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

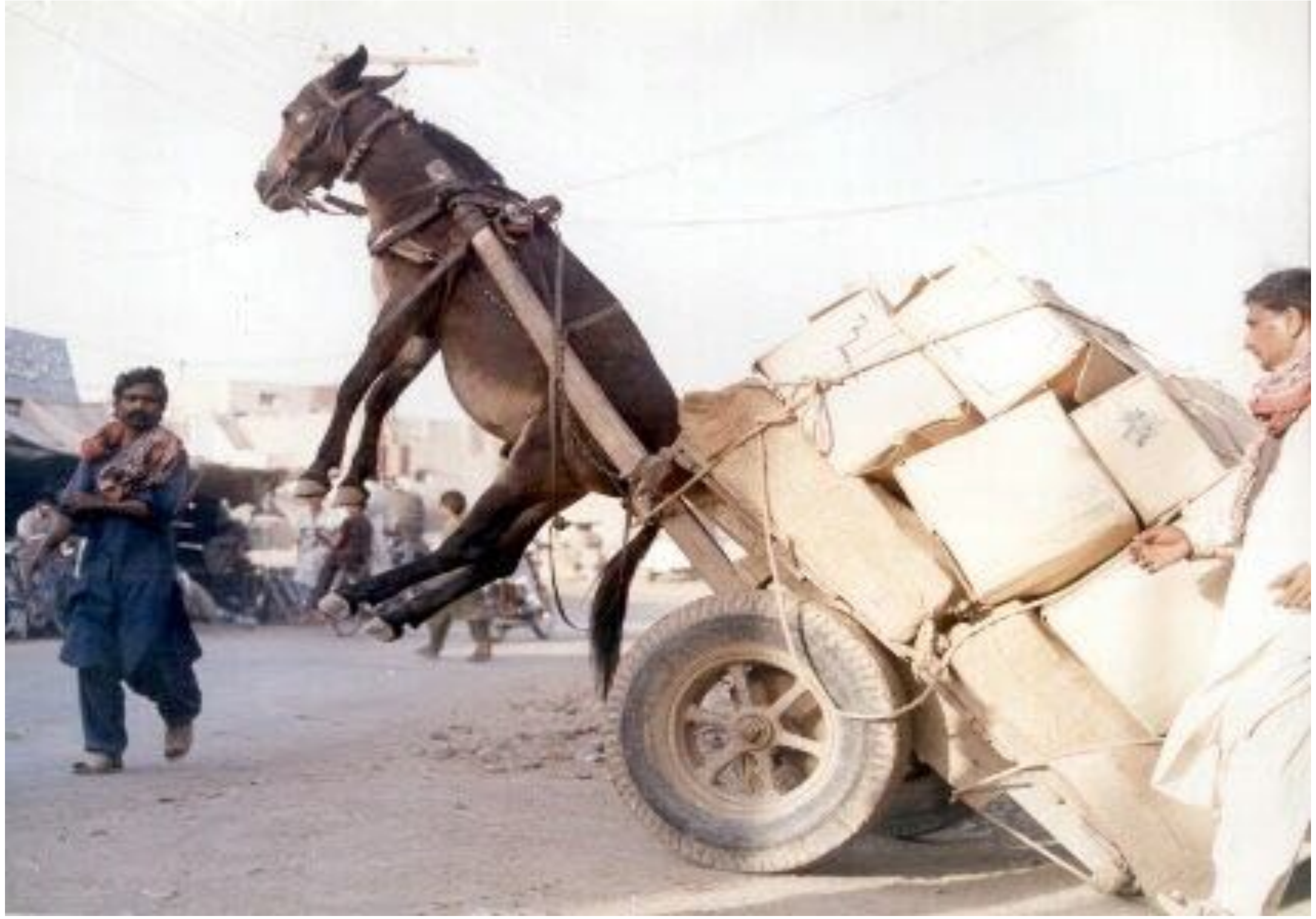
Why MySQL Performance ?...

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!... :-)

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 !!! ;-)

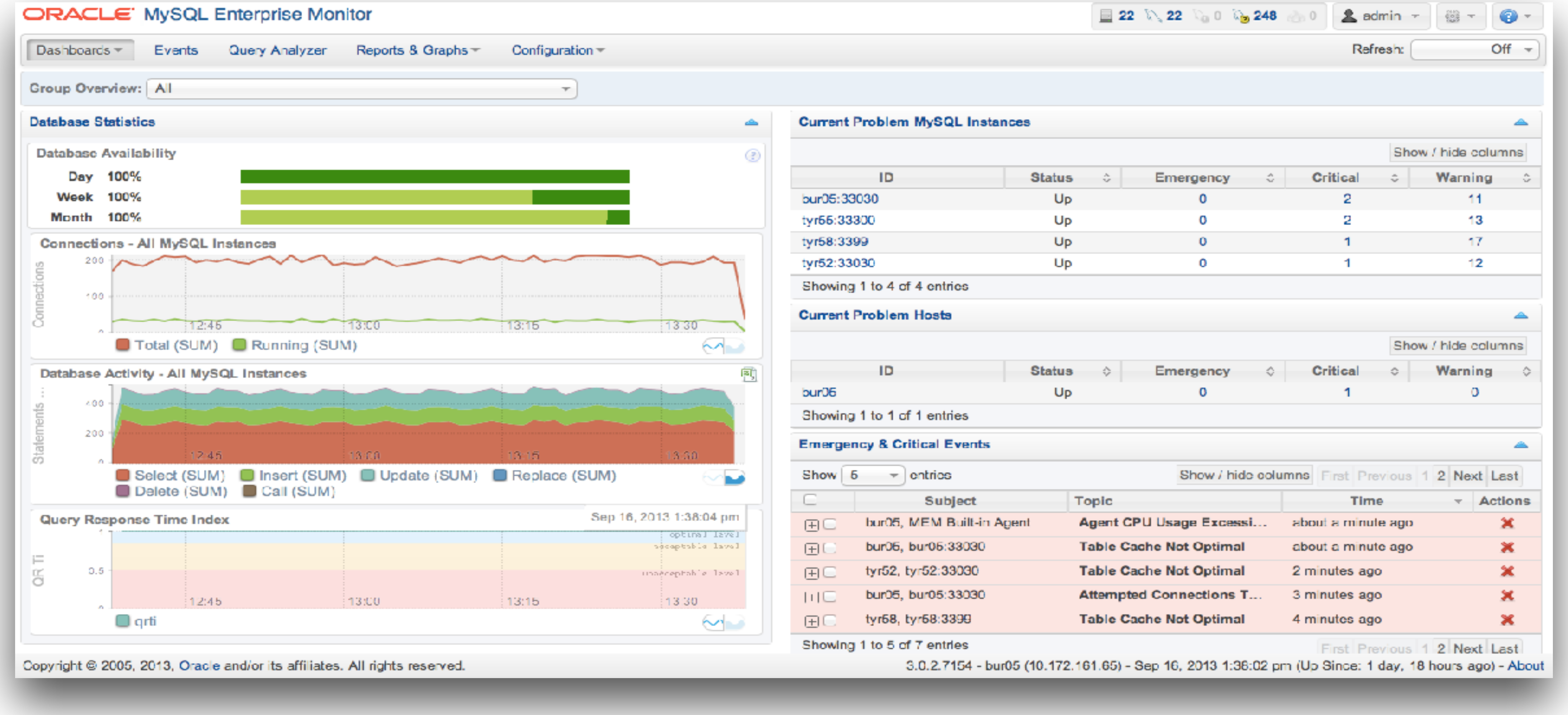
THE MAIN
SLIDE! ;-))

#2 - Monitoring is THE MUST !

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

MySQL Enterprise Monitor (MEM)

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



Database Statistics

Database Availability

Day	100%	
Week	100%	
Month	100%	

Current Problem MySQL Instances

Show / hide columns

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

Showing 1 to 4 of 4 entries

Current Problem Hosts

Show / hide columns

ID	Status	Emergency	Critical	Warning
bur05	Up	0	1	0

Showing 1 to 1 of 1 entries

Emergency & Critical Events

Show 5 entries Show / hide columns First Previous 1 2 Next Last

□	Subject	Topic	Time	Actions
⊞ □	bur05, MEM Built-in Agent	Agent CPU Usage Excessi...	about a minute ago	✗
⊞ □	bur05, bur05:33030	Table Cache Not Optimal	about a minute ago	✗
⊞ □	tyr52, tyr52:33030	Table Cache Not Optimal	2 minutes ago	✗
⊞ □	bur05, bur05:33030	Attempted Connections T...	3 minutes ago	✗
⊞ □	tyr58, tyr58:3399	Table Cache Not Optimal	4 minutes ago	✗

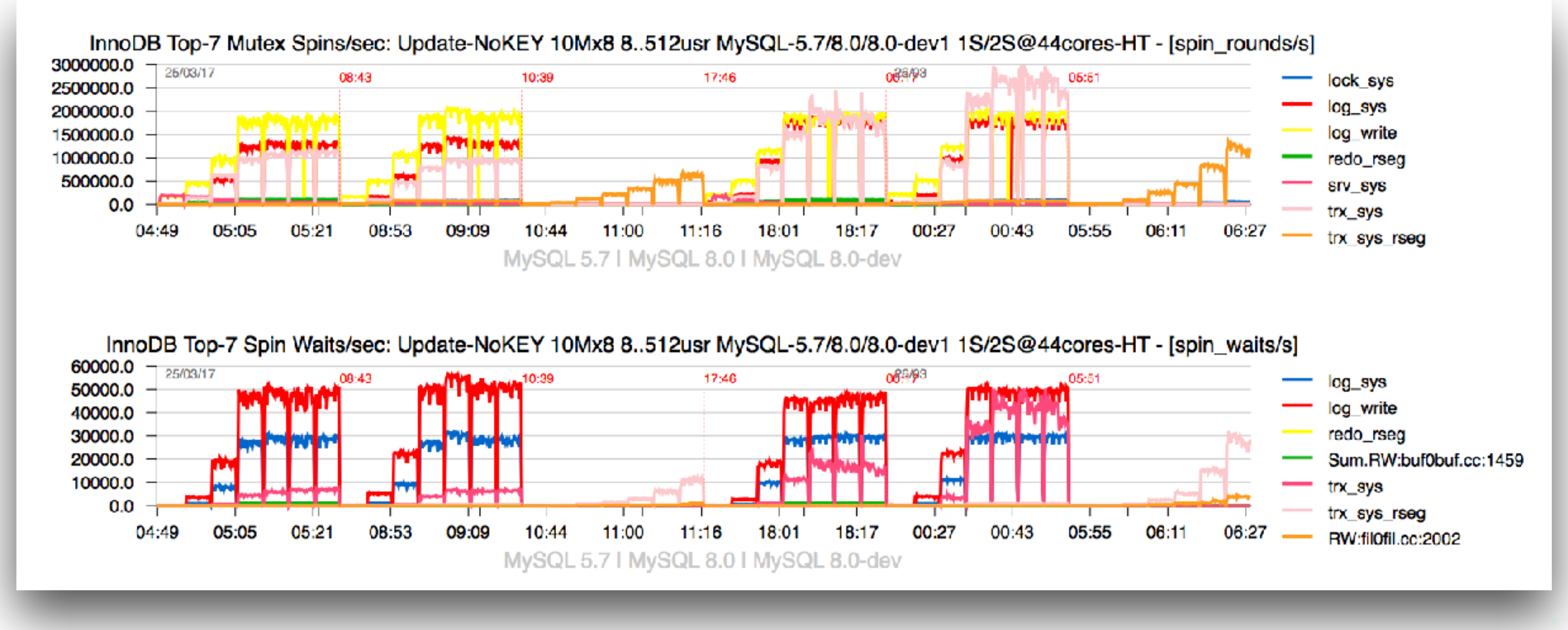
Showing 1 to 5 of 7 entries First Previous 1 2 Next Last

Copyright © 2005, 2013, Oracle and/or its affiliates. All rights reserved.

3.0.2.7154 - bur05 (10.172.161.65) - Sep 16, 2013 1:36:02 pm (Up Since: 1 day, 18 hours ago) - About

Other Monitoring Tools

- Cacti, Zabbix, Nagios, Solarwinds, VividCortex, PMM, etc…..
- ***dim_STAT***
  - yes, I'm using mine, 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 ;-))

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 ;-))

My main topic ;-)

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)
  - 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..

ORACLE

Example: iiBench (INSERT Benchmark)

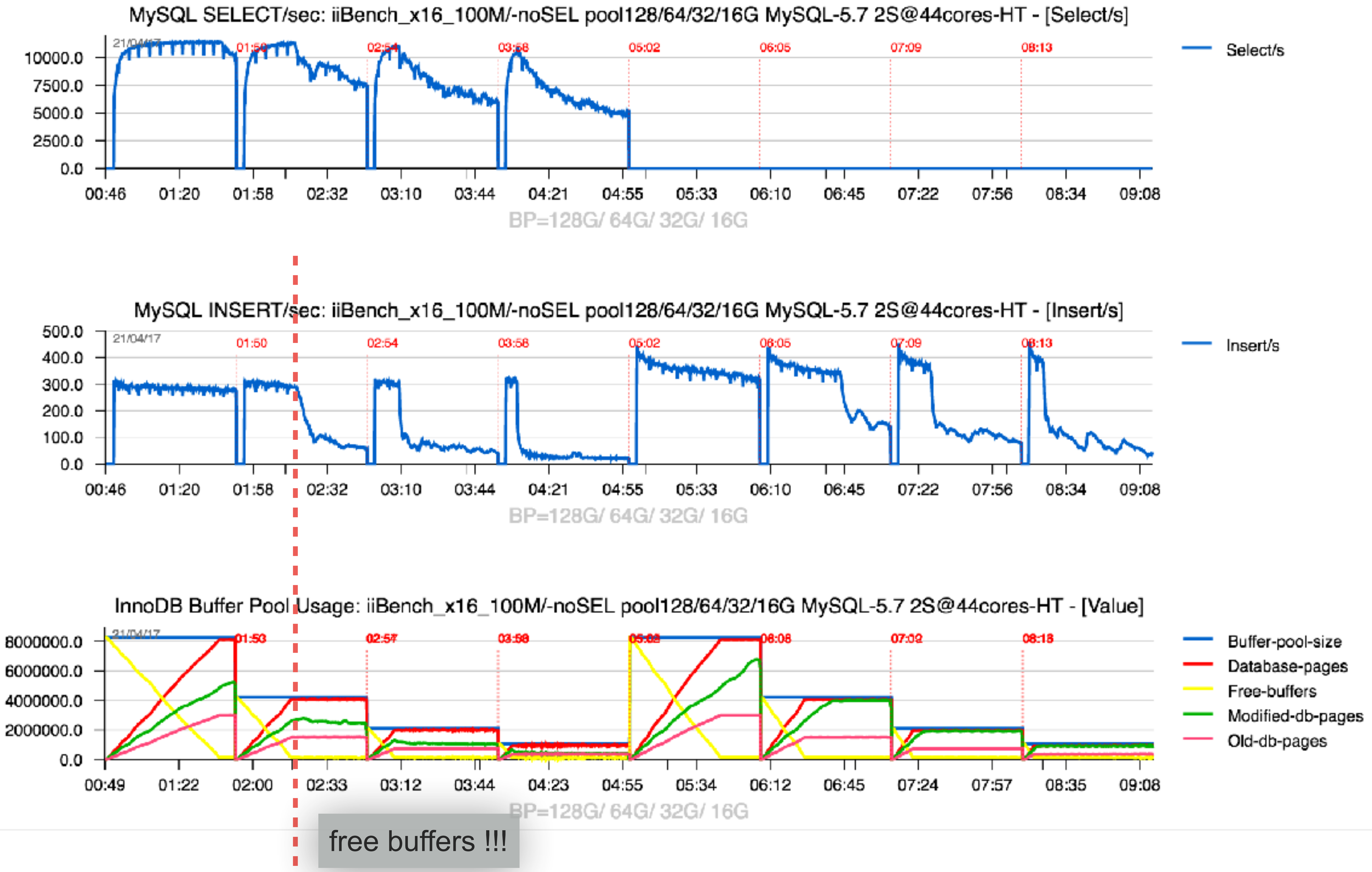
- Main claim :
  - InnoDB is xN times slower vs Write-oriented Engine XXX
  - so, use XXX, as it’s better

- 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)

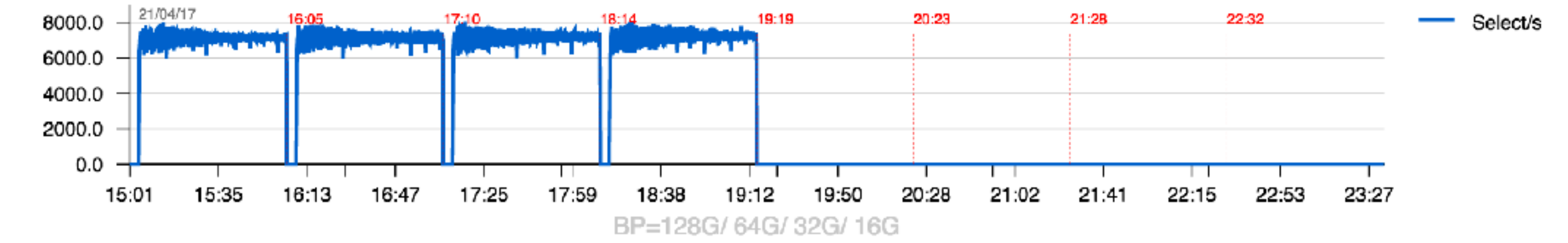


free buffers !!!

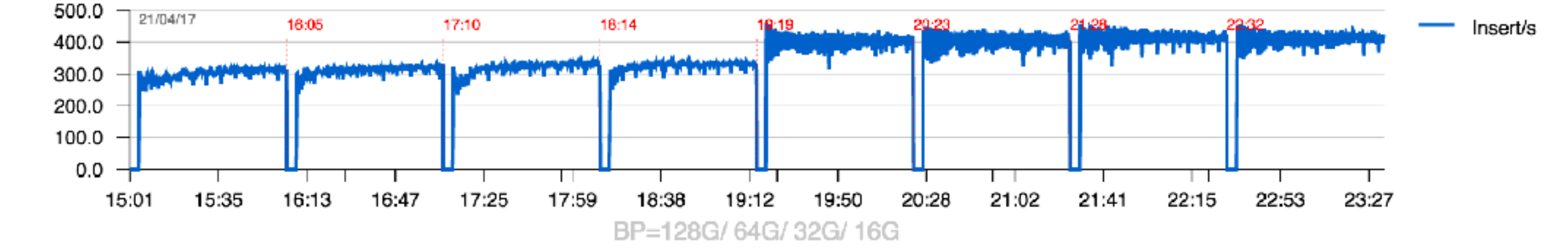
iiBench 100M x16 & 1M-parts : BP= 128G/ 64G/ 32G/ 16G

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

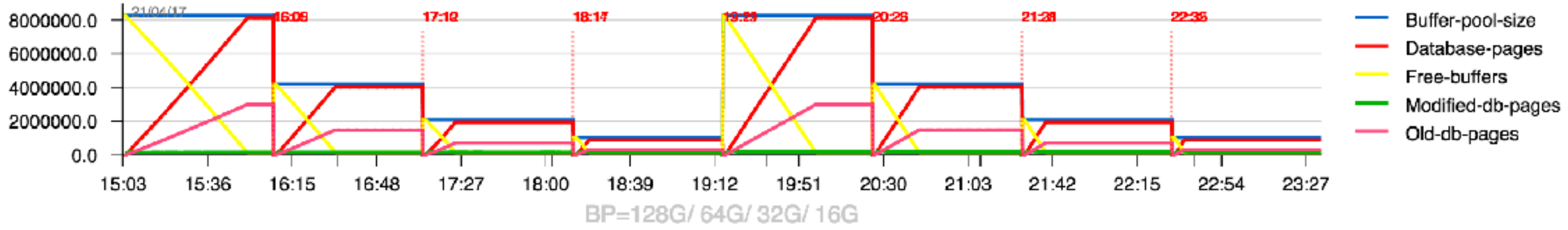
MySQL SELECT/sec: iiBench_x16_100M-1Mpart/-noSEL pool128/64/32/16G MySQL-5.7 2S@44cores-HT - [Select/s]



MySQL INSERT/sec: iiBench_x16_100M-1Mpart/-noSEL pool128/64/32/16G MySQL-5.7 2S@44cores-HT - [Insert/s]



InnoDB Buffer Pool Usage: iiBench_x16_100M-1Mpart/-noSEL pool128/64/32/16G MySQL-5.7 2S@44cores-HT - [Value]



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
  - “Entry Ticket” Workloads, looks simple, but still the most complete test kit !
  - OLTP, RO/RW, points on RO and RW issues
- DBT2 / TPCC-like
  - OLTP, RW, pretty complex, growing db, no options, deadlocks
  - in reality using mostly 2 tables only! (thanks Performance Schema ;-))
- DBT3
  - 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
- LinkBench (Facebook)
  - OLTP, RW, looks intensive and IO-hungry

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 ! ;-)

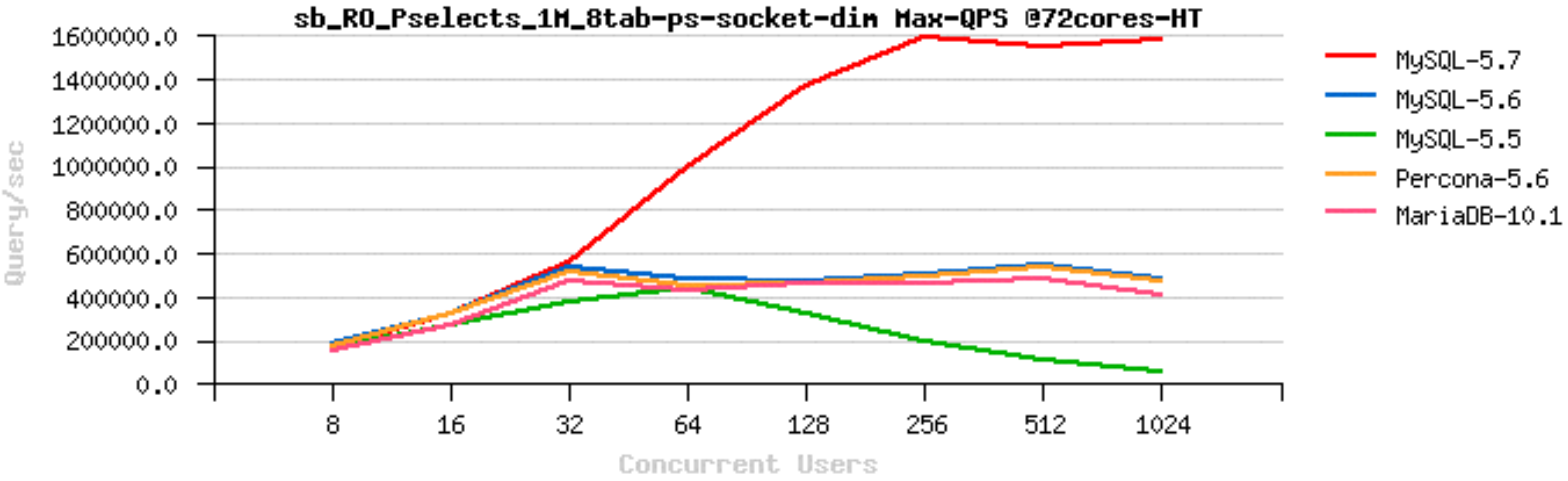
Historically main target : In-Memory Workloads

- What do you mean here ?..
  - have enough RAM for BP to keep all the data (or the "active data-set”) cached
  - e.g. => no I/O reads
  - e.g. => because the disks are so slow, keep as much as you can in RAM
  - historically => part of “best practice” to any RDBMS :
    - I/O reads most impacting
    - I/O writes => many solutions to speed-up

- Historical problems :
  - low load / high load
  - scalability

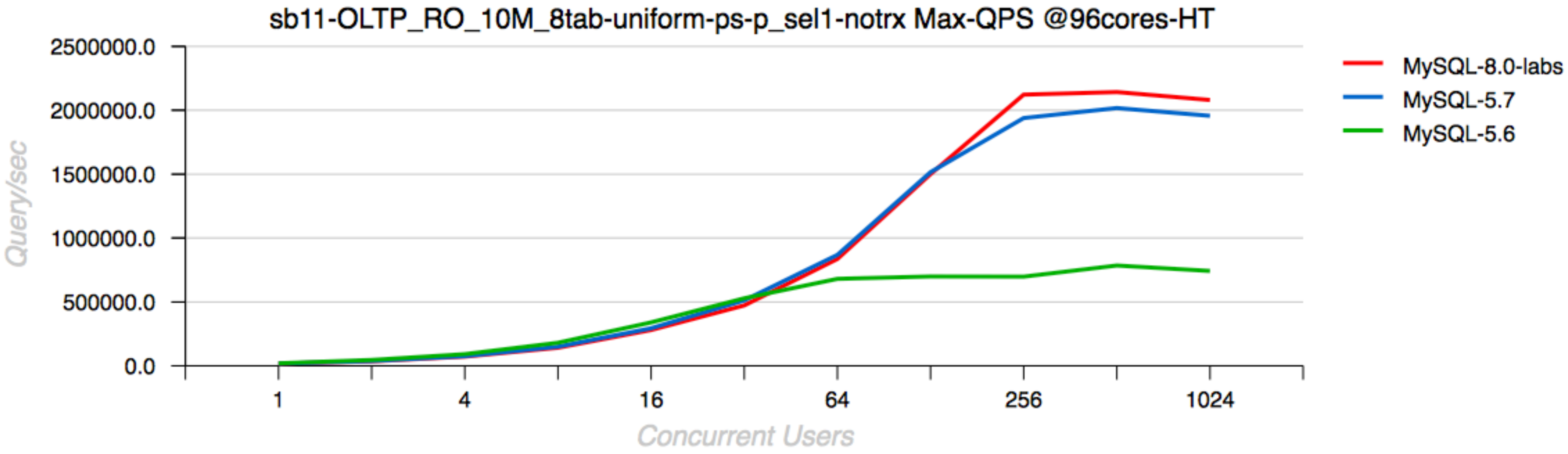
RO Point-Selects @MySQL 5.7 (Oct.2015)

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



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

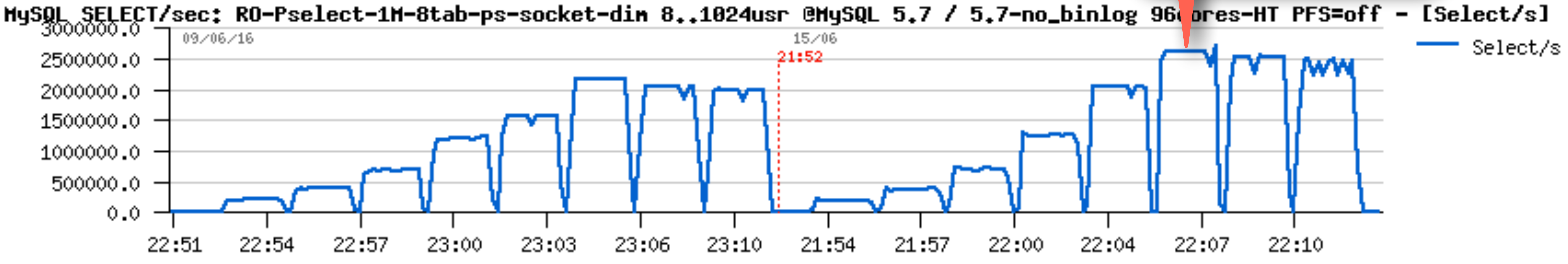
- **2.1M (!!) QPS** Sysbench Point-Selects 8-tab :
  - 96cores-HT Broadwell

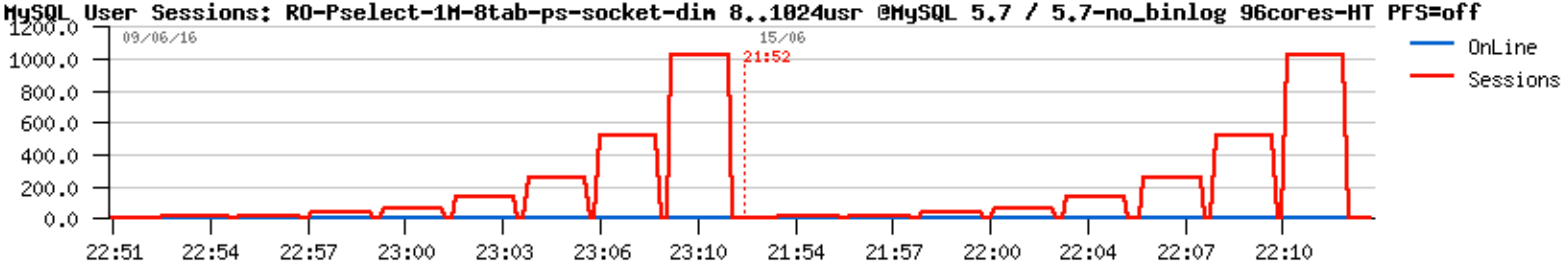


Potential RO Point-Selects @MySQL 5.7 (Jun.2016)

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

over 2.5M QPS


MySQL SELECT/sec: RO-Pselect-1M-8tab-ps-socket-dim 8..1024usr @MySQL 5.7 / 5.7-no_binlog 96cores-HT PFS=off - [Select/s]


MySQL User Sessions: RO-Pselect-1M-8tab-ps-socket-dim 8..1024usr @MySQL 5.7 / 5.7-no_binlog 96cores-HT PFS=off

Pending Scalability Issues after MySQL 5.7 GA..

- RO :
  - Block Locks
  - Lookups via Sec.IDX
  - UTF8

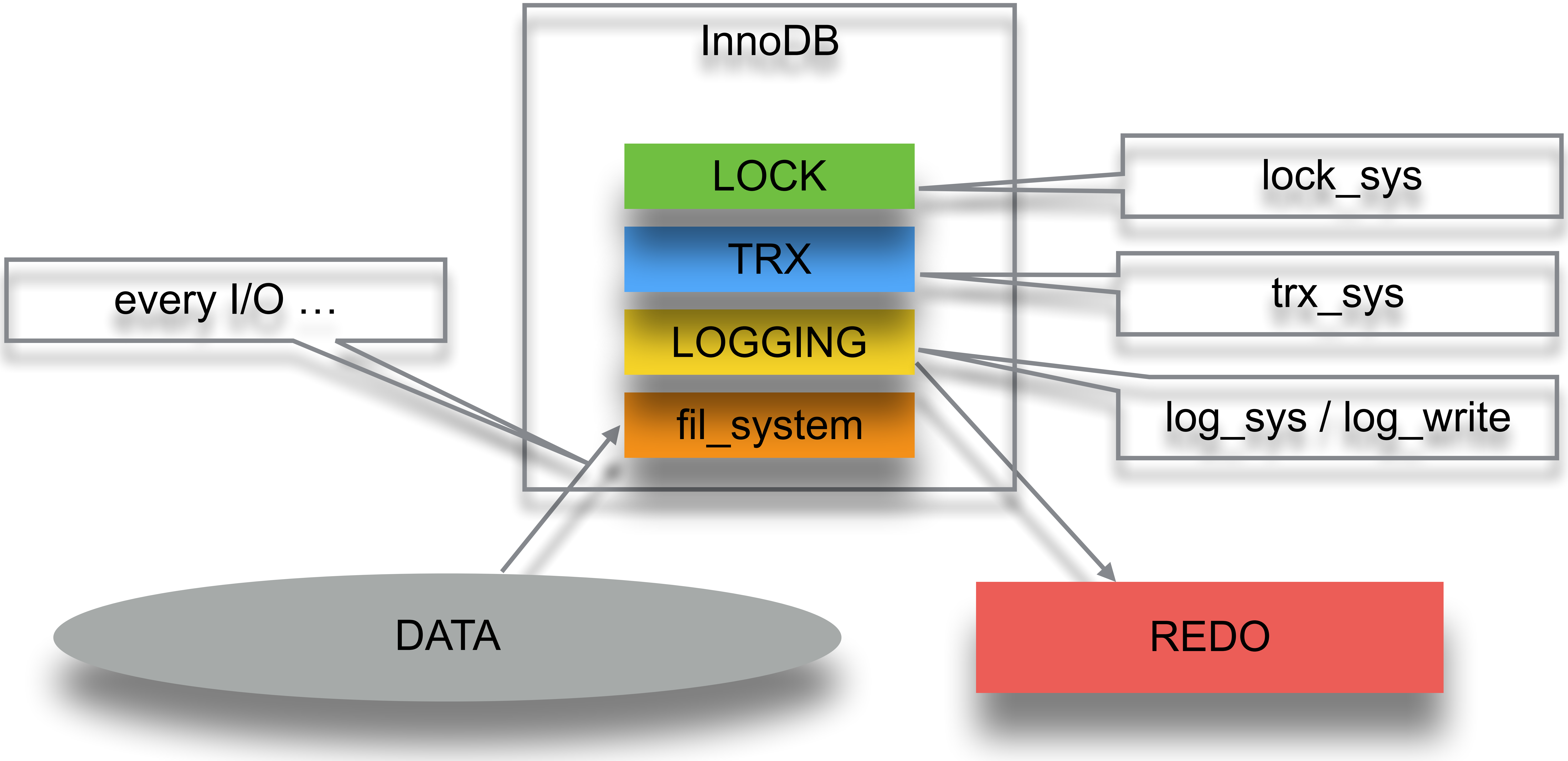
- RW :
  - Double Write..
  - REDO log related bottlenecks
  - TRX management contentions
  - LOCK management..
  - RR / RC isolation..
  - UPDATE Performance..
  - INSERT Performance..
  - Purge lagging..

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** ;-)

- RW :
  - Double Write.. <= **expected in 8.0**
  - REDO log related bottlenecks <= **new REDO 8.0-labs**
  - TRX management contentions <= work-in-progress, prototyped..
  - LOCK management.. <= work-in-progress, prototyped..
  - RR / RC isolation.. <= work-in-progress, prototyped..
  - UPDATE Performance.. <= **8.0-labs, more to come**
  - INSERT Performance.. <= possible workaround : use partitions
  - Purge lagging.. <= not yet solved, but you can truncate UNDO

MySQL-dev : New Design for InnoDB Fundamentals..



InnoDB

LOCK

TRX

LOGGING

fil_system

lock_sys

trx_sys

log_sys / log_write

every I/O …

DATA

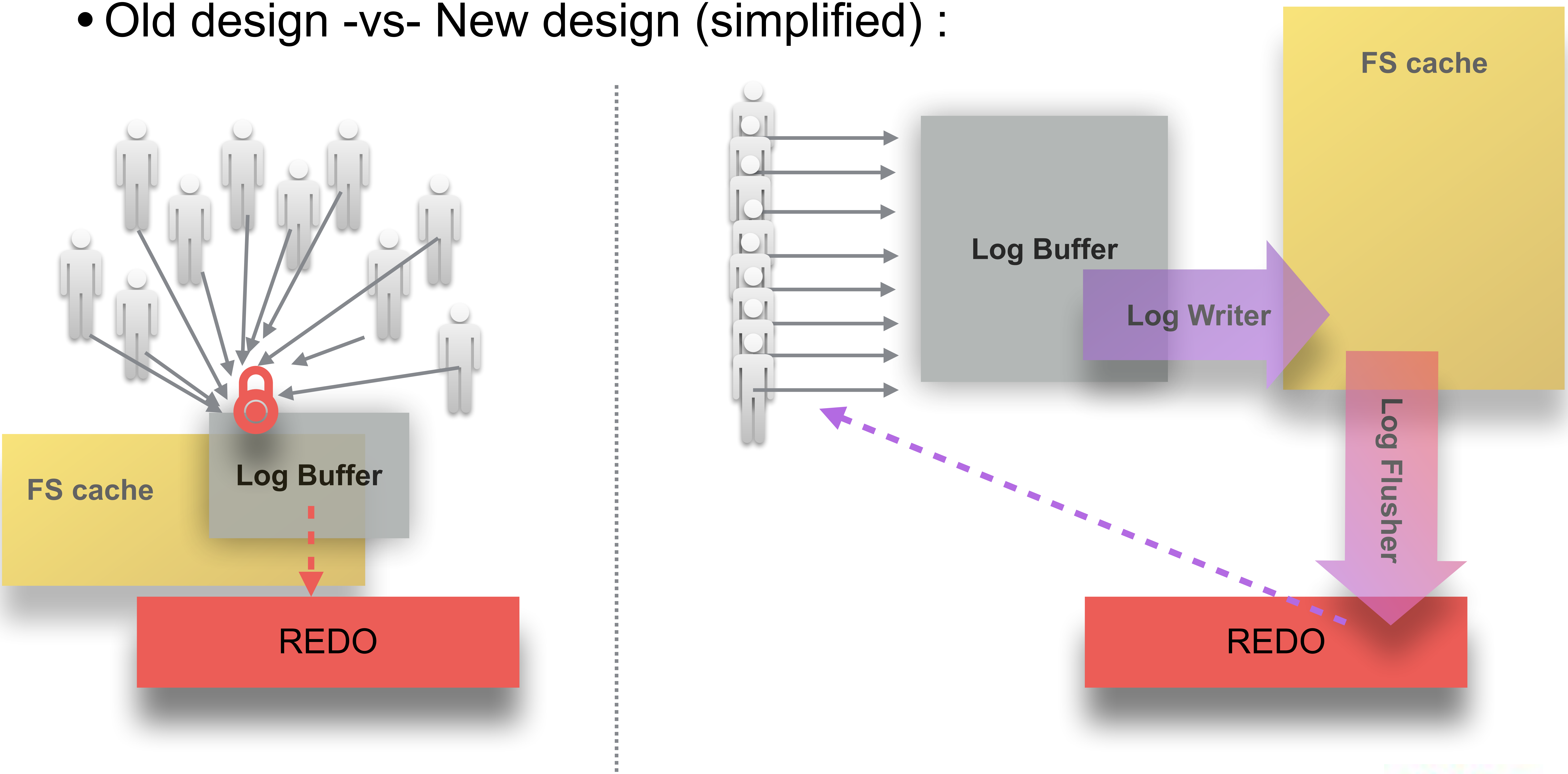
REDO

MySQL 8.0 : Re-Designed REDO

- InnoDB REDO writes :
  - FS cache buffered write() + fsync()
  - innodb_flush_log_at_trx_commit = 1 / 2 / 0
    - = 1 : fsync() on every COMMIT
    - = 2 : do write() on every COMMIT, but fsync() once per second
    - = 0 : do write() once per second, and fsync() once per second
  - historical supposition : the biggest impact is coming from fsync()
    - => group commit, etc.
  - **2015** : Sunny's probe patch is showing trx_commit=1 is faster than trx_commit=2
  - so, what is odd with REDO then ?..
    - user threads fight !
    - with faster storage fsync() becomes much less important -vs- internal contentions..

MySQL 8.0 : Re-Designed REDO

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

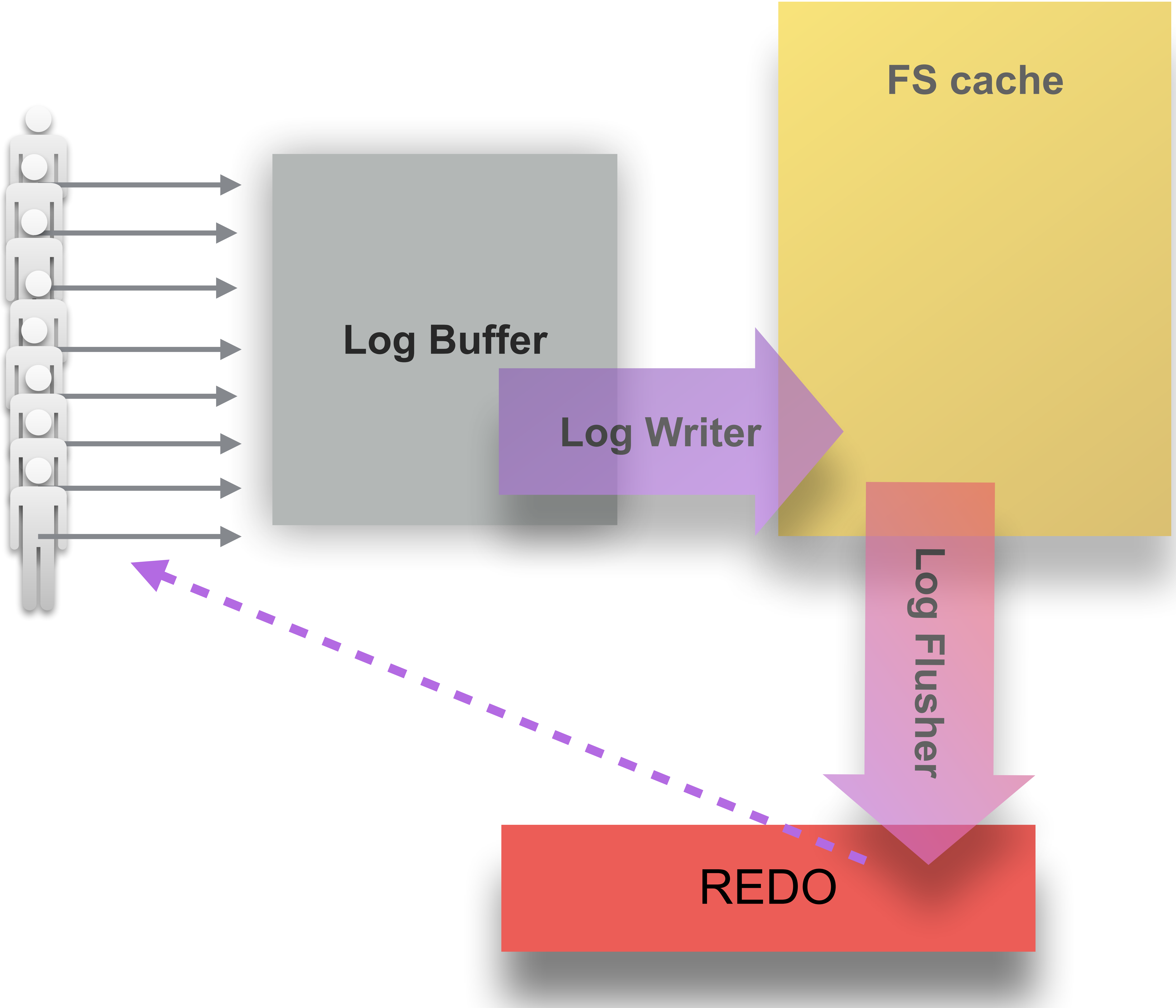


MySQL 8.0 : Re-Designed REDO

- New REDO design :
  - users are not fighting anymore !
  - self-driven processing..
  - self-driven by fsync() capacity

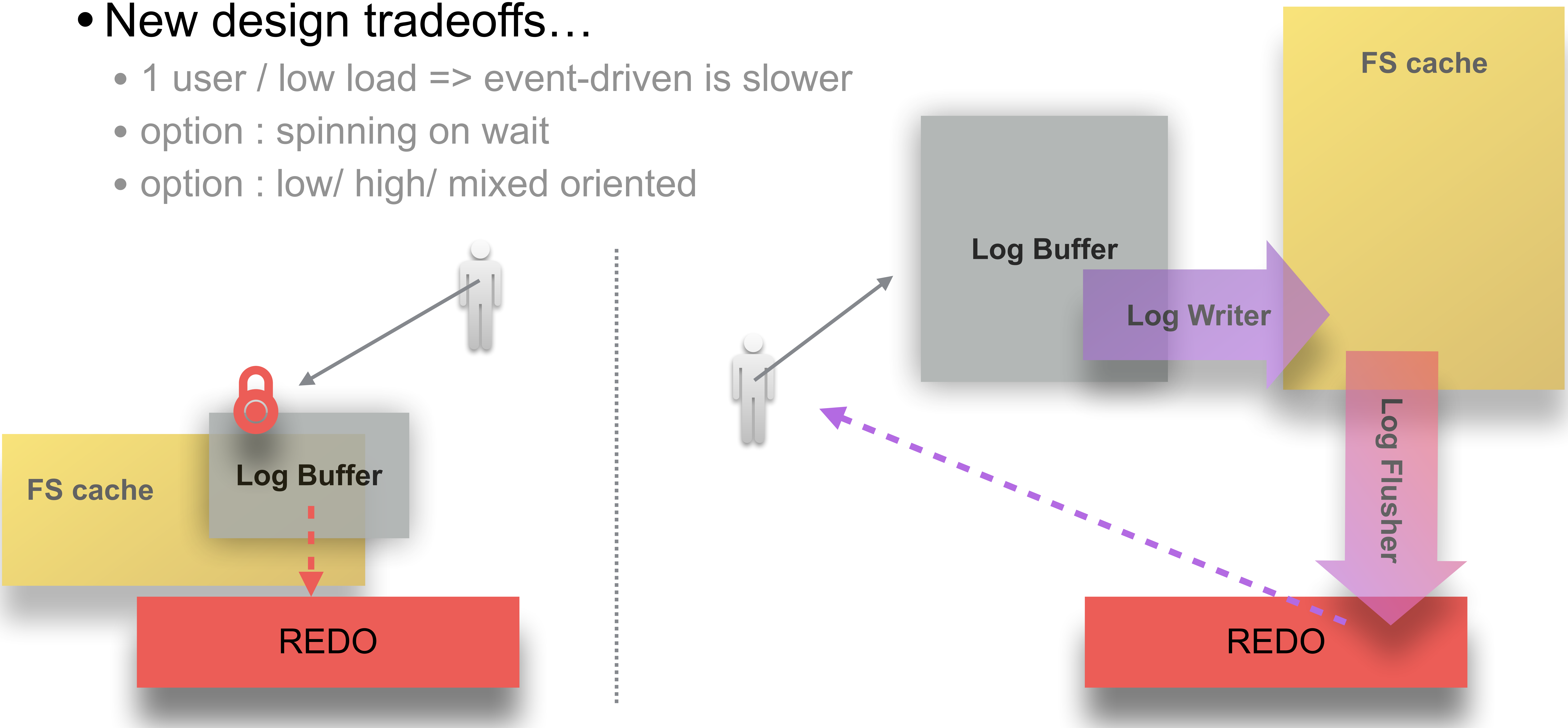
- Instrumented !
  - spins / waits
  - writer / flusher rates
  - max / avg flush times
  - etc..

- Configuration :
  - **mostly all dynamic !!!**
  - so you can play with it on-line ;-))



MySQL 8.0 : Re-Designed REDO

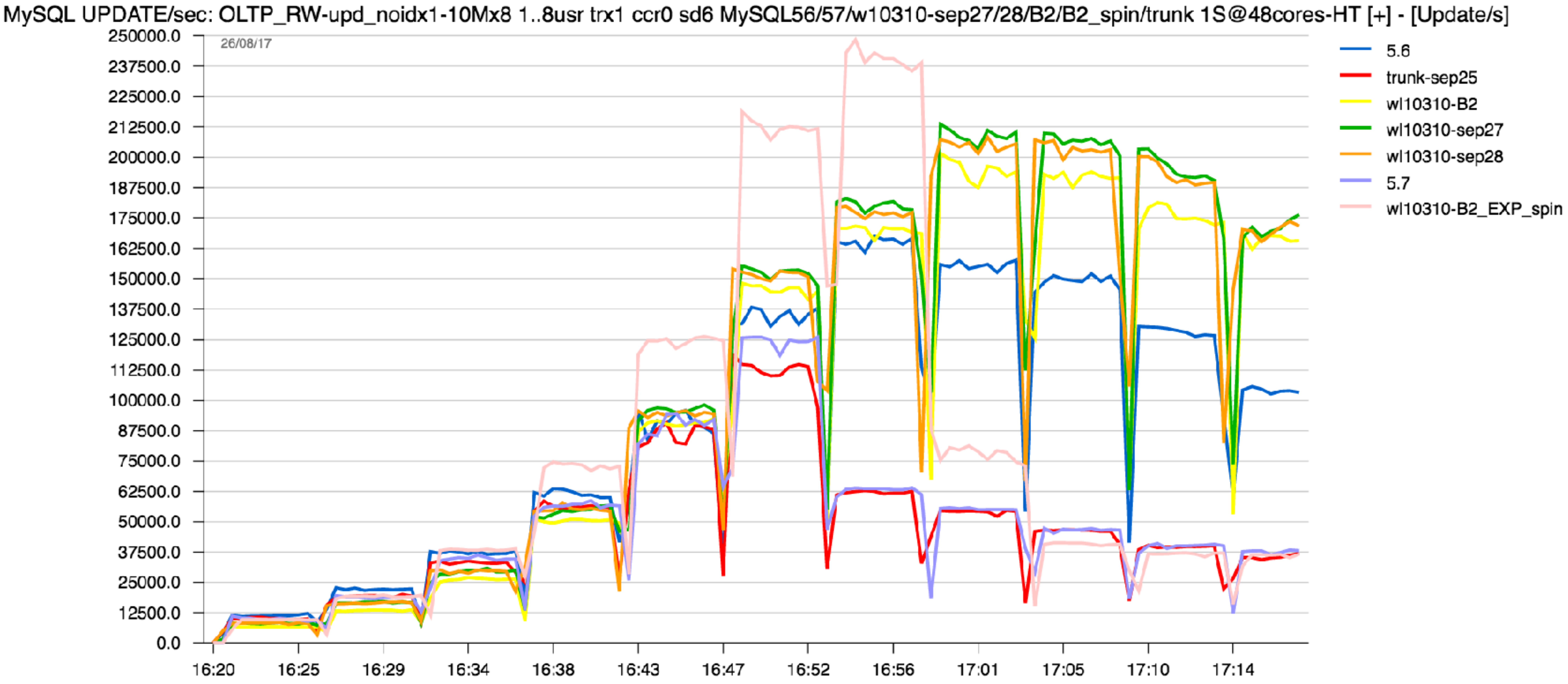
- New design tradeoffs…
  - 1 user / low load => event-driven is slower
  - option : spinning on wait
  - option : low/ high/ mixed oriented



ORACLE

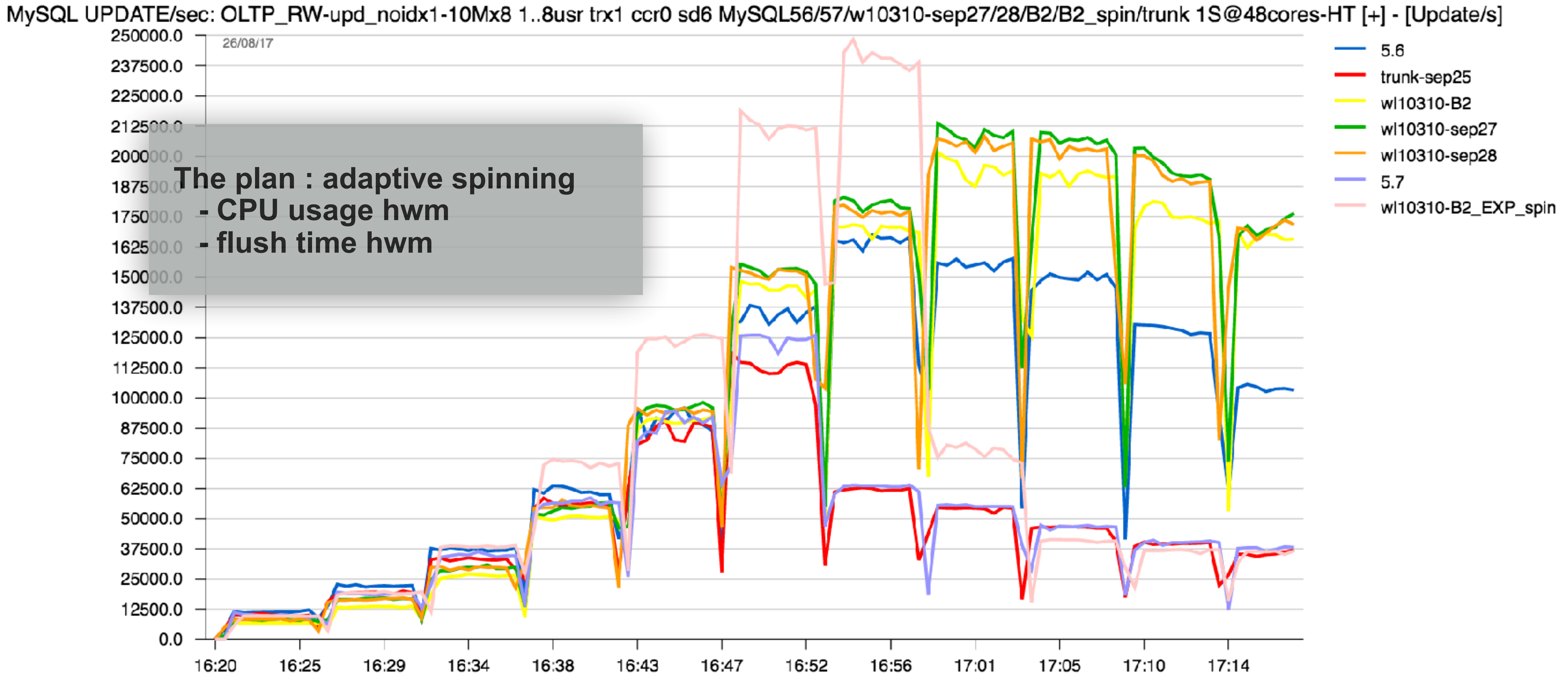
MySQL 8.0 : Re-Designed REDO

- New design tradeoffs…



MySQL 8.0 : Re-Designed REDO

- New design tradeoffs…

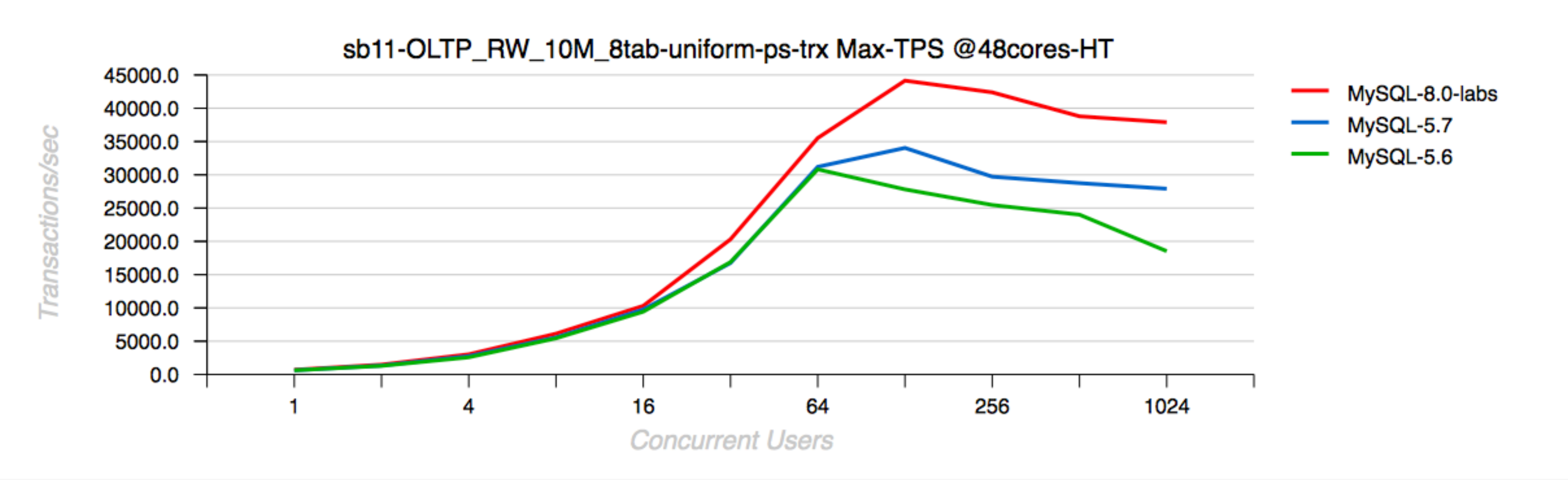


MySQL UPDATE/sec: OLTP_RW-upd_noidx1-10Mx8 1..8usr trx1 ccr0 sd6 MySQL56/57/w10310-sep27/28/B2/B2_spin/trunk 1S@48cores-HT [+] - [Update/s]

The plan : adaptive spinning
- CPU usage hwm
- flush time hwm

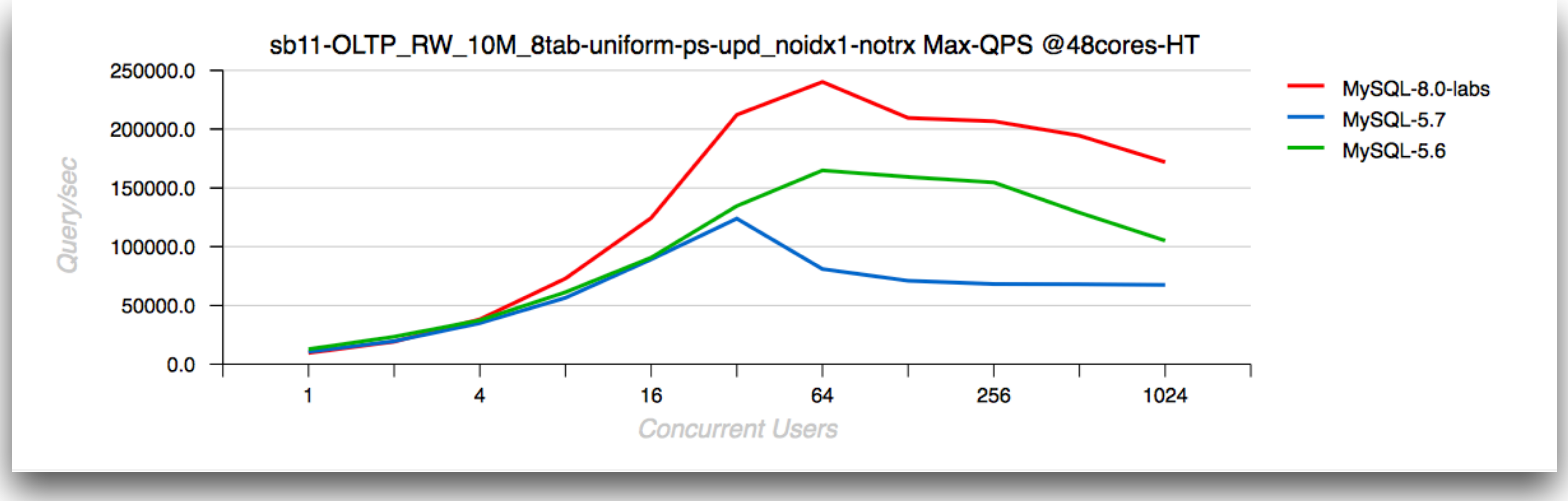
MySQL 8.0-labs Performance

- Sysbench OLTP_RW 10Mx8tab, trx_commit=1, 48cores-HT (Skylake)
  - 30% gain vs MySQL 5.7
  - 50% gain vs MySQL 5.6



MySQL 8.0-labs Performance

- Sysbench Updates-Nokey 10Mx8tab, trx_commit=1, 48cores-HT (Skylake)
  - 100% gain vs MySQL 5.7
  - 50% gain vs MySQL 5.6 (and yes, 5.7 is bad here.. => fixed !! ;-))

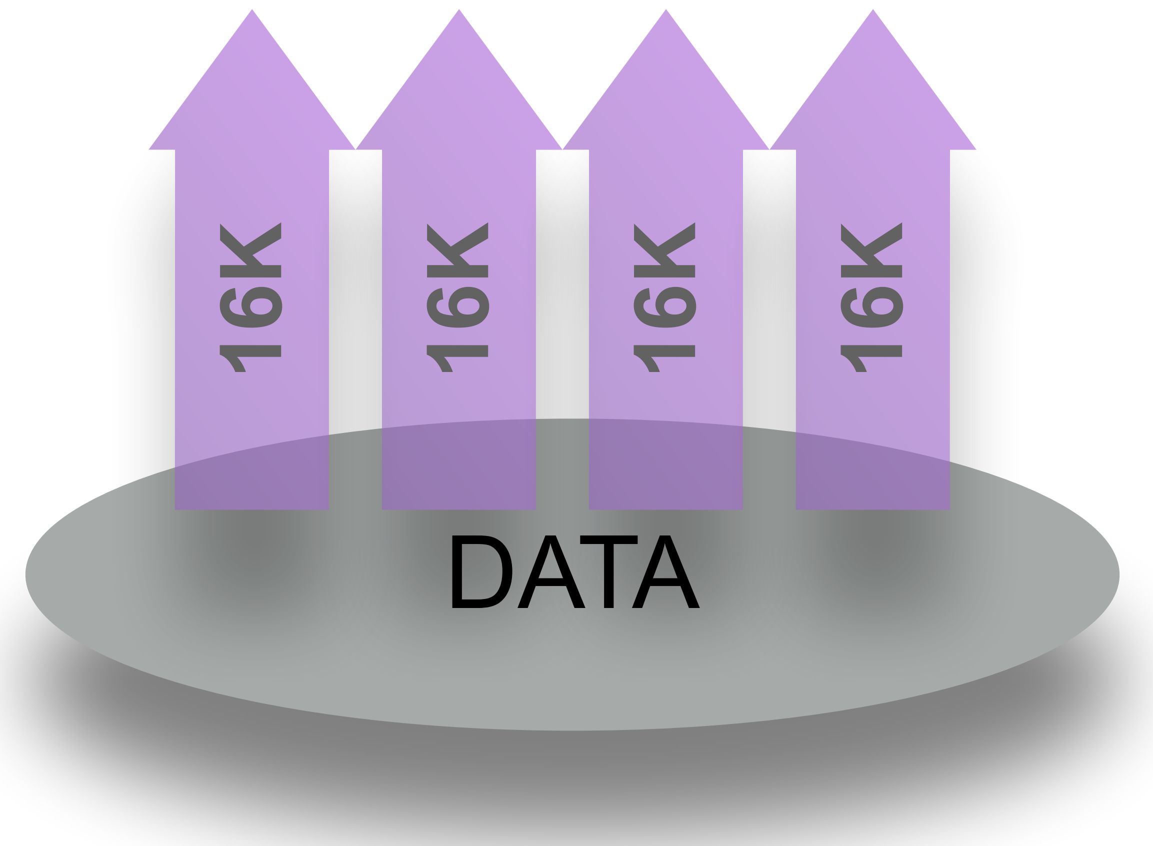


MySQL 8.0 Writes Scalability

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

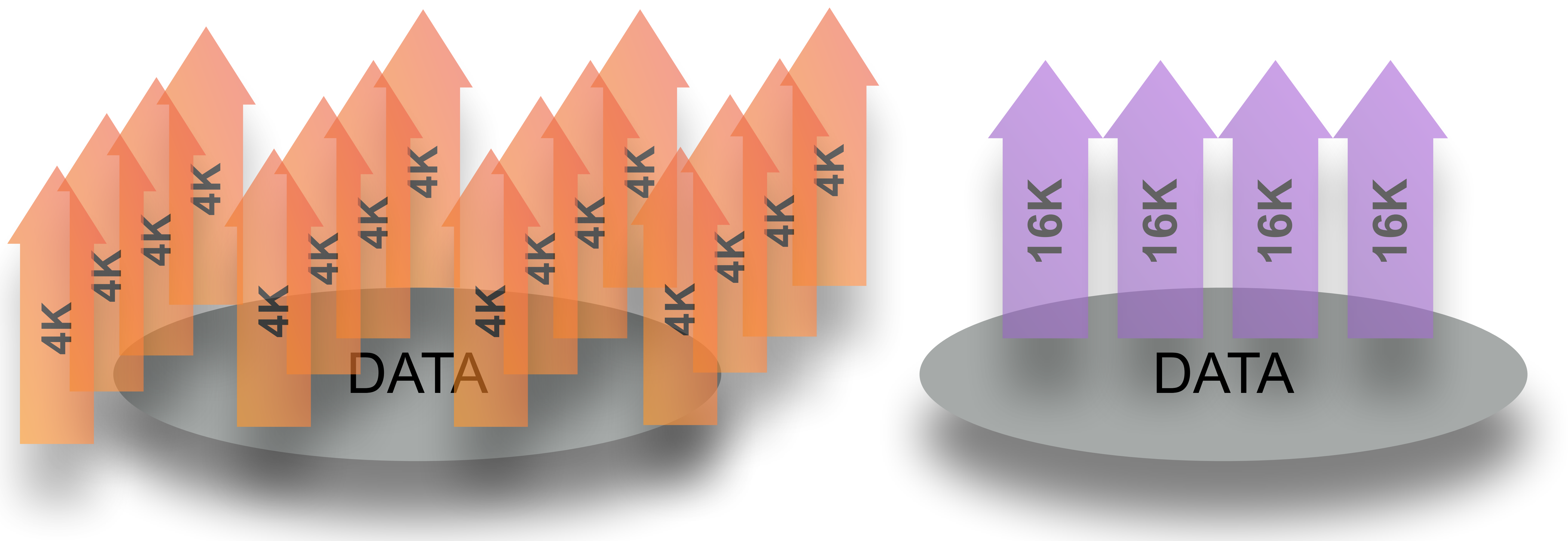
IO-bound Workloads : Game Changer..

- IO reads :
  - game changer : **FLASH** => goes faster / cheaper / more stable / living longer / etc..
  - e.g. no more “seek time” cost, the main IO limit : device throughput
  - supposing your max throughput is XXX MB/sec, what is the max IO-bound QPS possible ?
  - => driven by IO read **Operations/sec** …



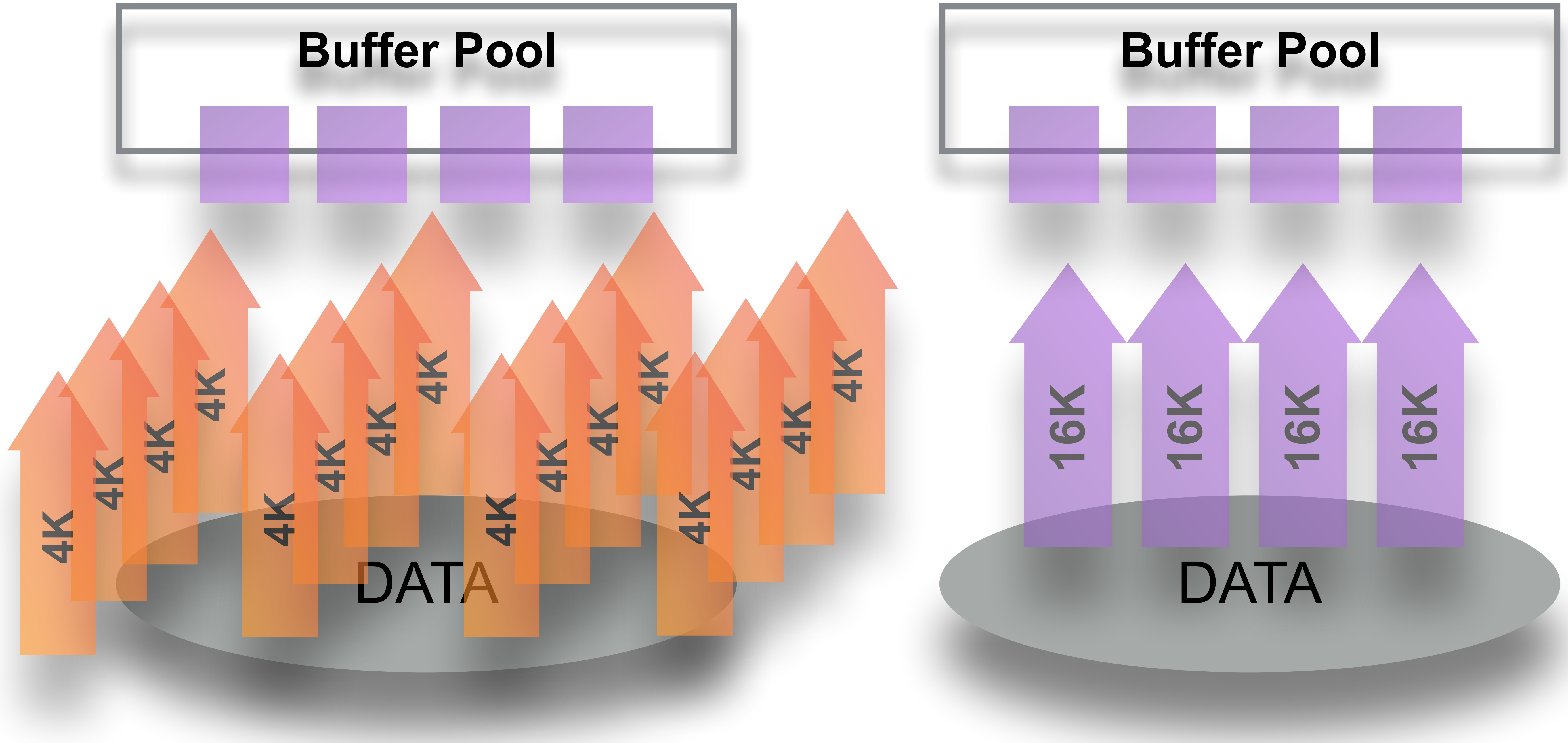
IO-bound Workloads : more in depth..

- IO reads :
  - game changer : FLASH => goes faster / cheaper / more stable / living longer / etc..
  - e.g. no more “seek time” cost, the main IO limit : device throughput
  - supposing your max throughput is XXX MB/sec, what is the max IO-bound QPS possible ?
  - => driven by IO read **Operations/sec** …
  - Compression ? => x4 times more IO reads !!!



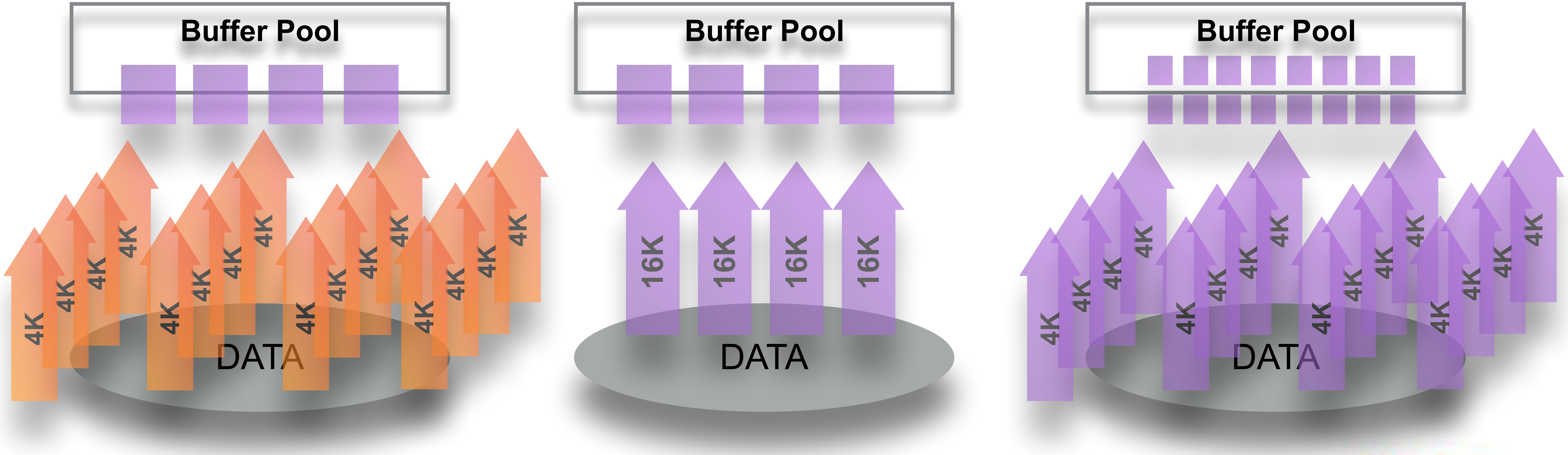
IO-bound Workloads : more in depth..

- IO reads :
  - game changer : FLASH => goes faster / cheaper / more stable / living longer / etc..
  - e.g. no more “seek time” cost, the main IO limit : device throughput
  - supposing your max throughput is XXX MB/sec, what is the max IO-bound QPS possible ?
  - => driven by IO read **Operations/sec** …
  - Compression ? => x4 times more IO reads !!! => **and QPS** ?..



IO-bound Workloads : more in depth..

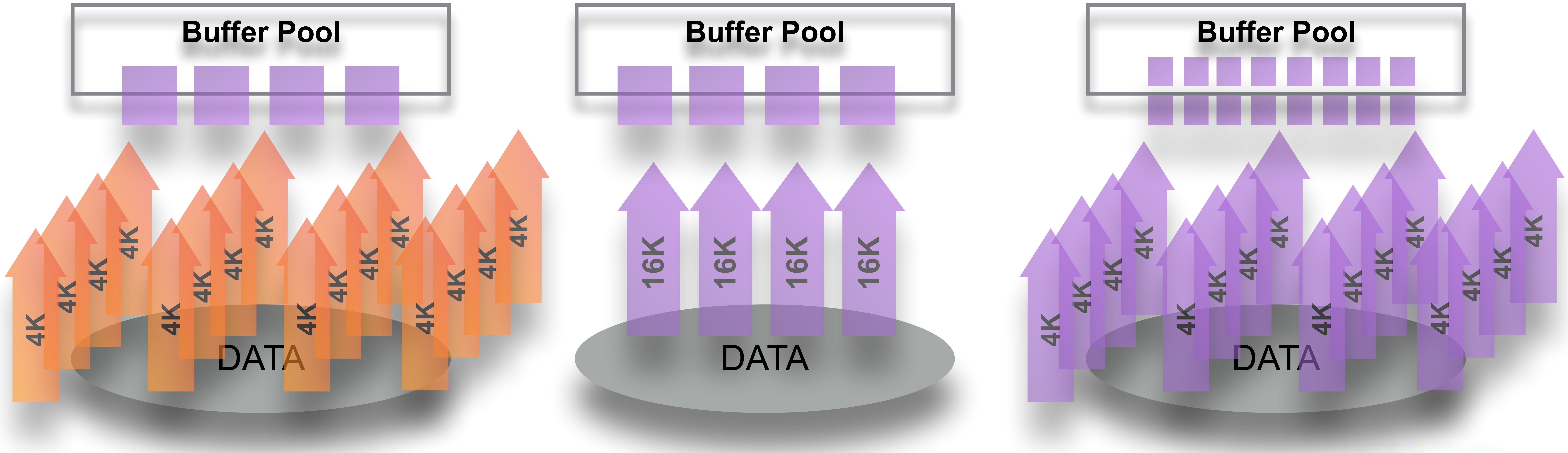
- IO reads :
  - game changer : FLASH => goes faster / cheaper / more stable / living longer / etc..
  - e.g. no more “seek time” cost, the main IO limit : device throughput
  - supposing your max throughput is XXX MB/sec, what is the max IO-bound QPS possible ?
  - => driven by IO read **Operations/sec** …
  - Compression ? => x4 times more IO reads !!! => and QPS ?.. **and what about 4K page ?**



ORACLE

IO-bound Workloads : more in depth..

- IO reads :
  - so, with fast FLASH + 4K page size => x4 times better RO performance vs default 16K ?
  - potentially YES ;-))
  - but.. => historically : **fil_system** global mutex lock on **every IO operation !!!**
  - good news : **fixed with 8.0 ! ;-))**



IO-bound Workloads : Test Case

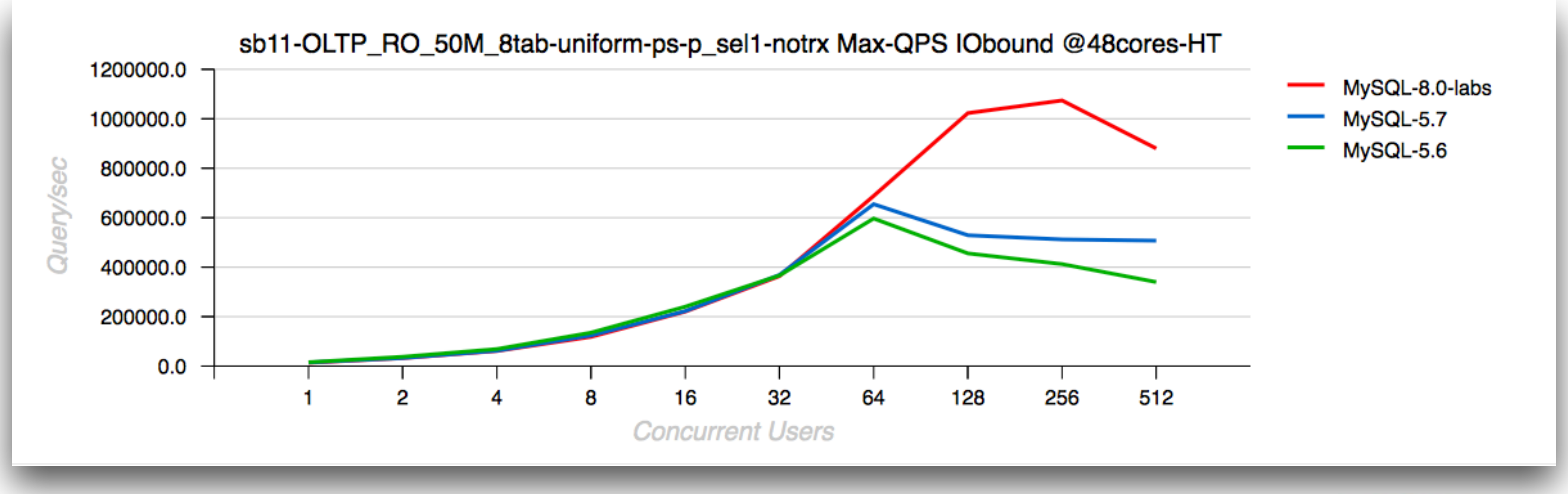
- Intel Optane drive :
  - IO read latency : 0,01ms (!!!)
  - 1 single process doing 16KB IO reads : ~65K reads/sec, 1000 MB/sec
  - however, the max throughput : 2000 MB/sec only (fix in progress by Intel)

- with x2 drives :
  - over 4000 MB/sec throughput
    - 16K page : ~260K IO reads/s
    - 8K page : over 500K IO reads/s
    - 4K page : **over 1M IO reads/s**

  - can MySQL get a profit of such an IO power ?..

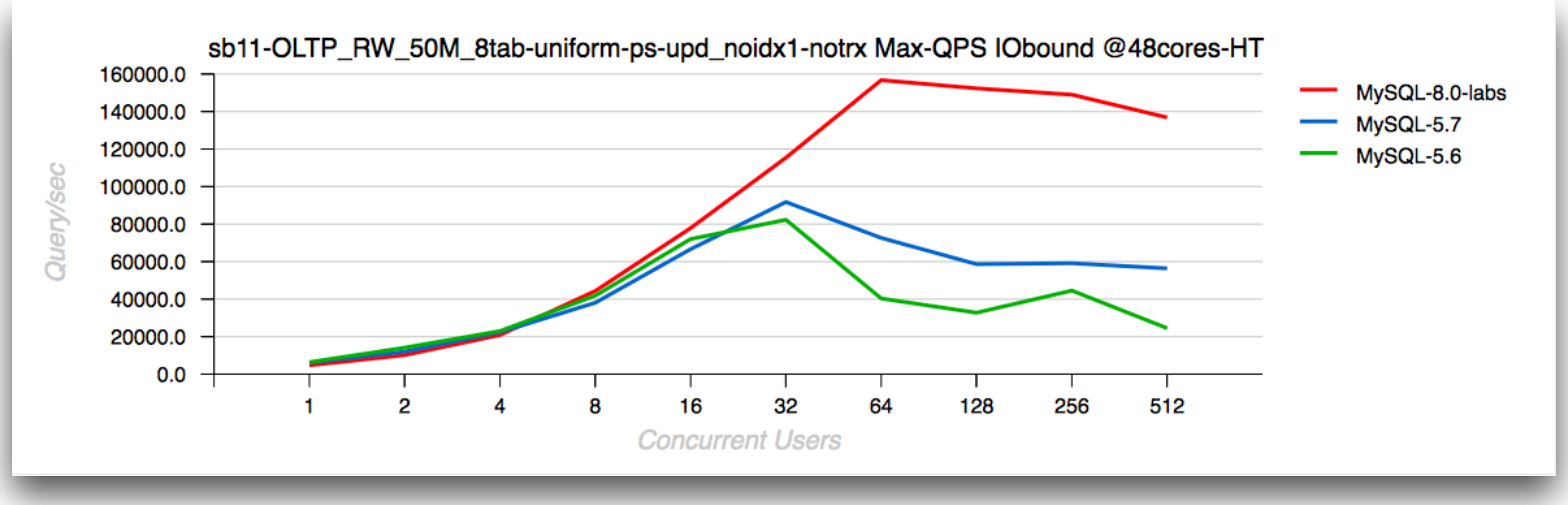
MySQL 8.0-labs Performance

- IO-bound Sysbench OLTP_RO Point-Selects
  - 50M x 8-tables, 48cores-HT, x2 Optane drives
  - NOTE : storage saturated & 100% CPU (new face of IO-bound ? ;-))
  - over **1M IO-bound QPS** with MySQL 8.0-labs !!!



MySQL 8.0-labs Performance

- IO-bound Sysbench OLTP_RW Update-NoKEY
  - 50M x 8-tables, 48cores-HT, x2 Optane drives
  - over **160K IO-bound QPS** with MySQL 8.0-labs !!!



ORACLE

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..)

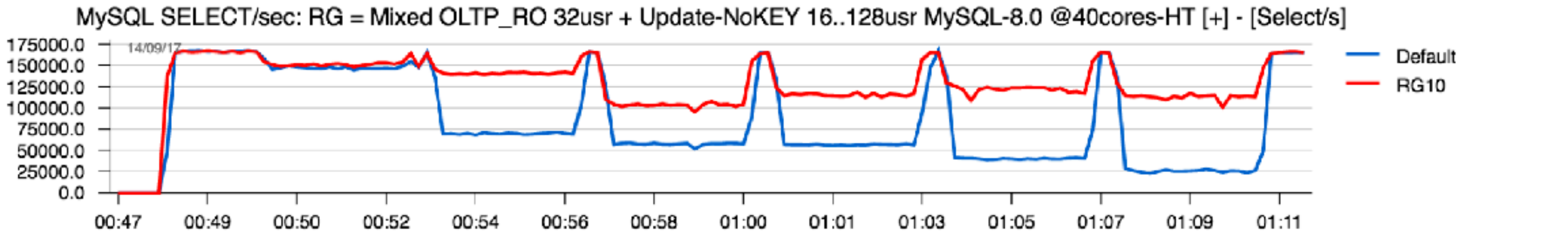
- Using : only by name !
  - mysql> SET RESOURCE GROUP name ; (also for any THREAD ID)
  - SELECT /*+ RESOURCE_GROUP(name) */ … ; (query hint)

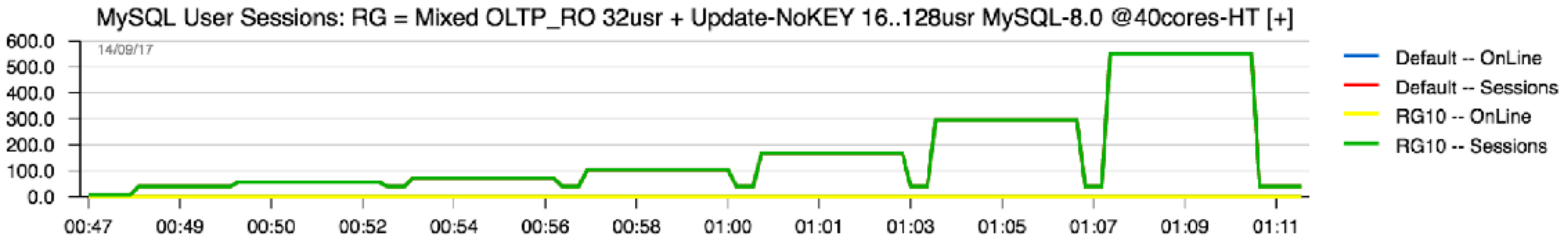
MySQL Resource Groups in Action

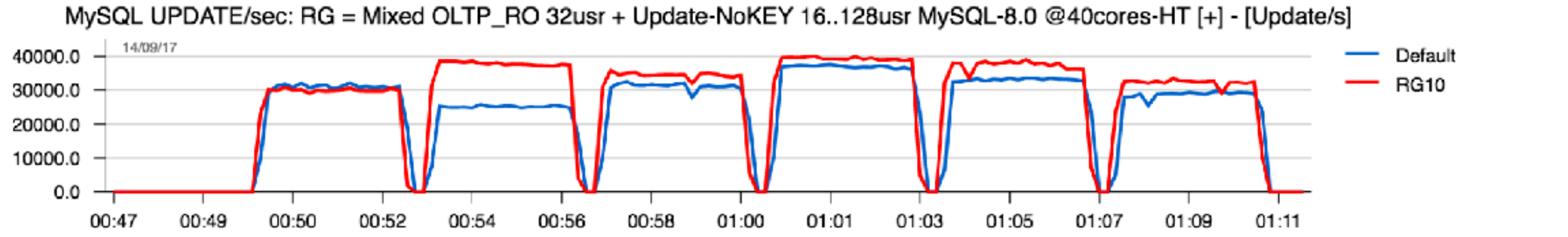
- Test case :
  - 40cores-HT 4S (Broadwell) server, OL7
  - 32 concurrent users are running SELECTs (Sysbench OLTP_RO)
  - other users are coming with UPDATEs (Sysbench Update-NoKEY)
    - 16 users, then 32, 64, 128, 256, 512
  - **Problem :** each workload is running well alone, but NOT together (yet)..
  - **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..

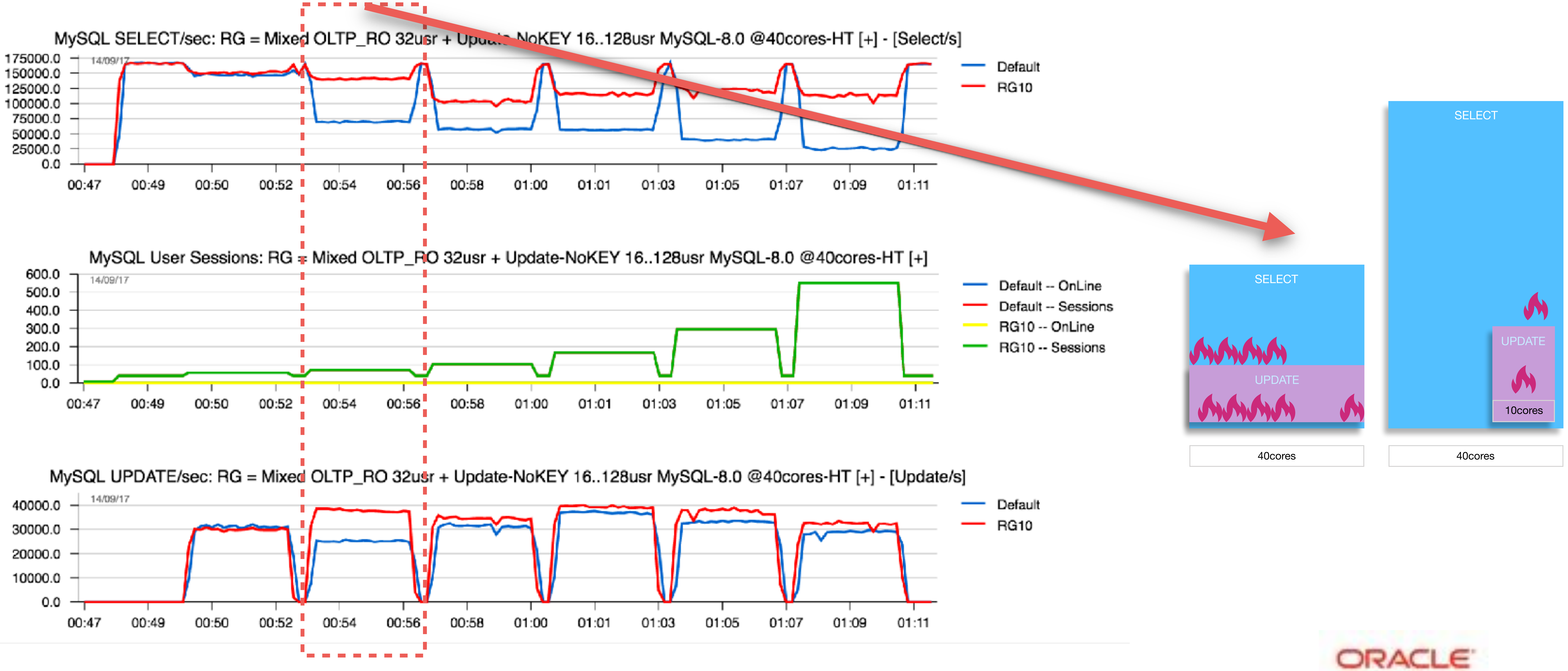

MySQL SELECT/sec: RG = Mixed OLTP_RO 32usr + Update-NoKEY 16..128usr MySQL-8.0 @40cores-HT [+] - [Select/s]


MySQL User Sessions: RG = Mixed OLTP_RO 32usr + Update-NoKEY 16..128usr MySQL-8.0 @40cores-HT [+]


MySQL UPDATE/sec: RG = Mixed OLTP_RO 32usr + Update-NoKEY 16..128usr MySQL-8.0 @40cores-HT [+] - [Update/s]

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..



TL;DR

- MySQL 8.0 :
  - huge amount of new features !!!

- MySQL 8.0 Performance & Scalability :
  - new REDO design
  - better IO-bound scalability
  - Resource Groups : a completely new angle in MySQL Workloads Tuning
  - yet more work in progress..

Hope you're seeing much more clear now ;-)

- Call To Action :
  - 2) download 8.0-rc / 8.0-labs
  - 3) test it in your own workloads
  - 4) send us feedback !!!

    …
  - 1) have fun ! ;-))

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.