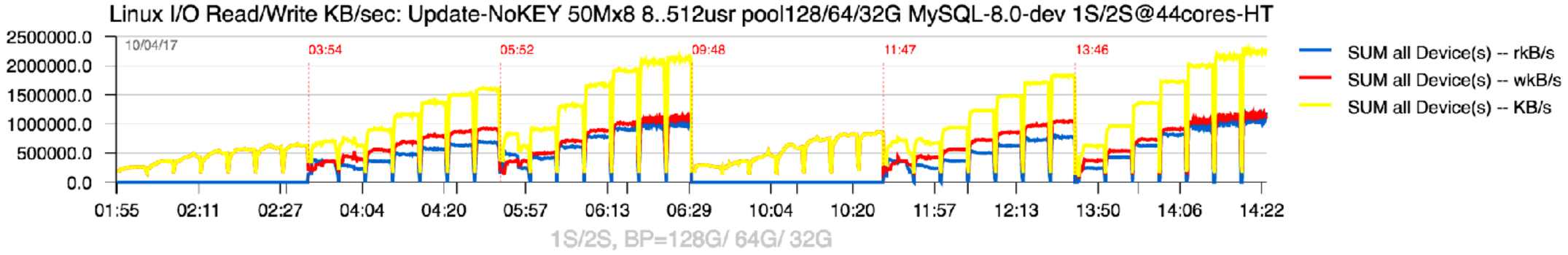




Sysbench Update-NoKEY 50Mx8 : BP=128G/ 64G/ 32G

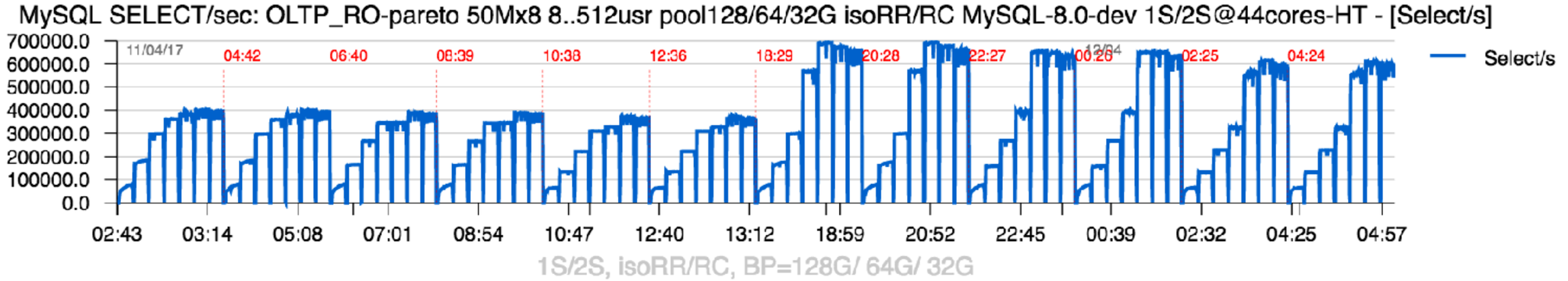
- Observations :
  - Clear I/O impact.. => significant TPS drop from 128G => 64G => 32G BP..
  - IO : **2200** MB/sec - the max possible “50/50 mixed” for a given NVMe ?..

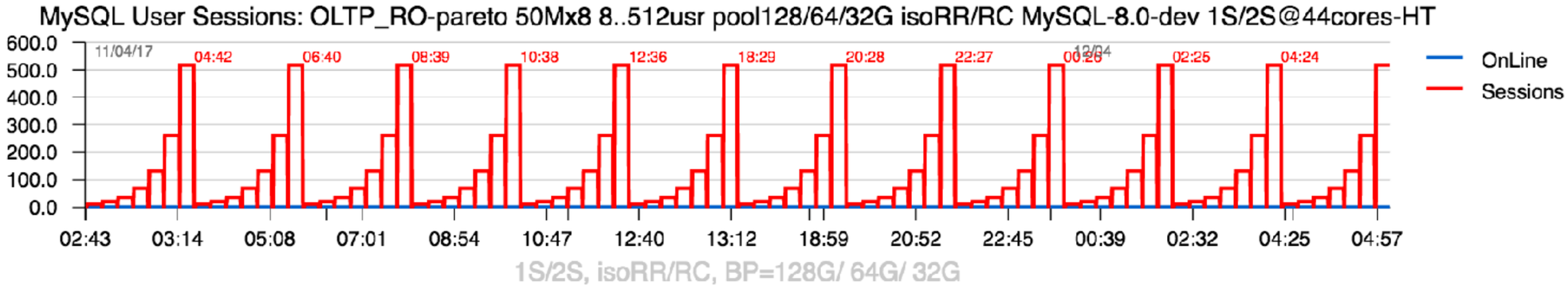




Sysbench OLTP_RO-pareto 50Mx8 : BP=128G/ 64G/ 32G

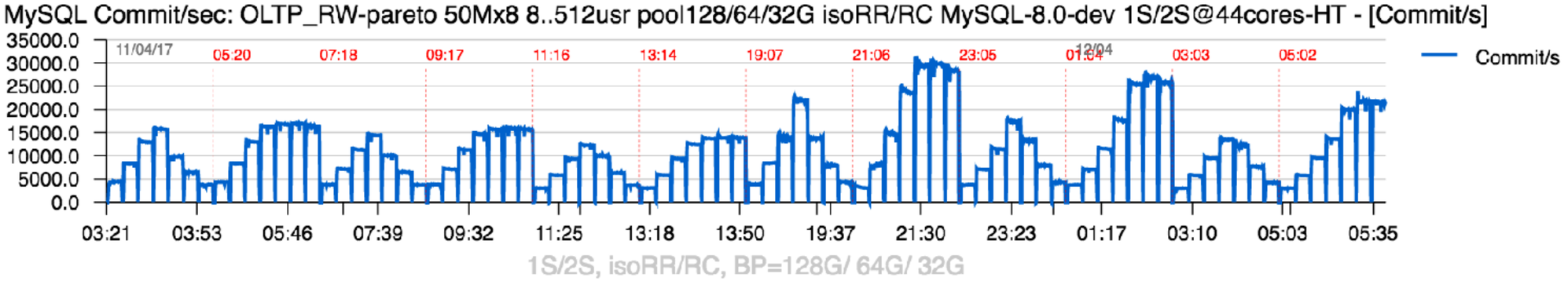
- Observations :
  - Very limited IO impact..
  - no difference between RR / RC isolation

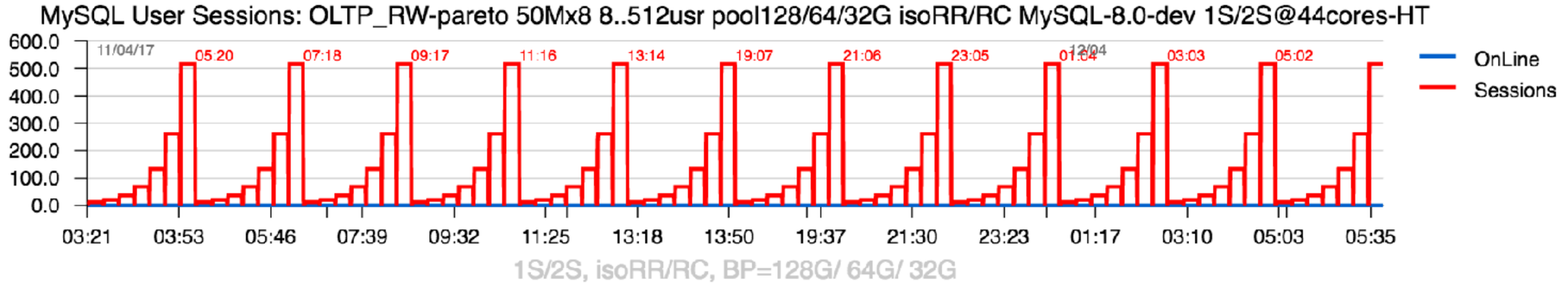




Sysbench OLTP_RW-pareto 50Mx8 : BP=128G/ 64G/ 32G

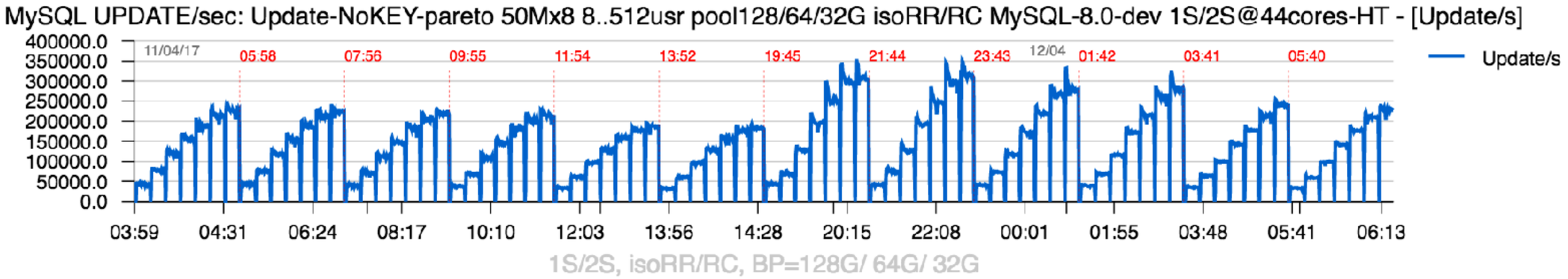
- Observations :
  - IO impact is present, but more than x2 times lower vs UNIFORM..
  - RC isolation is the must ;-)

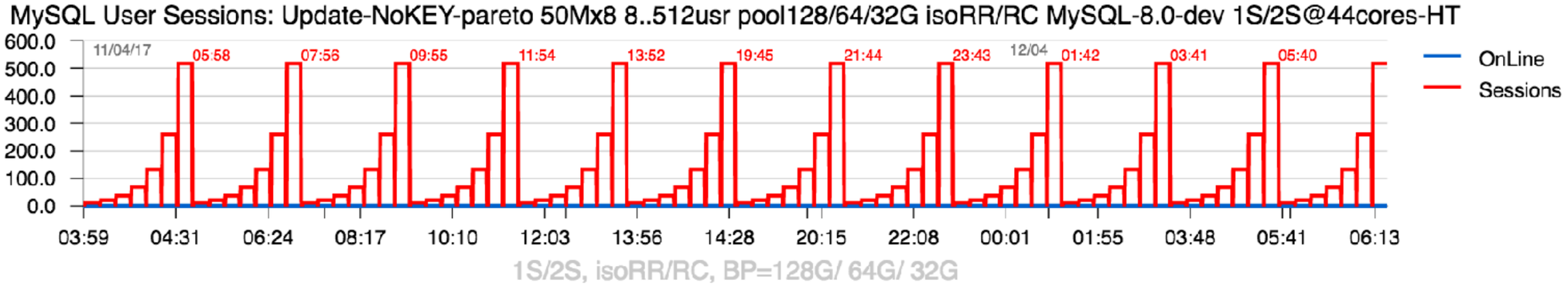




Sysbench Update-NoKEY-pareto 50Mx8 : BP=128G/ 64G/ 32G

- Observations :
  - IO impact is present, but more than x2 times lower vs UNIFORM..
  - why ?.. => significantly less IO reads with PARETO



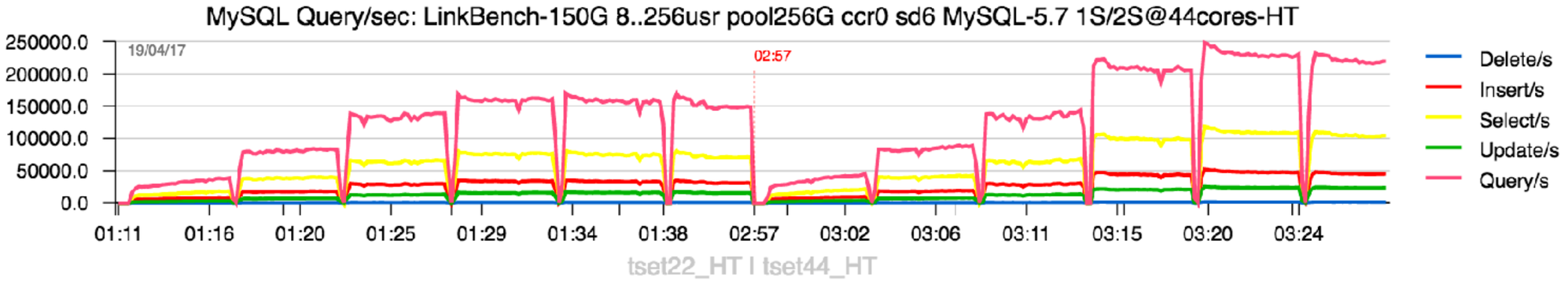


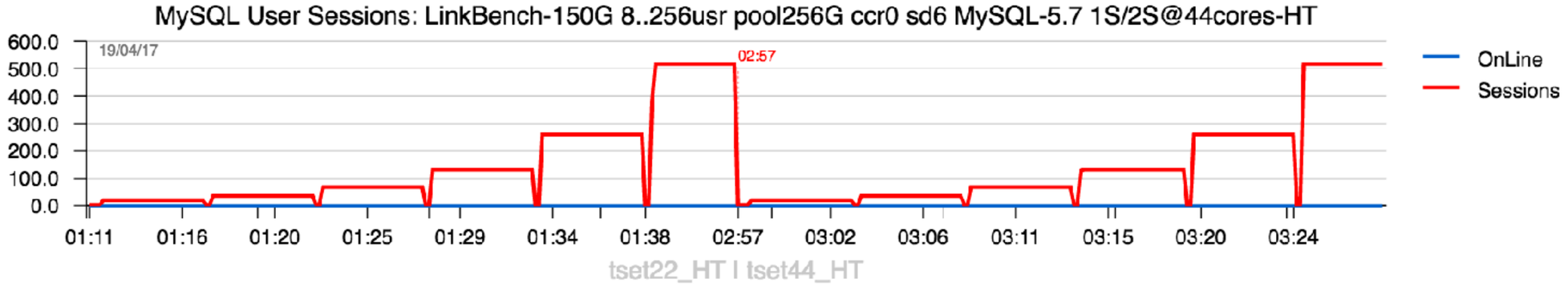
LinkBench

- Volume :
  - 150M (150GB)
- Engine :
  - LinkBench is not working anymore with 8.0
  - so, testing with MySQL 5.7
- Load :
  - 8, 16, .. 512 concurrent users

LinkBench-150G : BP=256GB (in-memory)

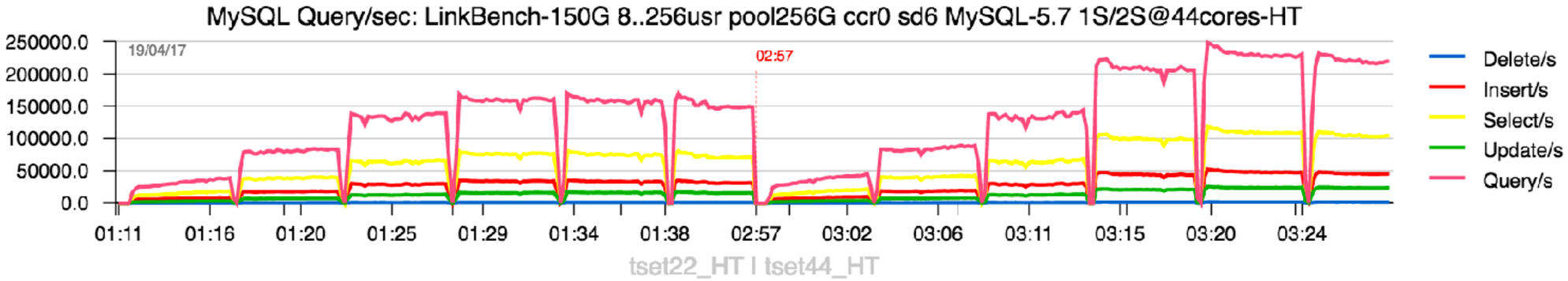
- Observations :
  - 1S => 2S : 50% performance improvement
  - what are the bottlenecks ?..

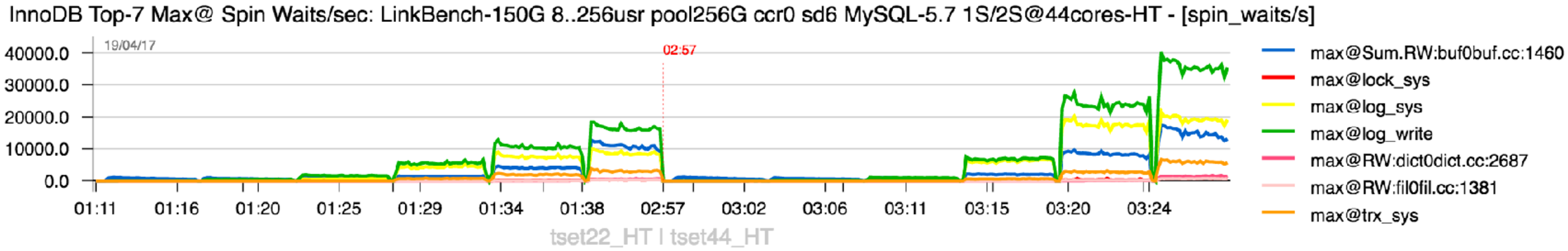




LinkBench-150G : BP=256GB (in-memory)

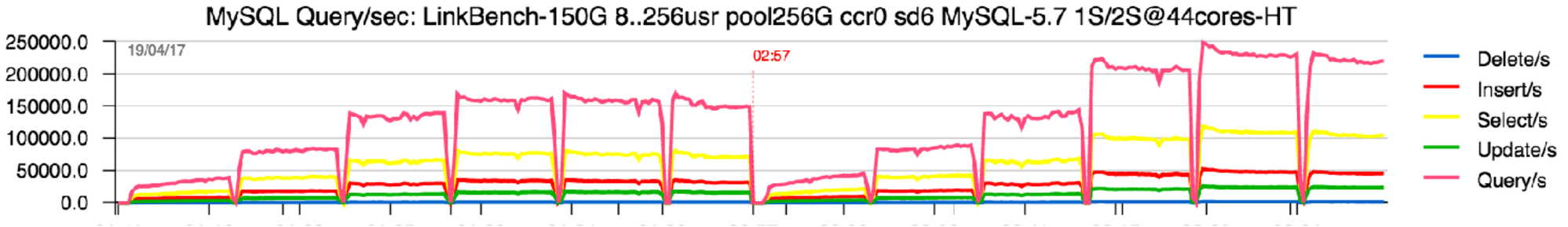
- Observations :
  - 1S => 2S : 50% performance improvement
  - what are the bottlenecks ?.. => 8.0 should fix, but block lock..

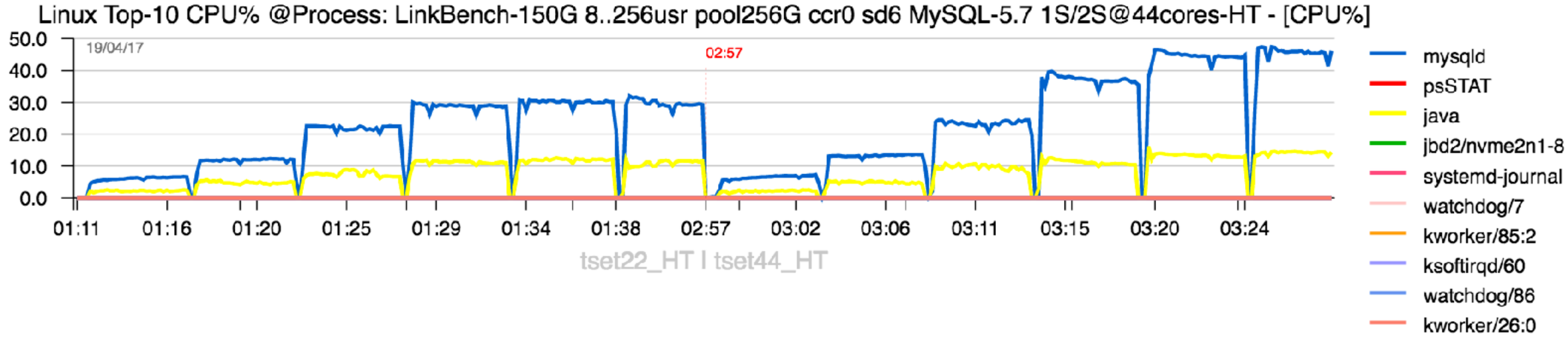




LinkBench-150G : BP=256GB (in-memory)

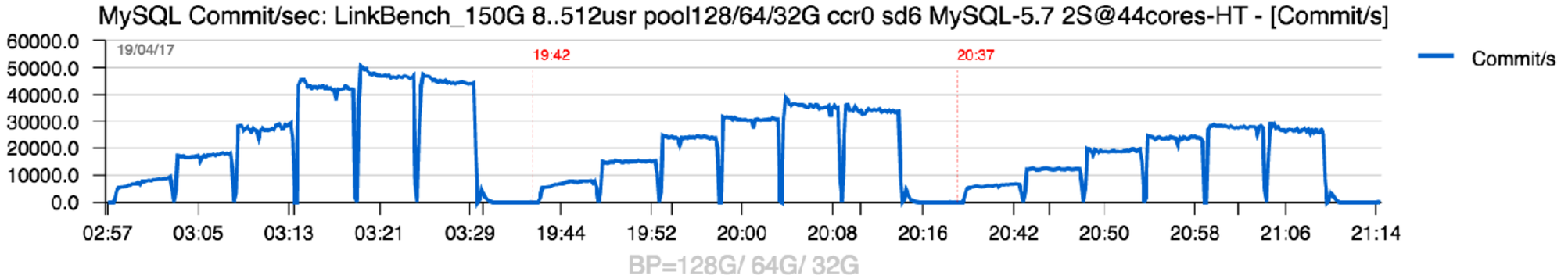
- Observations :
  - 1S => 2S : 50% performance improvement
  - low CPU usage.. => timeout in load generator itself ?..

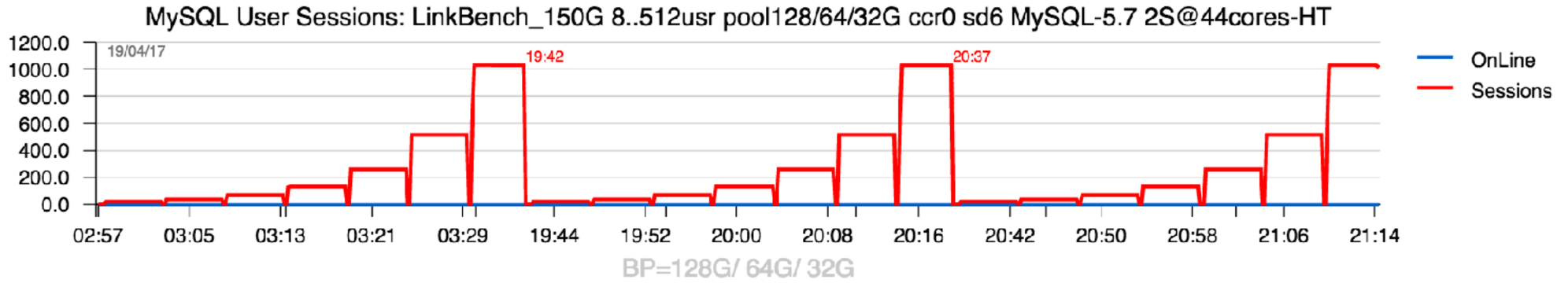




LinkBench-150G : BP=128GB/ 64GB/ 32GB

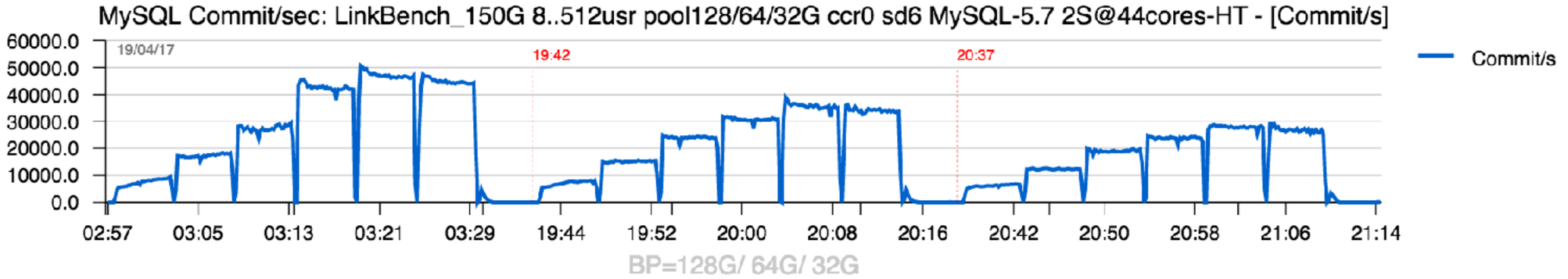
- Observations :
  - IO impact : significant, but much less than UNIFORM..
  - what about contentions ?..

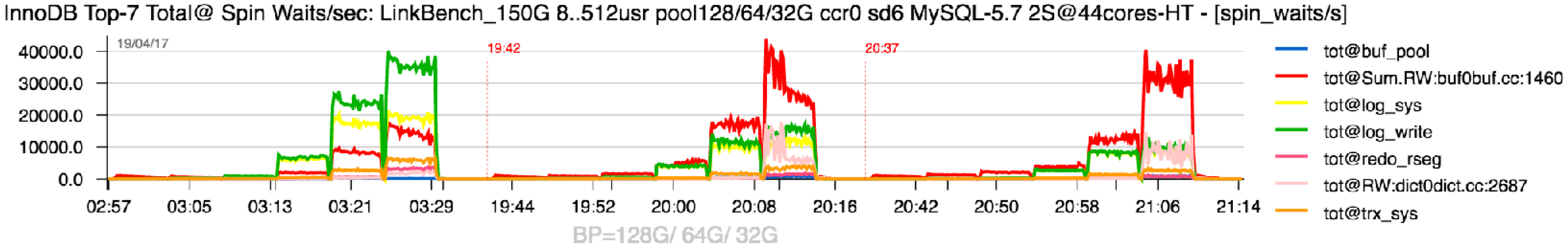




LinkBench-150G : BP=128GB/ 64GB/ 32GB

- Observations :
  - IO impact : significant, but much less than UNIFORM..
  - what about contentions ?.. => **block RW-lock** dominates on IO-bound !!!



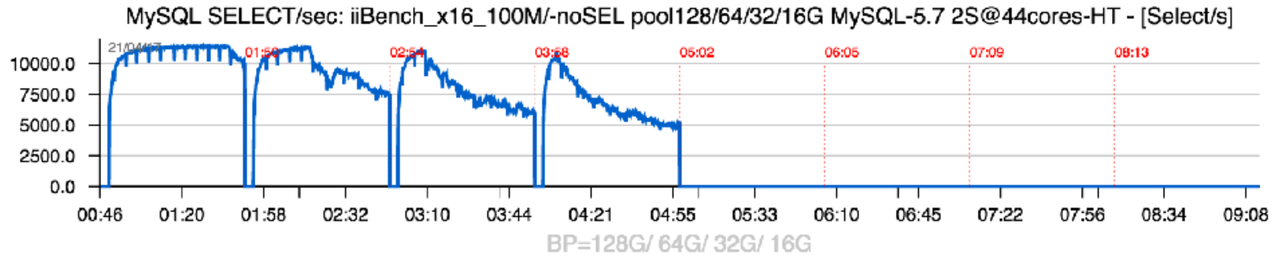


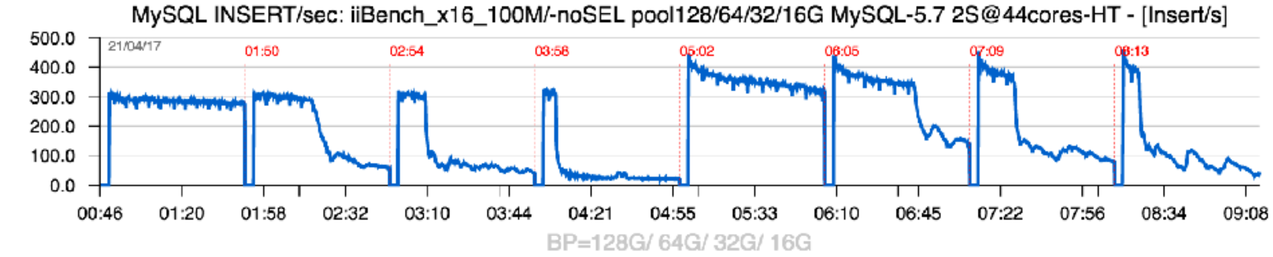
iiBench

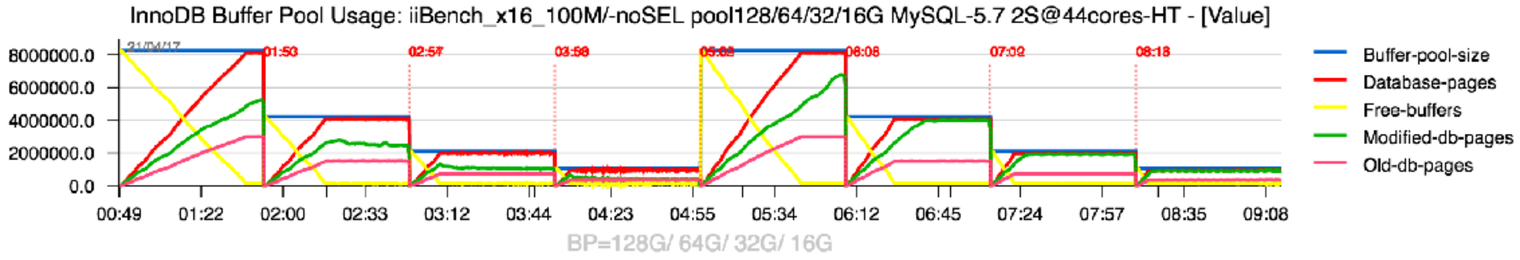
- Test Scenario :
  - x16 parallel iiBench processes running together during 1H
  - each process is using its own table
  - one test with SELECTs, another without..
- Side note :
  - iiBench is written in Python
  - during workload python process is using 60% CPU time itself (mysqld 40%)
  - makes sense to rewrite with sysbench 1.0 ?.. ;-)
- Key point :
  - during INSERT activity, B-Tree index in InnoDB growing quickly
  - as soon as index pages have no more place in BP and re-read from storage, performance is going down..

iiBench 100M x16 : BP= 128G/ 64G/ 32G/ 16G

- Observations :
  - until B-Tree remains in BP => 300K INSERT/sec.. (and if not, QPS drop)

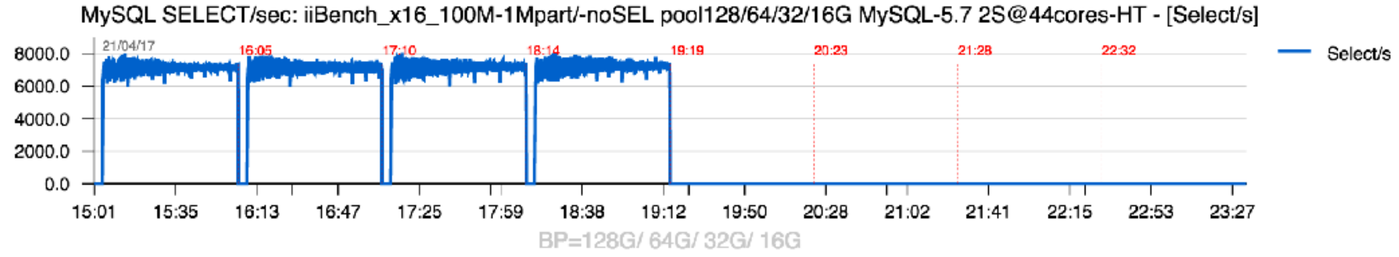


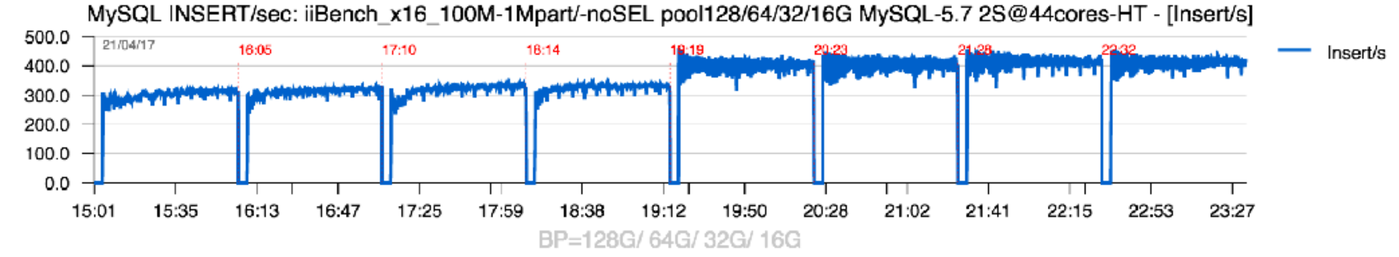


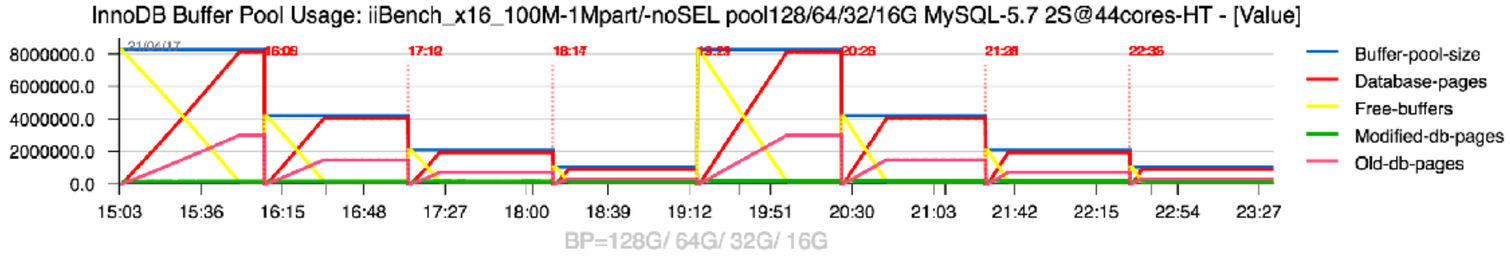


iiBench 100M x16 & 1M-parts : BP= 128G/ 64G/ 32G/ 16G

- Observations :
  - workaround : using partitions for table splits index B-Tree





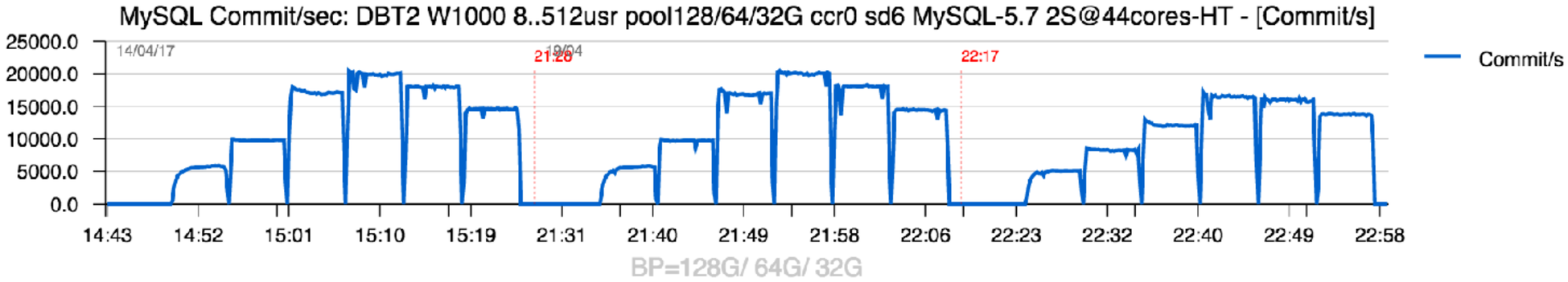


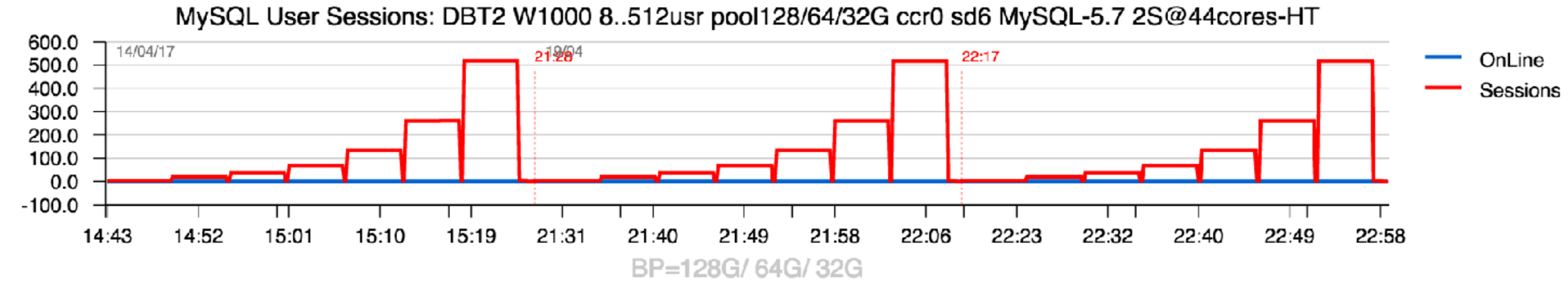
DBT2 (TPC-C)

- Data Volume :
  - 1000W (130G)
- Test Configurations :
  - BP = 128G/ 64G/ 32G
- Load :
  - 8, 16, … 512 concurrent users
- Engine :
  - MySQL 8.0-dev : has an issue on this workload..
  - so, testing with 5.7 to analyze potential gains

DBT2 1000W : BP = 128G/ 64G/ 32G

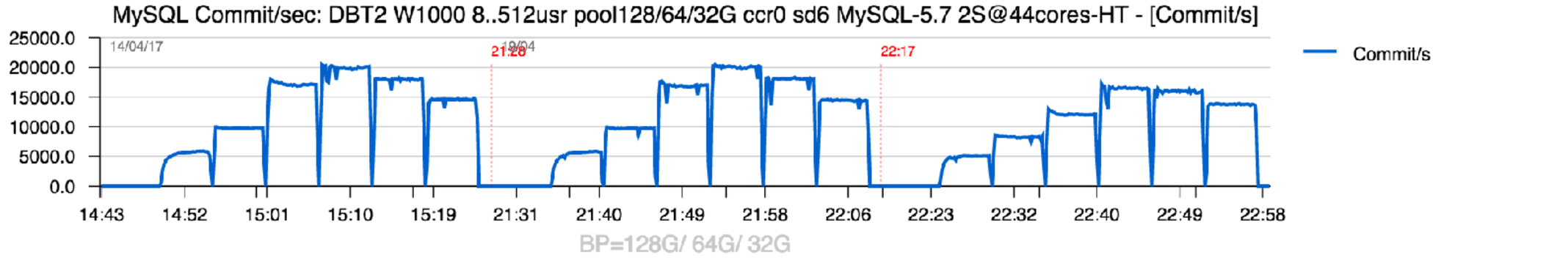
- Observations :
  - I/O impact : really small..
  - what is IO activity on 32GB BP ?..

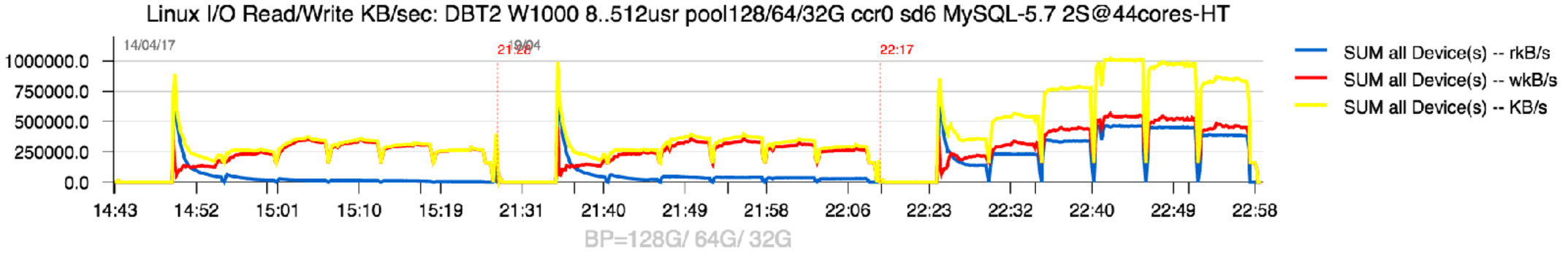




DBT2 1000W : BP = 128G/ 64G/ 32G

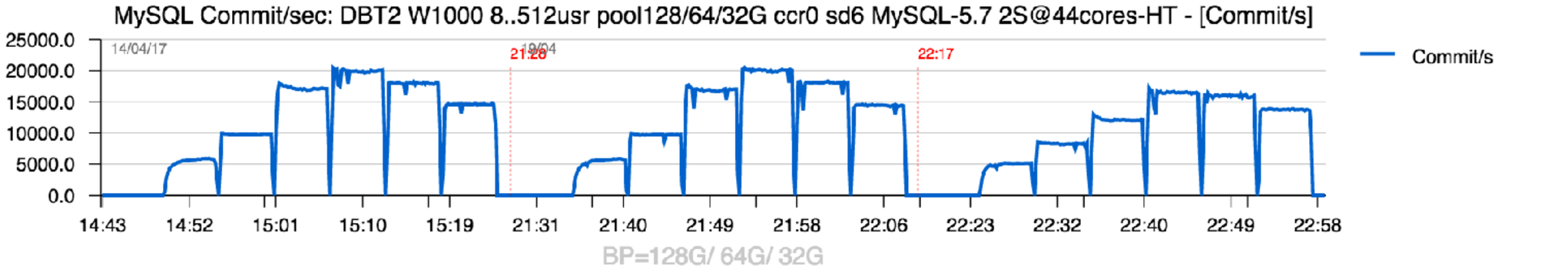
- Observations :
  - I/O impact : really small..
  - what is IO activity on 32GB BP ?.. => 1000 MB/s only (the lowest from all tests)

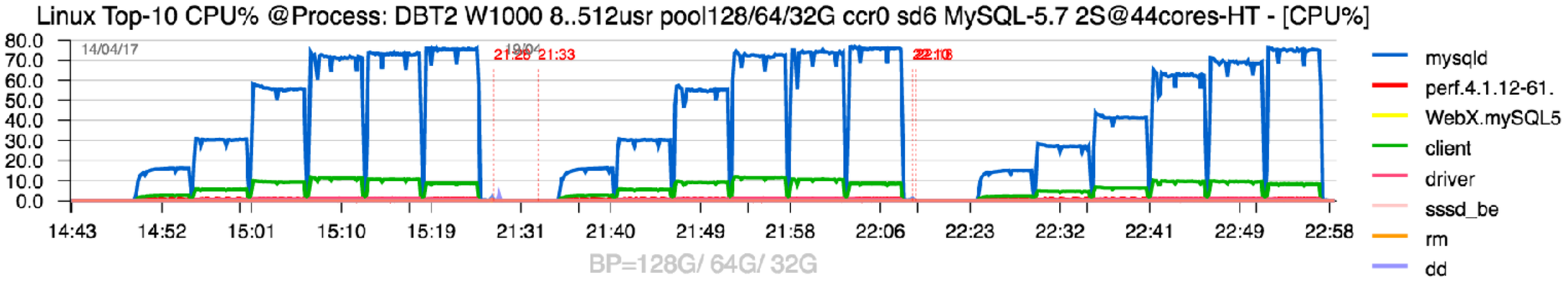




DBT2 1000W : BP = 128G/ 64G/ 32G

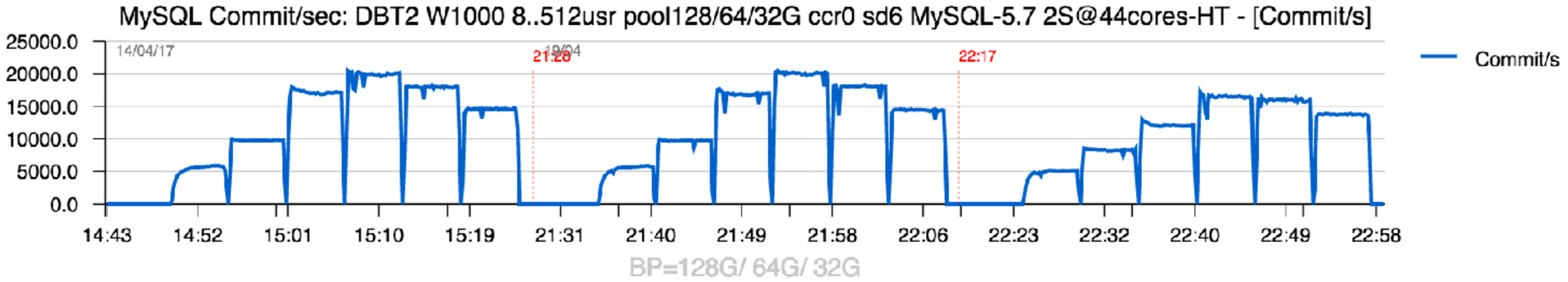
- Observations :
  - mostly CPU-bound => that's why so loved by CPU vendors ;-)
  - what about contentions ?..

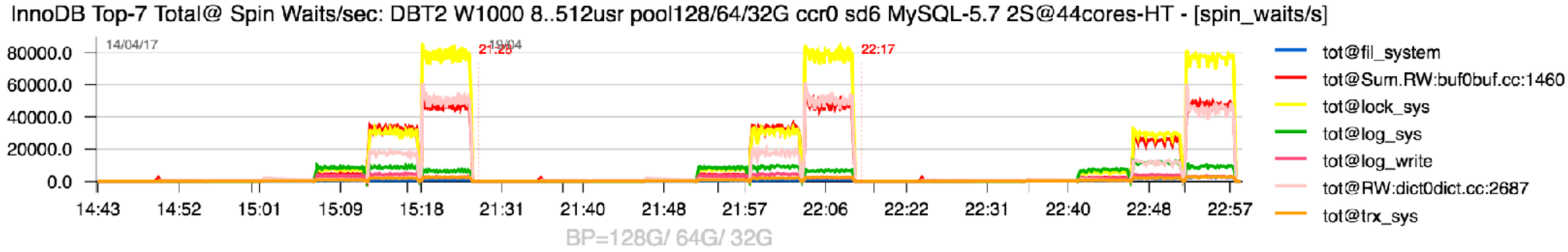




DBT2 1000W : BP = 128G/ 64G/ 32G

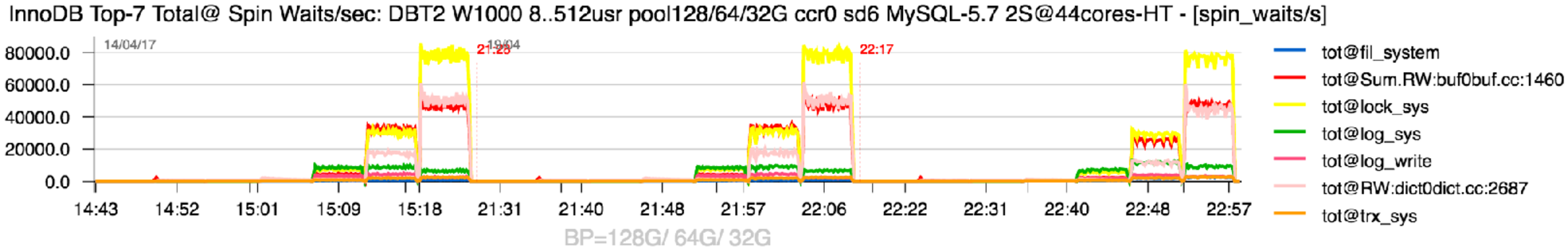
- Observations :
  - mostly CPU-bound => that's why so loved by CPU vendors ;-)
  - what about contentions ?.. => most are expected to gone with 8.0-dev..





DBT2 1000W : BP = 128G/ 64G/ 32G

- Observations :
  - mostly CPU-bound => that's why so loved by CPU vendors ;-)
  - what about contentions ?.. => most are expected to gone with 8.0-dev..
    - log_sys/ log_write/ trx_sys/ lock_sys ..
    - dict RW-lock => expected to gone with New DD
    - block RW-lock => … maybe not for 8.0 …
- Note :
  - maybe better not to use this workload for Storage validations ;-))




MAN
So, work in progress..
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.