MySQL

ORACLE

MySQL 8.0 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

- Benchmarks & Benchmarks..
- MySQL 8.0 Performance gains
- Pending issues & workarounds
- Work in progress
  - Regressions “mystery”
  - TPCC “mystery”
  - dbSTRESS “headache”
  - etc..
- Q & A

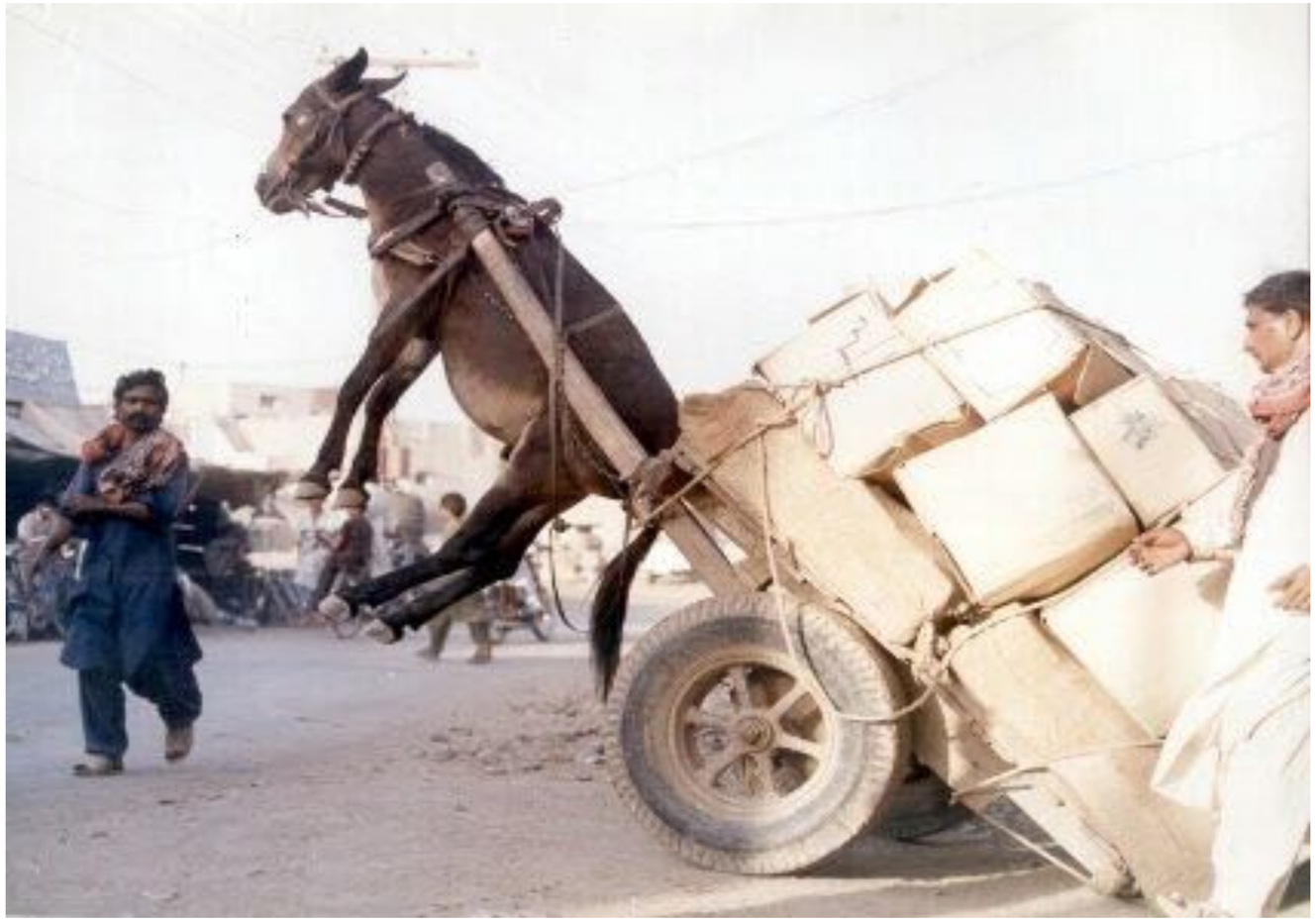
Why MySQL Performance ?...

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..

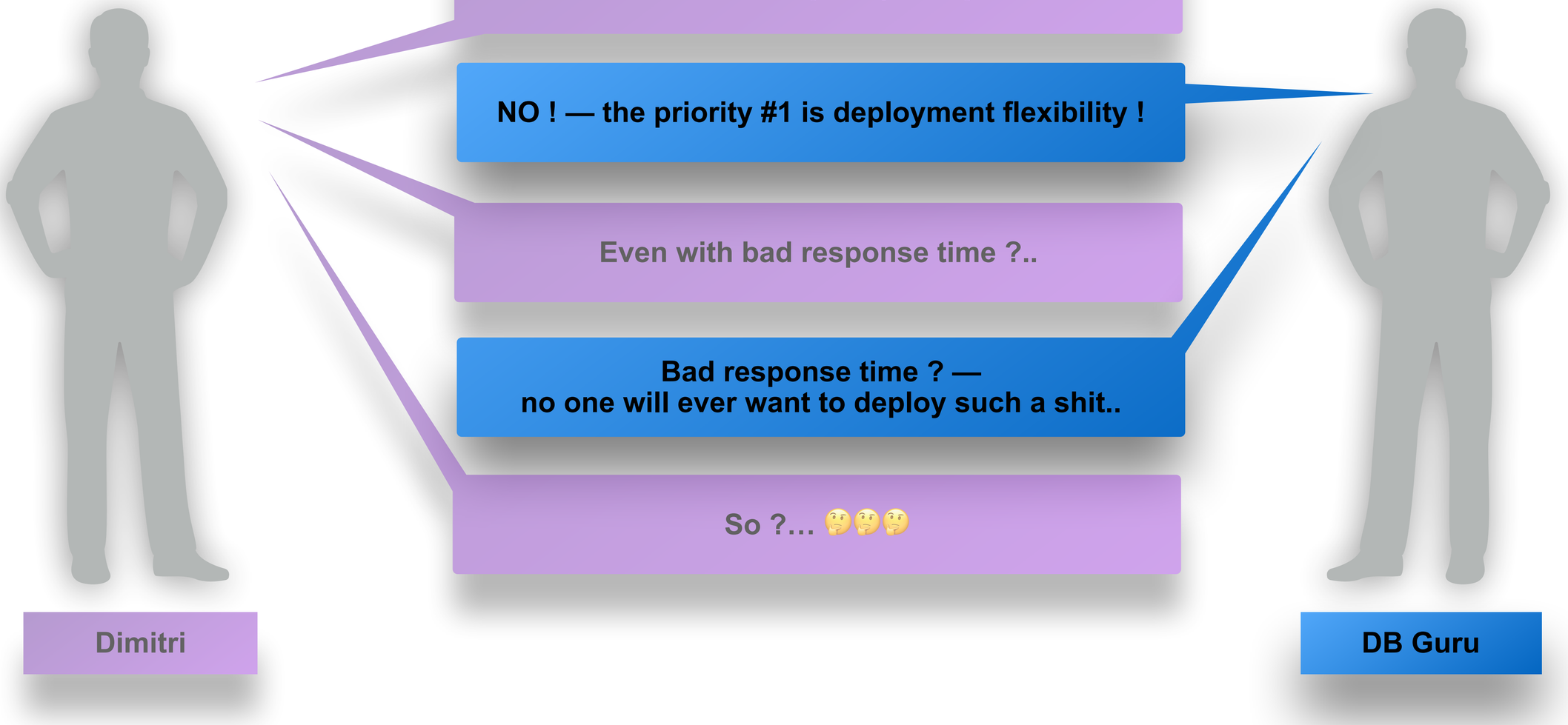
Why MySQL Performance ?..

- 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!... :-)

Why MySQL Performance ?



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

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

USE YOUR BRAIN !!! ;-)

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

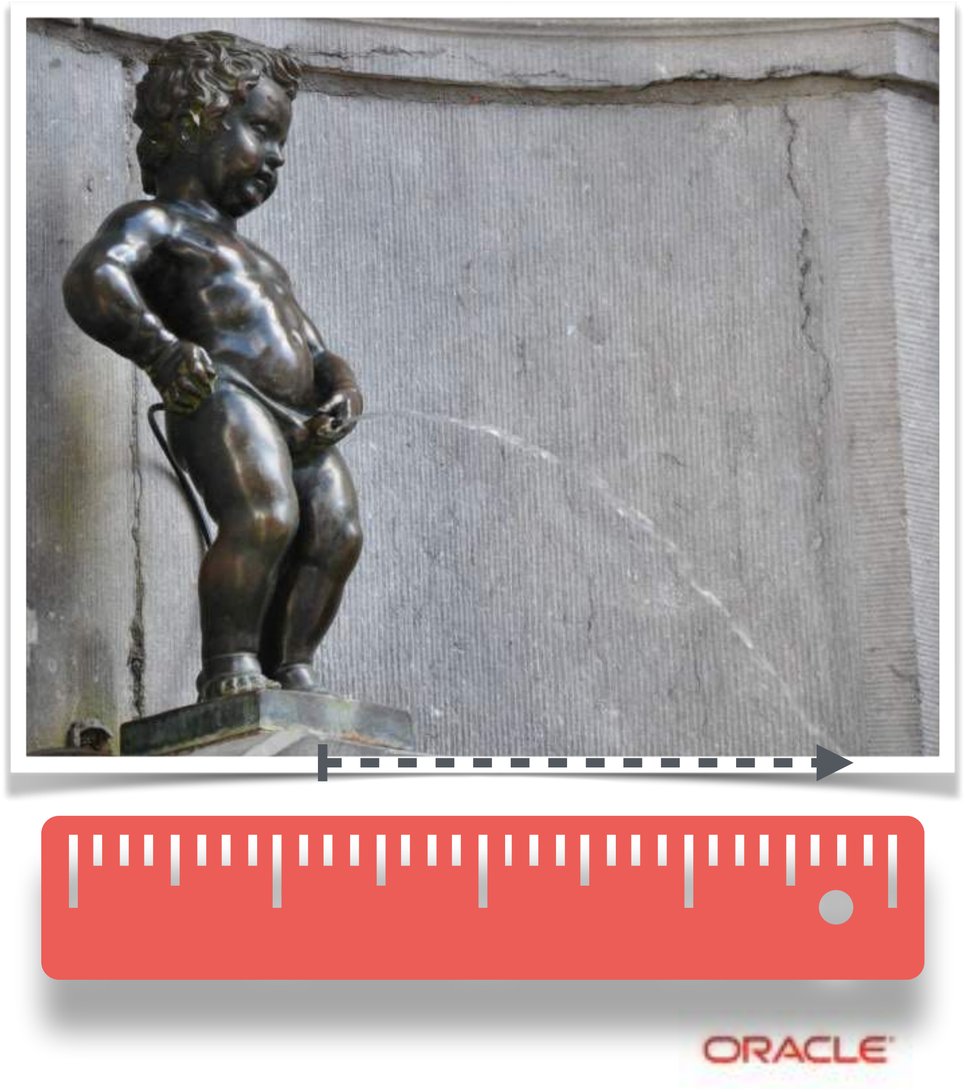
USE YOUR BRAIN !!! ;-)

Common Sources of MySQL Performance Problems..

- “Fixable” ones ;-)
  - DB Schema/ Indexes/ SQL query/ Optimizer plan/ Apps code/ etc. etc..
  - odd tuning/ wrong config setup/
  - e.g. generally can be fixed by => RTFM ! ;-)
- “By design” ones..
  - known ?..
  - workaround ?..
  - can be ever fixed ?..
  - heh…
  - work in progress.. <= and here is where we come ;-))

Why Benchmarks ?

- Common perception of benchmarks is often odd..
  - “not matching real world”…
  - “pure Marketing”..
  - “pure BenchMarketing”..
  - etc. etc. etc..
- well..
  - “it depends..” ©
  - get your own opinion by understanding of the tested workloads !
  - e.g. remind Best Practice #1 ;-))



Benchmarks & MySQL

- Every test workload is pointing to a problem to resolve !
  - evaluate & understand the problem(s)
  - then try to fix it
  - or propose a workaround
  - evaluate & confirm the fix / workaround
  - keep running in QA to discover any potential regression ON TIME !..

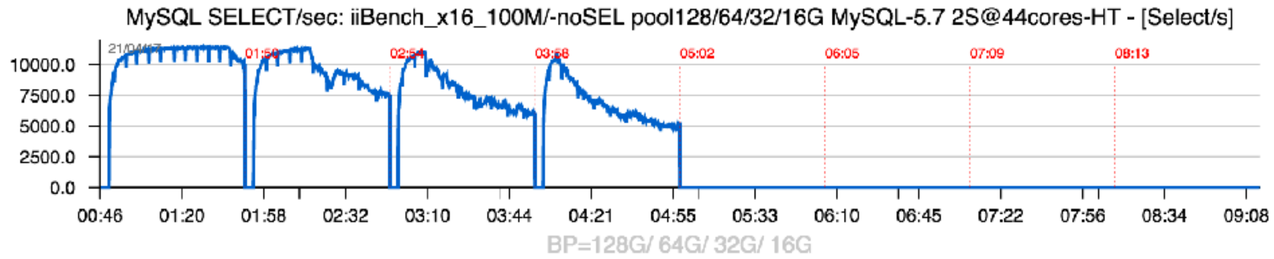
- As well :
  - kind of “reference” of what to expect
  - evaluate any new HW, systems, etc..

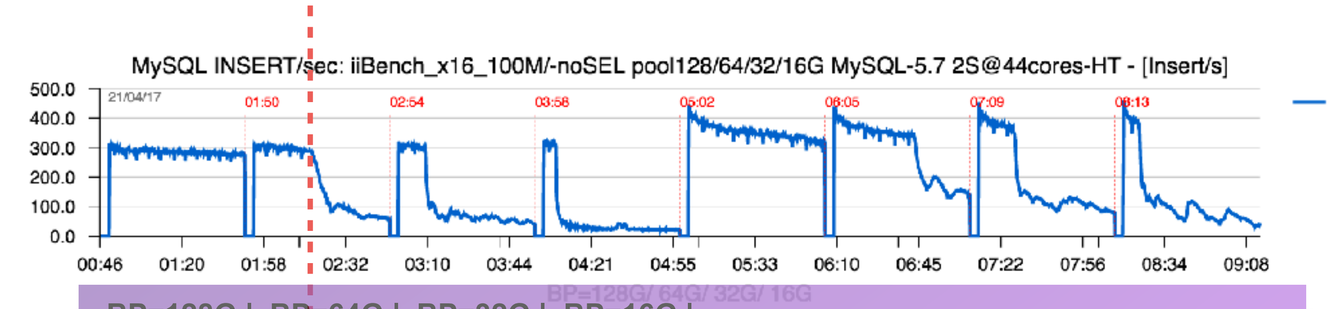
Example: iiBench (INSERT Benchmark)

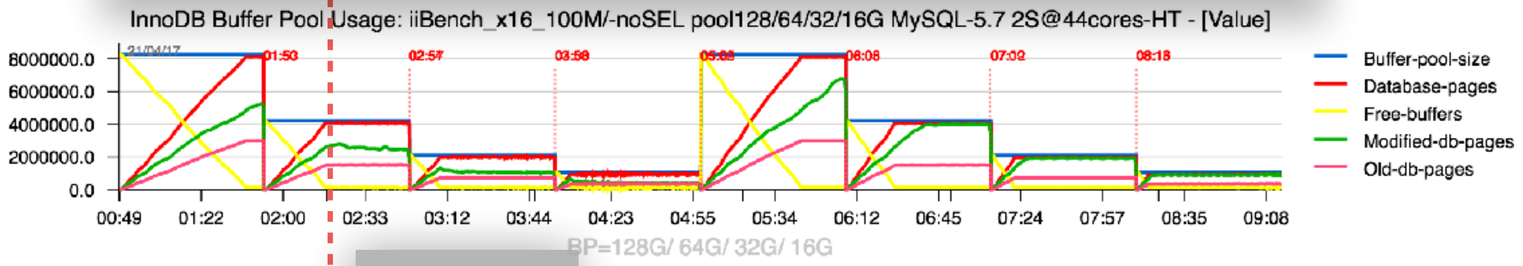
- Main claim :
  - InnoDB is xN times slower vs Write-oriented Engine XXX
  - so, use XXX, as it’s better
- Test Scenario :
  - x16 parallel iiBench processes running together during 1H
  - each process is using its own table
  - one test with SELECTs, another without..
- Key point :
  - during INSERT 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..
  - e.g. “by design” problem ;-))

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

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



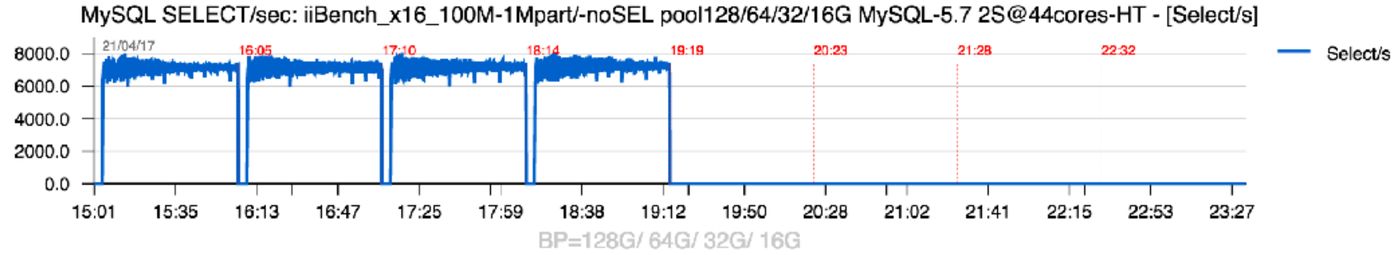


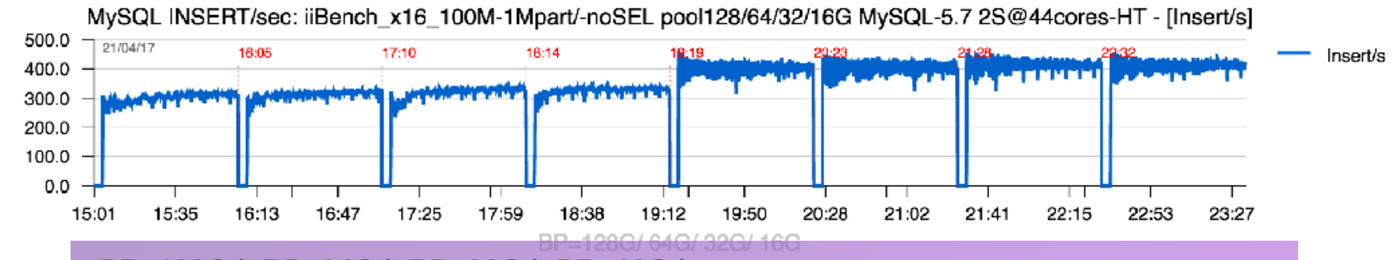


free buffers !!!

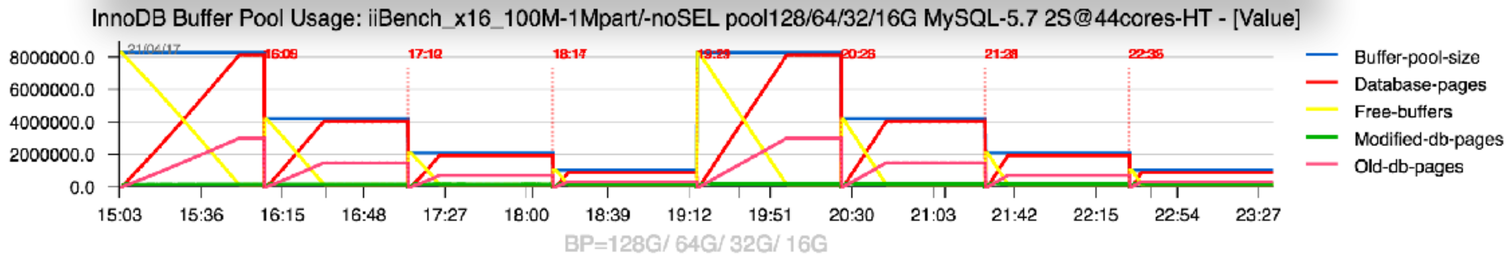
iiBench 100M x16 & 1M-parts : BP= 128G/ 64G/ 32G/ 16G

- Observations :
  - workaround : keep index cached in BP (via table partitions or other)



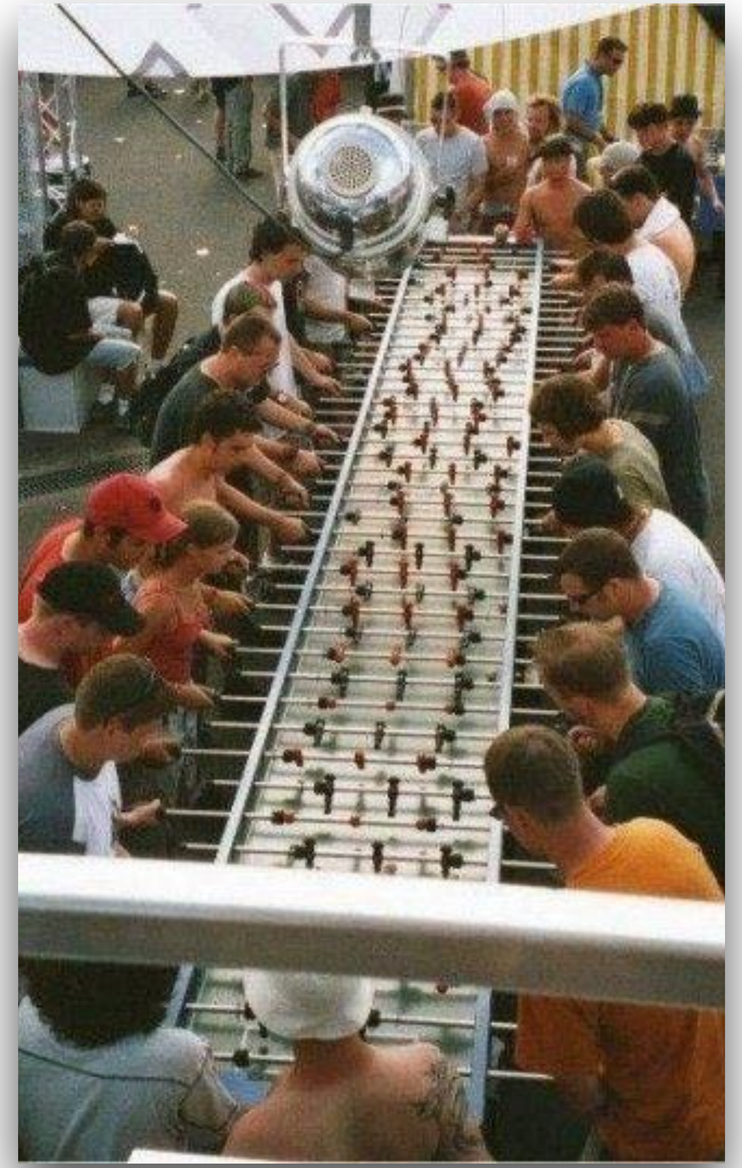


BP=128G | BP=64G | BP=32G | BP=16G |



The Best Test Workload

- For You :
  - workload simulating your Production !
  - JMetter (free)
  - LoadRunner ($$)
  - etc.
- otherwise :
  - use “generic” test workloads
  - change / adapt / extend..
  - **share** with us ! ;-))



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 - #1
  - The “Entry Ticket” Workloads - looks simple, but still the most complete test kit !
  - OLTP, RO/RW, points on various RO and RW issues
- TPC-C : DBT2 / TPCC-like / HammerDB / Sysbench-TPCC
  - OLTP, RW, pretty complex, growing db
  - HammerDB : fully based on Stored Procedures
- DBT-3 / TPC-H
  - DWH, RO, complex heavy queries, loved by Optimizer Team ;-)
- dbSTRESS
  - OLTP, RO/RW, several tables, points on RW and Sec.IDX issues
- iiBench
  - pure INSERT bombarding + optionally SELECTs, points on B-Tree issues

Understand what you're testing !!!

- Example of Proposed Benchmark Workload by DigitalOcean :
  - 1) deploy "employee" database set
  - 2) run "mysqlslap" with 5 SQL queries and "--concurrency=50"
  - ref: https://www.digitalocean.com/community/tutorials/how-to-measure-mysql-query-performance-with-mysqlslap

Understand what you're testing !!!

- Example of Proposed Benchmark Workload by DigitalOcean :
  - 1) deploy "employee" database set
  - 2) run "mysqlslap" with 5 SQL queries and "--concurrency=50"
  - ref: https://www.digitalocean.com/community/tutorials/how-to-measure-mysql-query-performance-with-mysqlslap
- Now, what exactly this test is doing ?
  - all 5 SQL queries => SELECT * from table;
  - e.g. 5 full scans of 5 tables concurrently executed by 50 users
- Does it represents your Production Workload ???
  - (e.g. mind "Best practice #1")

Why Sysbench is #1 ?..

- Historically :
  - the most simple to install, to use, most lightweight
  - why entry ticket : covers most important “key workload cases” in MySQL performance

- New Sysbench :
  - https://github.com/akopytov/sysbench
  - have fixed all past issues
  - high flexibility for any test scenario with LUA scripts
  - integrated LUA JIT => high execution speed + lightweight !
  - more various test scenarios are expected to come
  - excellent opportunity to write your own test cases !
  - move and use it now ! ;-)
  - => becomes a true dev **platform** for workload scenarios !

Your Own Test Scenario on Sysbench ?

- “Just do it!”

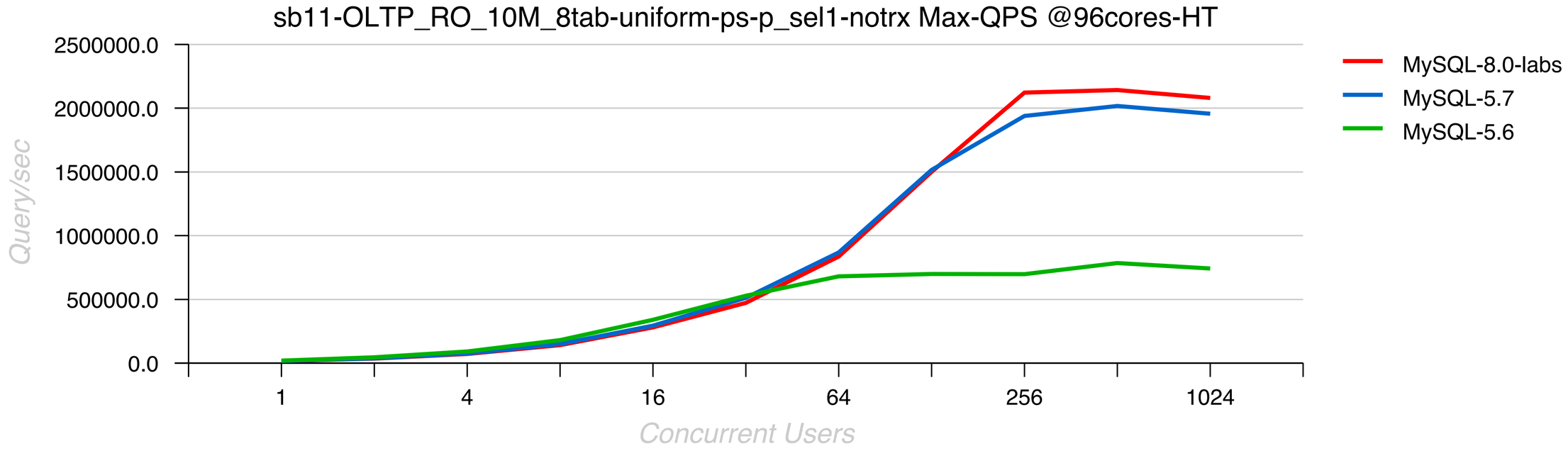
```
function thread_init()
  drv = sysbench.sql.driver()
  con = drv:connect()
end

function thread_done()
  con:disconnect()
end

function event()
  local id = sysbench.rand.default( 1, 1000000 )
  con:query( "select * from table1 where id = " .. id )
end
```

MySQL Scalability 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..)
  - but : RW workloads are faster than RO ! ;-))
- MySQL 5.7
  - finally fully unlocked Read-Only + no more contentions on the “Server” layer, etc..
  - and (finally) RO is faster than RW ;-))
- MySQL 8.0
  - main focus is on efficiency : do more on the same HW ;-))
  - main target HW : 2CPU Sockets systems
  - RW scalability.. & data security..
  - NOTE : Continuous Release Model !

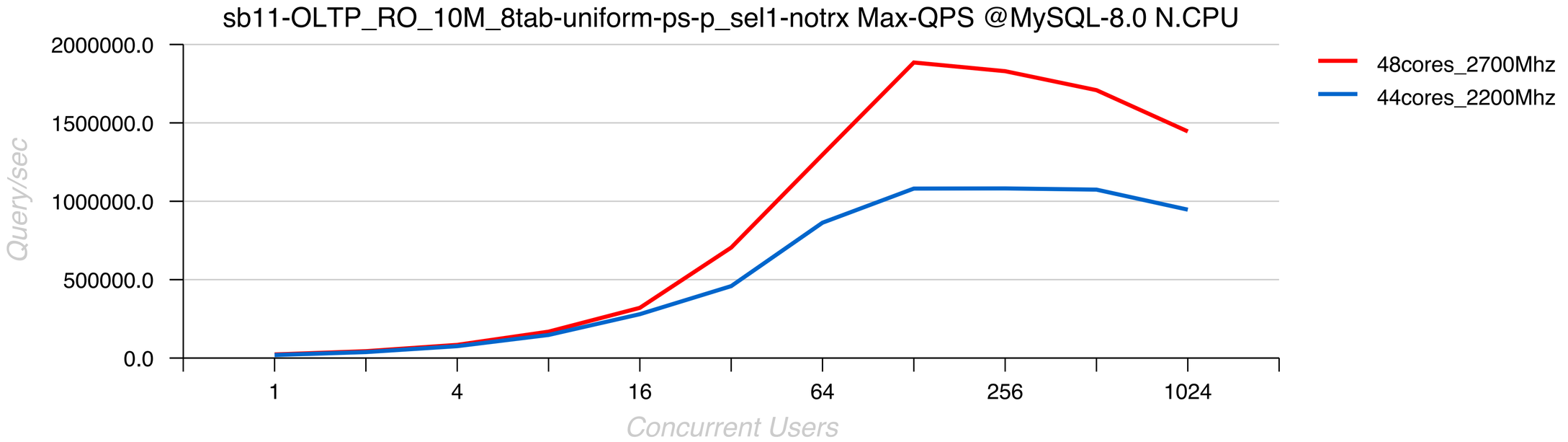
RO Point-Selects @MySQL 8.0-labs (Sep.2017)

- **2.1M (!!) QPS** Sysbench Point-Selects 10Mx8tab :
  - 4CPU Sockets (4S) : 96cores-HT Broadwell v4
  - the main gain : NO regression -vs- 5.7 !! (and I'm serious ;-))



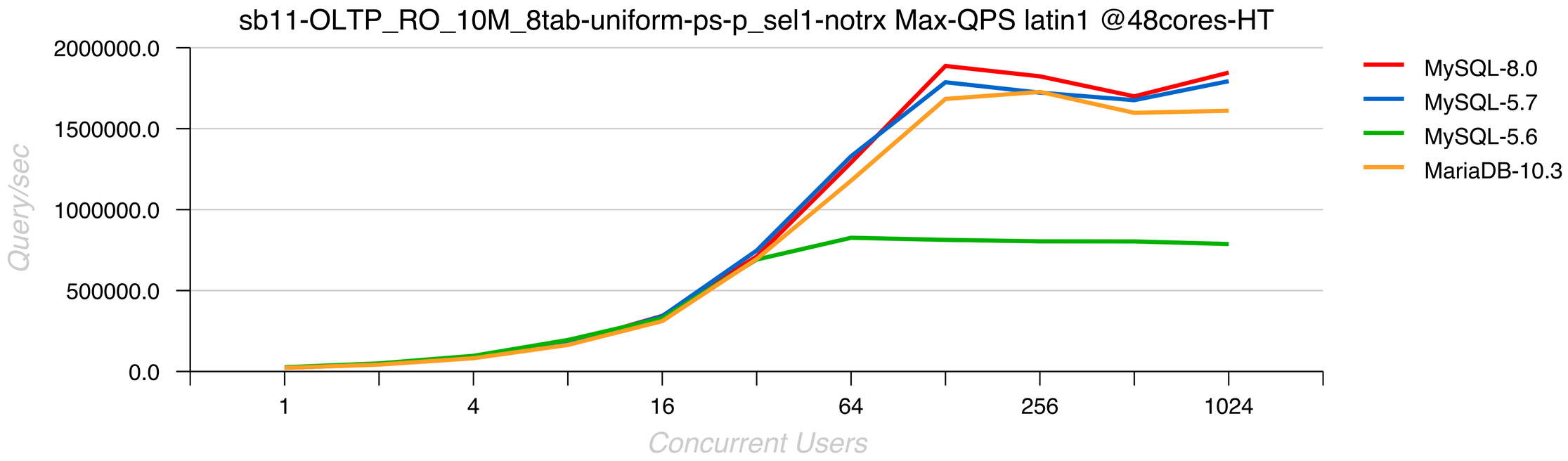
RO Point-Selects @MySQL 8.0 (Dec.2017)

- Sysbench Point-Selects 10Mx8tab @2CPU Sockets
  - 2CPU Sockets Broadwell v4 : 44cores-HT
  - 2CPU Sockets **Skylake** Platinum : 48cores-HT
  - => near **80**% gain in peak QPS !! already 50% on 32usr load !!



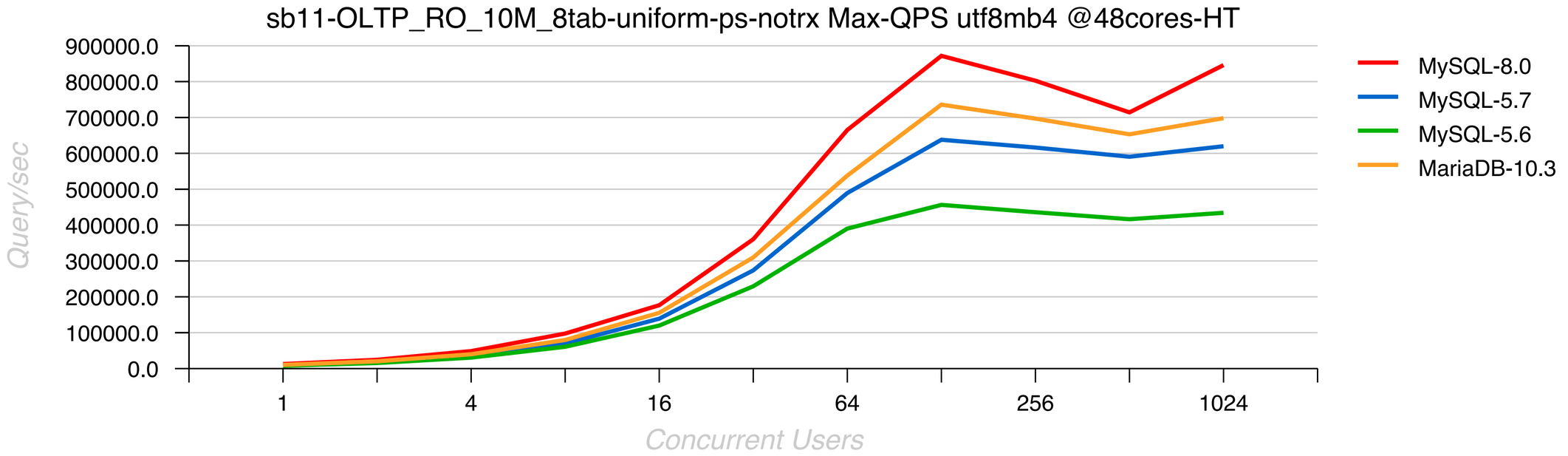
RO Point-Selects @MySQL 8.0 (Apr.2018)

- **1.8M+ (!!) QPS** Sysbench Point-Selects 10Mx8tab @2CPU Sockets
  - 2CPU Sockets **Skylake** Platinum : 48cores-HT
  - => currently our best results on 2S HW



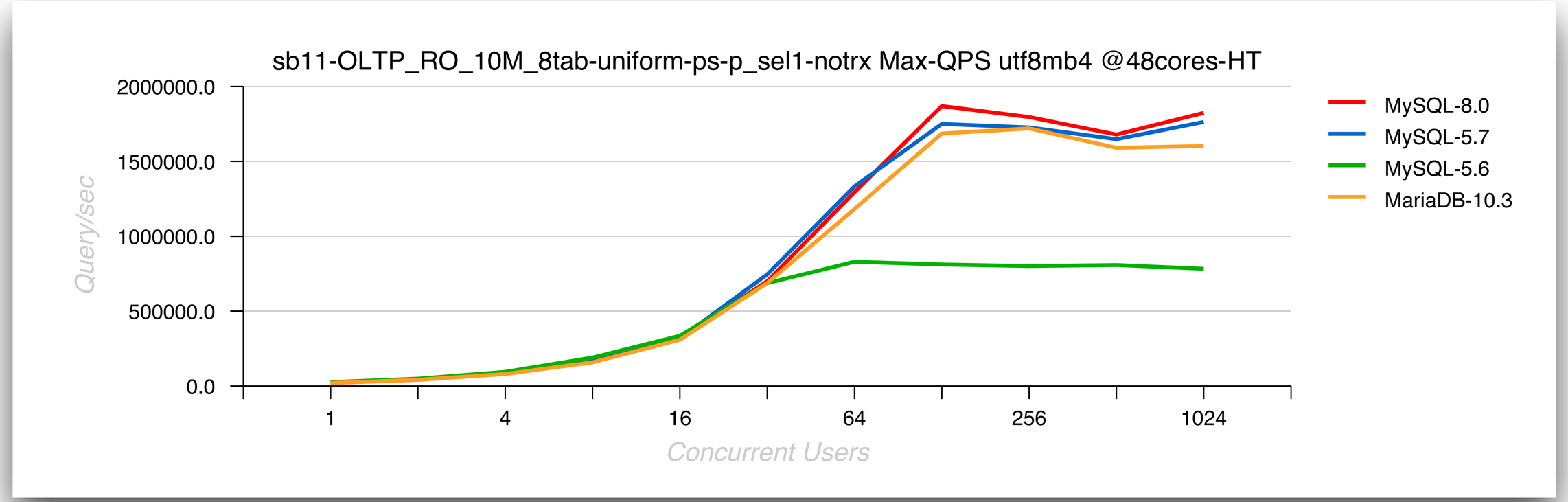
OLTP_RO UTF8mb4 @MySQL 8.0 (Apr.2018)

- **870K(!!) QPS** Sysbench OLTP_RO 10Mx8tab @2CPU Sockets
  - 2CPU Sockets **Skylake** Platinum : 48cores-HT
  - => up to 40% better than 5.7



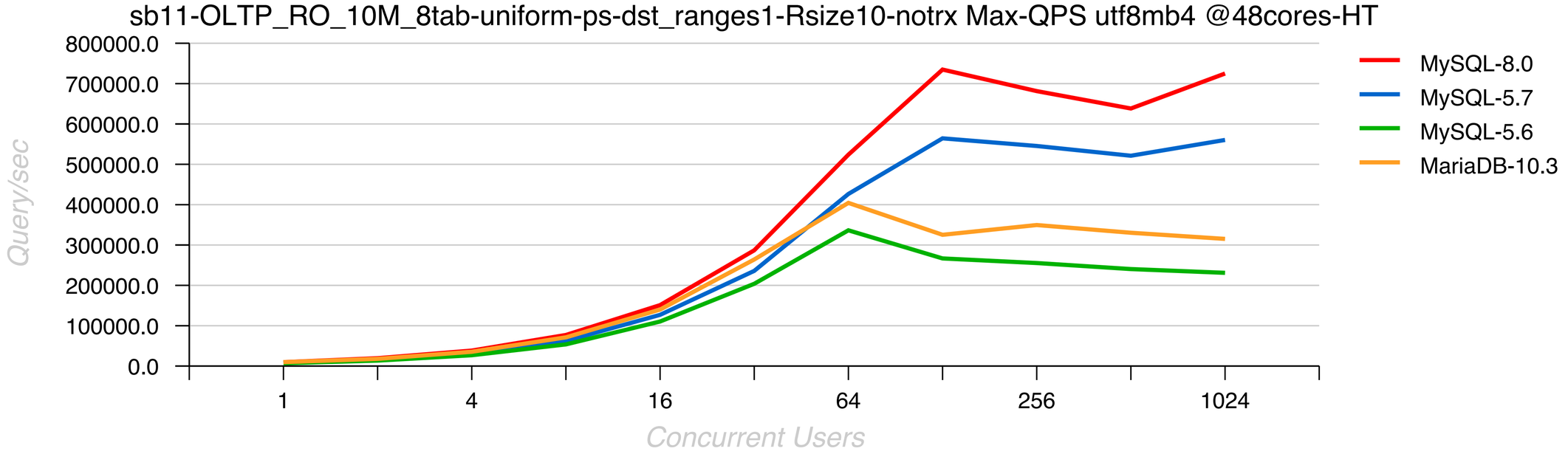
RO Point-Selects UTF8mb4 @MySQL 8.0 (Apr.2018)

- **Same QPS** Sysbench RO point-selects 10Mx8tab @2CPU Sockets
  - 2CPU Sockets **Skylake** Platinum : 48cores-HT
  - e.g. UTF8 mostly impacts when you have to compare data (order by / group by / etc.)



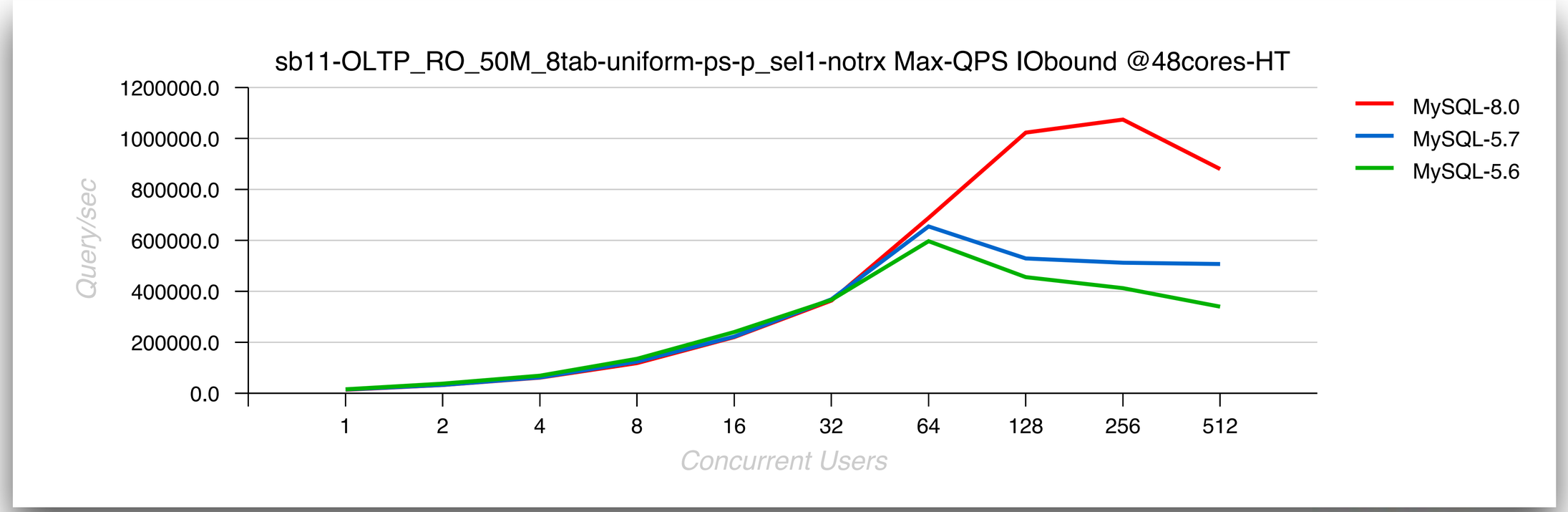
RO Distinct-Ranges UTF8mb4 @MySQL 8.0 (Apr.2018)

- **735K (!!) QPS** Sysbench RO dist-ranges 10Mx8tab @2CPU Sockets
  - 2CPU Sockets **Skylake** Platinum : 48cores-HT
  - => up to 30% better than 5.7

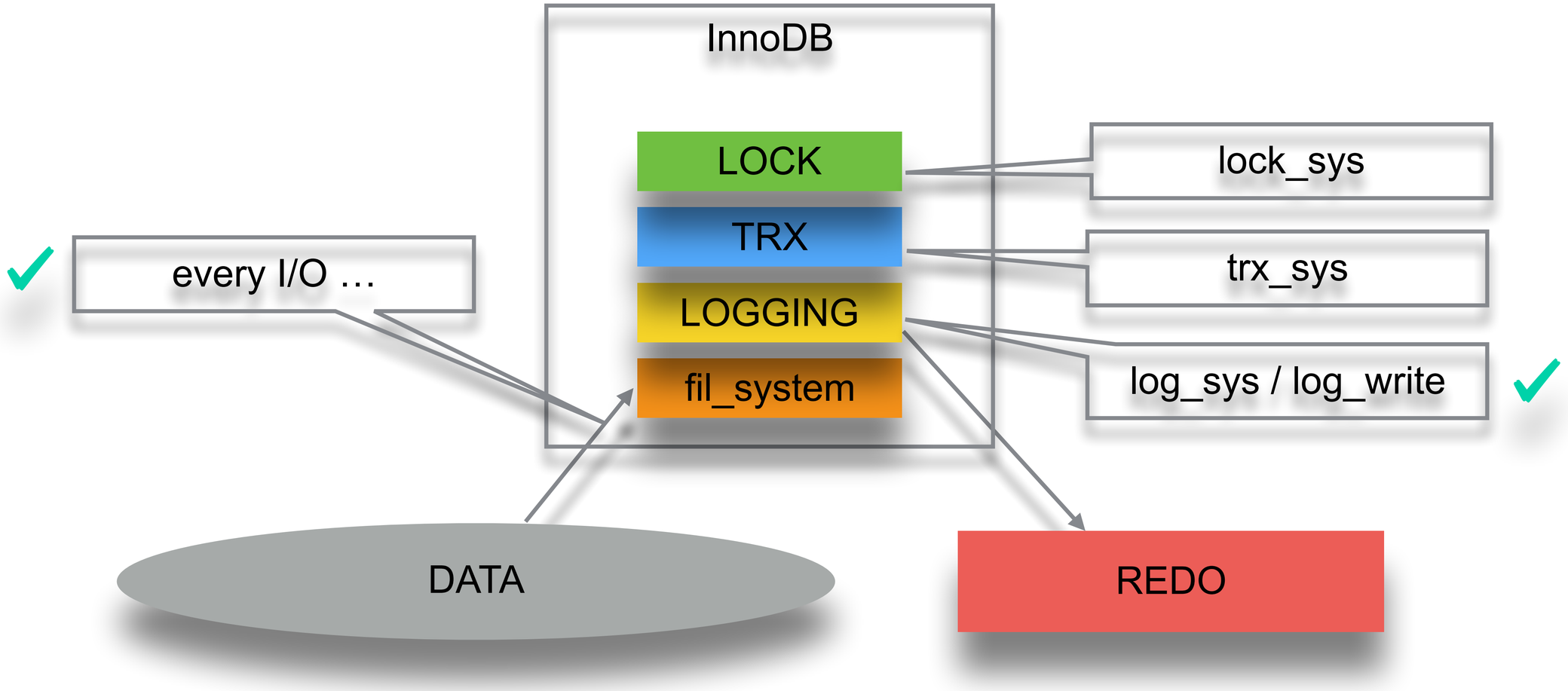


IO-bound Point-Selects @MySQL 8.0 (Apr.2018)

- IO-bound Sysbench OLTP_RO Point-Selects
  - 50M x 8-tables, 48cores-HT, x2 Optane drives
  - over **1M IO-bound QPS** with MySQL 8.0 !!!
  - all details : http://dimitrik.free.fr/blog/posts/mysql-performance-1m-iobound-qps-with-80-ga-on-intel-optane-ssd.html

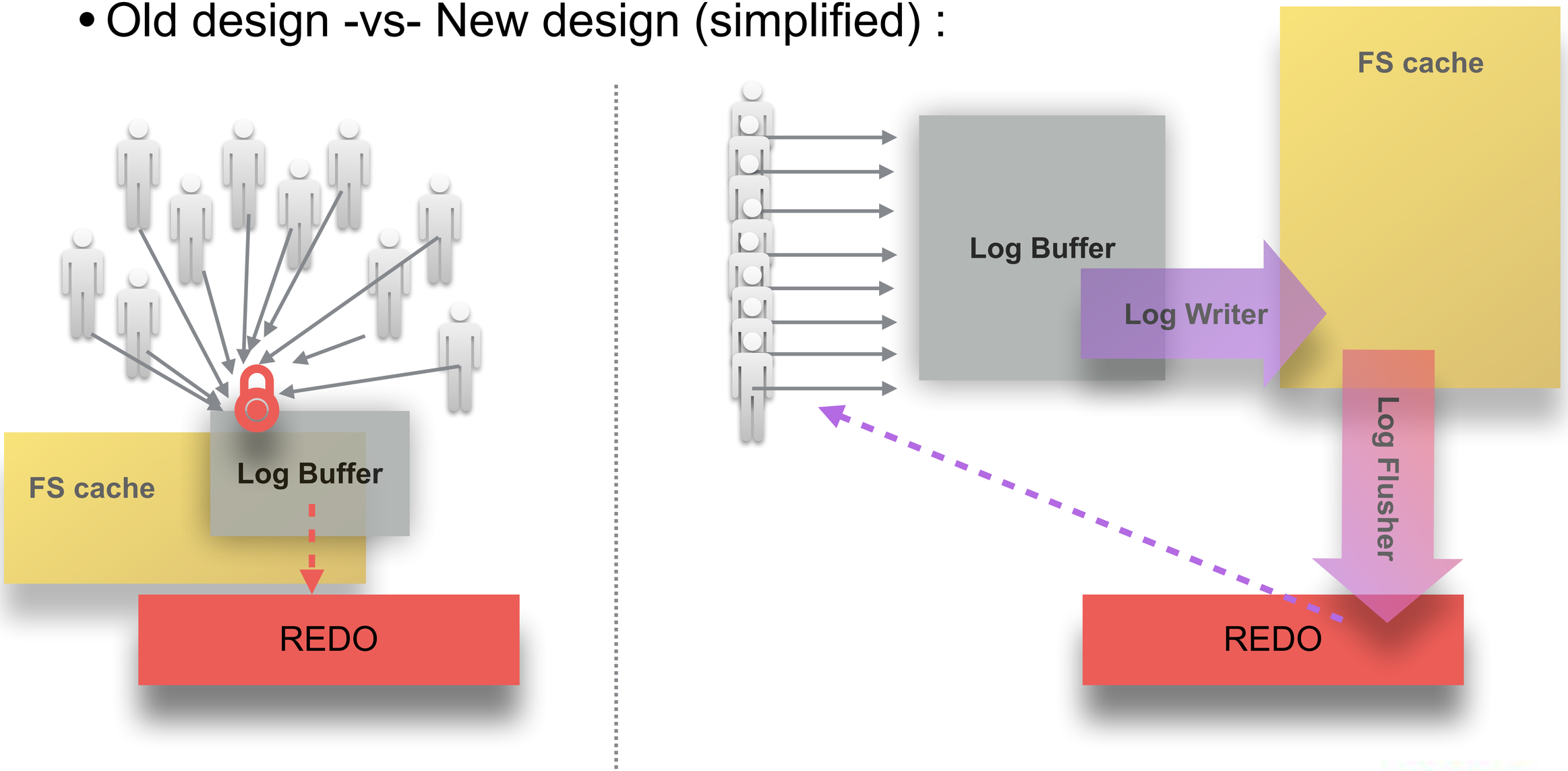


MySQL 8.0 : New Design for InnoDB Fundamentals..



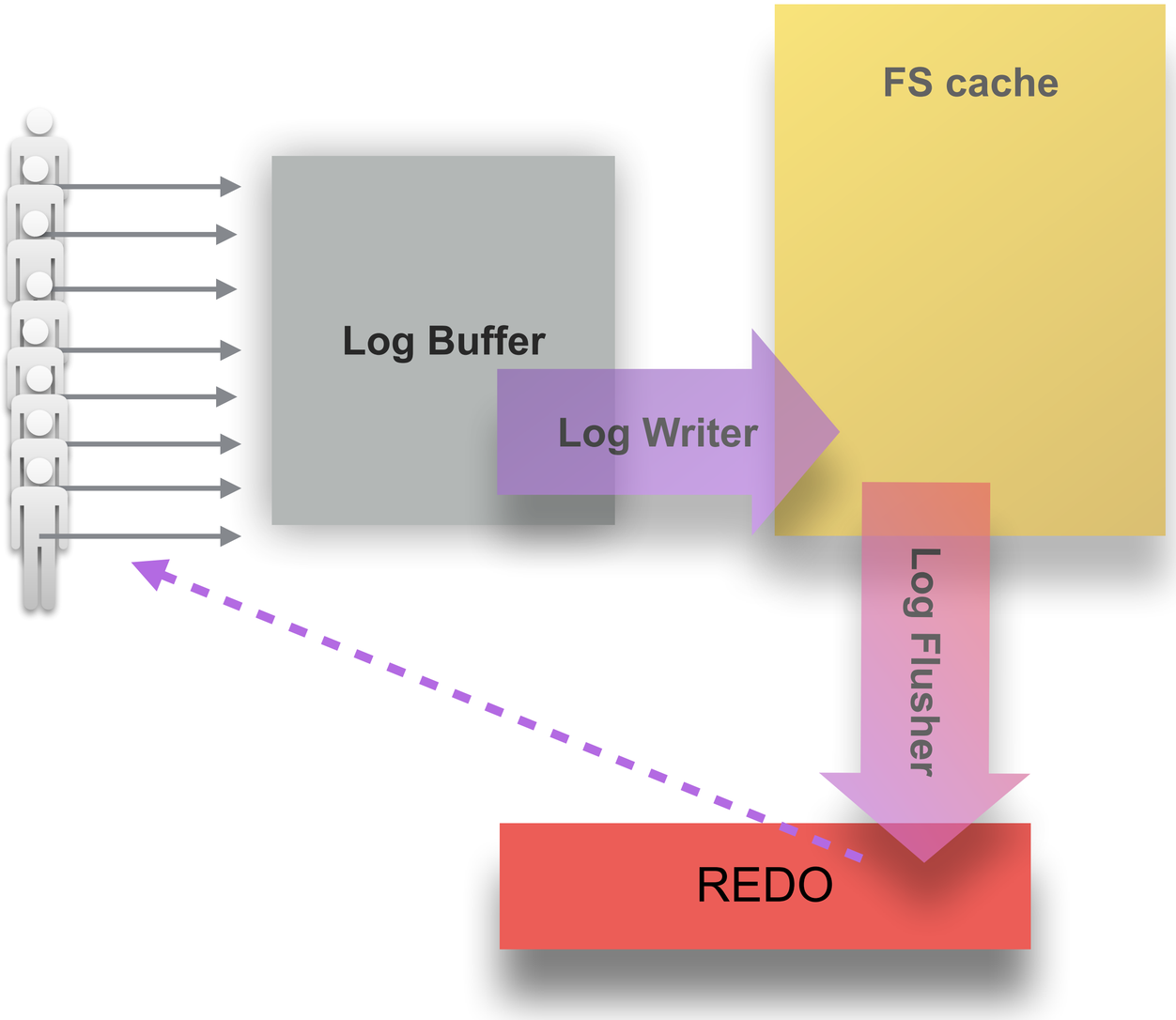
MySQL 8.0 : Re-Designed REDO

- Old design -vs- New design (simplified) :



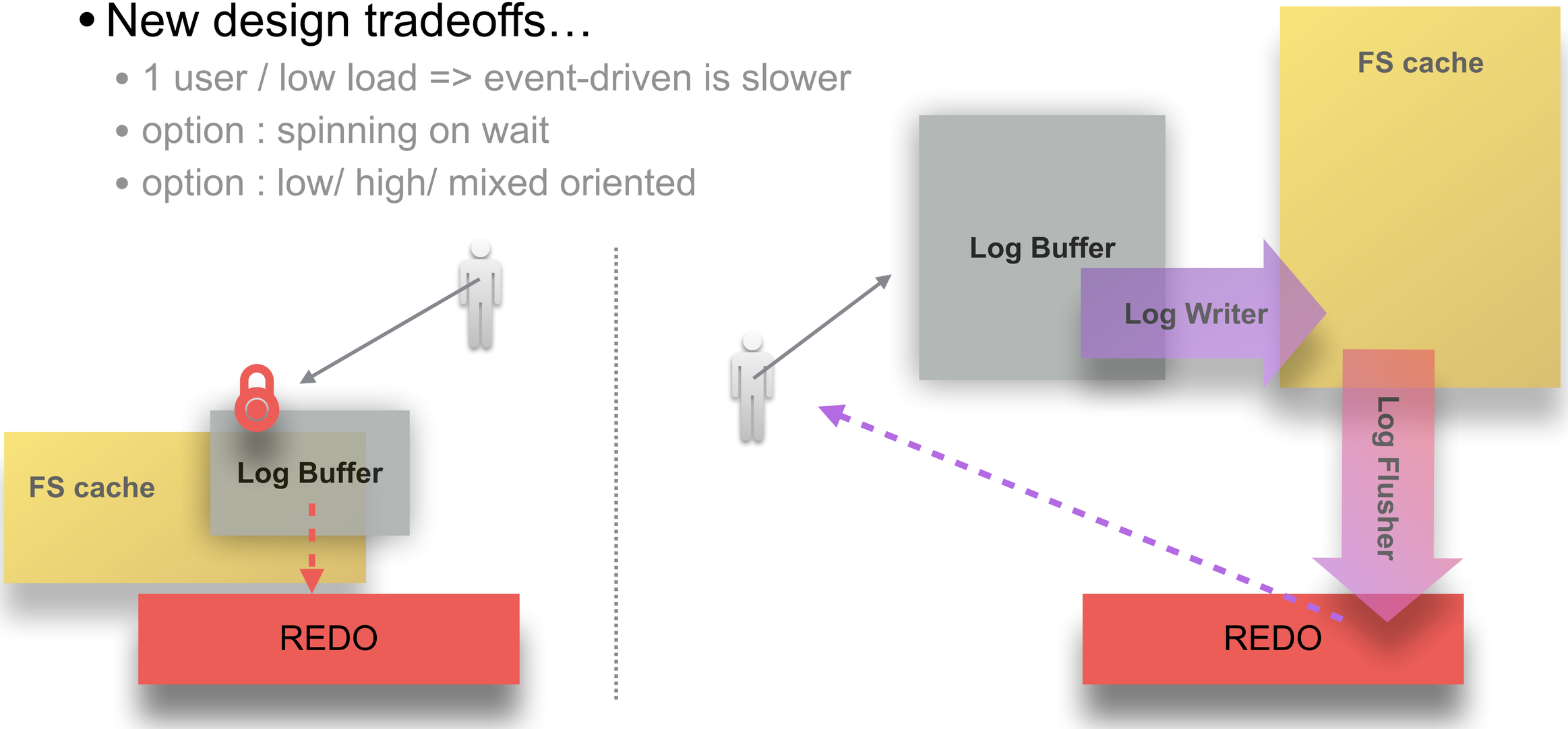
MySQL 8.0 : Re-Designed REDO

- New REDO design :
  - users are not fighting anymore !
  - self-driven processing..
  - self-driven by fsync() capacity
- Instrumented !
  - spins / waits
  - writer / flusher rates
  - max / avg flush times
  - etc..
- Configuration :
  - **mostly all dynamic !!!**
  - so you can play with it on-line ;-))



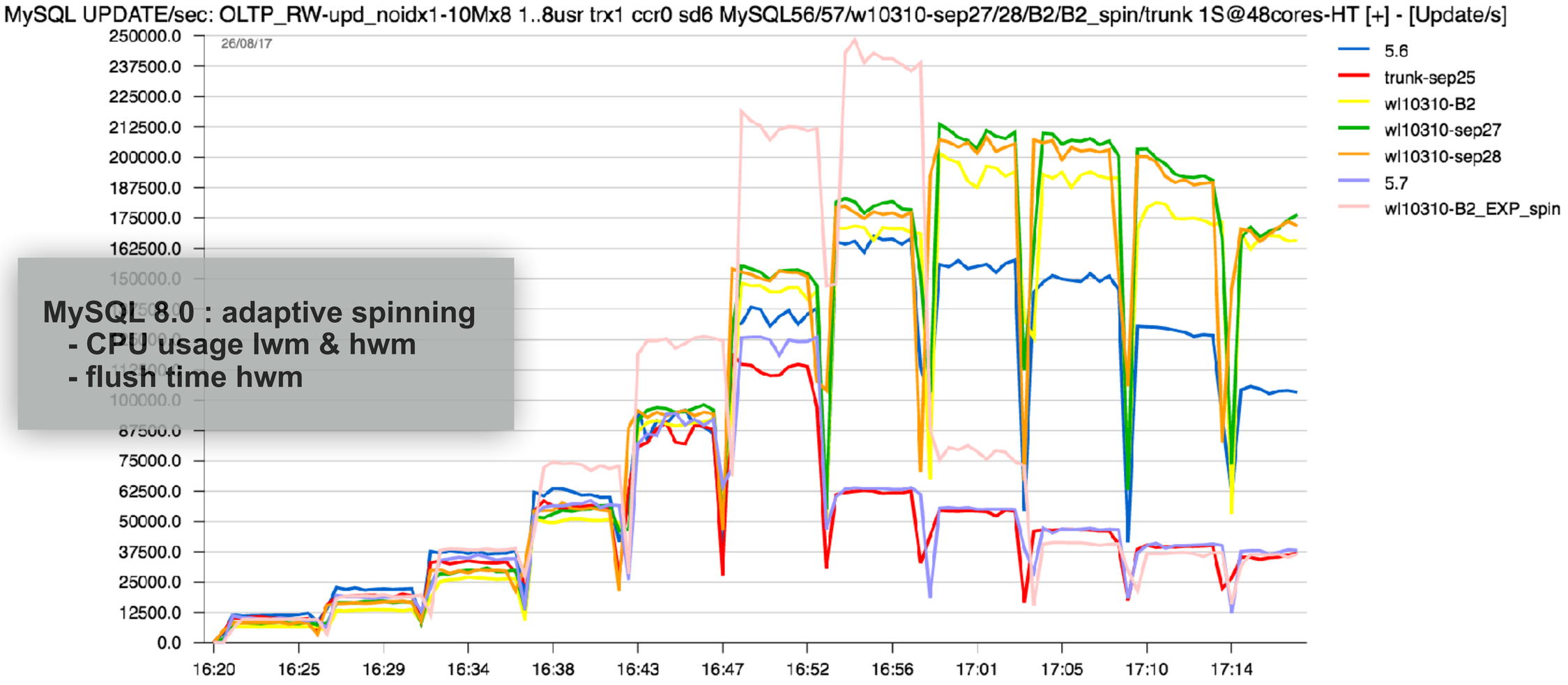
MySQL 8.0 : Re-Designed REDO

- New design tradeoffs…
  - 1 user / low load => event-driven is slower
  - option : spinning on wait
  - option : low/ high/ mixed oriented



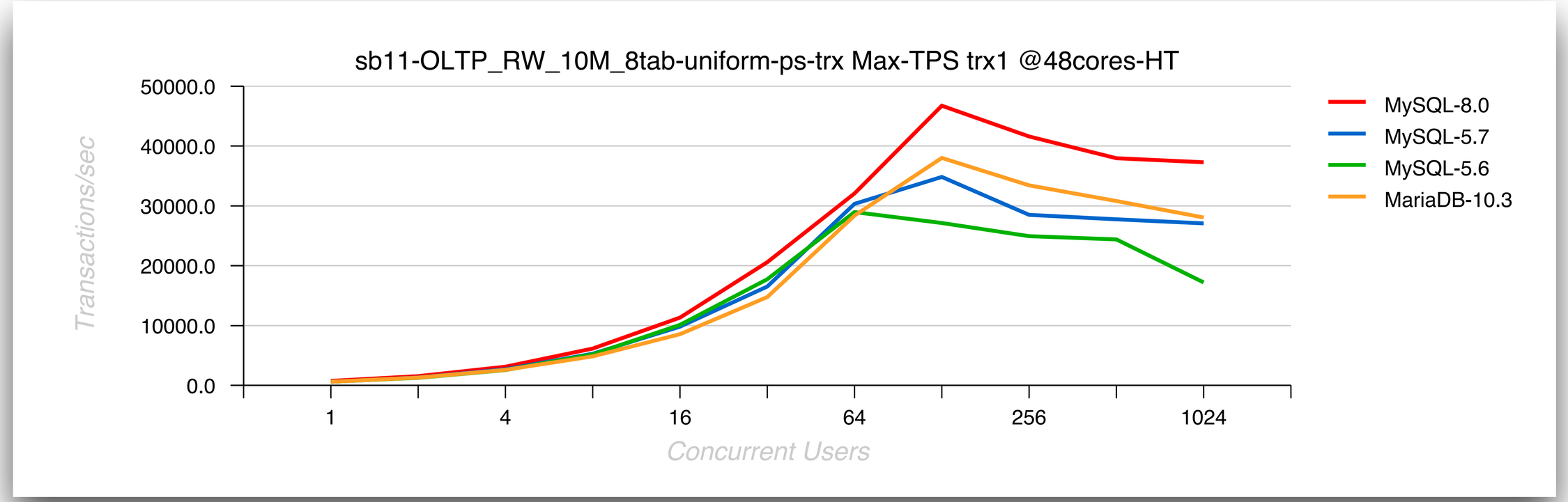
MySQL 8.0 : Re-Designed REDO

- New design tradeoffs…



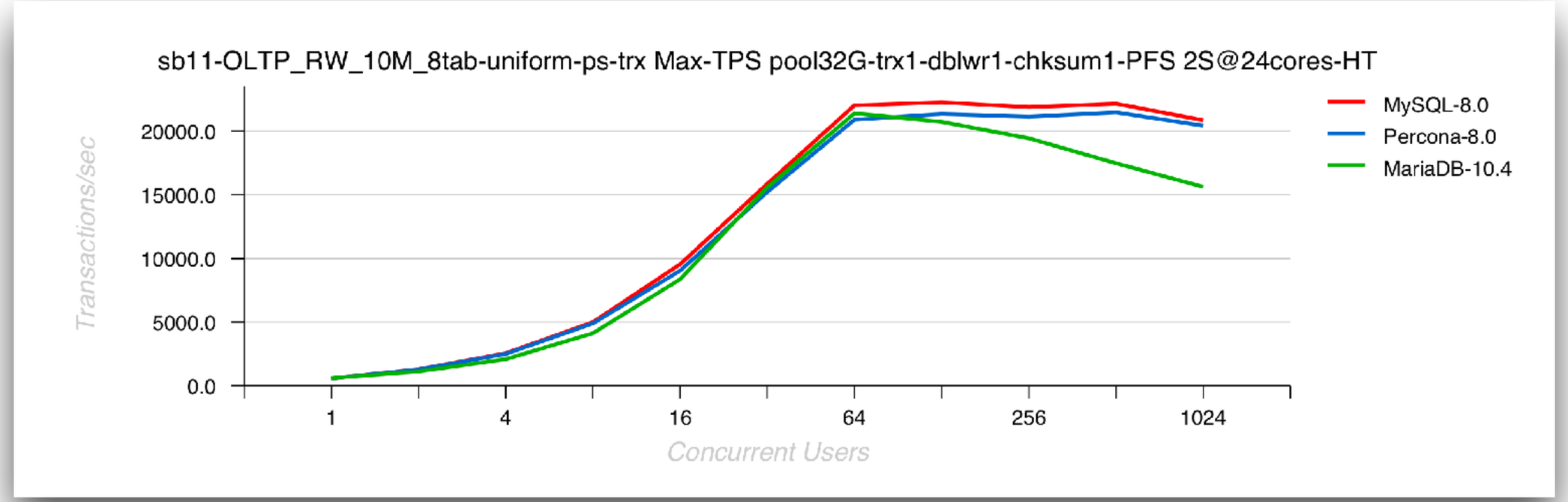
OLTP_RW latin1 @MySQL 8.0 (Apr.2018)

- **45K (!!) TPS** Sysbench OLTP_RW 10Mx8tab, trx_commit=1, 2S Skylake
  - **30**% gain vs MySQL 5.7
  - **50**% gain vs MySQL 5.6



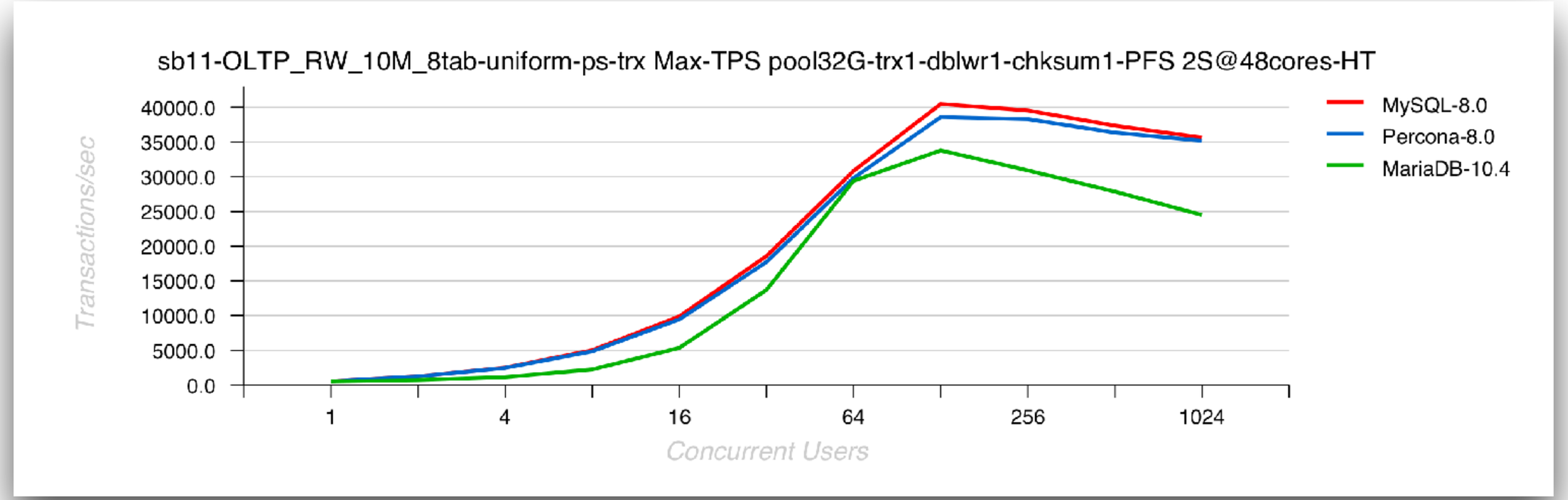
OLTP_RW latin1 @MySQL 8.0 (Sep.2019)

- Sysbench OLTP_RW 10Mx8tab, trx_commit=1
  - 2S Dell 24cores-HT 2.6 Ghz



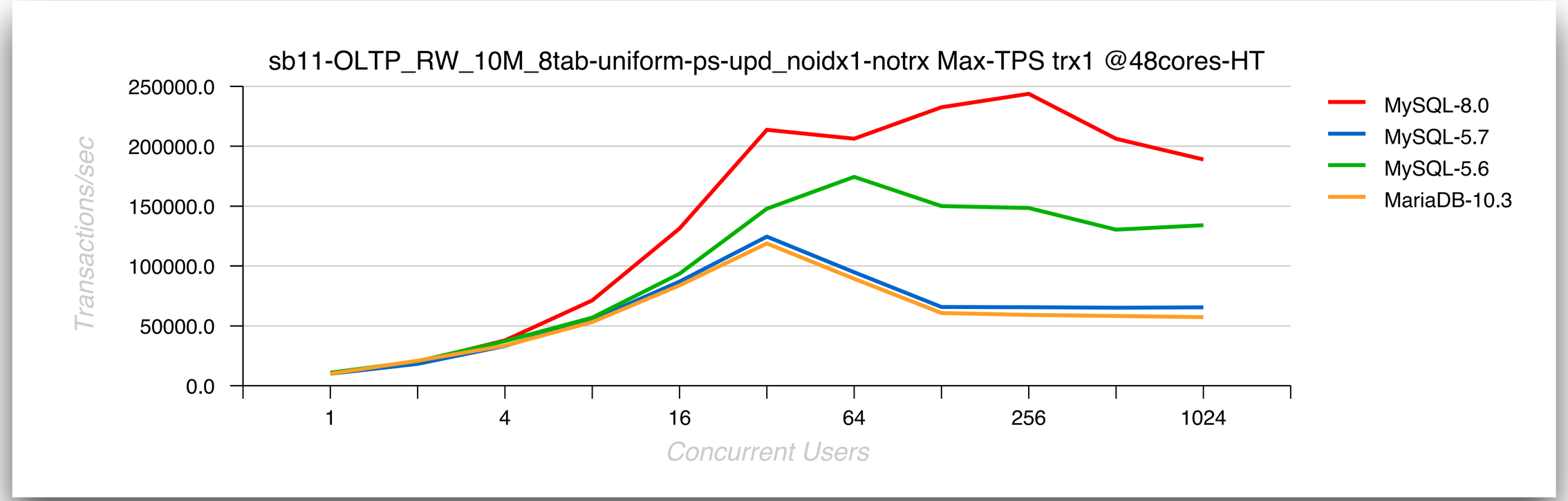
OLTP_RW latin1 @MySQL 8.0 (Sep.2019)

- Sysbench OLTP_RW 10Mx8tab, trx_commit=1
  - 2S Intel 48cores-HT Cascade Lake 2.9 Ghz



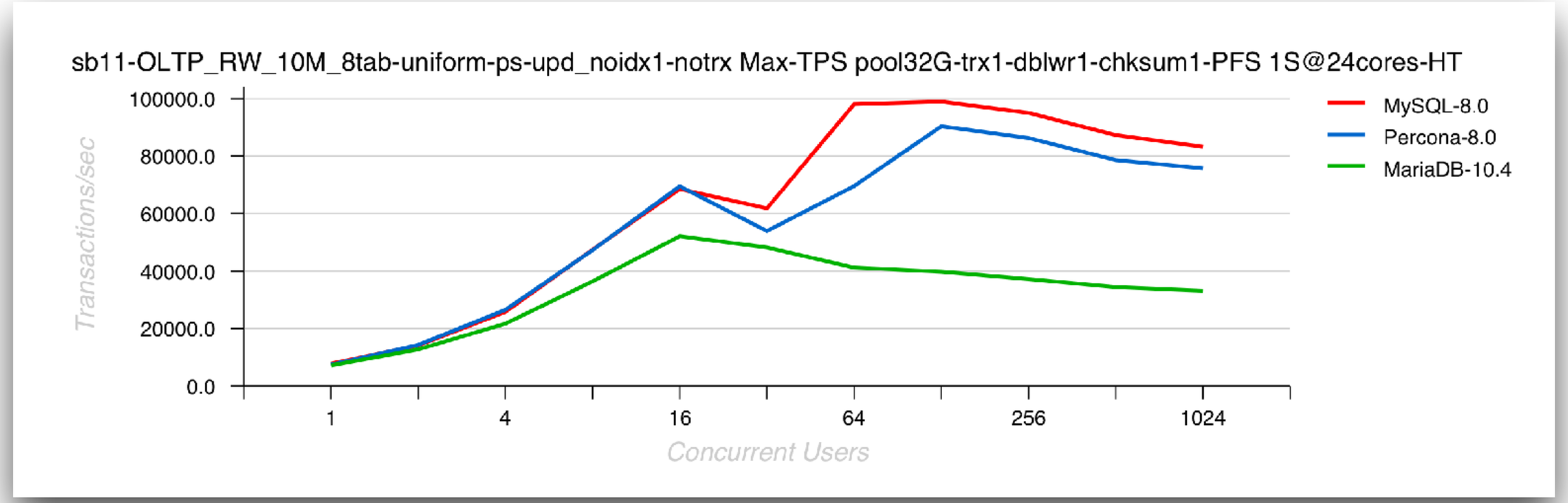
Updates-NoKEY latin1 @MySQL 8.0 (Apr.2018)

- **250K (!!) QPS** Sysbench Update-nokey 10Mx8tab, trx_commit=1, Skylake
  - **100**% gain vs MySQL 5.7
  - **50**% gain vs MySQL 5.6 (and yes, 5.7 is bad here.. => fixed !! ;-))
  - and clearly seen MariaDB's adoption of InnoDB 5.7..



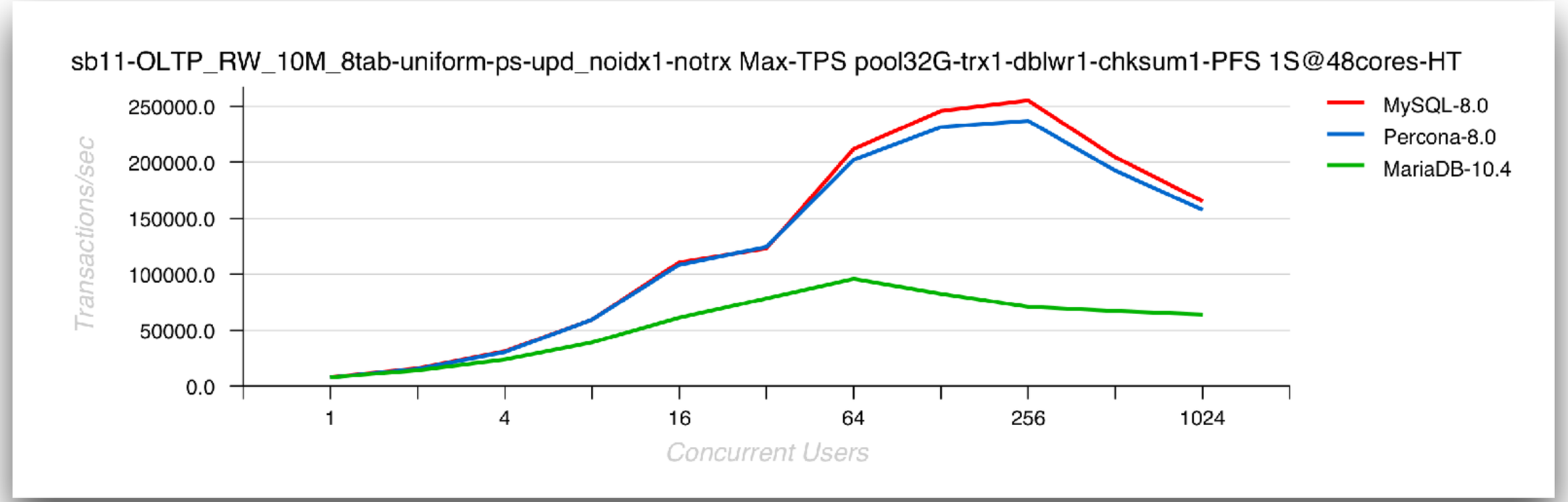
Updates-NoKEY latin1 @MySQL 8.0 (Sep.2019)

- Sysbench Update-nokey 10Mx8tab, trx_commit=1
  - 2S Dell 24cores-HT 2.6 Ghz



Updates-NoKEY latin1 @MySQL 8.0 (Sep.2019)

- Sysbench Update-nokey 10Mx8tab, trx_commit=1
  - 2S Intel 48cores-HT Cascade Lake 2.9 Ghz



MySQL 8.0 Writes Scalability

- IMPORTANT :
  - MySQL 8.0 overall WRITE performance is way better comparing to all we have before !
  - but : we're still NOT scaling !..
- Going from 1S => 2S (CPU Sockets) :
  - OLTP_RW : somewhat 50% better TPS only, and it's due RO scaling..
  - Update-NoKEY : just same or worse TPS..
- Why ?
  - 1) next-level bottlenecks (TRX / LOCK Management)
  - 2) + something else (yet to discover)..
  - so, still a lot of work ahead ;-))

New: MySQL Resource Groups

- What :
  - starting codebase for our future Resource Management solutions
  - flexible and proper thread / query isolation
  - dynamic, integrated, fun ! ;-))
- Why :
  - protect background threads, provide them optimal conditions for processing
  - run batches on low priority, OLTP on higher (and opposite on night)
  - isolate DDL orders from other activity
  - allow to move long running queries to low priority / isolate (live !! ;-))
  - apply particular execution conditions for any SQL query via Optimizer Hint
    - => Query Rewrite, ProxySQL, etc..
  - automatically assign RG to users / databases / workloads via ProxySQL
  - potential workaround for many CPU cache related issues
  - **huge opportunity to all kind of new tools !!!**

MySQL Resource Groups

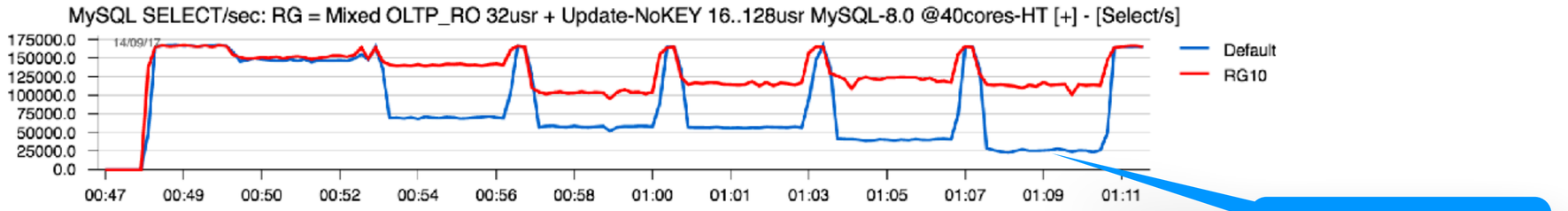
- Implementation Details :
  - currently : USER and SYSTEM groups
  - attributes : CPU (vcpu) affinity & thread priority
  - **thread priority** :
    - SYSTEM : [-19, 0] normal or **higher**
    - USER : [0, 20] normal or **lower**
- Admin :
  - permissions : none / can use / can use + admin
  - mysql> create RESOURCE GROUP RG10 type=user vcpu=0-9,40-49 thread_priority=0 ;
  - mysql> alter … ; drop … ; (also DISABLE / ENABLE / etc..)
- Using : only by name !
  - mysql> SET RESOURCE GROUP name ; (also for any THREAD ID)
  - SELECT /*+ RESOURCE_GROUP(name) */ … ; (query hint)

MySQL Resource Groups in Action

- Test case :
  - 40cores-HT 4S (Broadwell) server, OL7
  - 32 concurrent users are running SELECTs (Sysbench OLTP_RO)
  - other users are coming with UPDATEs (Sysbench Update-NoKEY)
    - 16 users, then 32, 64, 128, 256, 512
  - **Problem :** each workload is running well alone, but NOT together (yet)..
  - **Workaround :**
    - UPDATEs are not scaling and mixed with SELECTs creating yet more contentions
    - let's limit UPDATE queries to 10cores-HT only
    - mysql> create RESOURCE GROUP RG10 type=user vcpu=0-9,40-49 thread_priority=0 ;
    - and add a hint to UPDATE queries :
      UPDATE /*+ RESOURCE_GROUP(RG10) */ … ;

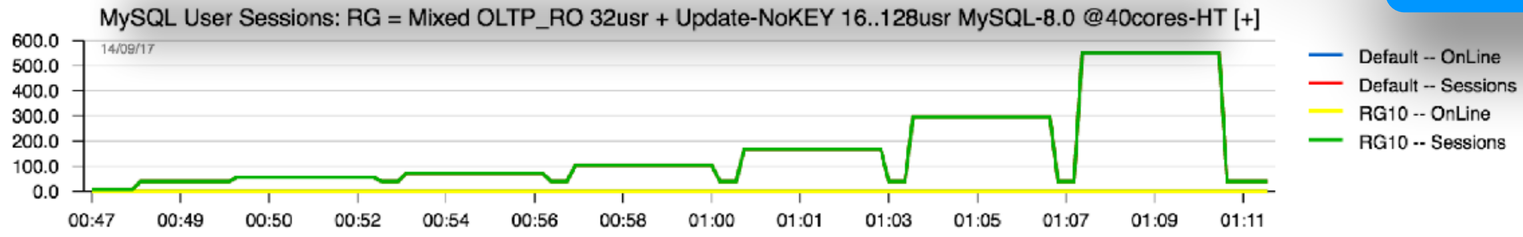
MySQL Resource Groups in Action

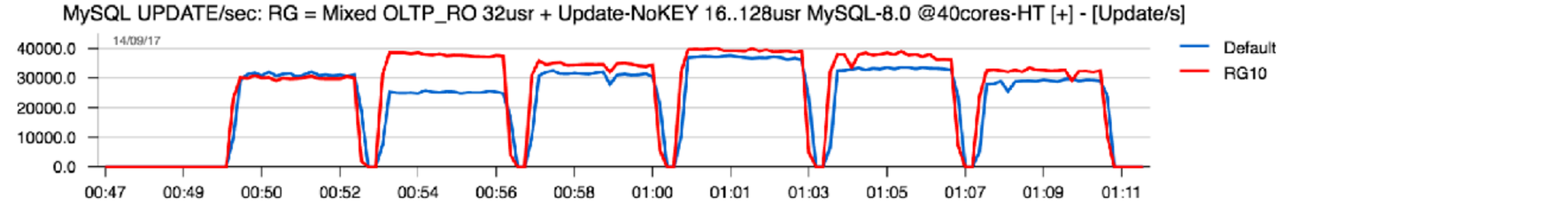
- Test case :
  - 32 concurrent users are running SELECTs
  - other users are coming with UPDATEs : 16 users, then 32, 64, 128, 256, 512..



none | 16usr | 32usr | 64usr | 128usr | 256usr | 512usr

Default

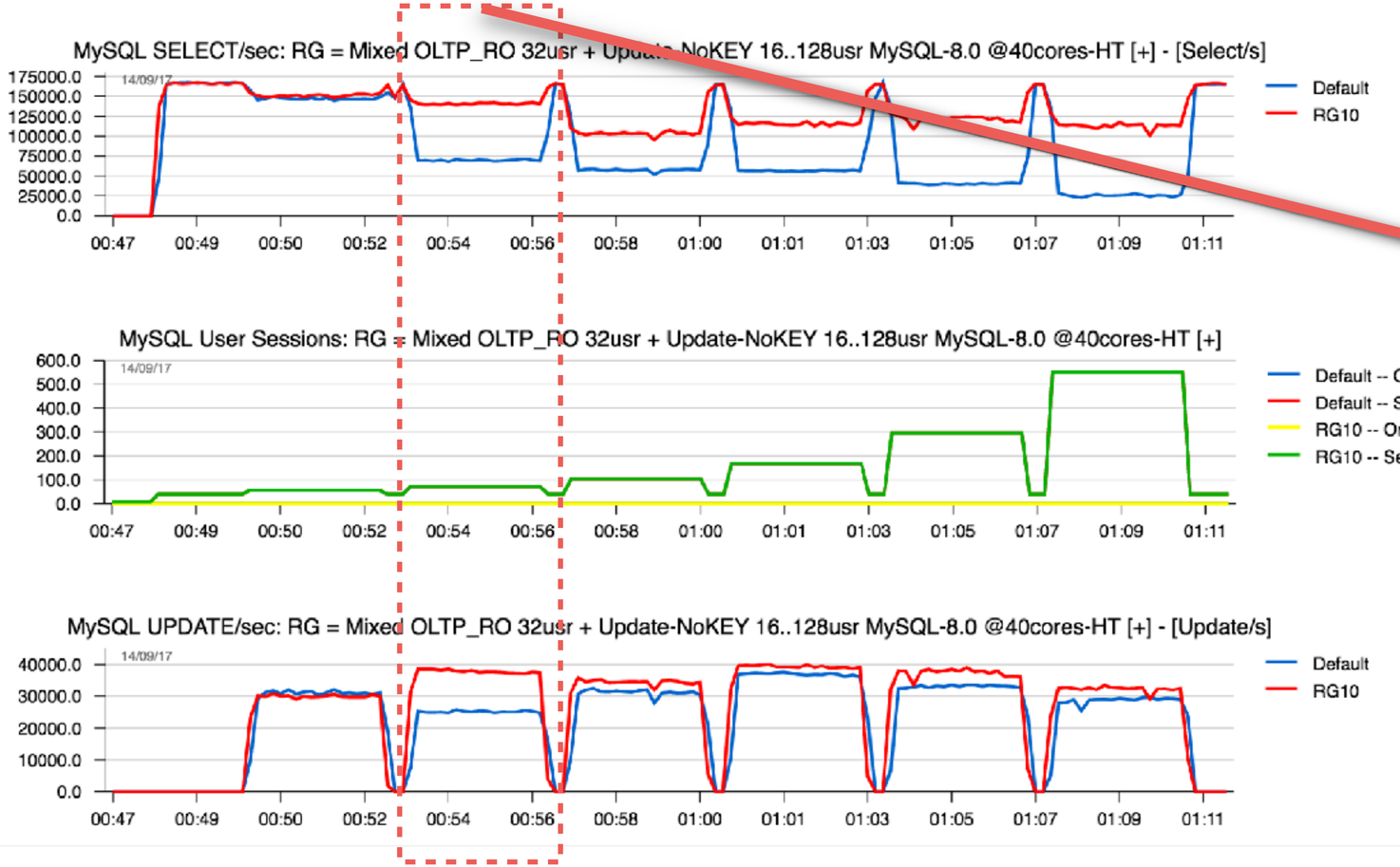


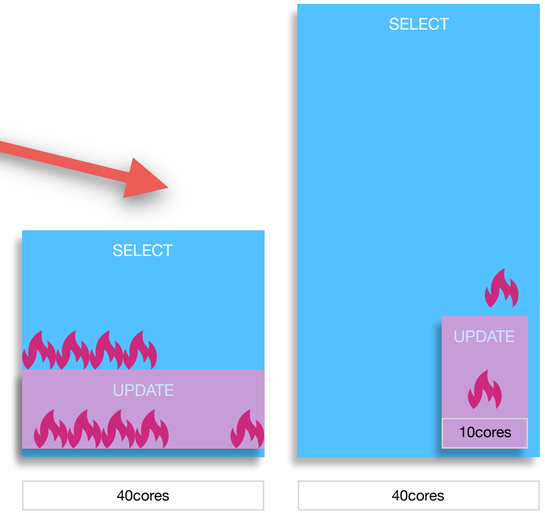


ORACLE

MySQL Resource Groups in Action

- Test case :
  - 32 concurrent users are running SELECTs
  - other users are coming with UPDATEs : 16 users, then 32, 64, 128, 256, 512..





Please, attention ! the HOT stuff is coming ;-))

Pending Scalability Issues after MySQL 5.7 GA..

- RO :
  - Block Locks
  - Lookups via Sec.IDX
  - UTF8
  - Global lock on every IO read

Pending Scalability Issues after MySQL 5.7 GA..

- RO :
  - Block Locks <= workaround : ProxySQL Query Cache
  - Lookups via Sec.IDX <= possible workaround : use PK, AHI
  - UTF8 <= **use 8.0** ;-)
  - Global lock on every IO read <= **use 8.0** ;-)

Pending Scalability Issues after MySQL 5.7 GA..

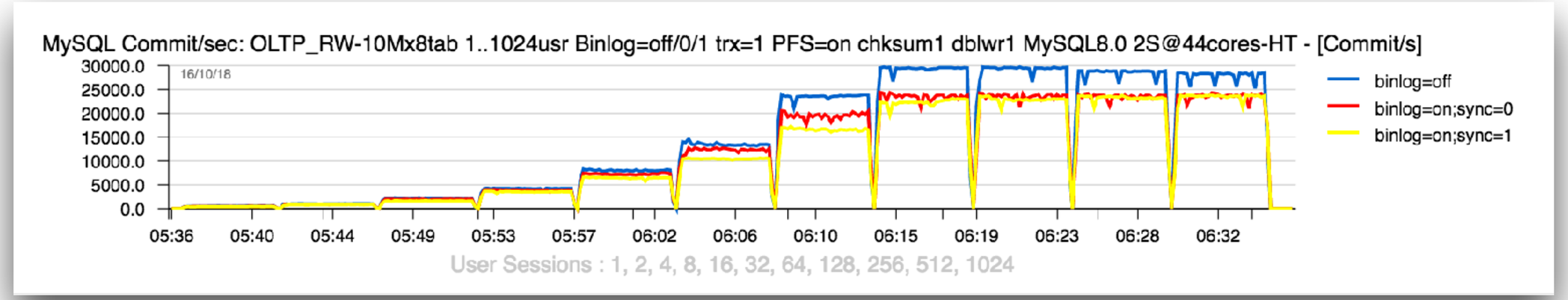
- RO :
  - Block Locks <= workaround : ProxySQL Query Cache
  - Lookups via Sec.IDX <= possible workaround : use PK, AHI
  - UTF8 <= **use 8.0** ;-)
  - Global lock on every IO read <= **use 8.0** ;-)
- RW :
  - REDO log related bottlenecks..
  - Double Write..
  - TRX management contentions..
  - LOCK management..
  - RR / RC isolation..
  - UPDATE Performance..
  - INSERT Performance..
  - Purge lagging..

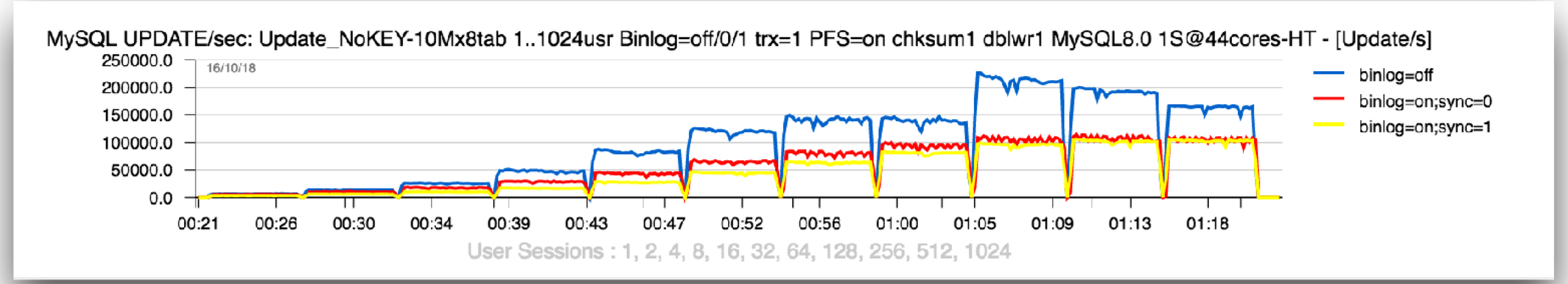
Pending Scalability Issues after MySQL 5.7 GA..

- RO :
  - Block Locks <= workaround : ProxySQL Query Cache
  - Lookups via Sec.IDX <= possible workaround : use PK, AHI
  - UTF8 <= **use 8.0** ;-)
  - Global lock on every IO read <= **use 8.0** ;-)
- RW :
  - REDO log related bottlenecks.. <= **new REDO since 8.0 !!**
  - Double Write.. <= **coming soon with 8.0 update (!!)**
  - TRX management contentions.. <= work-in-progress, prototyped..
  - LOCK management.. <= work-in-progress, prototyped..
  - RR / RC isolation.. <= work-in-progress, prototyped..
  - UPDATE Performance.. <= **use 8.0, and stay tuned ;-)**
  - INSERT Performance.. <= possible workaround : use partitions
  - Purge lagging.. <= not yet solved, but you can truncate UNDO

“Binlog sucks!” (c)

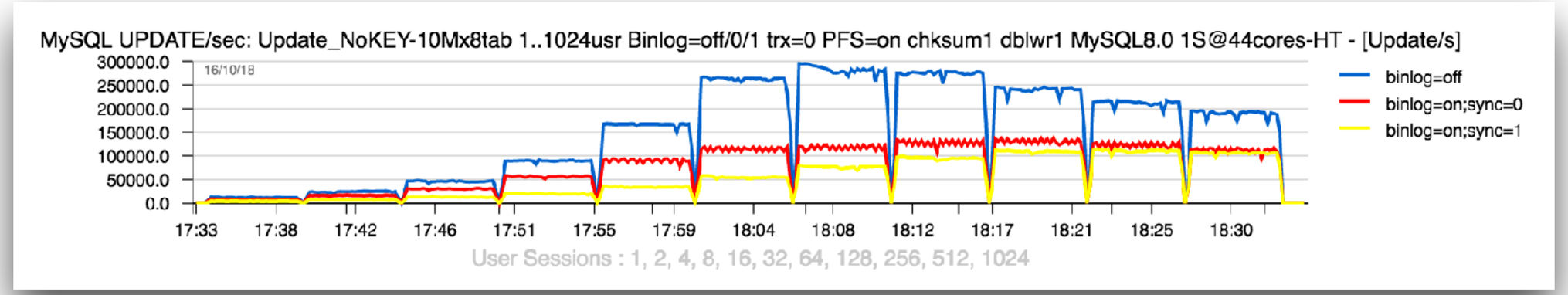
- OLTP_RW / Update_NoKEY 10Mx8tab, BP=32G
  - “yes it sucks, and we know why..” (c)
  - workaround ? … 🤔 …

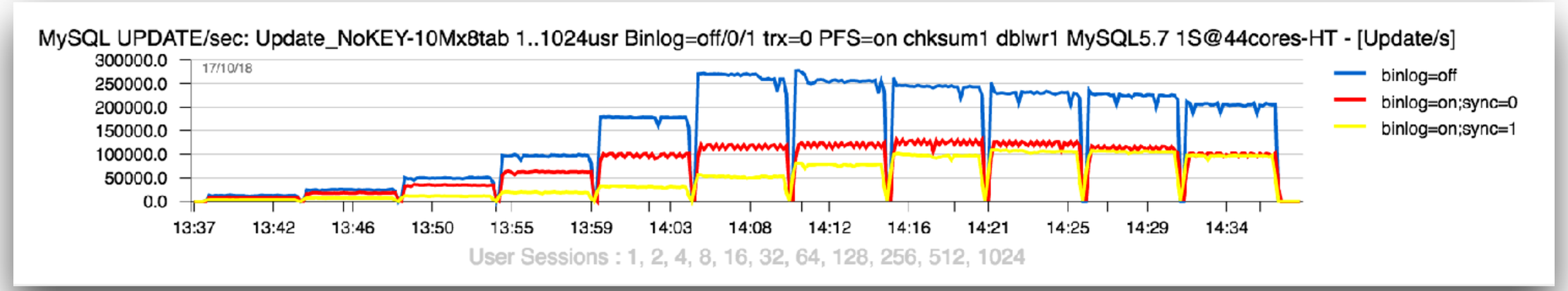




“Binlog sucks!” (c)

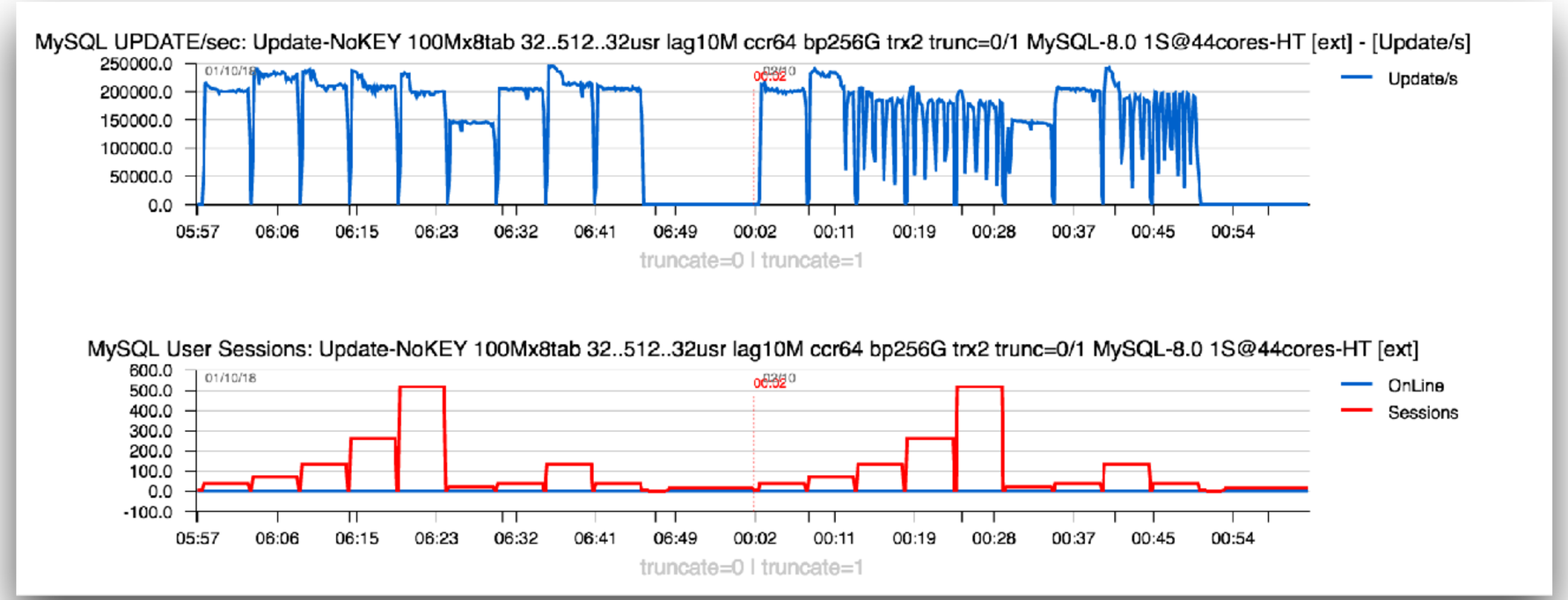
- Update_NoKEY 10Mx8tab, BP=32G
  - … and it’s not from today !
  - the same was already in MySQL 5.7 as well :





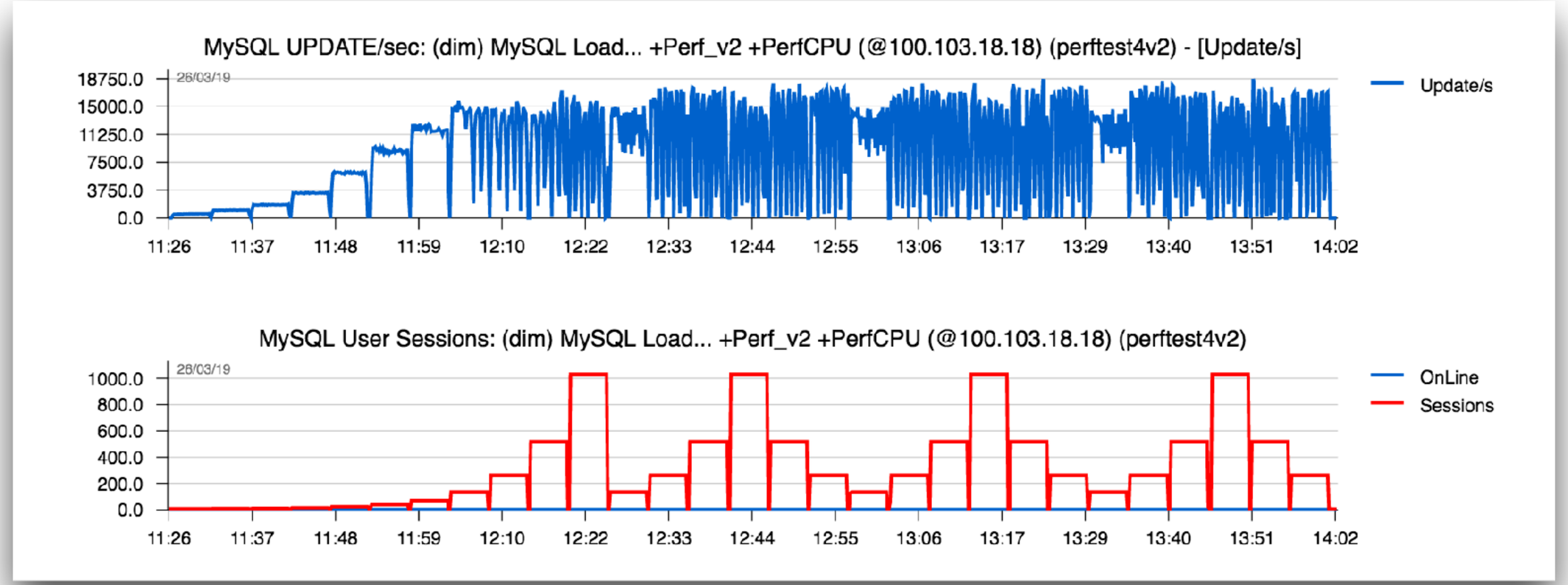
UNDO Auto-Truncate Impact

- Stalls on heavy RW activity
  - if you see it in your own workload => ping Sunny directly ! ;-))
  - workaround :
    - innodb_undo_log_truncate=off
    - and truncate manually when no user activity



Slow Storage

- Stalls on slow storage
  - REDO writes can go faster now than you write your DATA pages to disk..
  - fix : since MySQL 8.0.18 update !

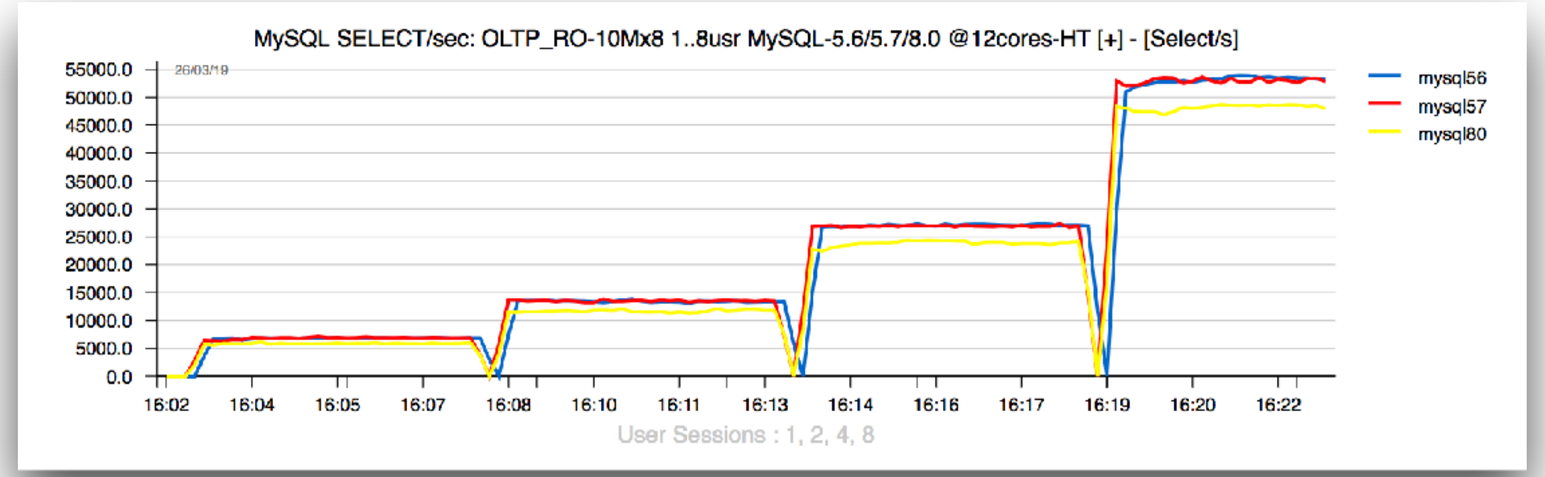


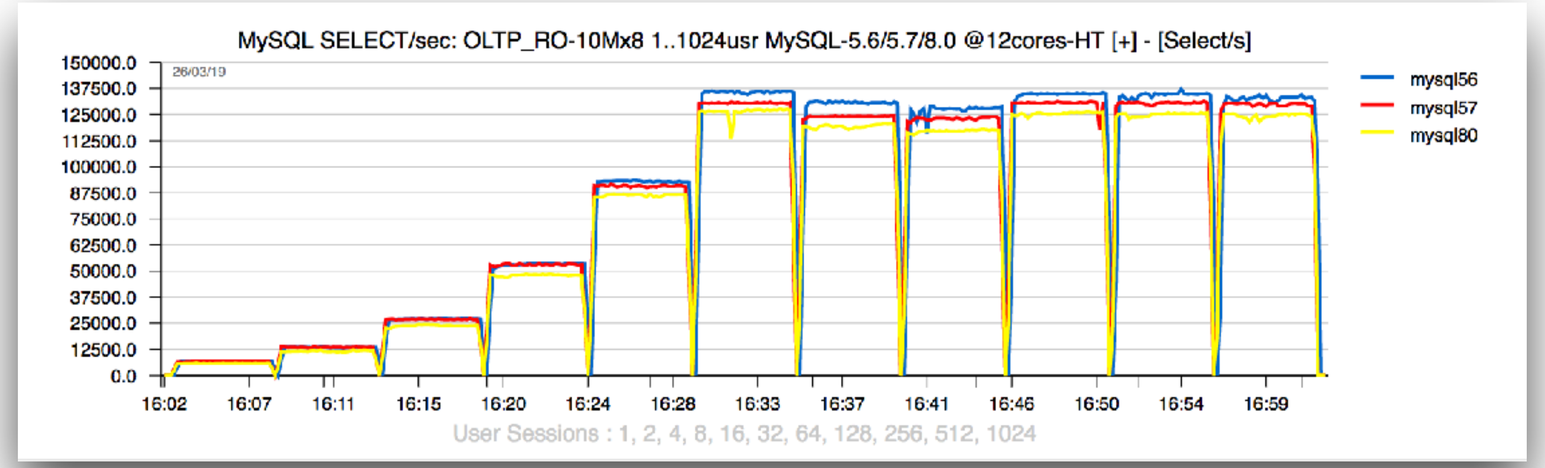
Small HW

- Unexpected Performance Regressions..
  - historically : single thread only (more new features => longer code path)
  - currently : **even on high load**..
- Looking closer :
  - Server : 12cores-HT Intel
  - Workloads (all in-memory) :
    - OLTP_RO
    - point-selects
    - OLTP_RW
    - Update_NoKEY

Small HW

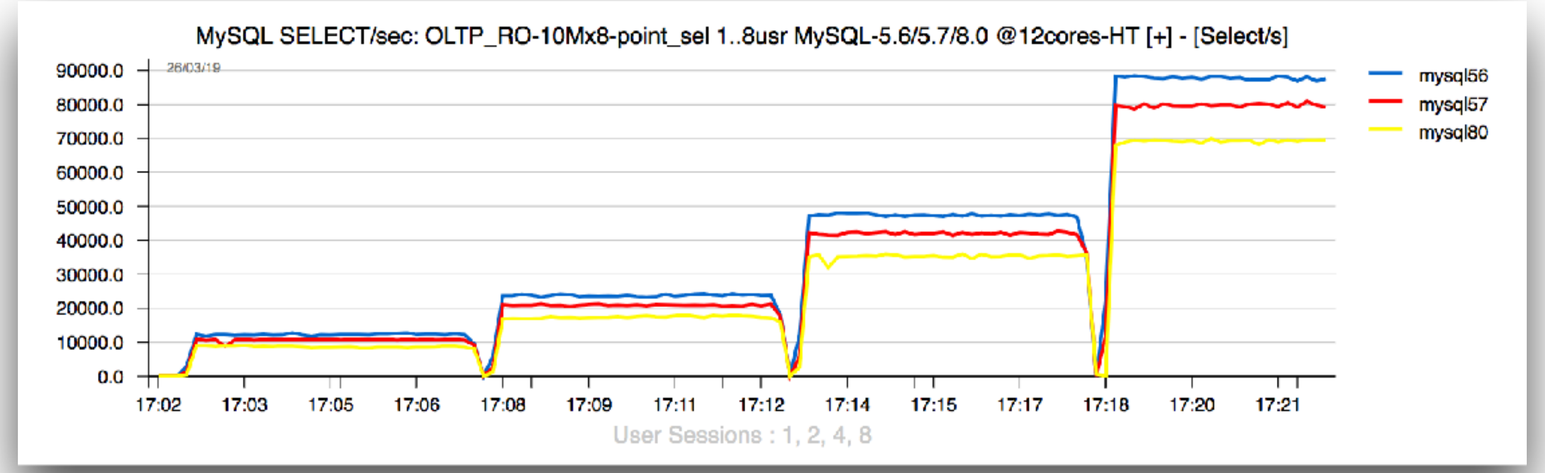
- OLTP_RO 10Mx8tab-uniform latin1 :

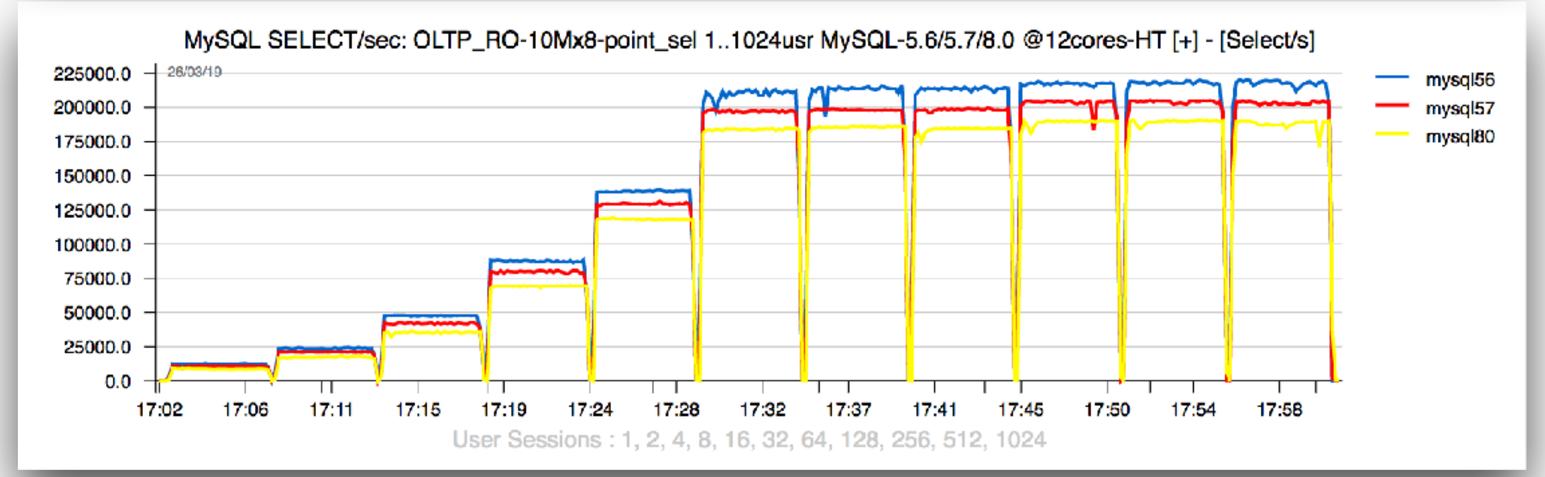




Small HW

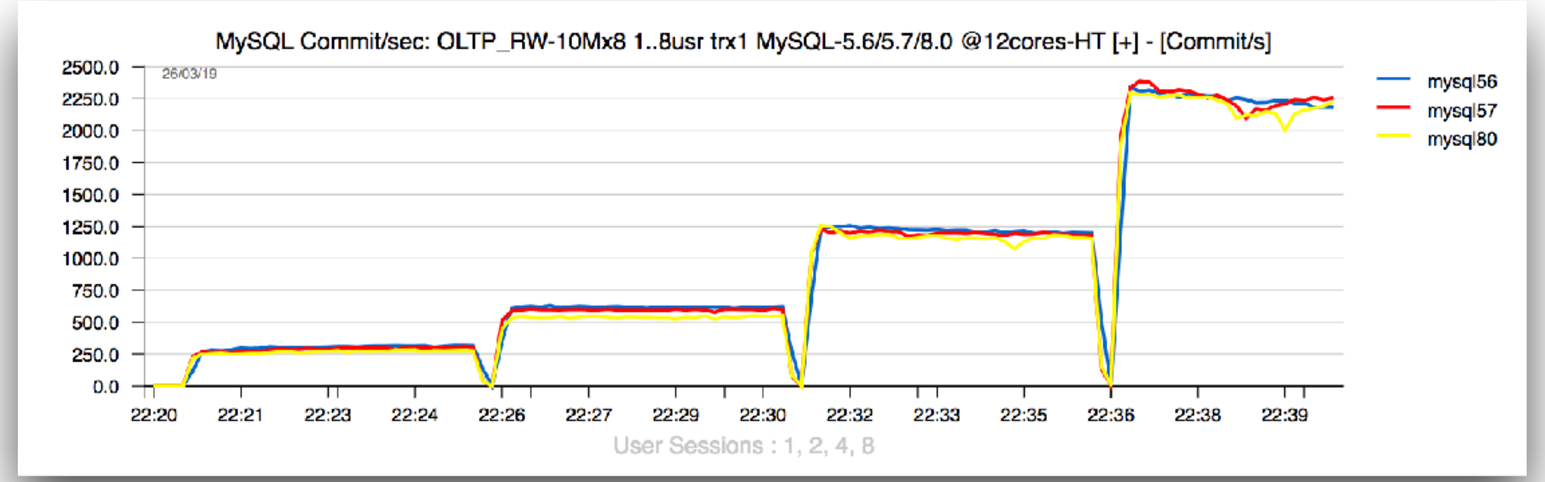
- OLTP_RO point-selects 10Mx8tab-uniform latin1 :

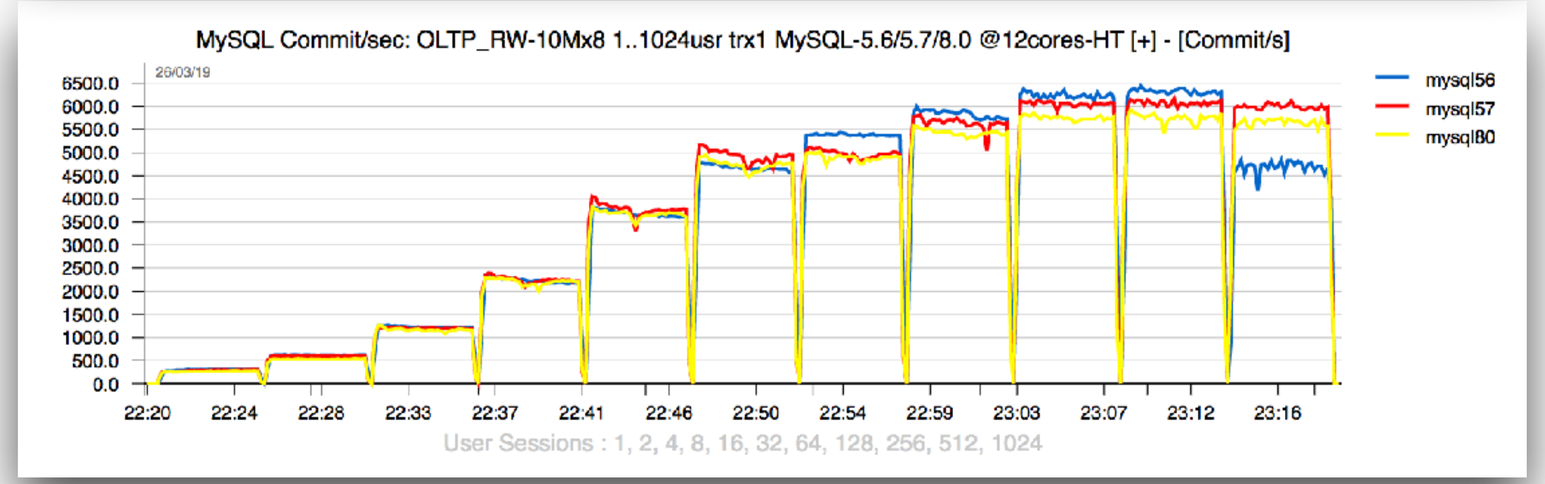




Small HW

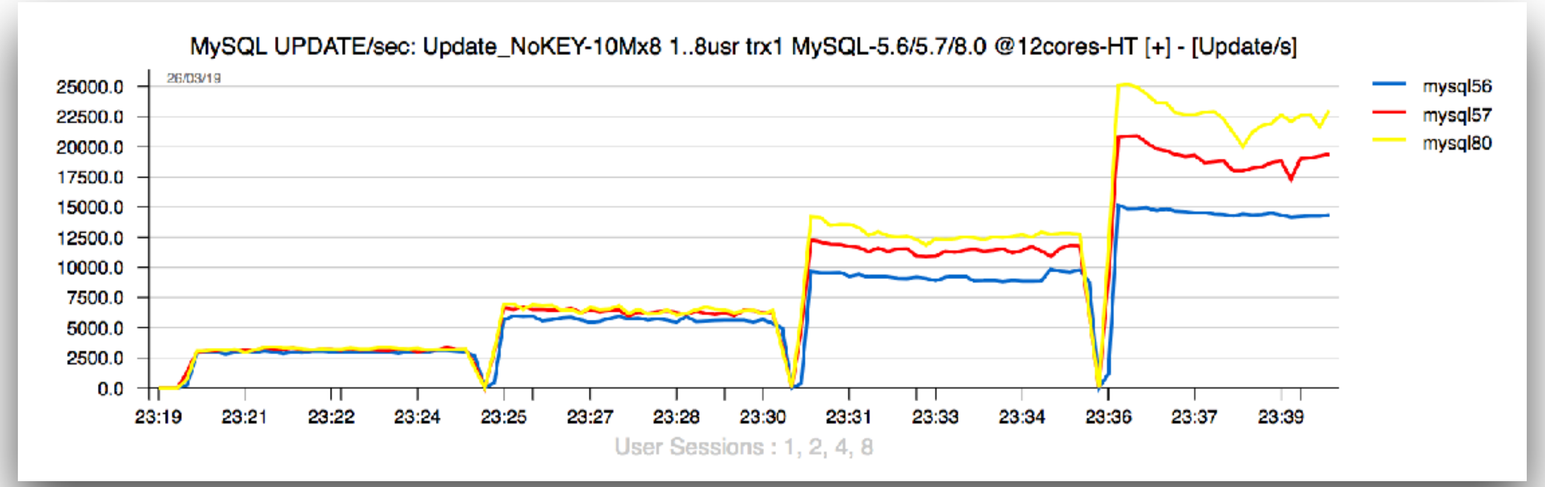
- OLTP_RW 10Mx8tab-uniform latin1 :

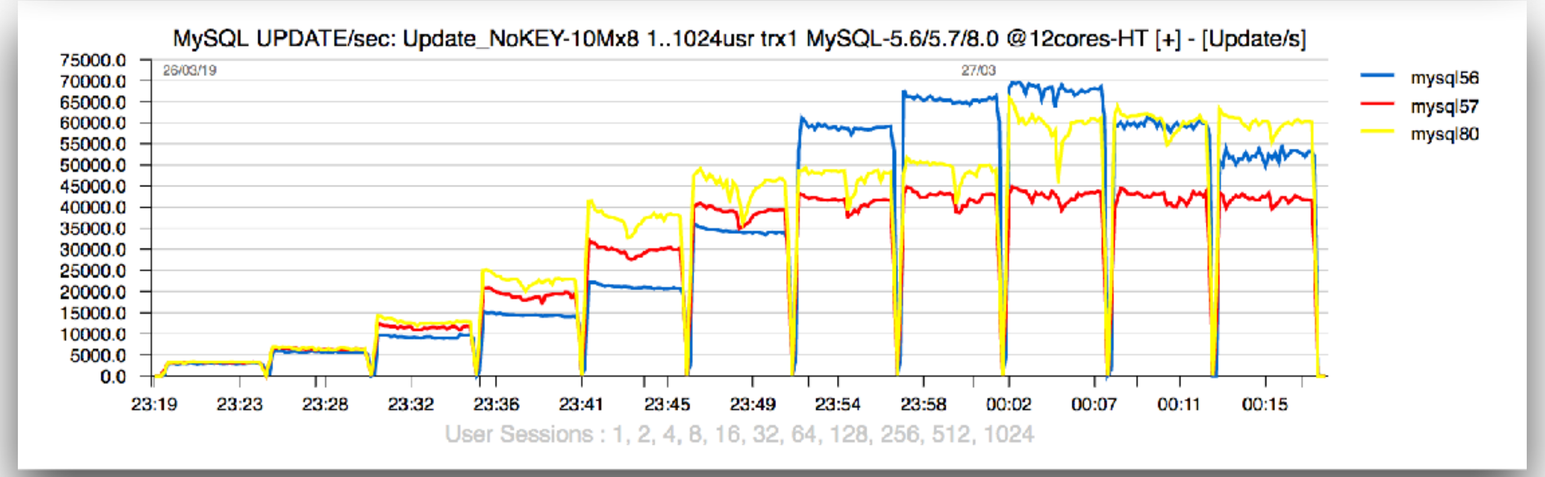




Small HW

- Update_NoKEY 10Mx8tab-uniform latin1 :



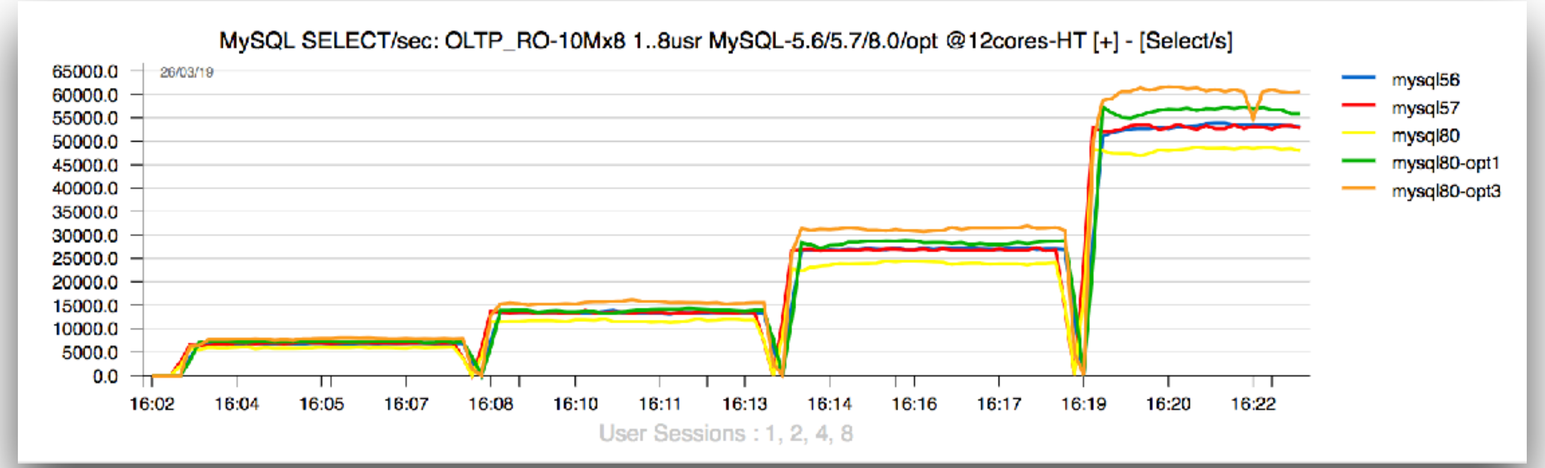


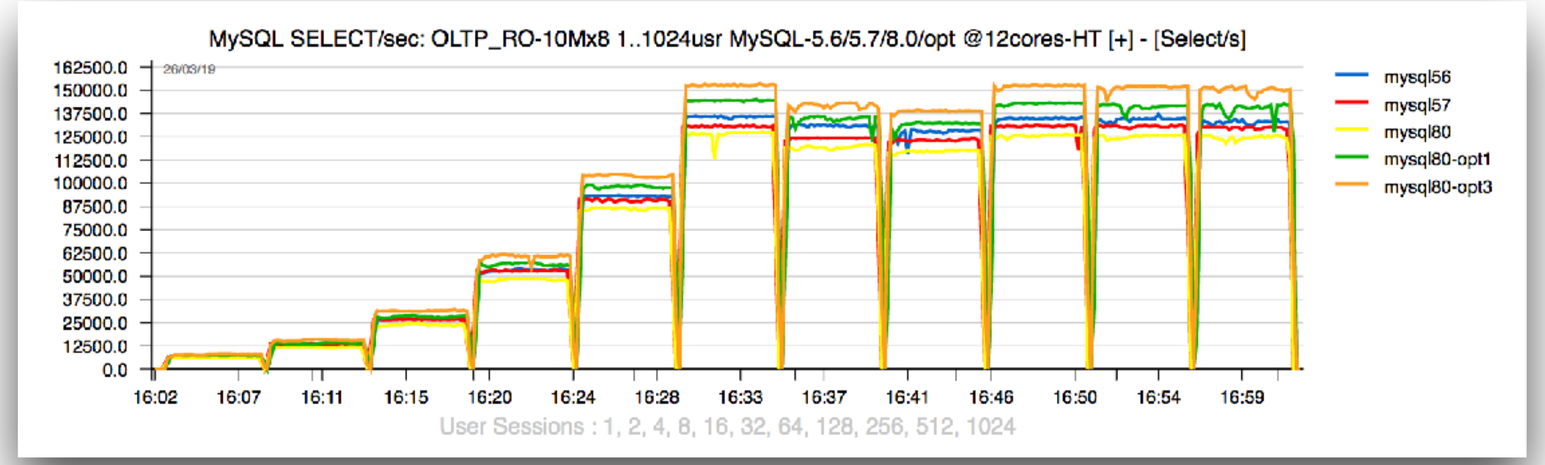
Small HW

- Unexpected Performance Regressions..
  - historically : single thread only (more new features => longer code path)
  - currently : **even on high load**..
- Looking closer :
  - Server : 12cores-HT Intel
  - Workloads (all in-memory) :
    - OLTP_RO
    - point-selects
    - OLTP_RW
    - Update_NoKEY
- So, **WHY** ???
  - the story is no more about longer code path..
  - the story is about bigger and bigger amount of instructions in binary (and icache miss) !
  - solution : optimize the MySQL 8.0 binary itself !
  - with strong collaboration with Oracle Linux Team ! (KUDOS Karsten !!! ;-))

Small HW

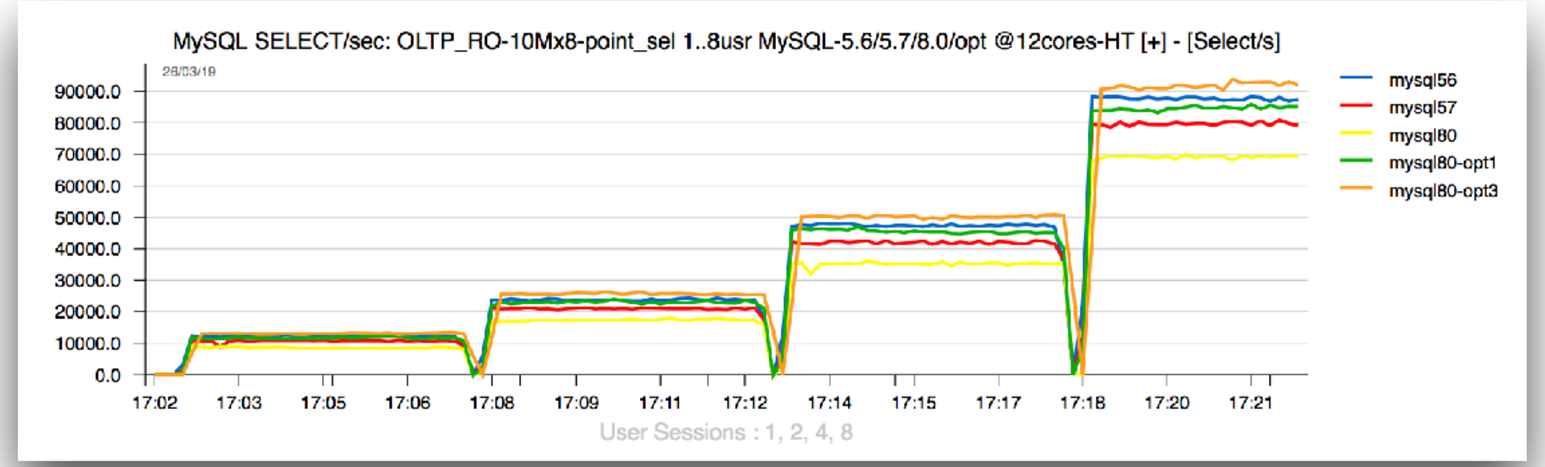
- OLTP_RO 10Mx8tab-uniform latin1 + optimized :

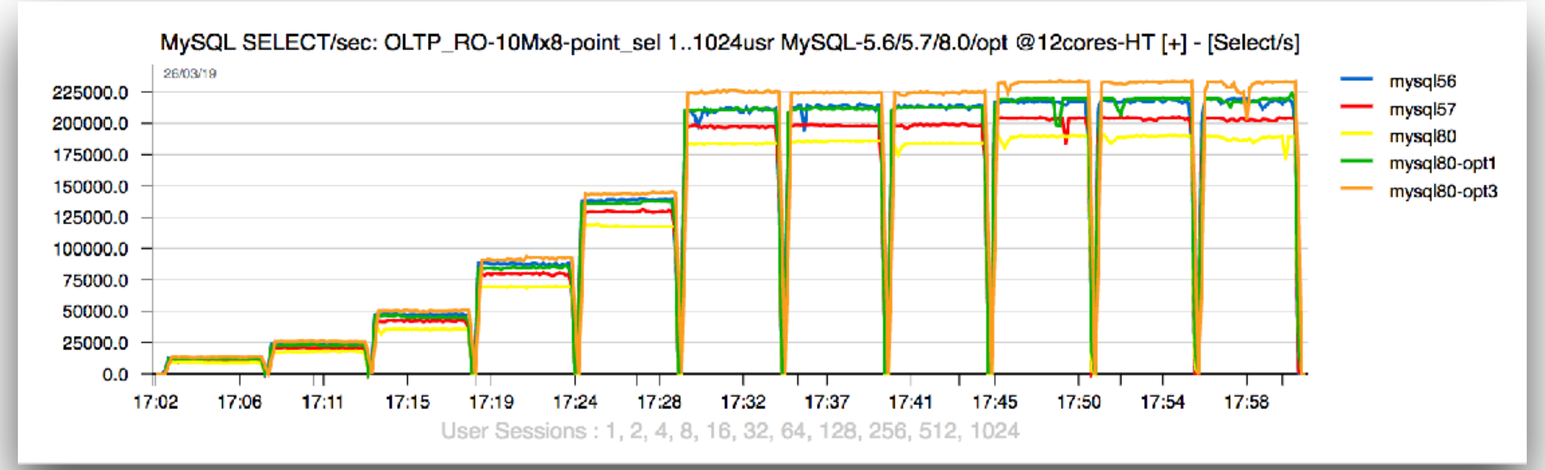




Small HW

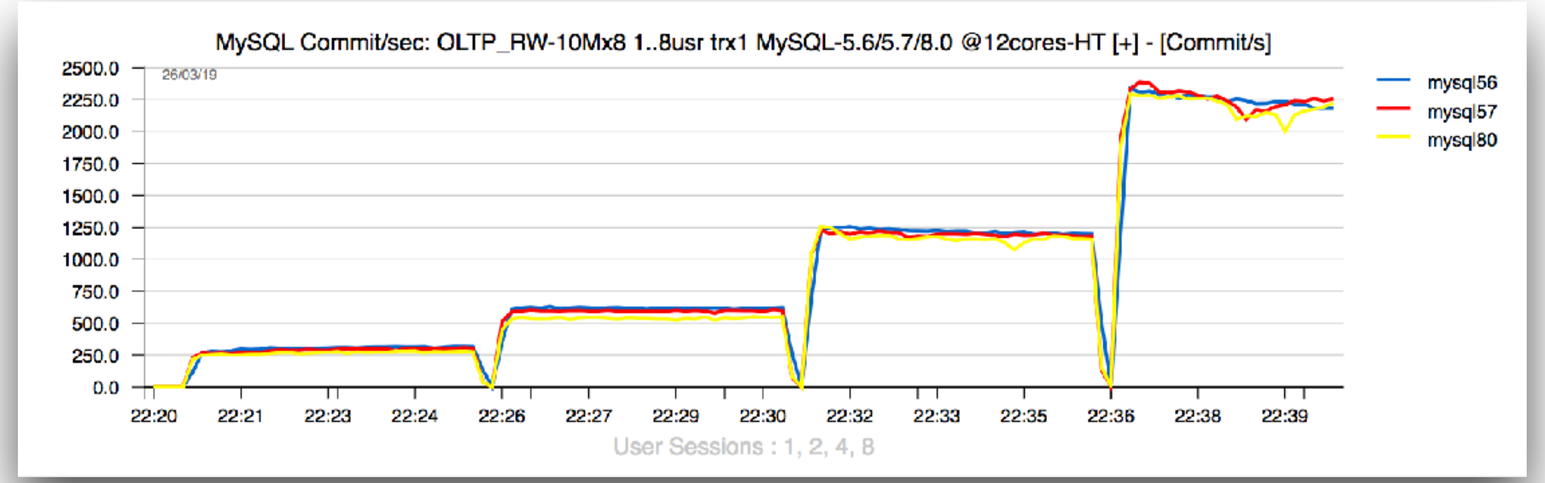
- OLTP_RO point-selects 10Mx8tab-uniform latin1 + optimized :

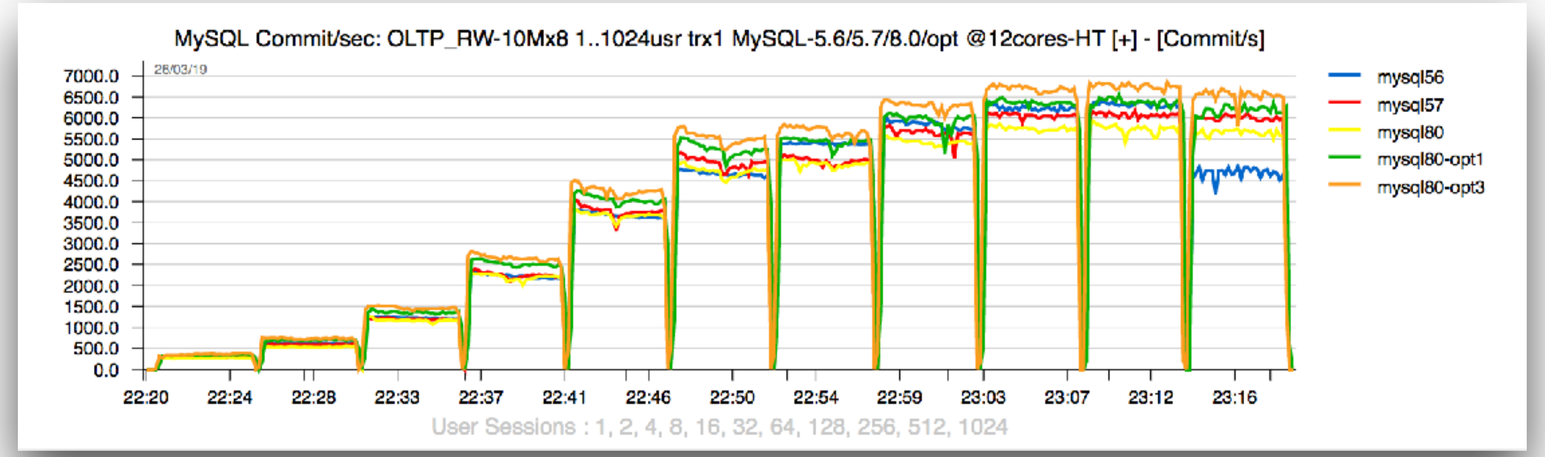




Small HW

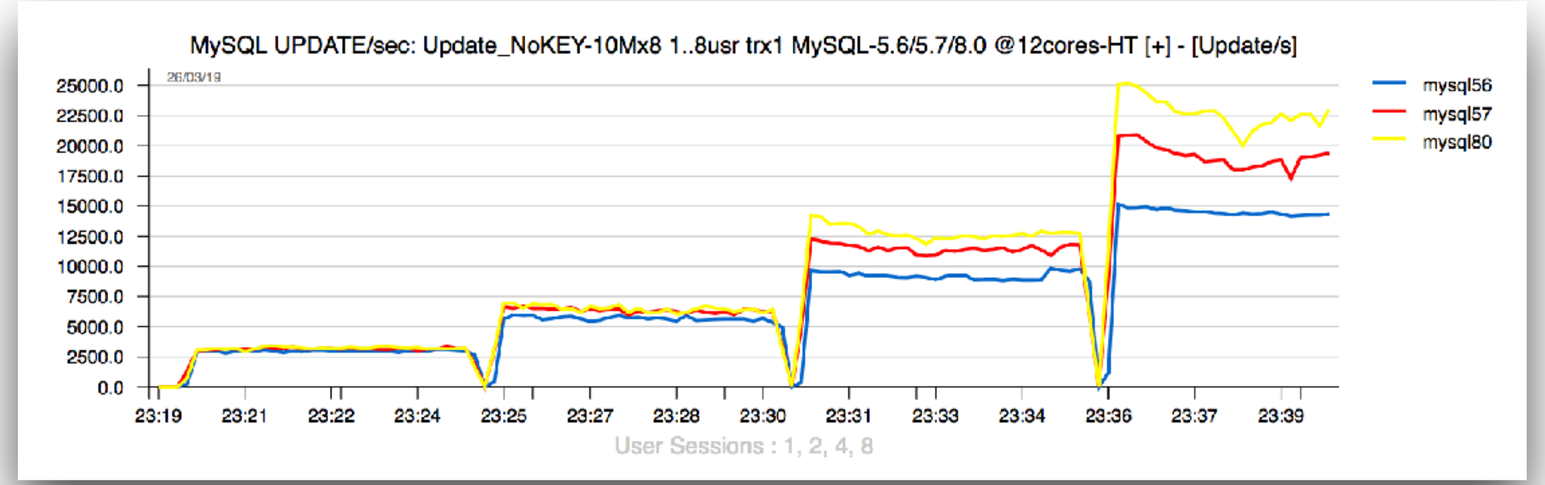
- OLTP_RW 10Mx8tab-uniform latin1 + optimized :

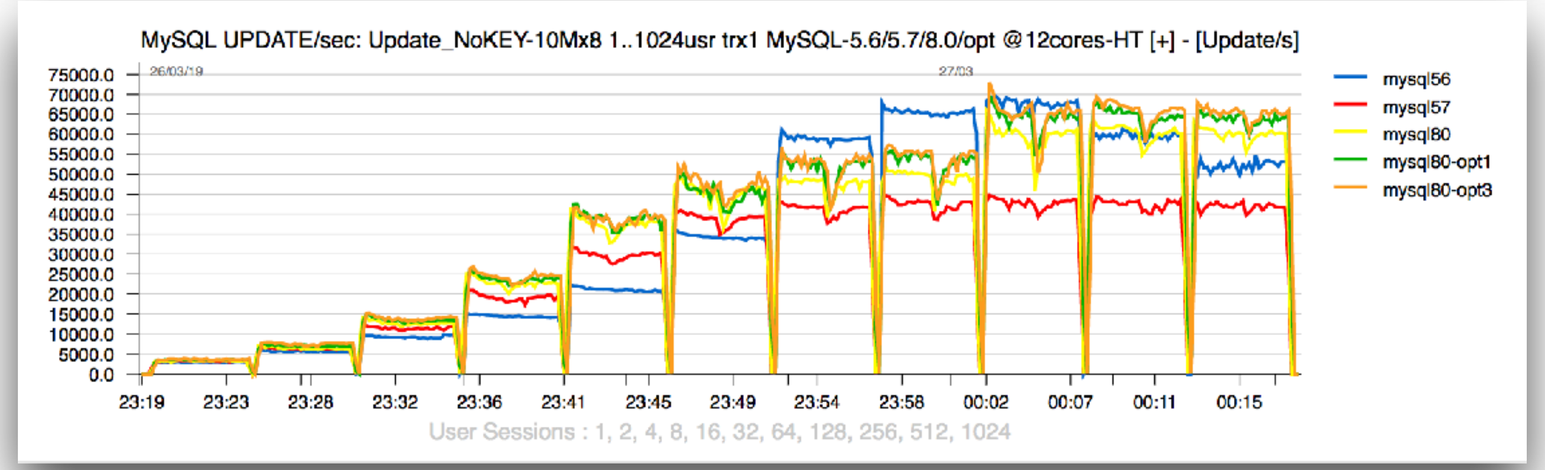




Small HW

- Update_NoKEY 10Mx8tab-uniform latin1 :





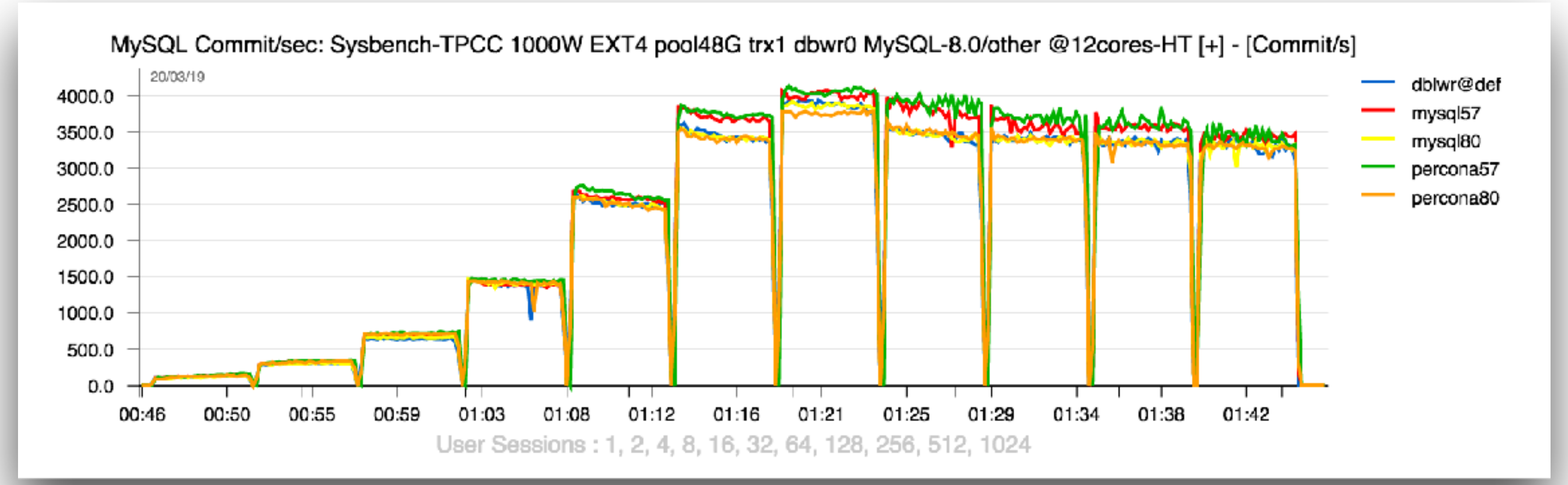
ORACLE

InnoDB Double-Write (DBLWR) Saga..

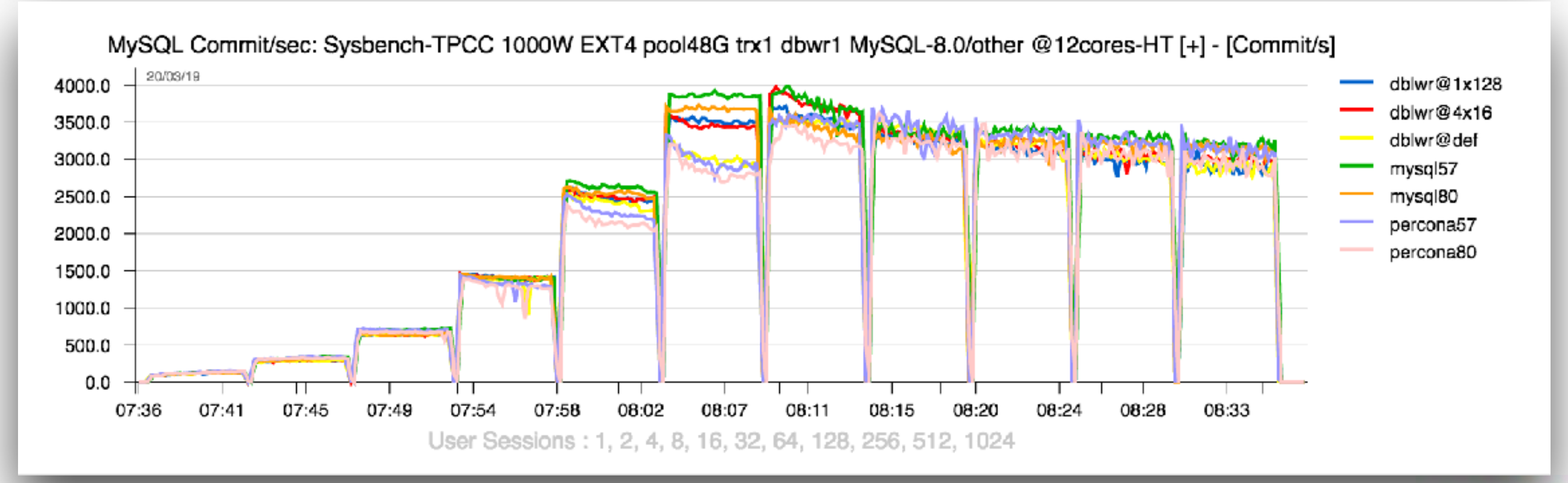
- DBLWR impact :
  - In-Memory Workloads : if page writes can follow => zero impact ! (background activity)
  - IO-bound Workloads :
    - NOTE : you have to Write before you can Read !
    - so, mostly no impact on low load
    - but huge impact on high load (due internal DBLWR contention)
- Percona
  - delivered their own new DBLWR for Percona-5.7
- MySQL
  - first new improved DBLWR was ready for MySQL 5.7, but missed GA timeframe..
  - MySQL 8.0 : continuous delivery !
  - Aug.2018 => new DBLWR is ready for MySQL 8.0 (and doing better than Percona)
  - Oct.2018 => final validations of new DBLWR on the latest HW + latest OL7 & kernel..
    - the surprises are starting..
    - …and investigation is still in progress ;-))

InnoDB Double-Write (DBLWR) Saga..

- DBLWR impact on IO-bound TPCC_1000W @12cores-HT :
  - EXT4 dblwr=0 :

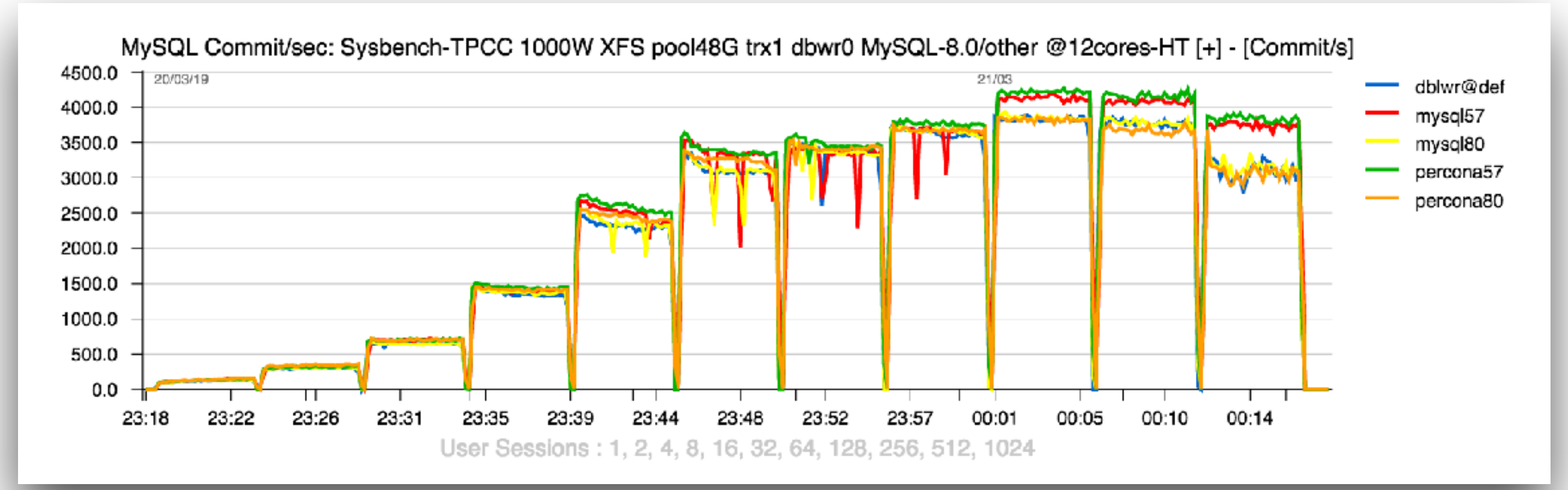


  - EXT4 dblwr=1 :

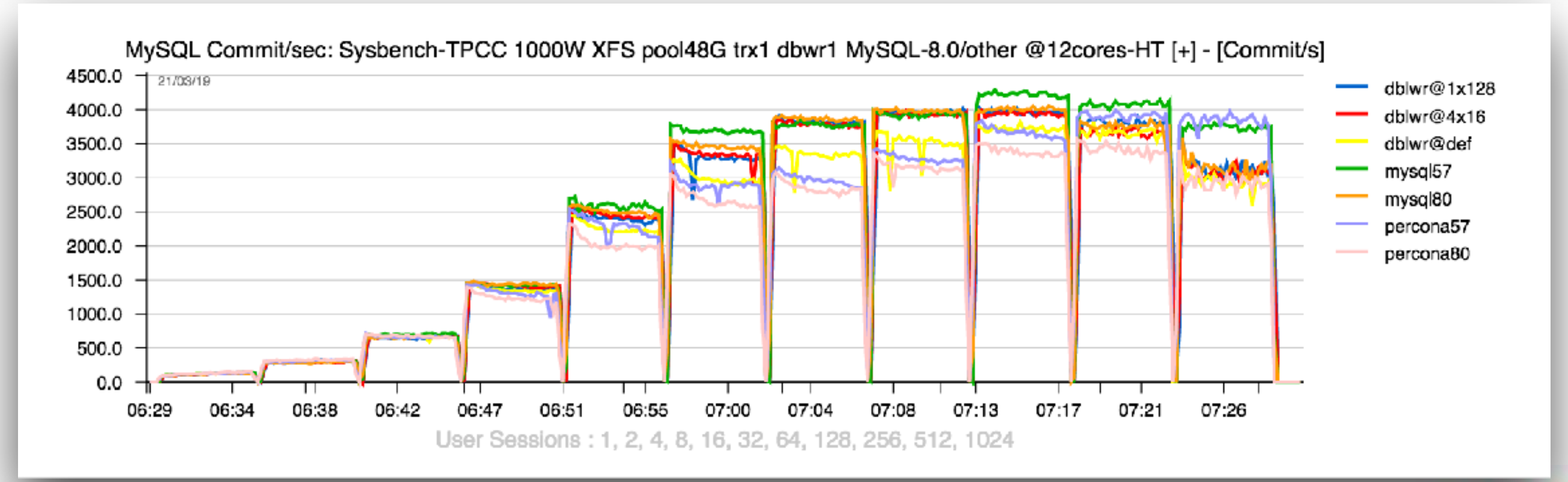


InnoDB Double-Write (DBLWR) Saga..

- DBLWR impact on IO-bound TPCC_1000W @12cores-HT :
  - XFS dblwr=0 :

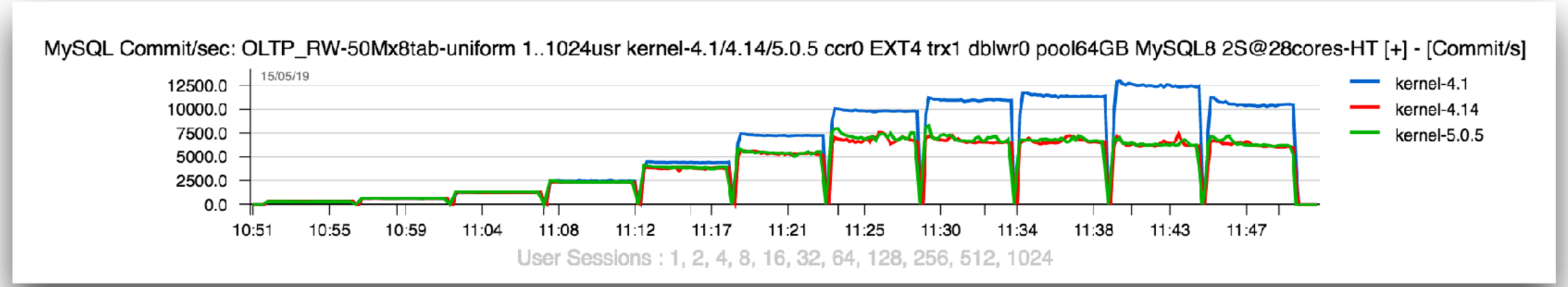


  - XFS dblwr=1 :



InnoDB Double-Write (DBLWR) Saga..

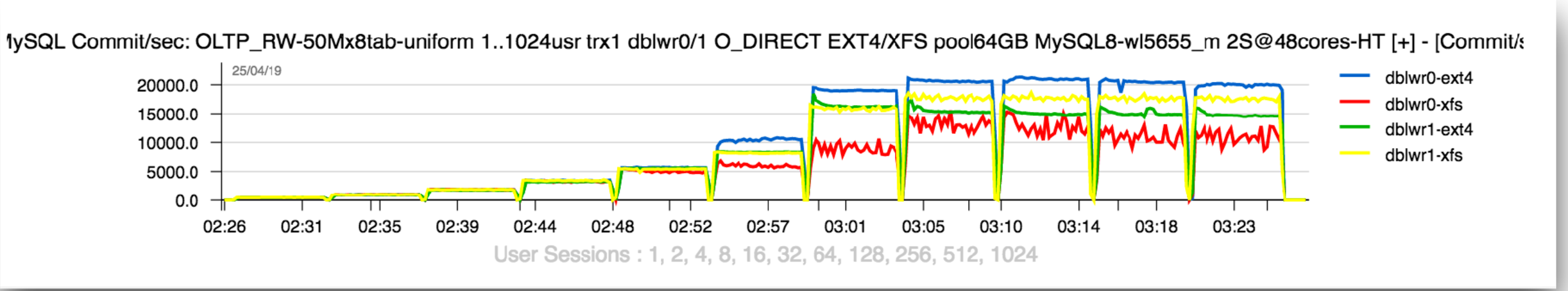
- Massive regression with EXT4 on IO-bound OLTP_RW @28cores-HT
  - Note : this is yet without DBLWR..
  - EXT4 and Linux kernels :



  - UPDATE : seems like the issue is confirmed by EXT4 devs..
  - (and also additionally similar observation on Oracle DB)

InnoDB Double-Write (DBLWR) Saga..

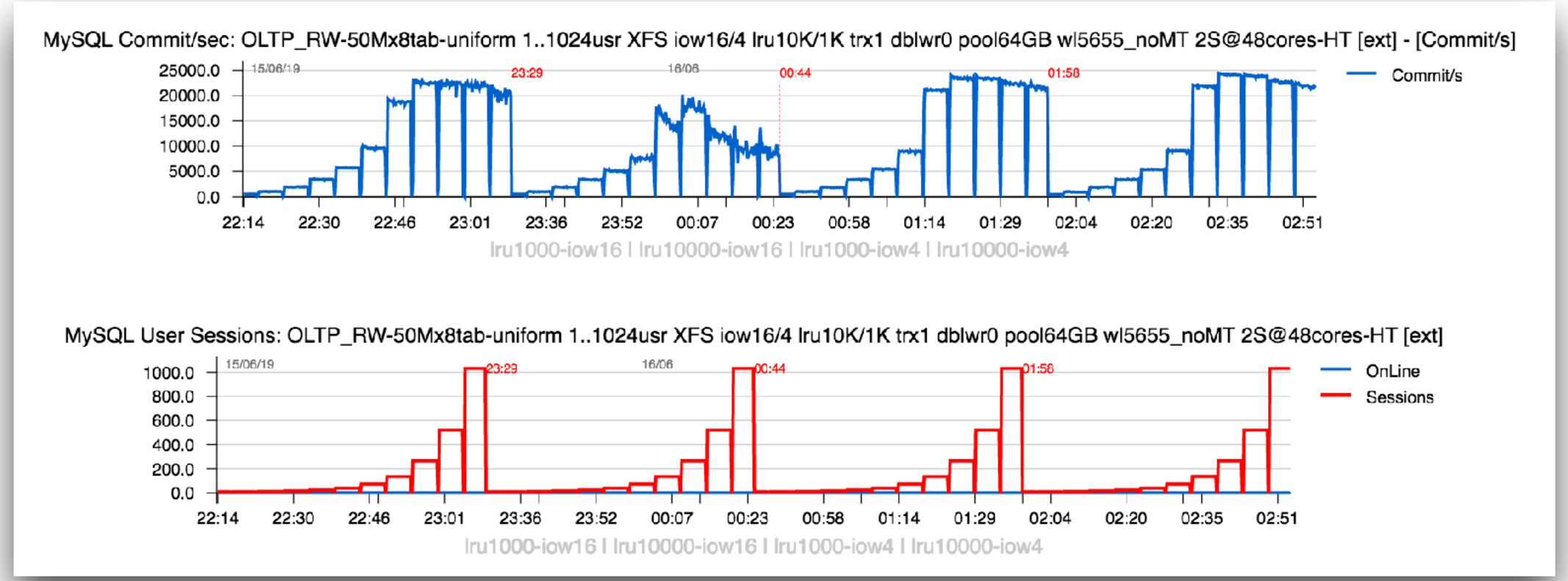
- DBLWR impact on IO-bound OLTP_RW @48cores-HT & Fast SSD :
  - dblwr=0 : EXT4 is doing better than XFS
  - dblwr=1 : XFS is doing better than EXT4
  - XFS : doing better with dblwr=1 than with dblwr=0 !!! => WTF ?..



- Still under investigation…

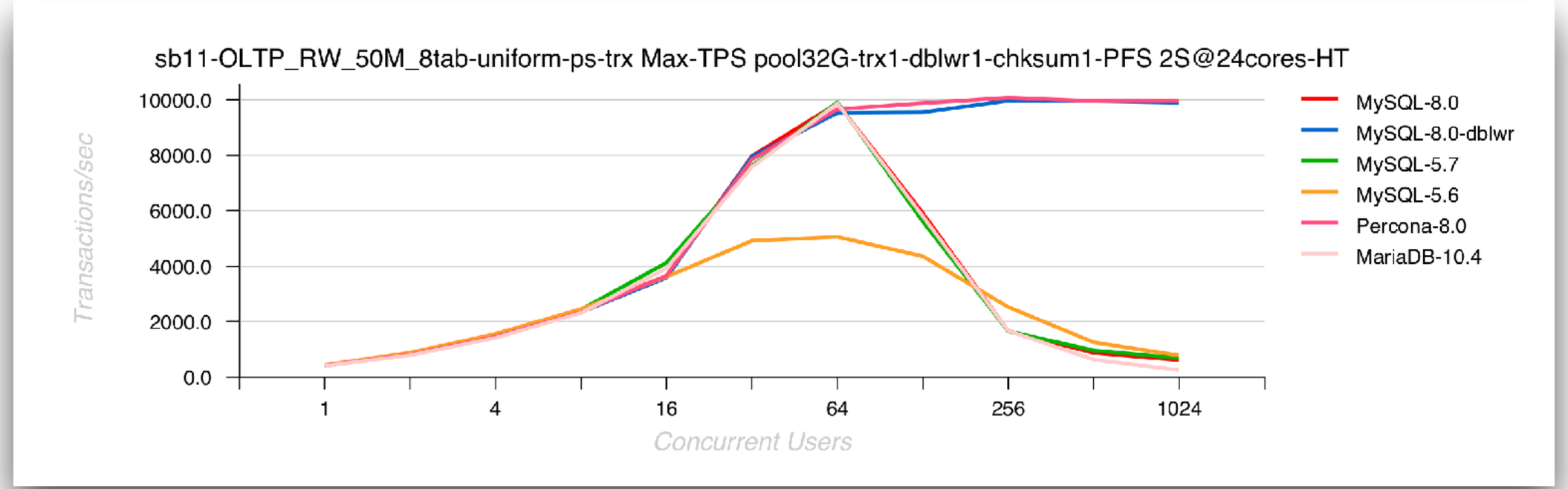
InnoDB Double-Write (DBLWR) Saga..

- XFS : DBLWR=0 on IO-bound OLTP_RW @48cores-HT & Fast SSD :
  - Workaround : lower the number of InnoDB IO Write threads !
  - iow 16 => iow 4 : clearly visible impact, but WHY ?.. ;-))



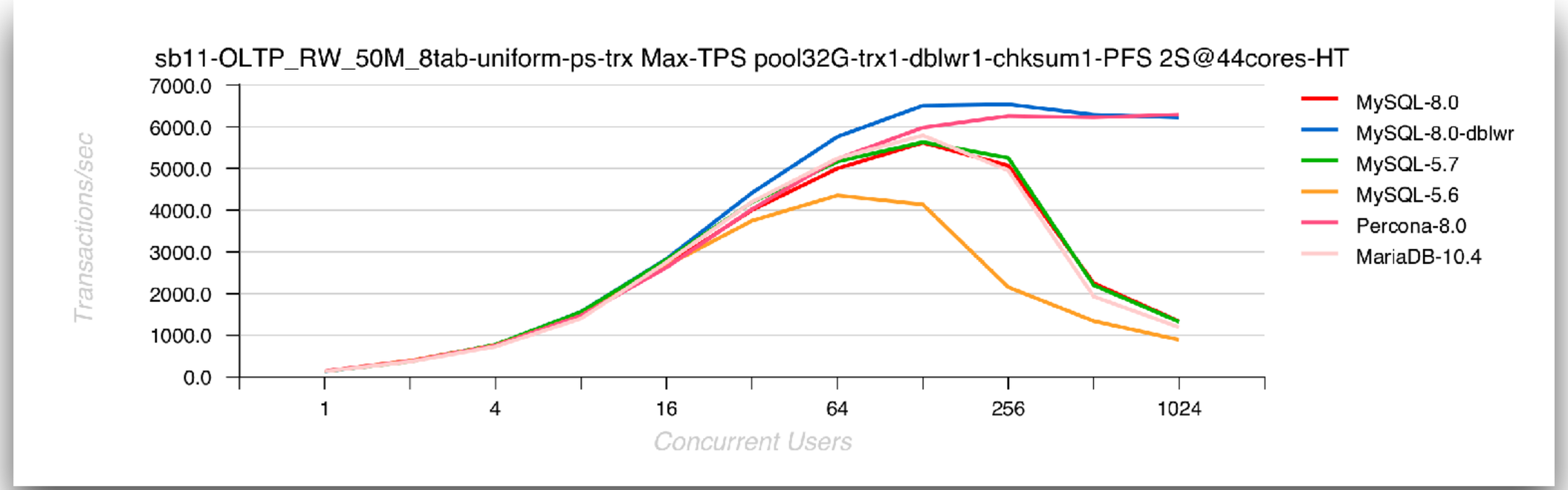
InnoDB Double-Write (DBLWR) Saga..

- IO-bound Sysbench OLTP_RW 50Mx8tab (Sep.2019)
  - Config : trx=1, dblwr=1, iow=4, pool=32G
  - 2S Dell 24cores-HT & Dell NVMe :



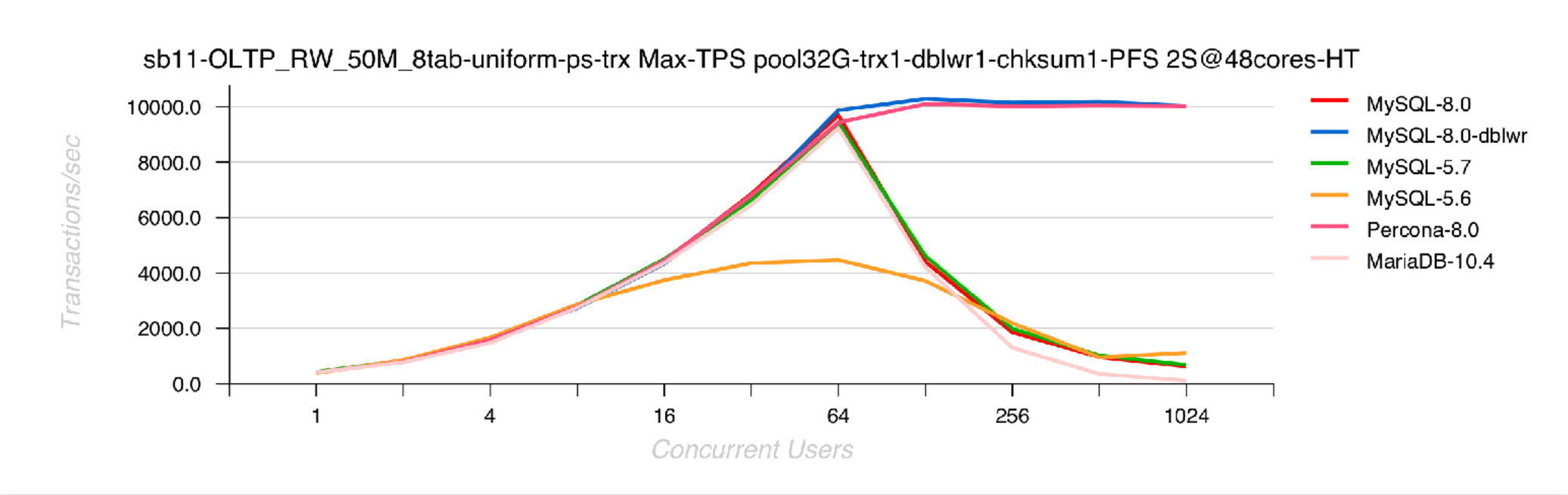
InnoDB Double-Write (DBLWR) Saga..

- IO-bound Sysbench OLTP_RW 50Mx8tab (Sep.2019)
  - Config : trx=1, dblwr=1, iow=4, pool=32G
  - 2S Broadwell v4 44cores-HT & Samsung NVMe :



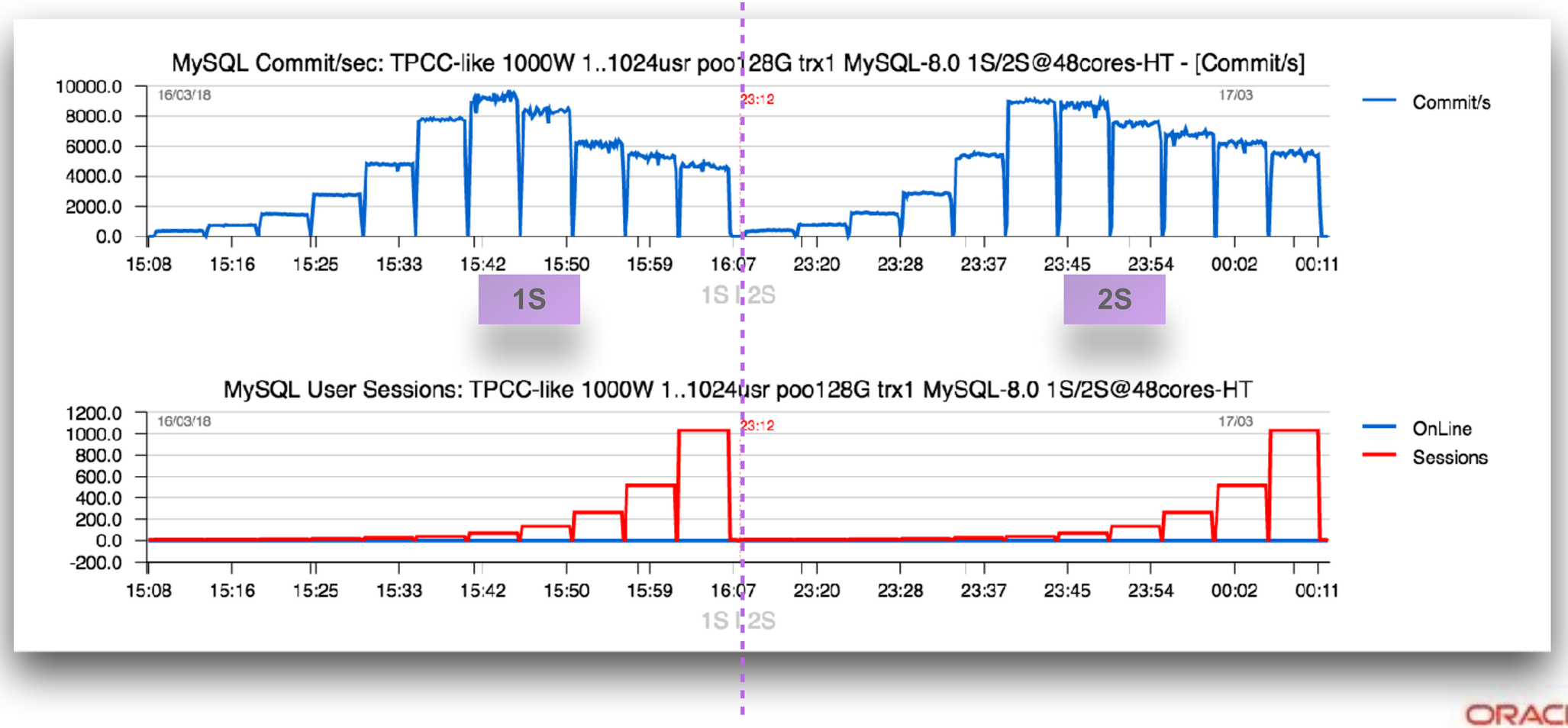
InnoDB Double-Write (DBLWR) Saga..

- IO-bound Sysbench OLTP_RW 50Mx8tab (Sep.2019)
  - Config : trx=1, dblwr=1, iow=4, pool=32G
  - 2S Intel Cascade Lake 48cores-HT & 2xOptane NVMe :



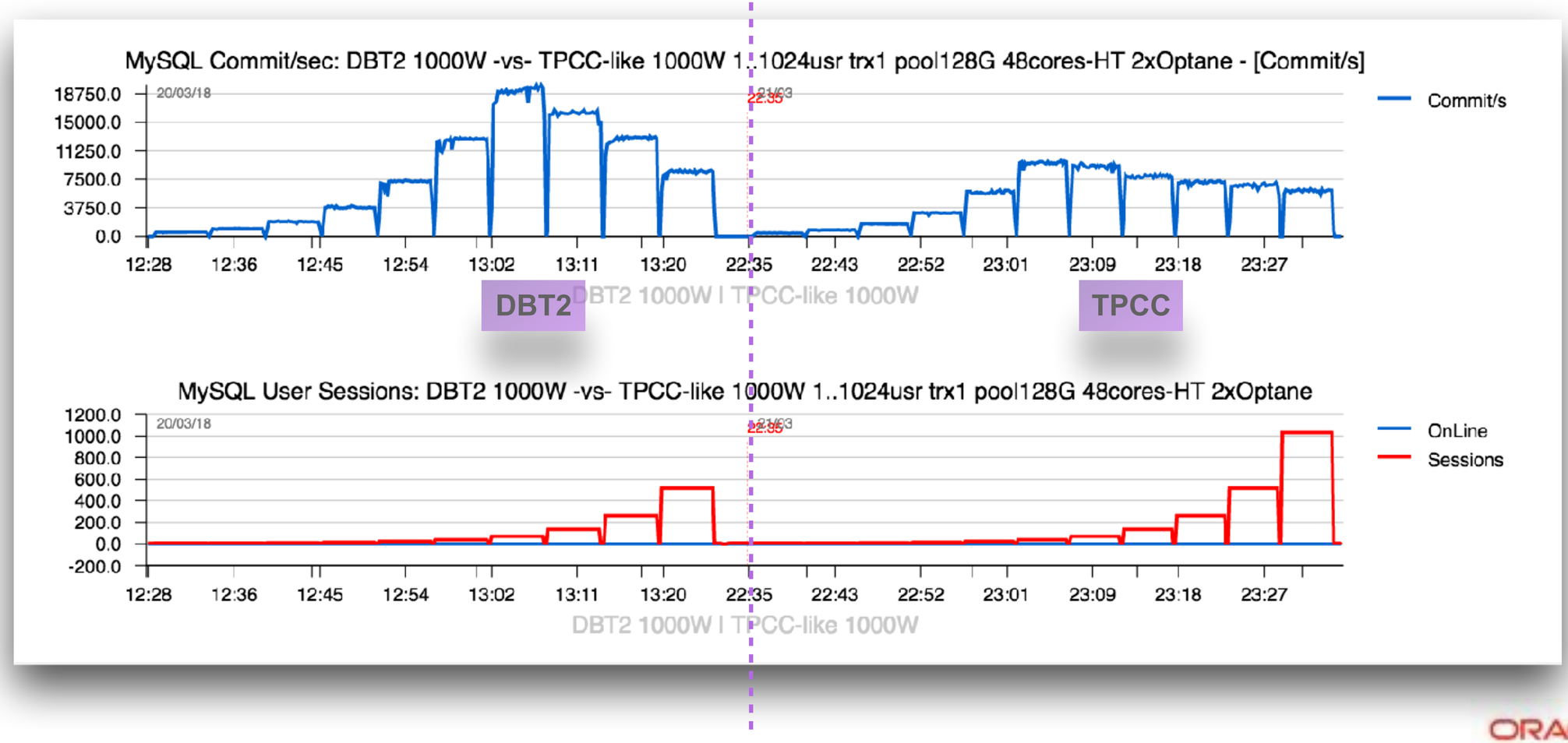
TPCC “mystery” : 1000W Workload @Sysbench-TPCC

- MySQL 8.0 :
  - small or no gain between 1S -vs- 2S
  - scalability is totally blocked by “index RW-lock” contention..



TPCC “mystery” : DBT2 -vs- Sysbench-TPCC (Apr.2018)

- BP = 128GB : in-memory
  - nearly the same workloads, but different “execution profile”
  - no index RW-lock contention in DBT2.. => **to investigate**



TPCC “mystery” : Investigation

- Motivation
  - even if we could not “fix” index RW-lock contention in InnoDB on TPCC…
  - …but we can find a “workaround” on how to avoid it => **already a huge gain !!!**

TPCC “mystery” : Investigation

- Motivation
  - even if we could not “fix” index RW-lock contention in InnoDB on TPCC…
  - …but we can find a “workaround” on how to avoid it => **already a huge gain !!!**
- 1) Ranger (ex-MySQL, now Percona, DBT2 lover in the past ;-))
  - => who changed the DBT2 schema ???
  - => as the result : 1/2 transactions are rejected on INSERT of NULL value !!!
  - => at least it became clear why in my case DBT2 was doing better ;-))

Anything could we do with INDEX lock contention ???

TPCC “mystery” : Investigation

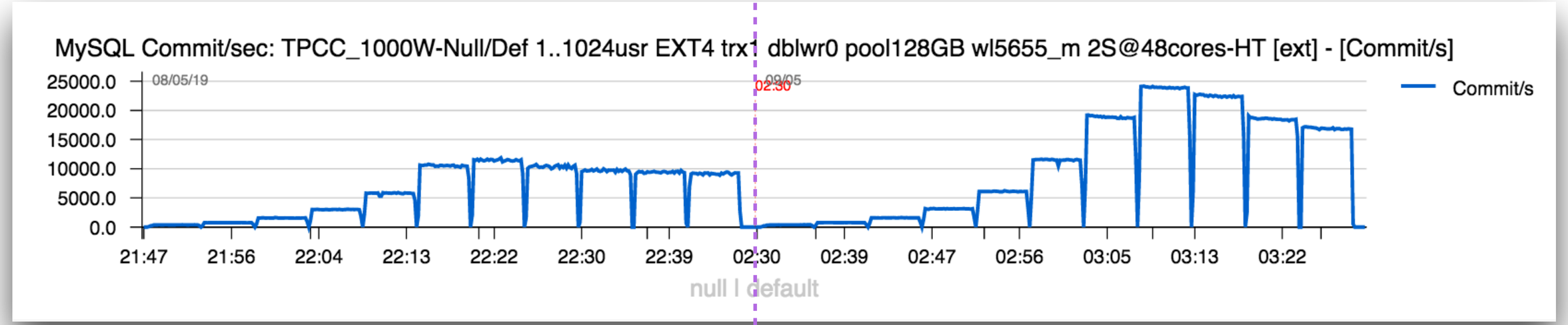
- Motivation
  - even if we could not “fix” index RW-lock contention in InnoDB on TPCC…
  - …but we can find a “workaround” on how to avoid it => **already a huge gain !!!**
- 1) Ranger (ex-MySQL, now Percona, DBT2 lover in the past ;-))
  - => who changed the DBT2 schema ???
  - => as the result : 1/2 transactions are rejected on INSERT of NULL value !!!
  - => at least it became clear why in my case DBT2 was doing better ;-))
- 2) Yasufumi (ex-Percona, ex-MySQL, now MySQL again, TPCC lover ;-))
  - why INDEX lock ? => excessive page split !
  - why page split ? => …
  - TPCC : … INSERT val=NULL
  - TPCC: … UPDATE val=DATE
  - NULL => DATE => no space in page => page split !!! ;-))

TPCC “mystery” : Solution

- Workaround :
  - instead of NULL use DEFAULT (and assign an “good size” value to DEFAULT)
  - then : INSERT values(…, DEFAULT, …) /* instead of NULL */
  - then : UPDATE … var = DEFAULT … /* instead of NULL */
  - then : SELECT … WHERE var != DEFAULT /* instead of NULL */
  - and so on.. => no more page split (or very minimal)
- Final solution :
  - work in progress..

TPCC “mystery” : test results

- Sysbench-TPCC 1000W
  - NULL -vs- DEFAULT

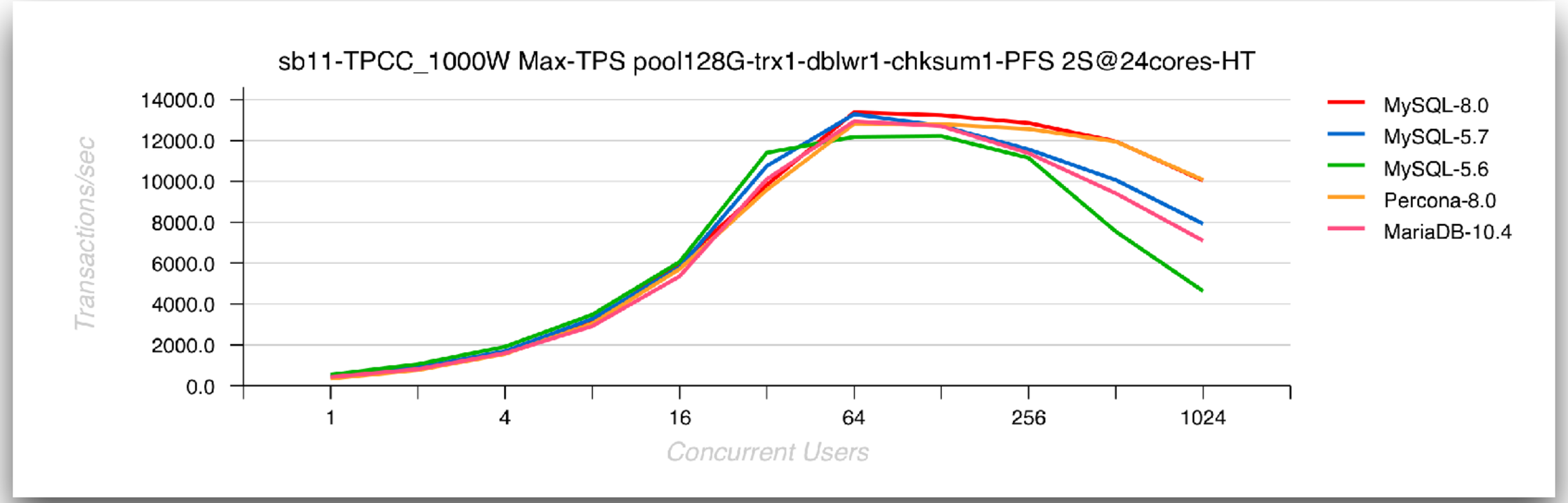


NULL

DEFAULT

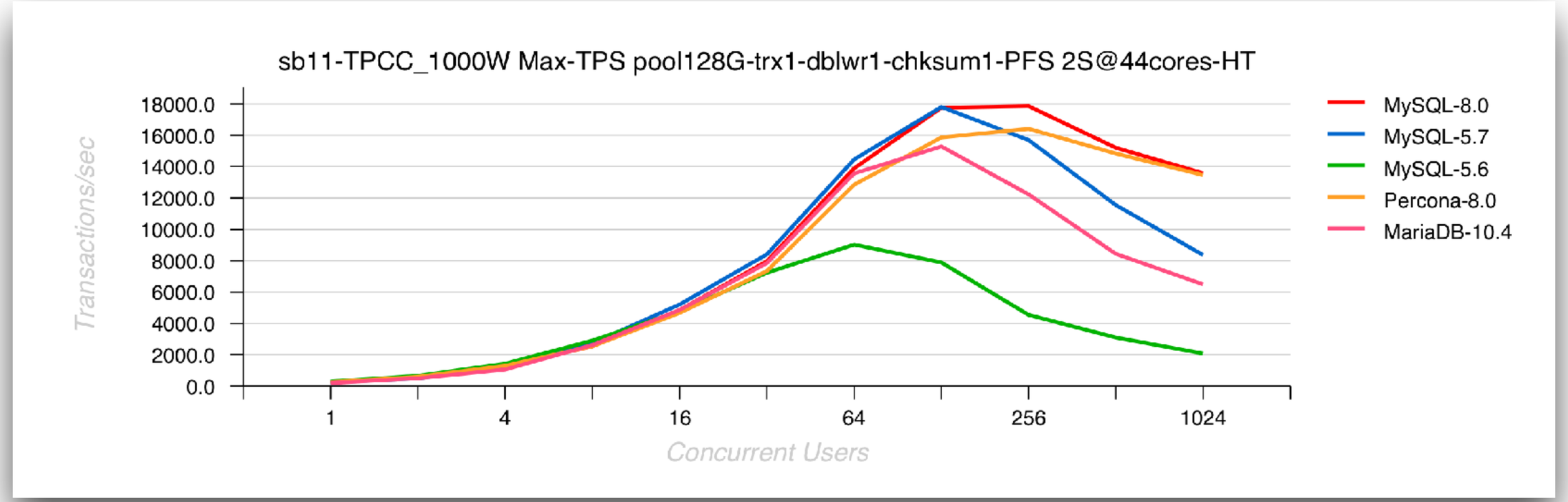
TPCC “mystery” : test results (Sep.2019)

- Sysbench-TPCC 1000W, using DEFAULT
  - **in-memory** (pool=128G)
  - 2S Dell 24cores-HT & Dell NVMe :



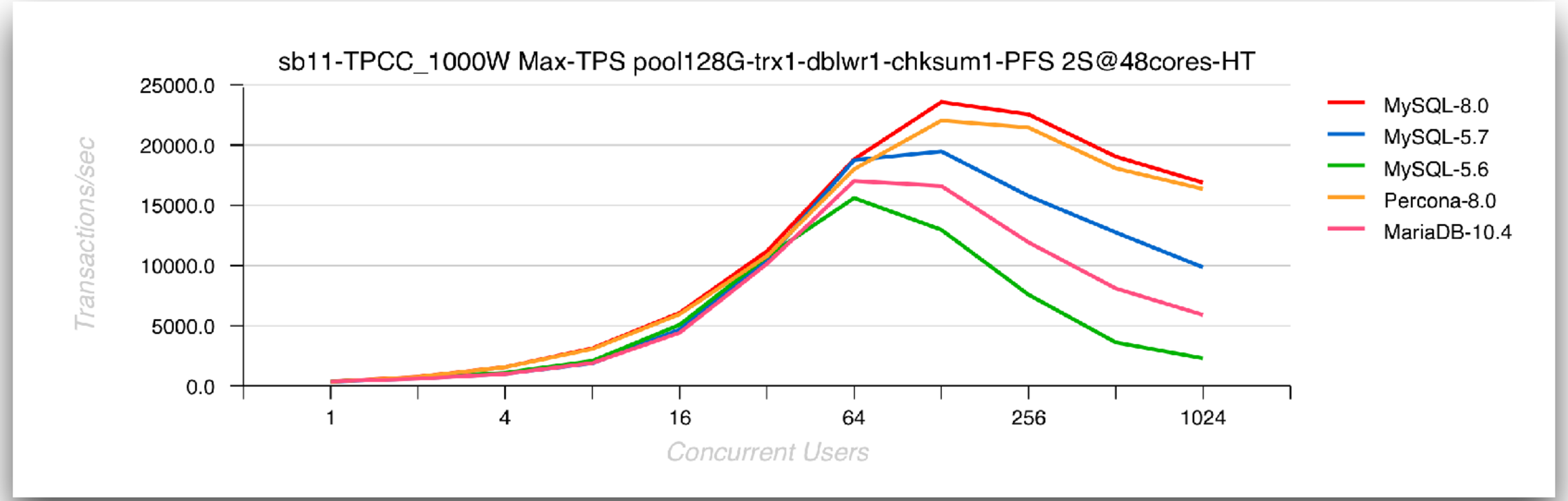
TPCC “mystery” : test results (Sep.2019)

- Sysbench-TPCC 1000W, using DEFAULT
  - **in-memory** (pool=128G)
  - 2S Broadwell v4 44cores-HT & Samsung NVMe :



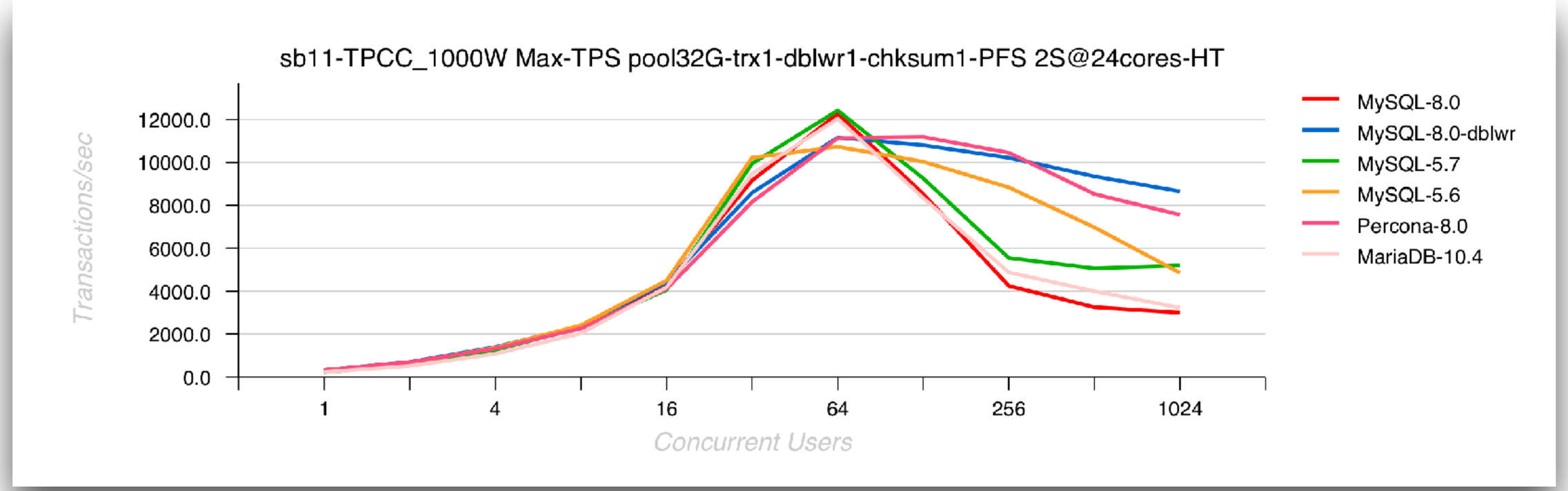
TPCC “mystery” : test results (Sep.2019)

- Sysbench-TPCC 1000W, using DEFAULT
  - **in-memory** (pool=128G)
  - 2S Intel Cascade Lake 48cores-HT & 2xOptane NVMe :



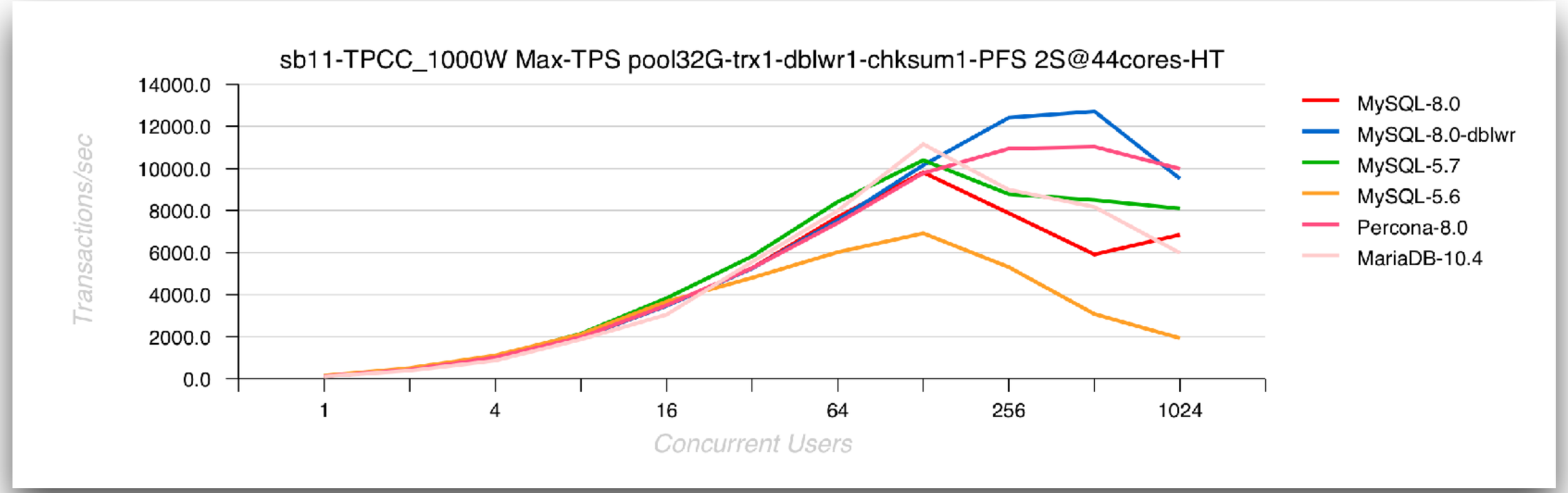
TPCC “mystery” : test results (Sep.2019)

- Sysbench-TPCC 1000W, using DEFAULT
  - **IO-bound** (pool=32G)
  - 2S Dell 24cores-HT & Dell NVMe :



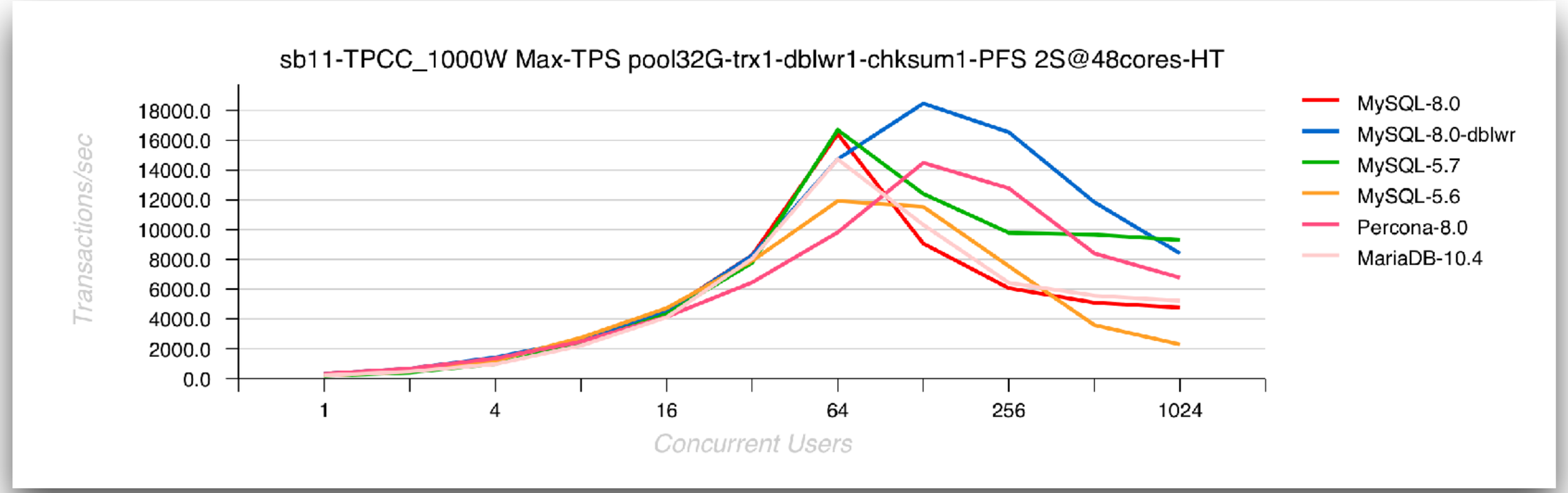
TPCC “mystery” : test results (Sep.2019)

- Sysbench-TPCC 1000W, using DEFAULT
  - **IO-bound** (pool=32G)
  - 2S Broadwell v4 44cores-HT & Samsung NVMe :



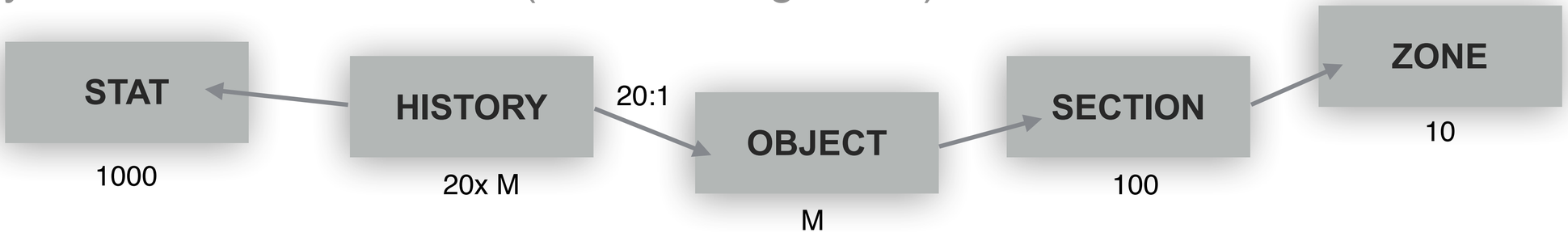
TPCC “mystery” : test results (Sep.2019)

- Sysbench-TPCC 1000W, using DEFAULT
  - **IO-bound** (pool=32G)
  - 2S Intel Cascade Lake 48cores-HT & 2xOptane NVMe :



dbSTRESS Workload

- Developed & Maintained by me ;-))
  - fully ported to Sysbench, but not yet published (in progress)
  - can be executed as x1 dataset or xN datasets (similar to xN db, except it's the same db)
  - inspired by real customer workload (stock management)
  - schema :

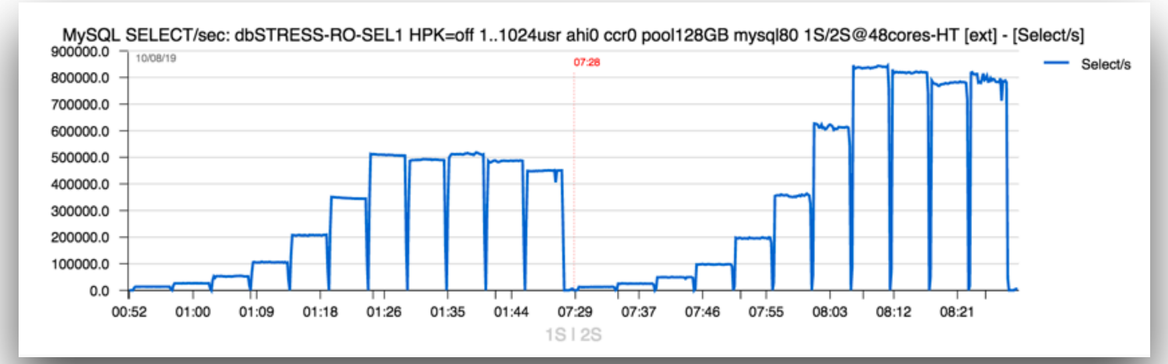


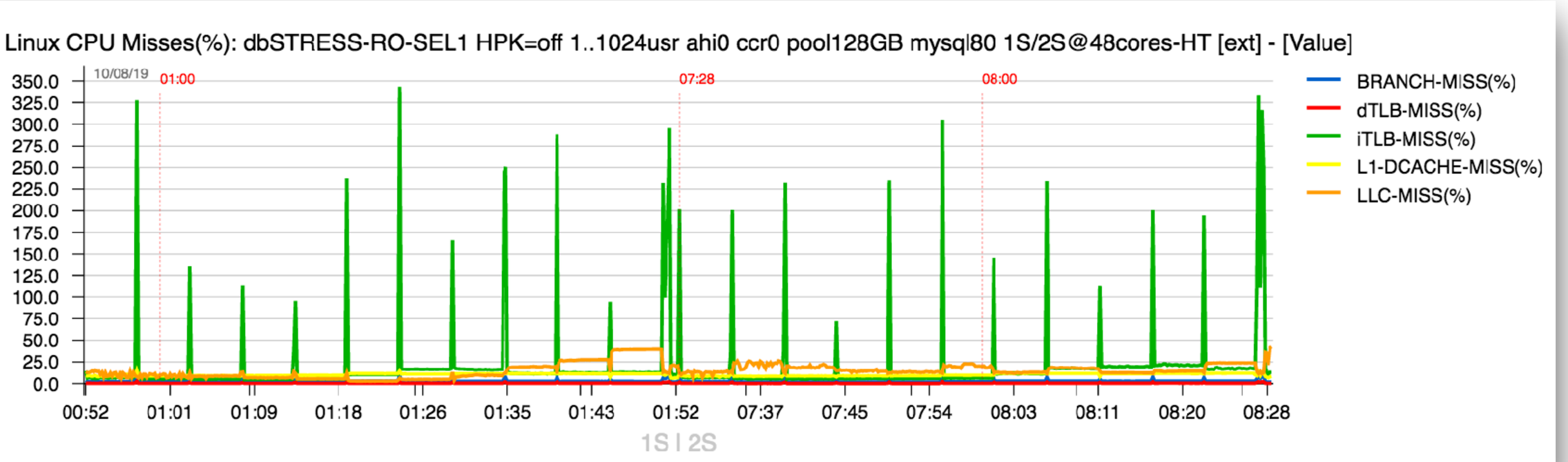
  - queries :
    - SEL1 : join of 3 tables by OBJECT REF (OBJECT => SECTION => ZONE)
    - SEL2 : join of 2 tables by OBJECT REF (OBJECT => HISTORY)
    - WRITE: DELETE/ INSERT/ UPDATE of single HISTORY record by OBJECT REF

- The current "mystery" :
  - SEL1 : scaling well, QPS on 2S is higher than on 1S
  - SEL2 : not scaling ! QPS on 2S is worse than on 1S !!
    - NOTE: using PK instead of Sec.IDX is helping in efficiency, but not in scalability..

dbSTRESS-RO-SEL1

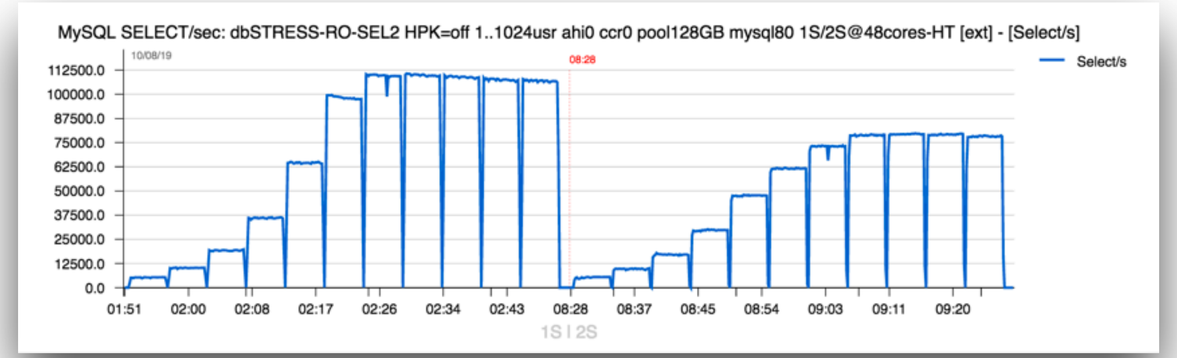
- SEL1 :
  - scaling from S1 to S2

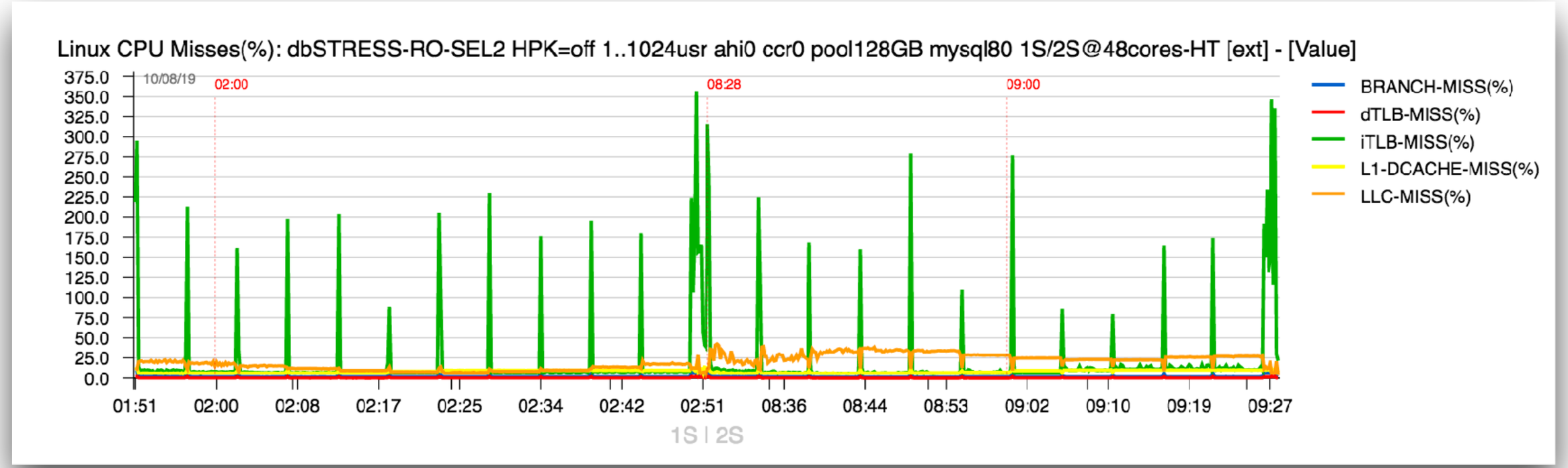




dbSTRESS-RO-SEL2

- SEL2 :
  - NOT scaling from S1 to S2

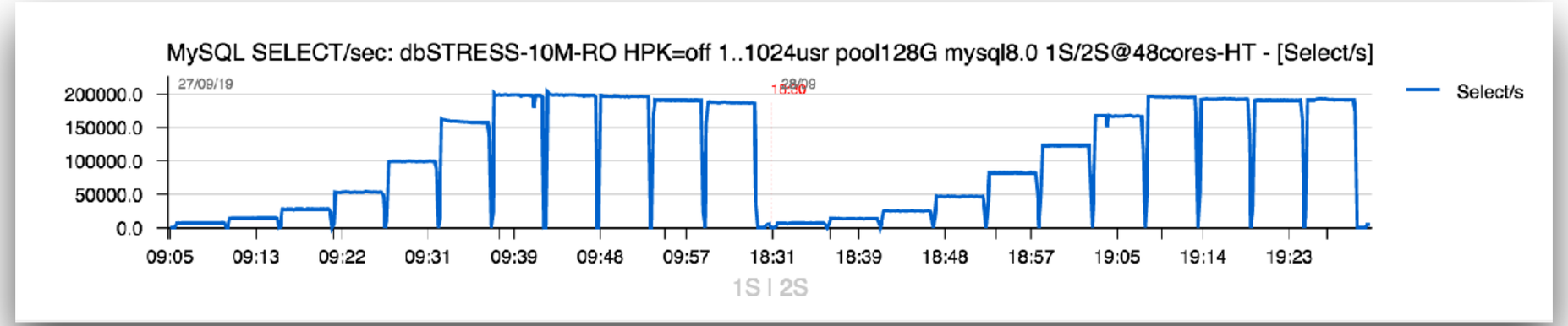




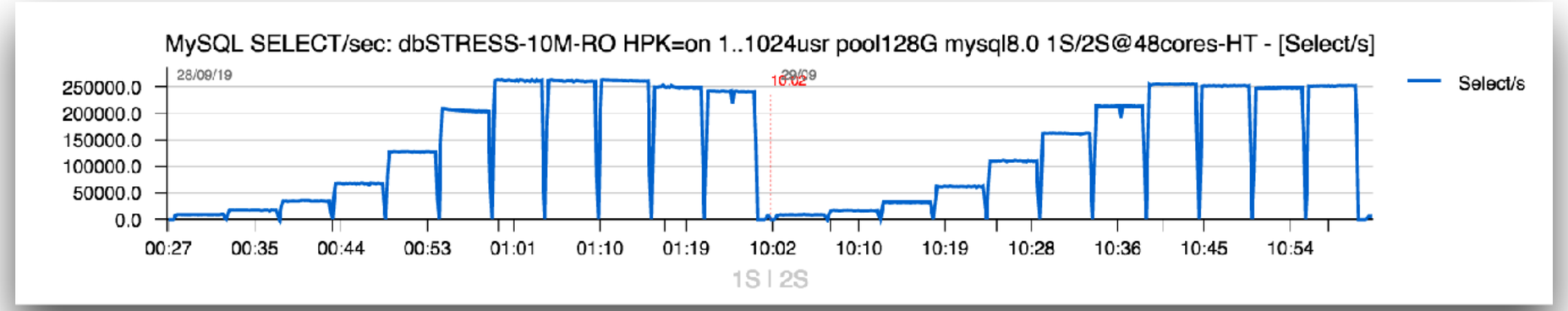
ORACLE

dbSTRESS-RO (SEL1 + SEL2)

- HISTORY table NOT using PK (HPK=off) :



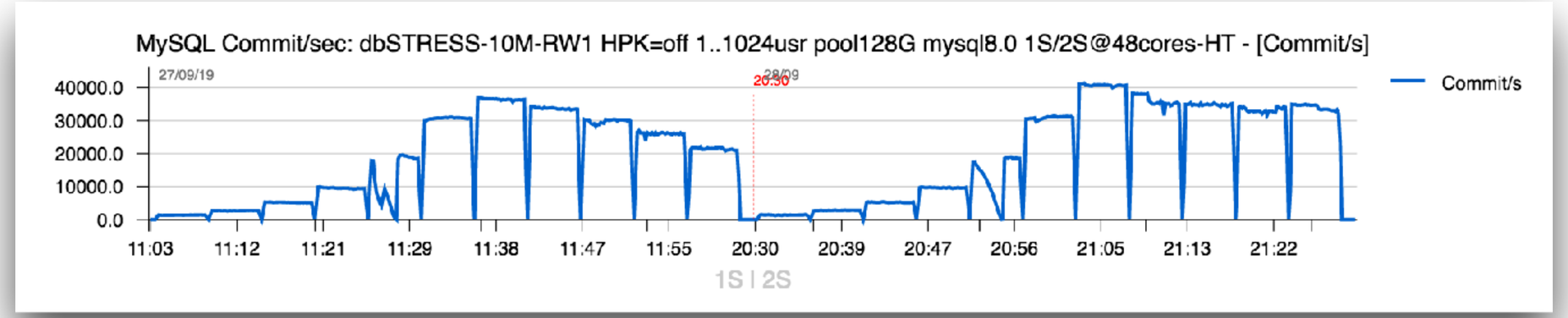
- HISTORY table using PK (HPK=on) :



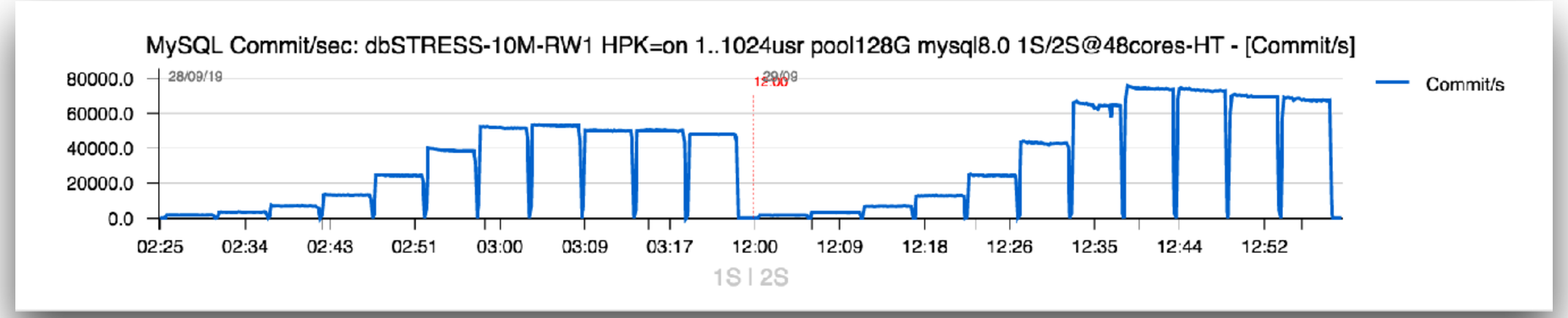
ORACLE

dbSTRESS-RW1 (R/W ratio= 1:1)

- HISTORY table NOT using PK (HPK=off) :



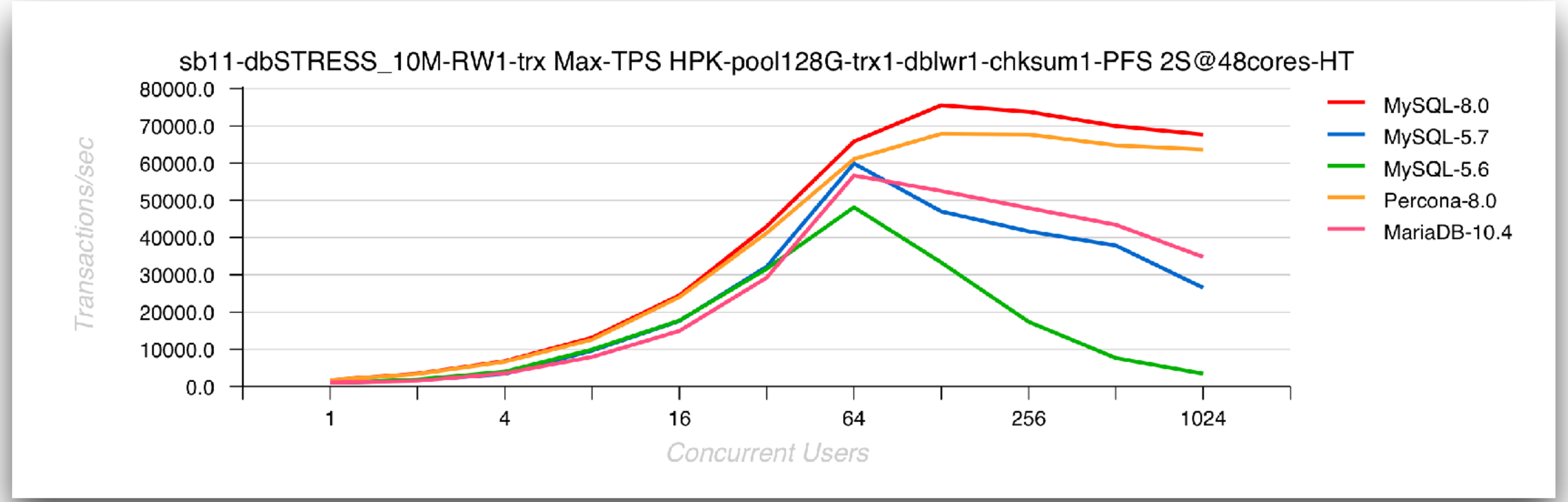
- HISTORY table using PK (HPK=on) :



ORACLE

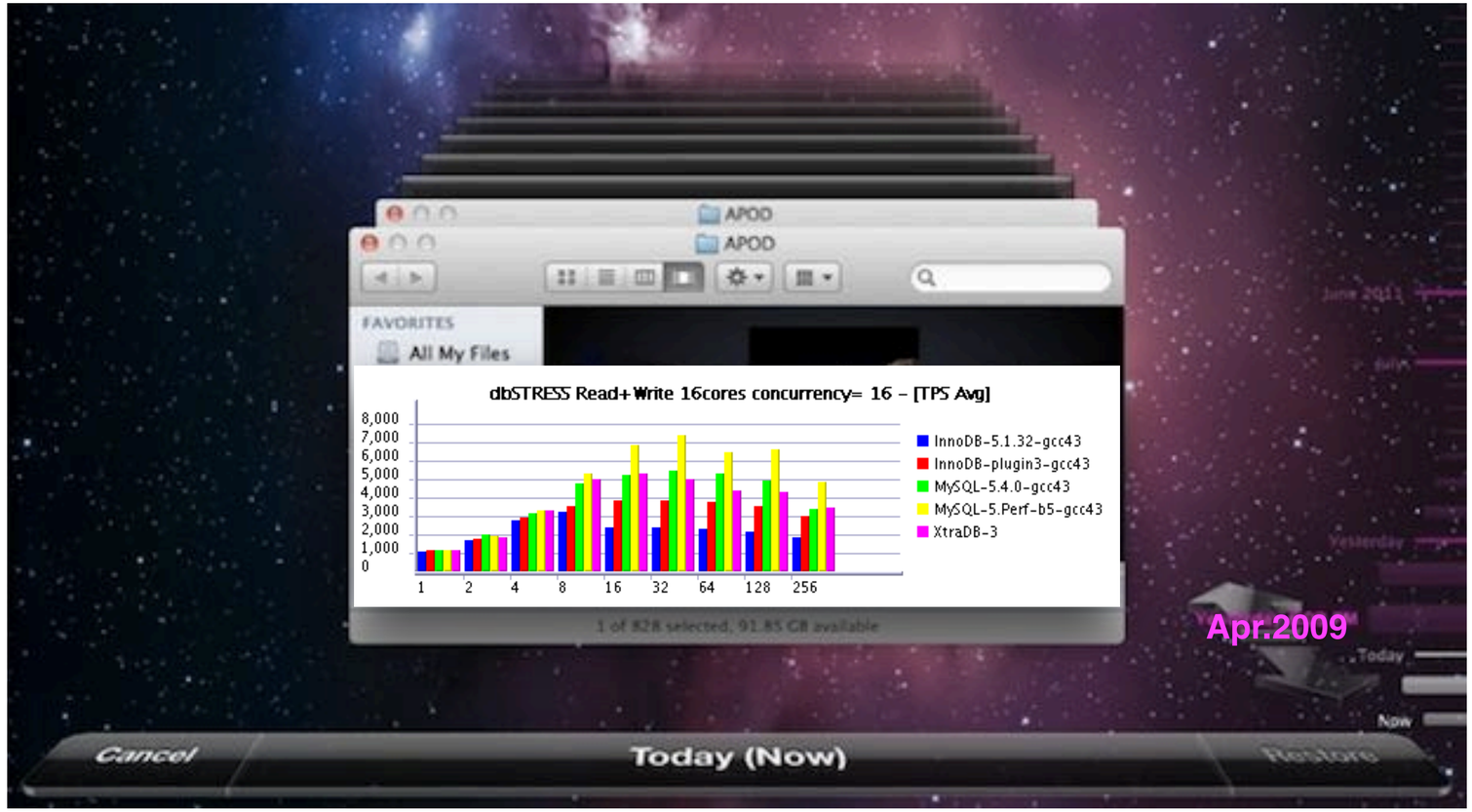
dbSTRESS 10M RW1 Test + HPK=on

- Dataset : 10M (200M HISTORY+PK (HPK)) @48cores-HT Cascade Lake
  - as usual, “everything is relative”..
  - while we blame MySQL 8.0 for scalability limits, it’s still doing better than all we have ;-))



dbSTRESS RW1 Test & Time Machine..

- Looking back with Time Machine :
  - even more once again — “everything is relative”..
  - **Apr.2009** : dbSTRESS 10M RW1 => **7.5K** TPS **only** as the best ever possible result !!
  - 10 years => and so huge jump in SW/HW + MySQL 8.0 !!! => x10 times diff..



Thank You !!! 🍾🍾🍾

ORACLE

Appendix

- System details :
  - 24cores-HT Dell : Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz
  - 44cores-HT : Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
  - 48cores-HT : Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz

One more thing ;-)

- All graphs are built with dim_STAT (http://dimitrik.free.fr)
  - All System load stats (CPU, I/O, Network, RAM, Processes,...)
    - Mainly for Linux, Solaris, OSX (and any other UNIX too :-)
    - Add-Ons for MySQL, Oracle RDBMS, PostgreSQL, Java, etc.
    - Linux : PerfSTAT (“perf” based), mysqlSTACK (quickstack based)
  - 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.