MySQL

ORACLE

MySQL 8.0 Performance: InnoDB Re-Design

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

- To tell you in 15min where we’re & where we’re going ;-))
- Q & A

Common Sources of MySQL Performance Problems..

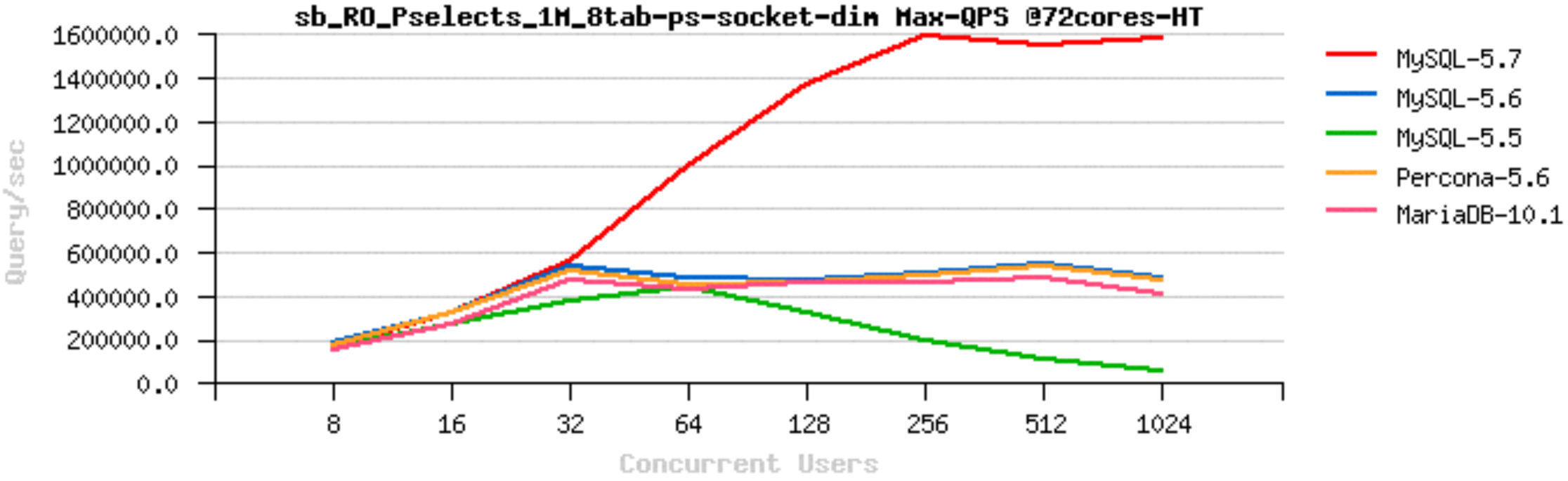
- “Fixable” ones ;-)
  - DB Schema/ Indexes/ SQL query/ Optimizer plan/ Apps code/ etc. etc..
  - odd tuning/ wrong config setup/
  - e.g. generally can be fixed by => RTFM ! ;-)
- “By design” ones..
  - known ?..
  - workaround ?..
  - can be ever fixed ?..
  - heh…
  - work in progress.. <= and here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

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..
  - so RO is faster than RW ! ;-))
- MySQL 8.0
  - main focus is on efficiency : do more on the same HW ;-))
  - work in progress..

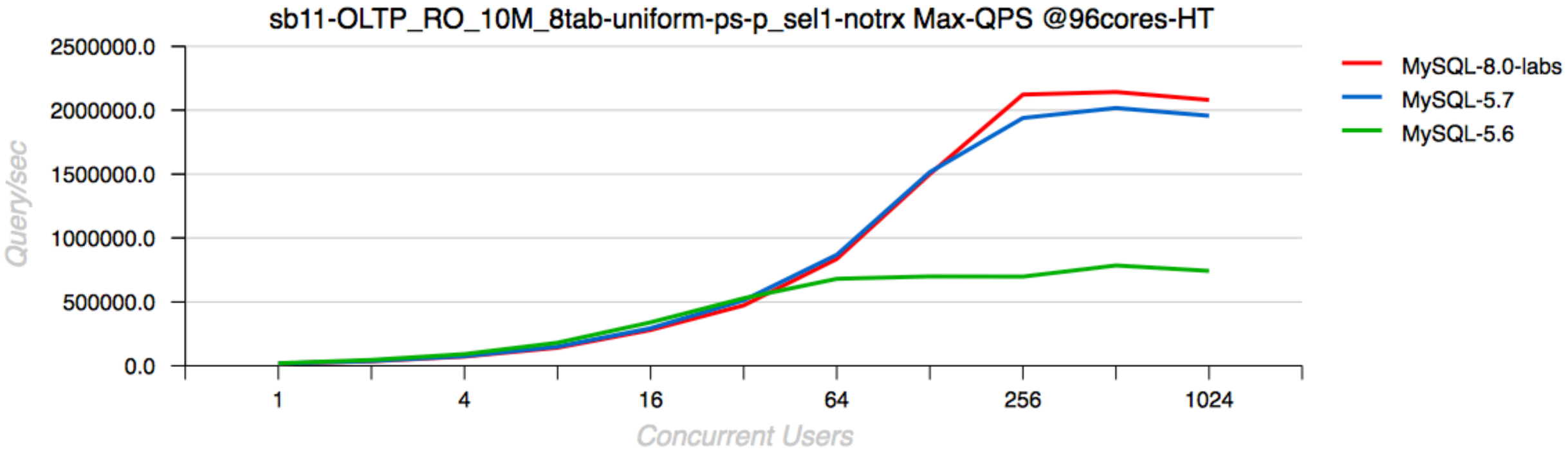
RO Point-Selects @MySQL 5.7 (Oct.2015)

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



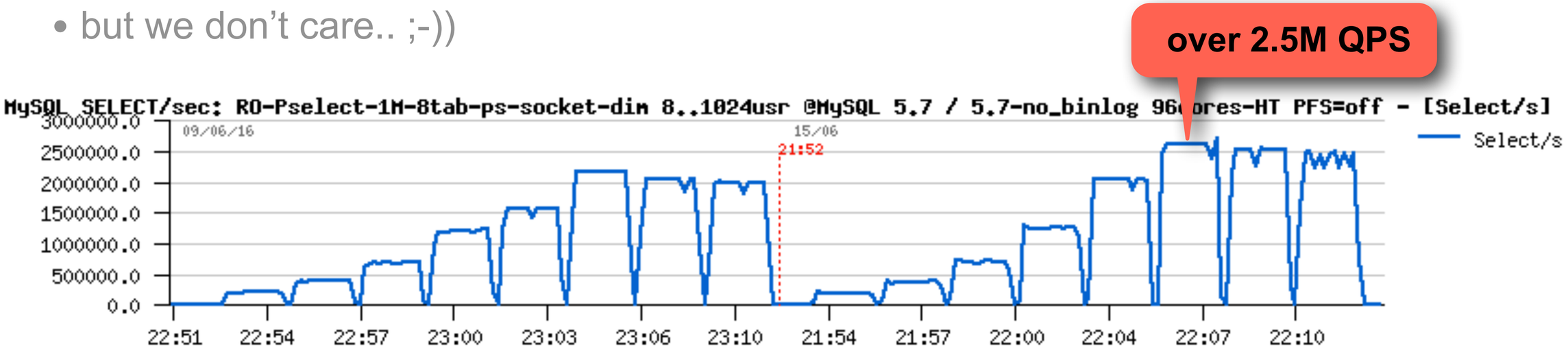
RO Point-Selects @MySQL 8.0 (Sep.2017)

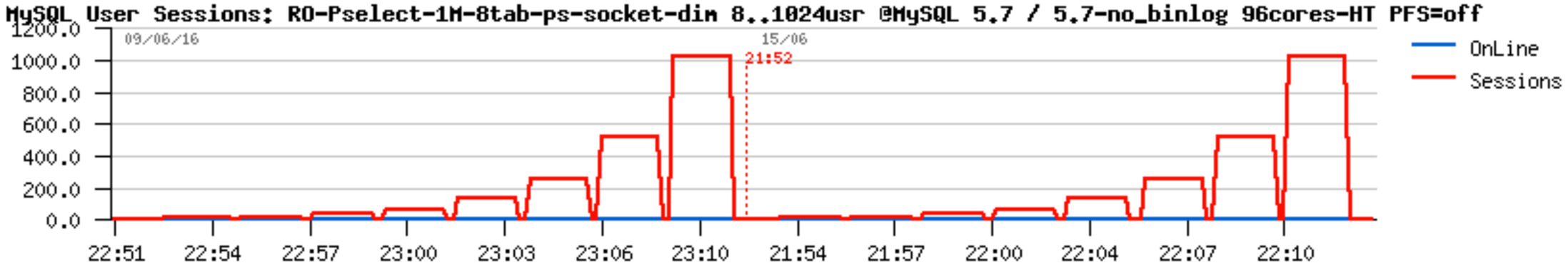
- **2.1M (!!) QPS** Sysbench Point-Selects 8-tab :
  - 96cores-HT Broadwell



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.. ;-))





Pending Scalability Issues after MySQL 5.7 GA..

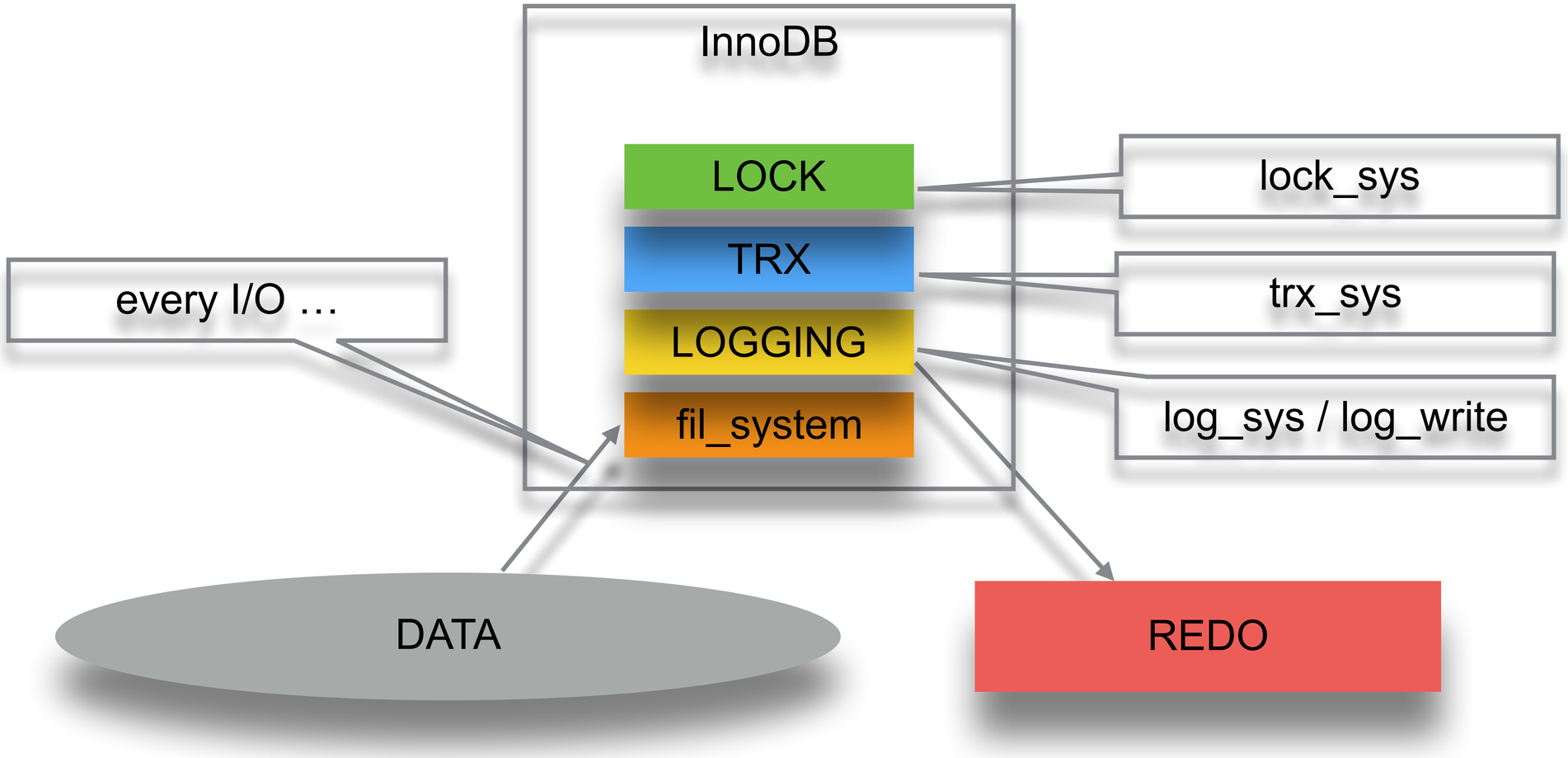
- RO :
  - Block Locks
  - Lookups via Sec.IDX
  - UTF8
- RW :
  - Double Write..
  - REDO log related bottlenecks
  - TRX management contentions
  - LOCK management..
  - RR / RC isolation..
  - UPDATE Performance..
  - INSERT Performance..
  - Purge lagging..

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.. + **CATS** since 8.0.3
  - 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..

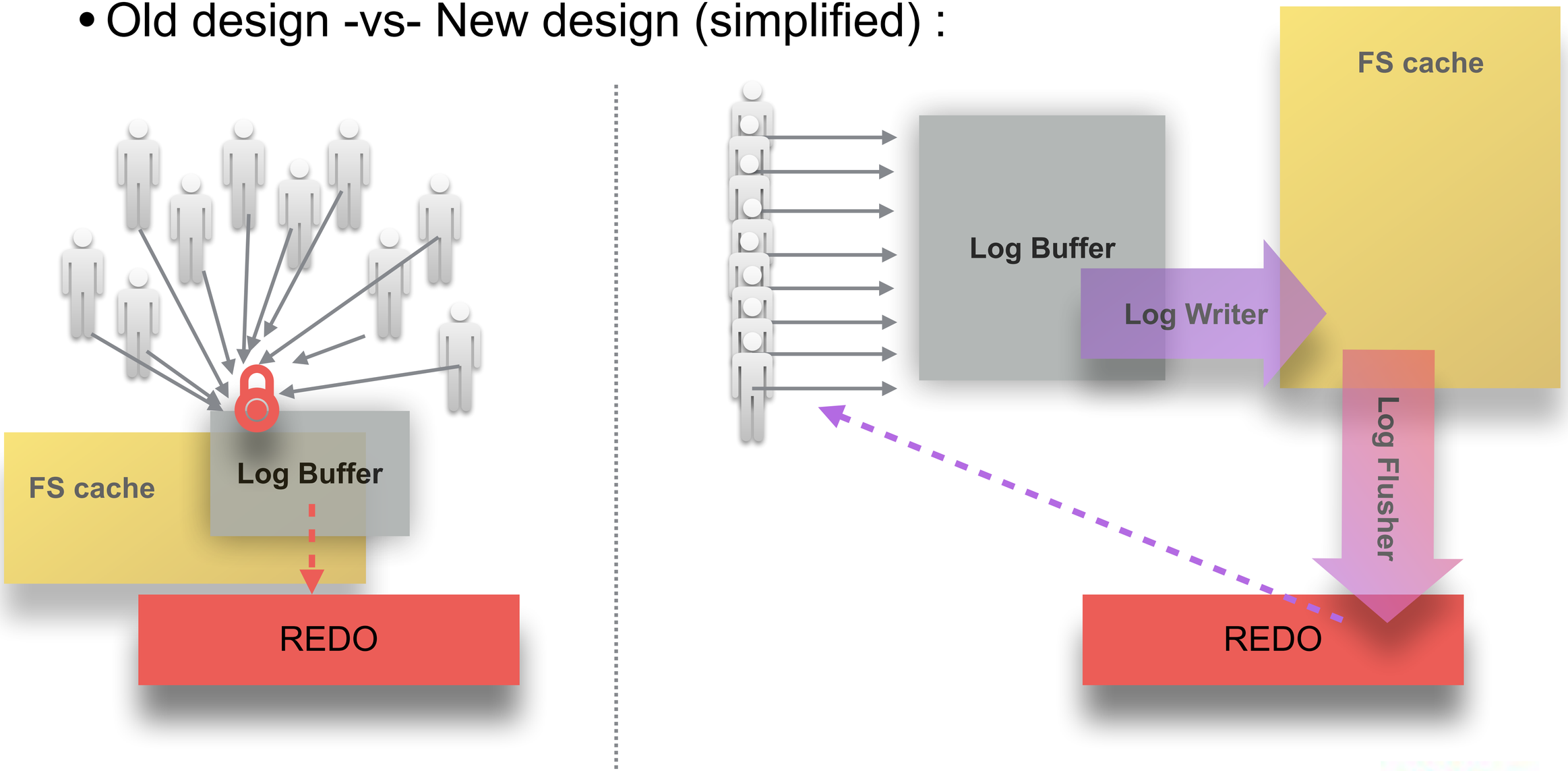


MySQL 8.0 : Re-Designed REDO

- 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..

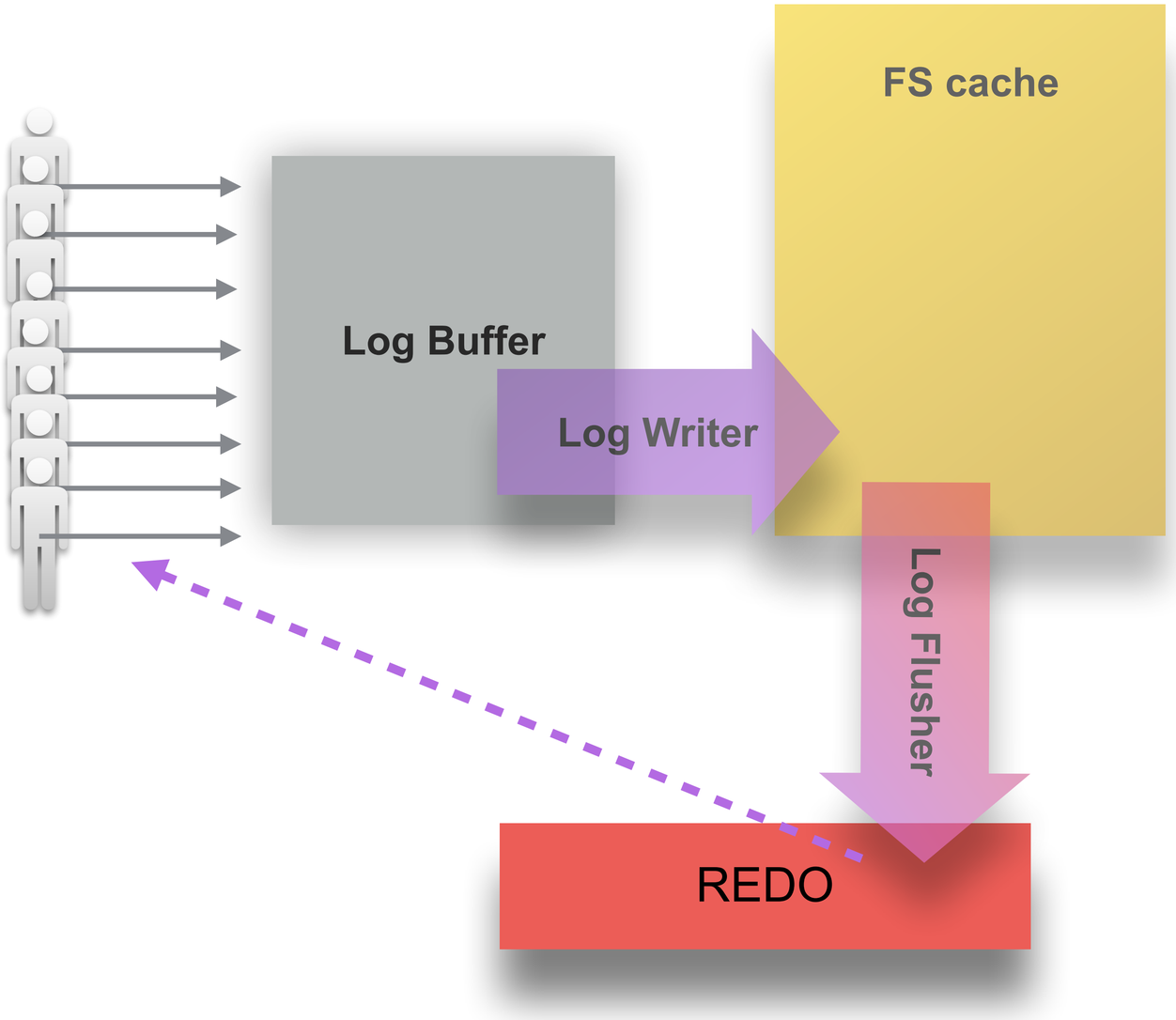
MySQL 8.0 : Re-Designed REDO

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



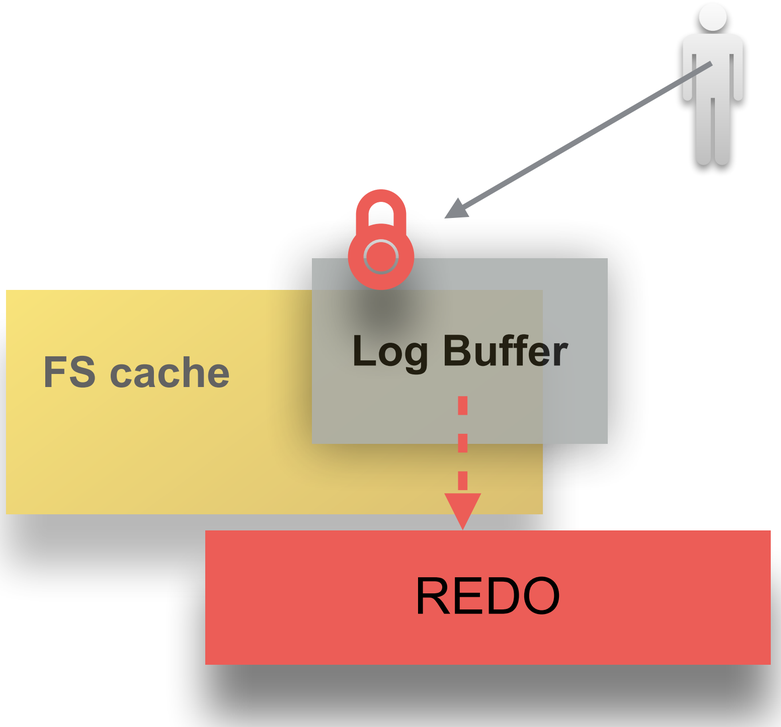
MySQL 8.0 : Re-Designed REDO

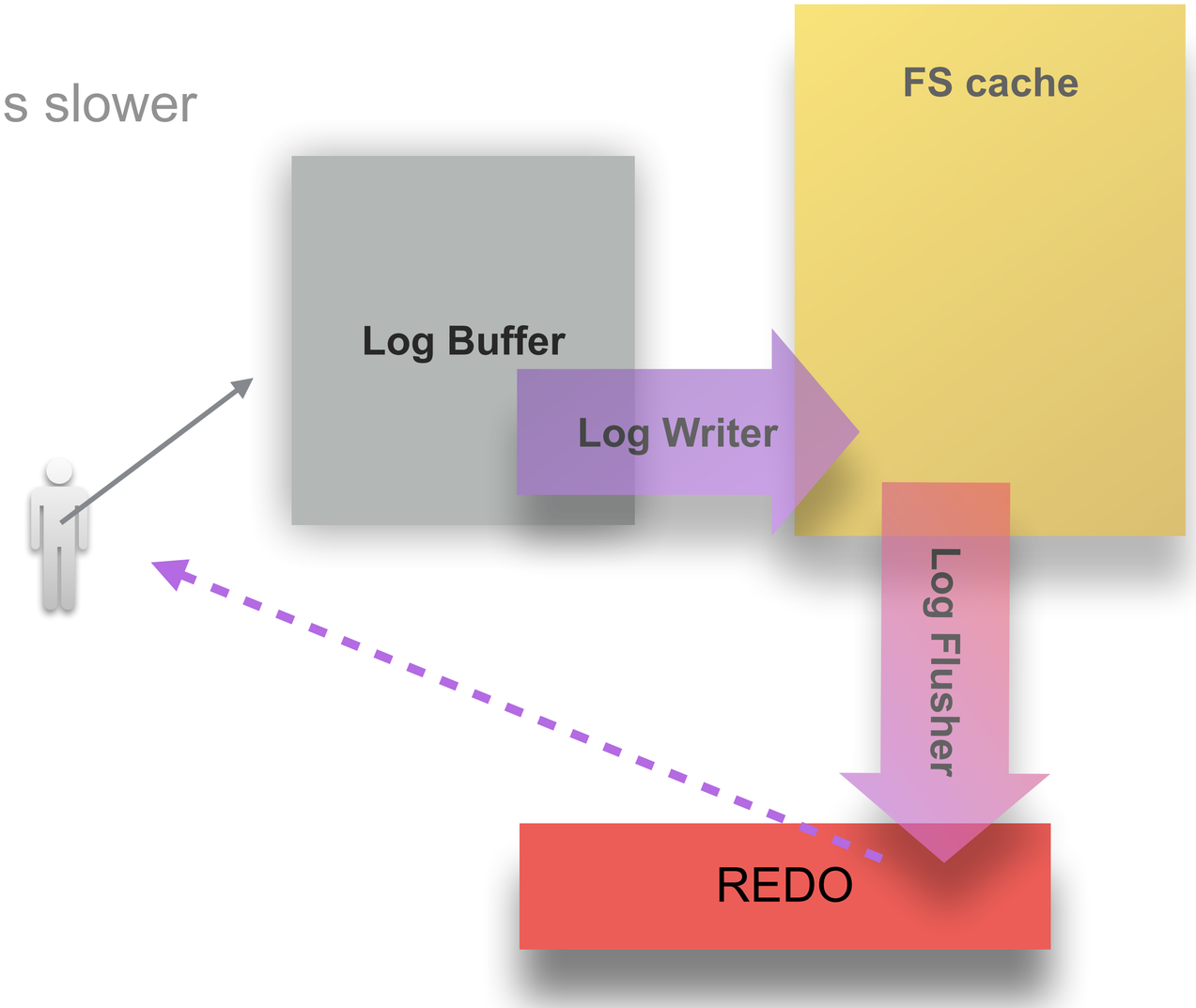
- New REDO design :
  - users are not fighting anymore !
  - self-driven processing..
  - self-driven by fsync() capacity
- Instrumented !
  - spins / waits
  - writer / flusher rates
  - max / avg flush times
  - etc..
- Configuration :
  - **mostly all dynamic !!!**
  - so you can play with it on-line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



MySQL 8.0 : Re-Designed REDO

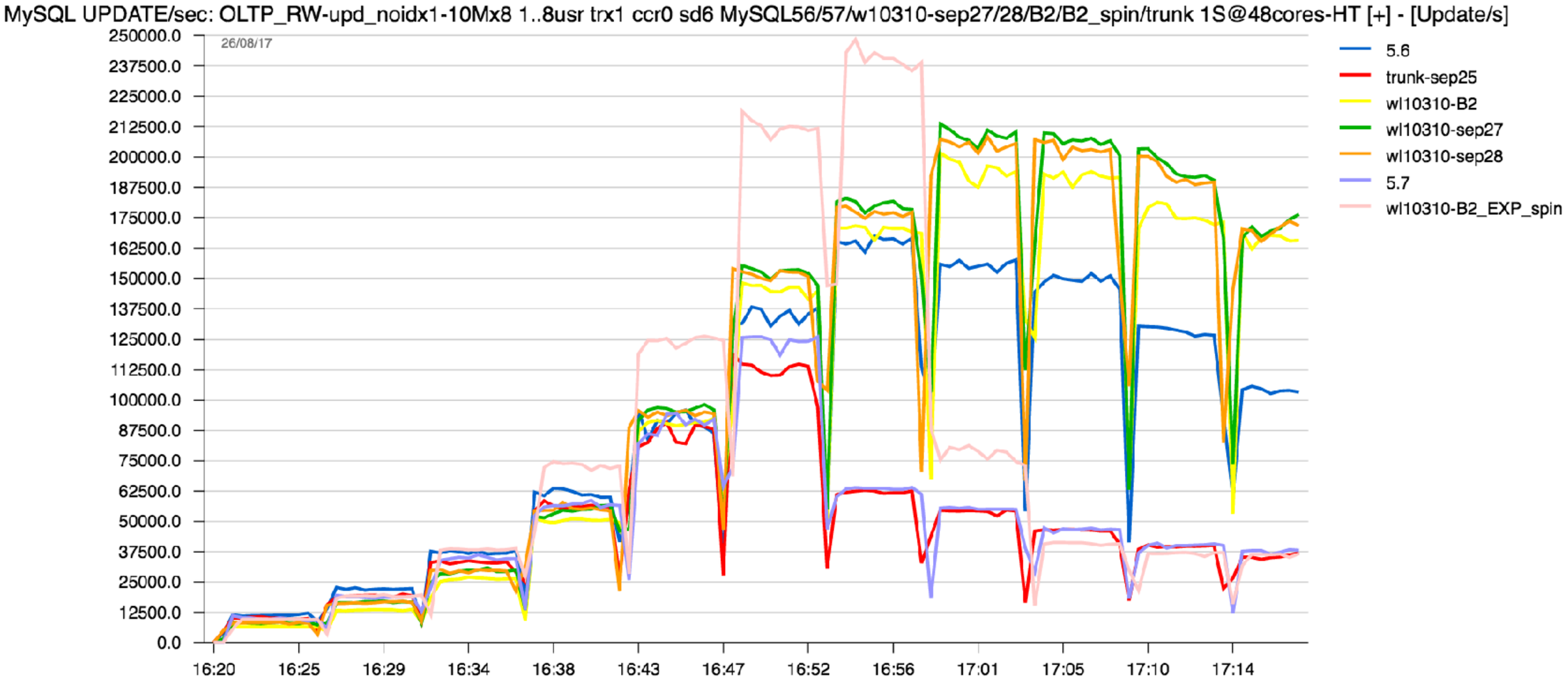
- New design tradeoffs…
  - 1 user / low load => event-driven is slower
  - option : spinning on wait
  - option : low/ high/ mixed oriented





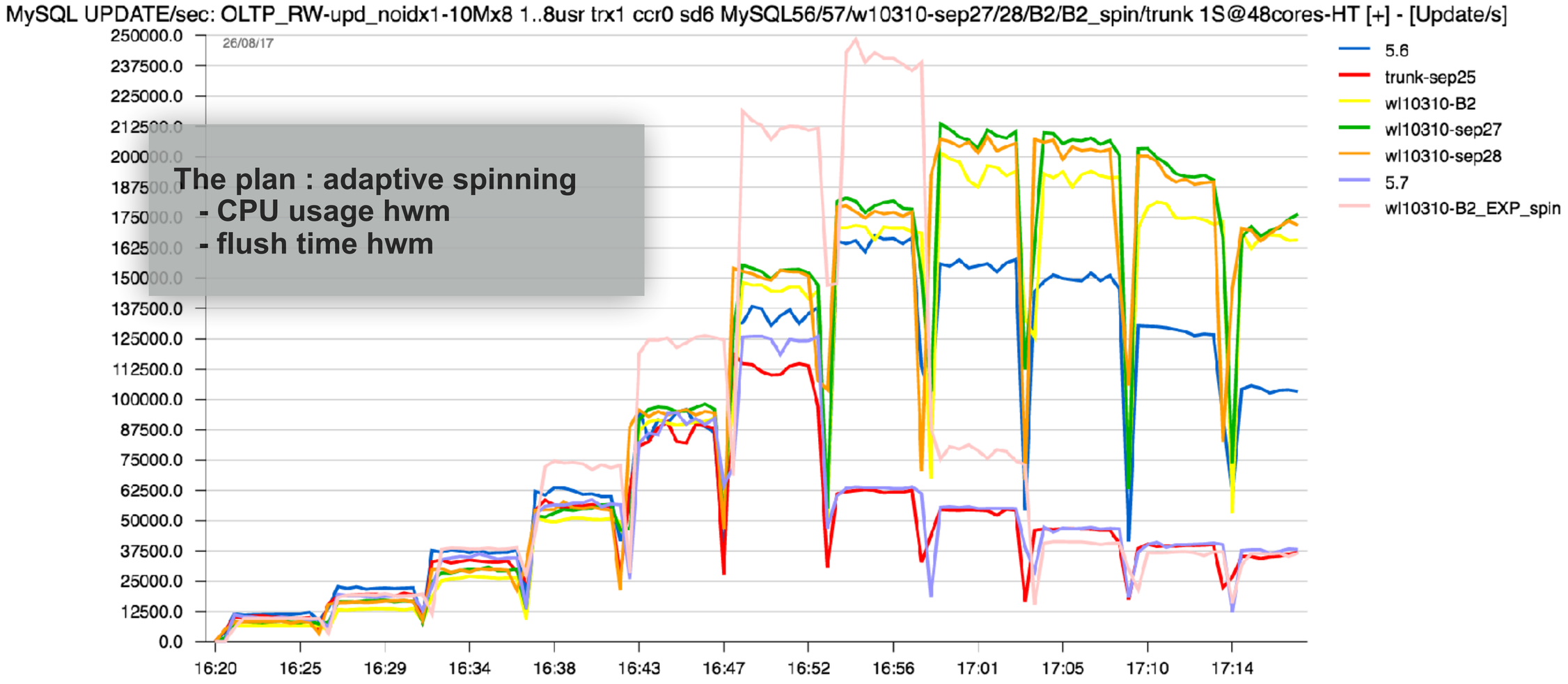
MySQL 8.0 : Re-Designed REDO

- New design tradeoffs…



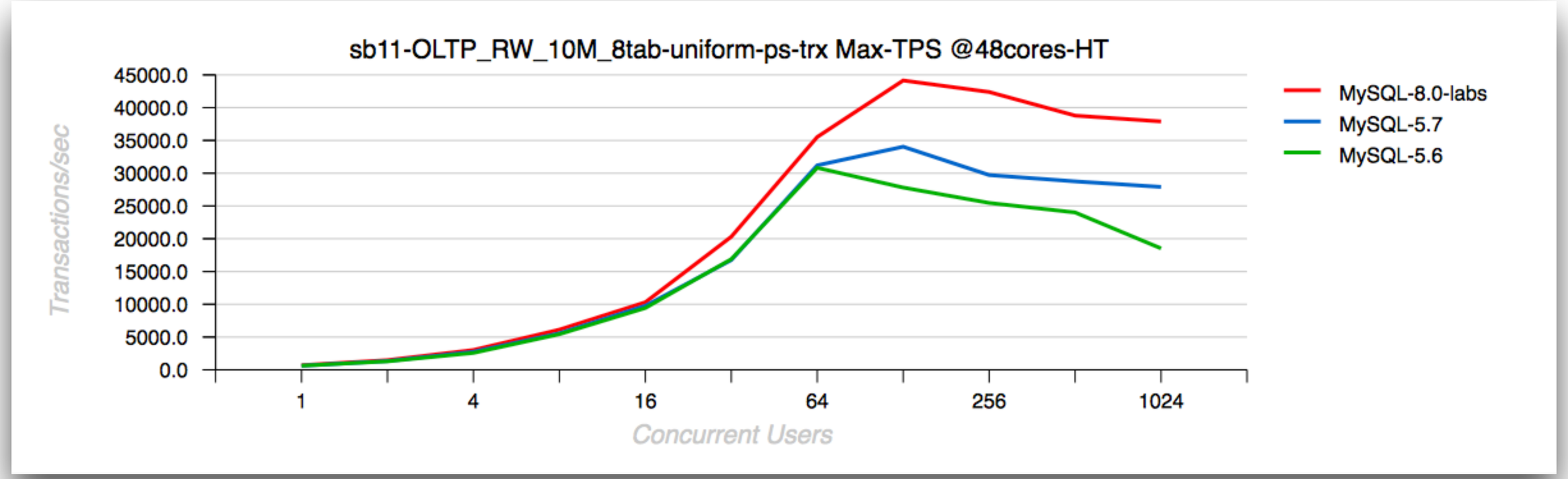
MySQL 8.0 : Re-Designed REDO

- New design tradeoffs…



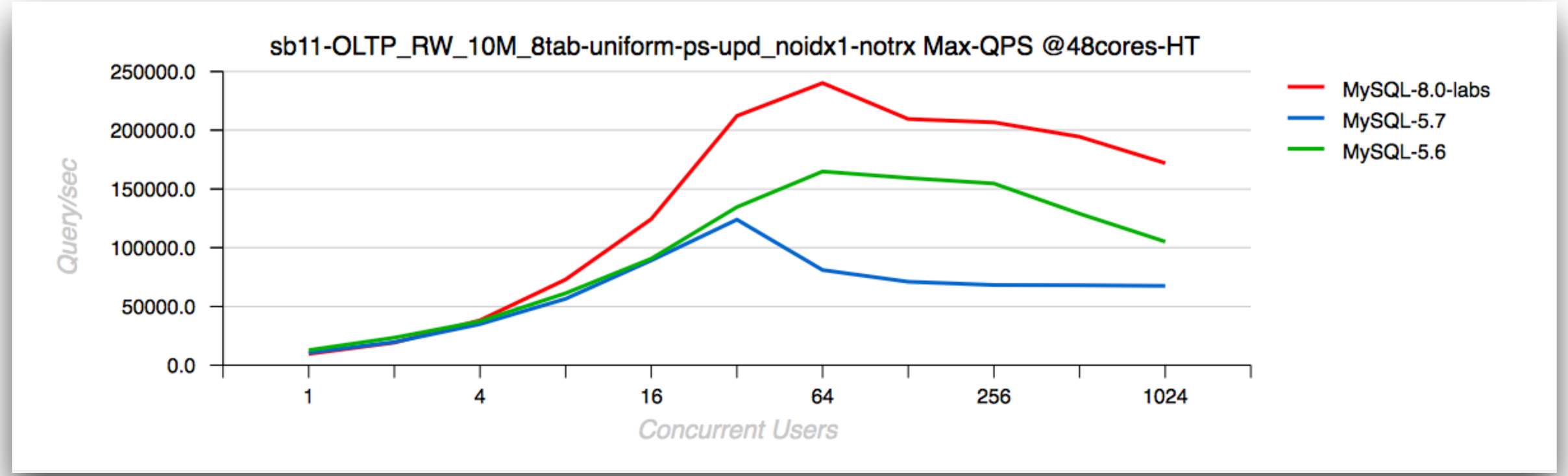
MySQL 8.0-labs Performance

- 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 !! ;-))

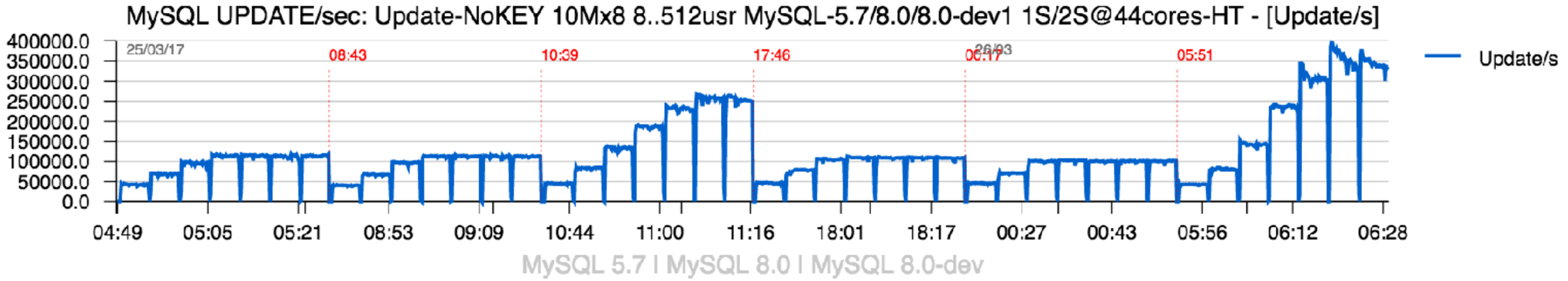


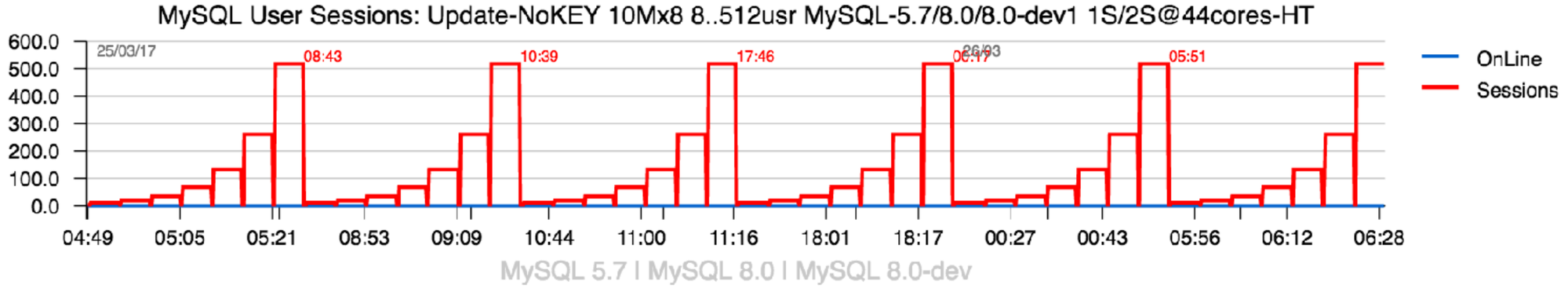
MySQL 8.0 Writes Scalability

- 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 ;-))

MySQL-dev preview : Sysbench Update-NoKEY 10Mx8-tables

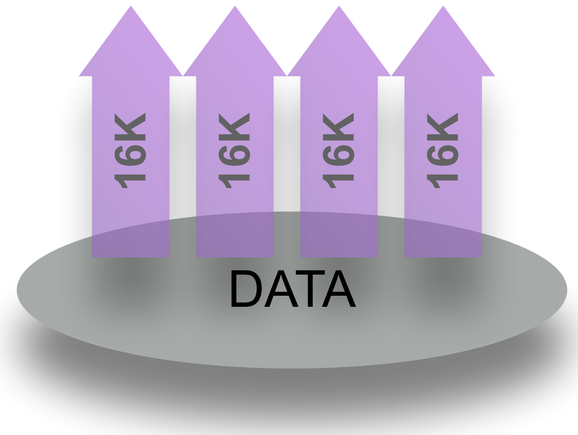
- Observations :
  - MySQL-dev : **x2** times better on 1 CPU socket, **x3** times on 2 CPU !!!
  - NOTE : the gain becomes visible already since **4usr** load level !!!





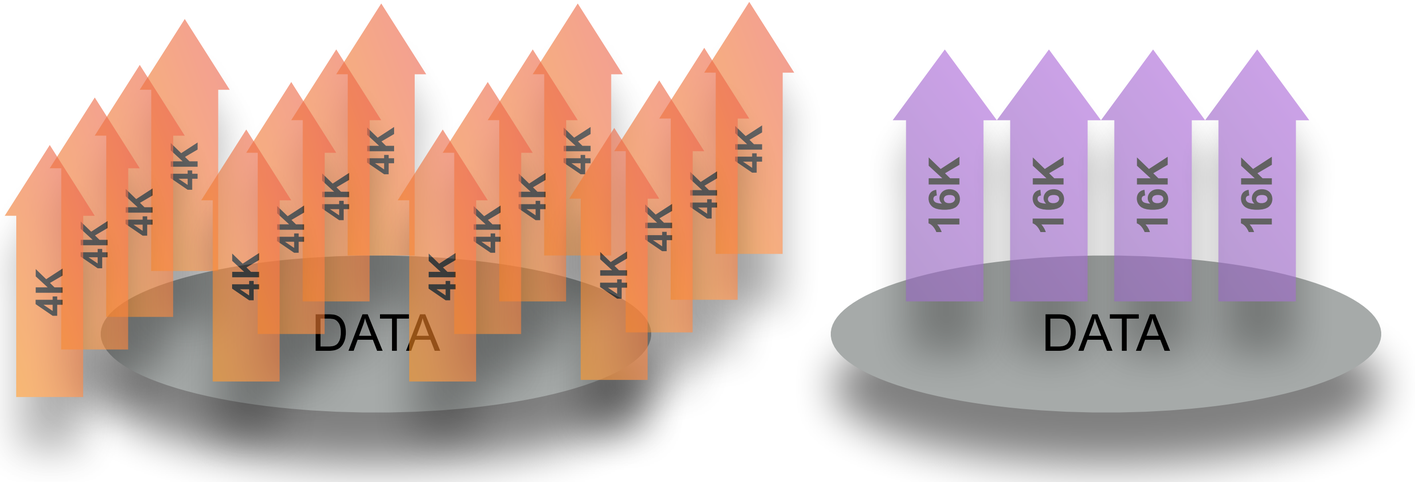
IO-bound Workloads : The Game Changer..

- 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** …



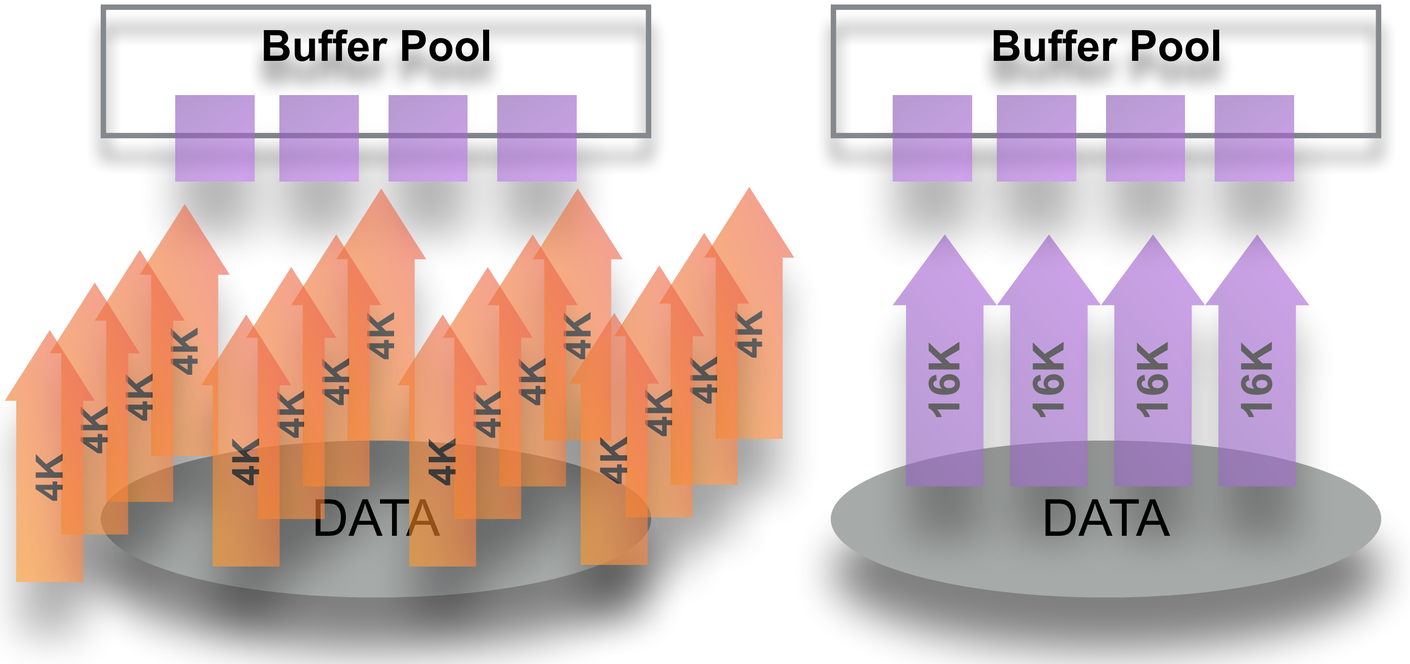
IO-bound Workloads : more in depth..

- 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 !!!



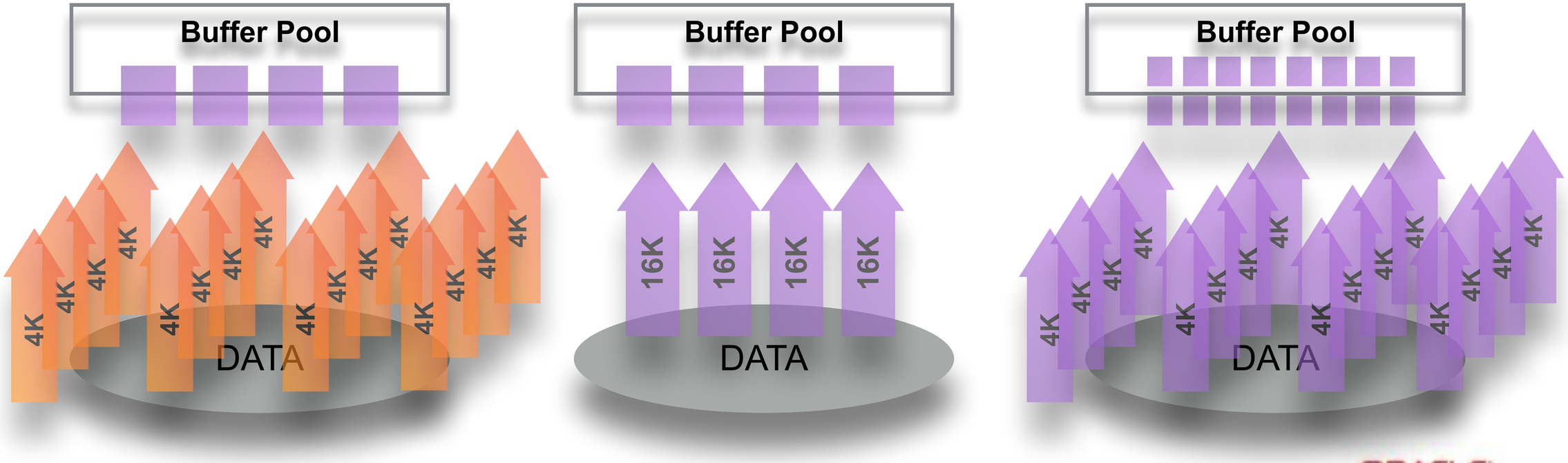
IO-bound Workloads : more in depth..

- 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** ?..



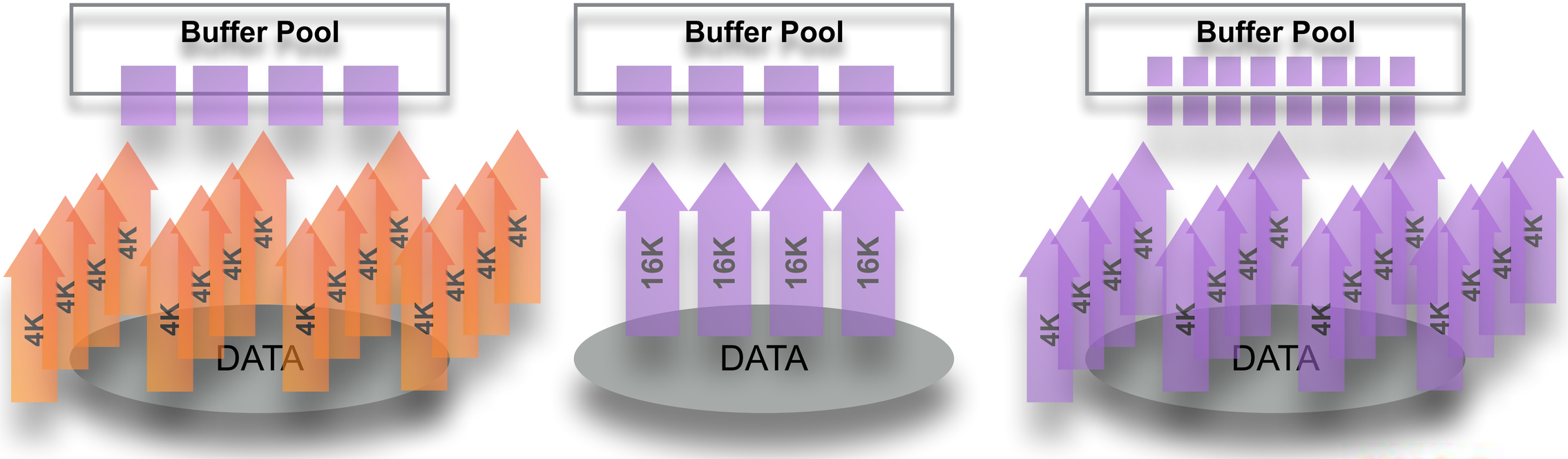
IO-bound Workloads : more in depth..

- 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 ?**



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

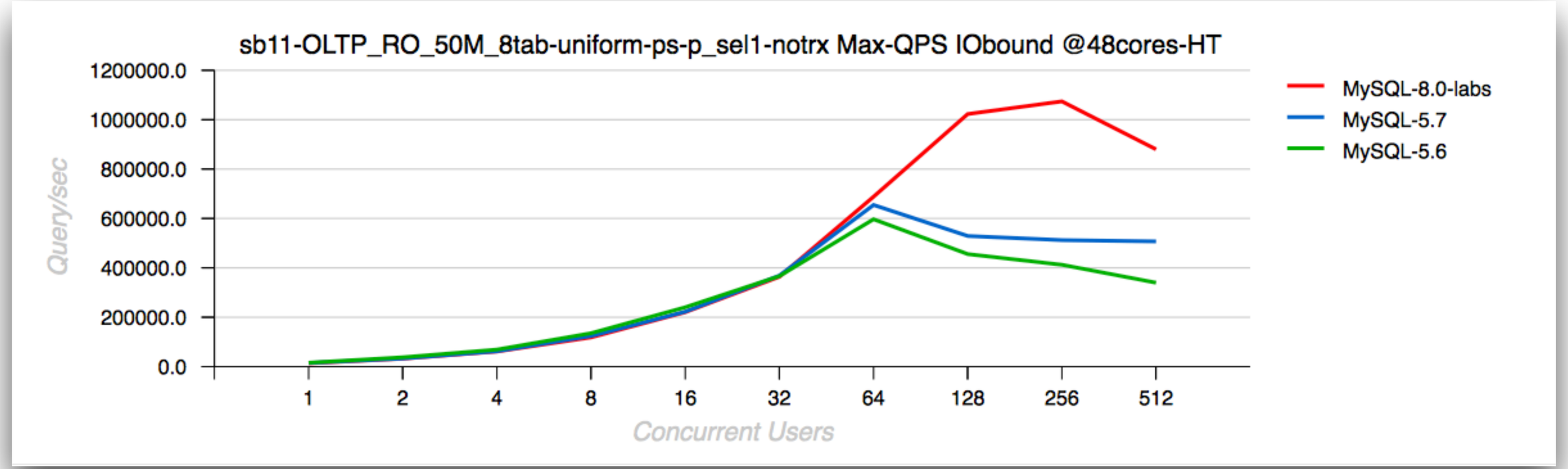
- Intel Optane drive :
  - IO read latency : 0,01ms (!!!)
  - 1 single process doing 16KB IO reads : ~65K reads/sec, 1000 MB/sec
  - however, the max throughput : 2000 MB/sec only (fix in progress by Intel)

- 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 ?..

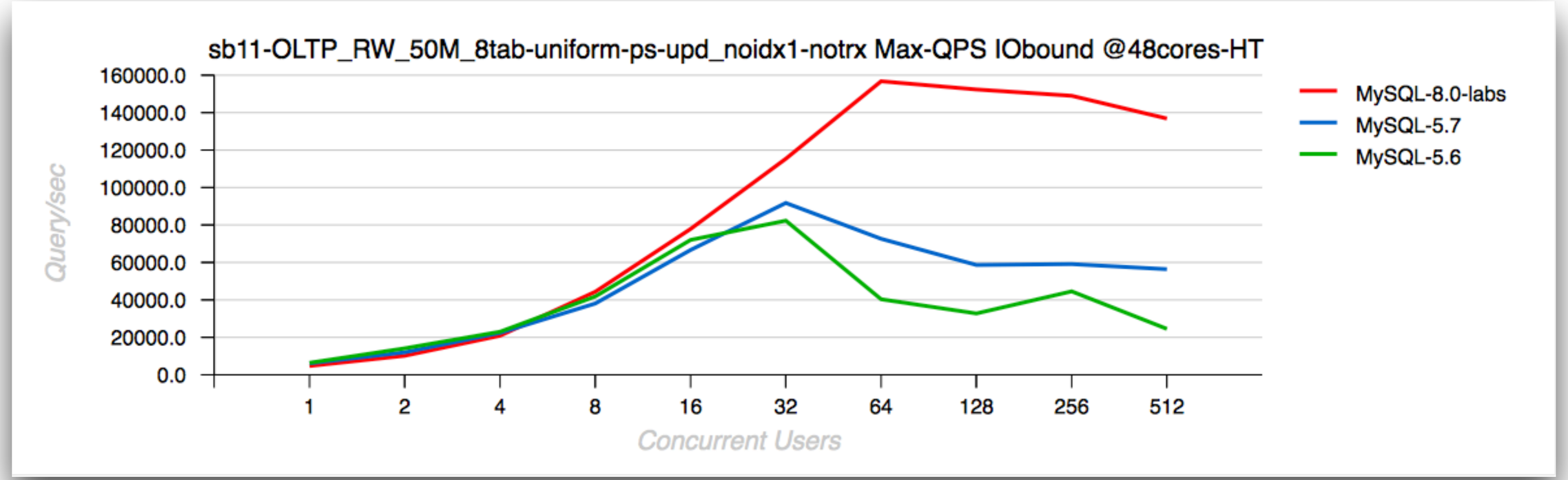
MySQL 8.0-labs Performance

- 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 !!!



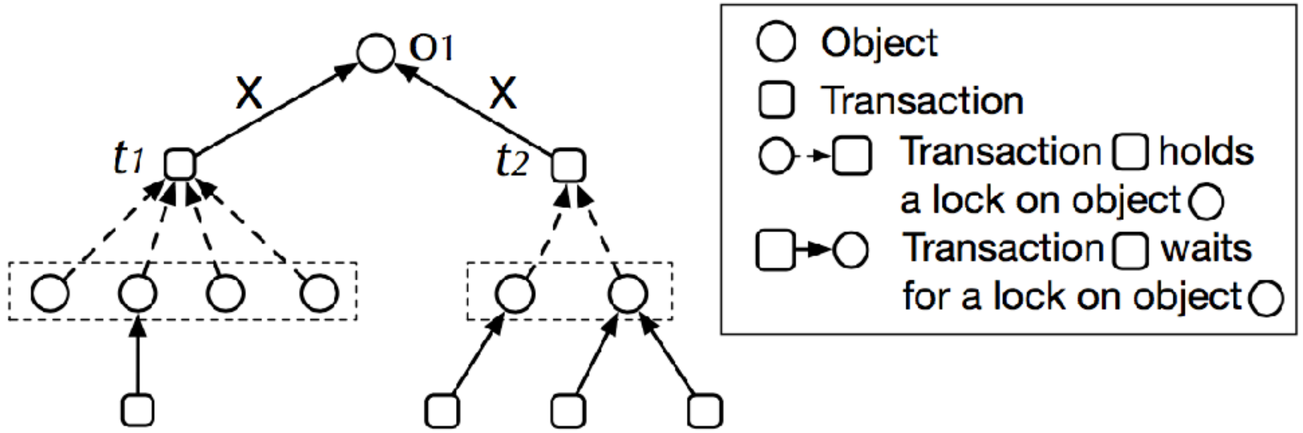
MySQL 8.0-labs Performance

- 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 !!!



MySQL 8.0 : InnoDB CATS (VATS)

- CATS : Contention-Aware Transactions Scheduling
  - invention : University of Michigan
  - adopted and integrated by InnoDB Team, available since MySQL 8.0.3
- Idea :
  - not all transactions are equal
  - FIFO could be not optimal..
  - unblock the most blocking transaction first



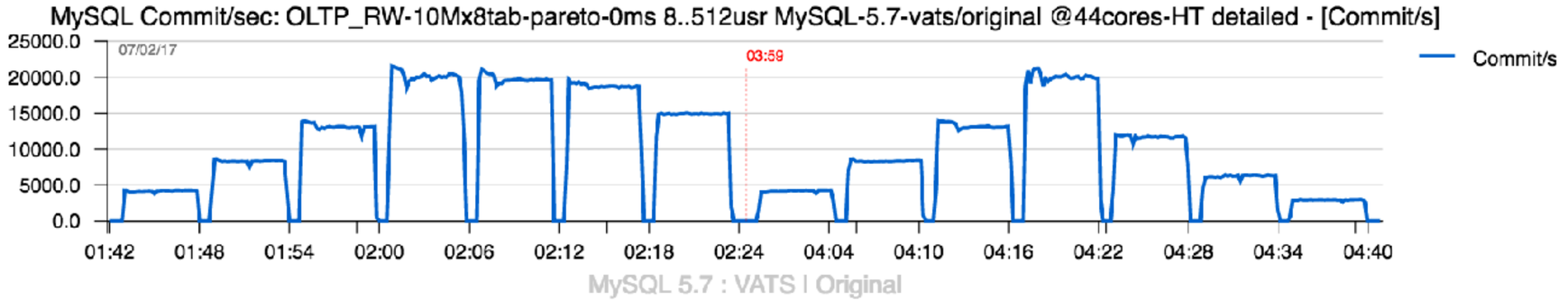
see :
- https://mysqlserverteam.com/contention-aware-transaction-scheduling-arriving-in-innodb-to-boost-performance/
- http://www.vldb.org/pvldb/vol11/p648-tian.pdf

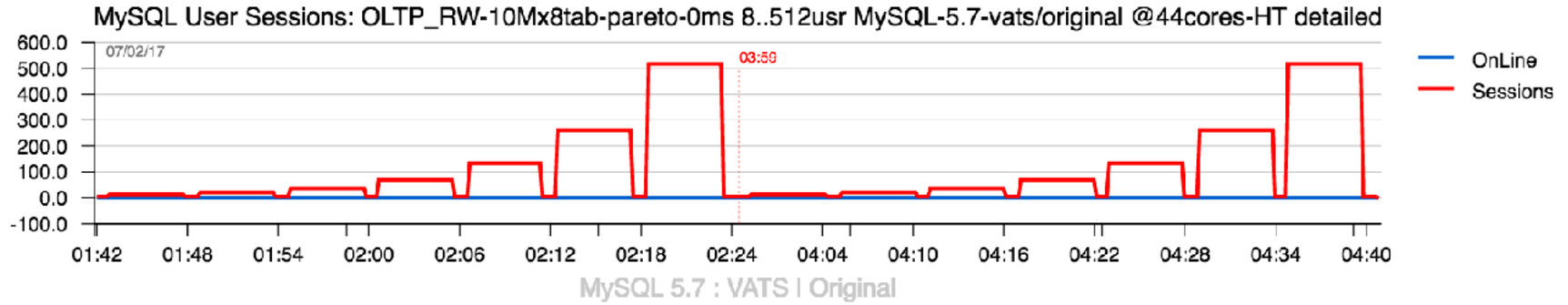
MySQL 8.0 : InnoDB CATS (VATS)

- CATS : Contention-Aware Transactions Scheduling
  - invention : University of Michigan
  - adopted and integrated by InnoDB Team, available since MySQL 8.0.3
- Kind of a detective story ;-))
  - claim : huge performance improvement
  - initial probe tests of patched code on all test workloads we have around : **zero** gain..
  - long investigation and deep discussions with authors to understand what kind of problems they're expecting to solve.. (they are not kidding, right ? ;-))
  - finally able to build a test scenario showing a visible gain ! - **Yes !** ;-))
  - Sunny analyzing the patch => several serious bugs..
  - loop : bug fix => remastering => retesting => goto begin..
  - finally stable ! => but brings regression on "normal" workloads..
  - solution ? => auto-tuned detection on switching to FIFO or CATS

MySQL 8.0 : InnoDB CATS (VATS)

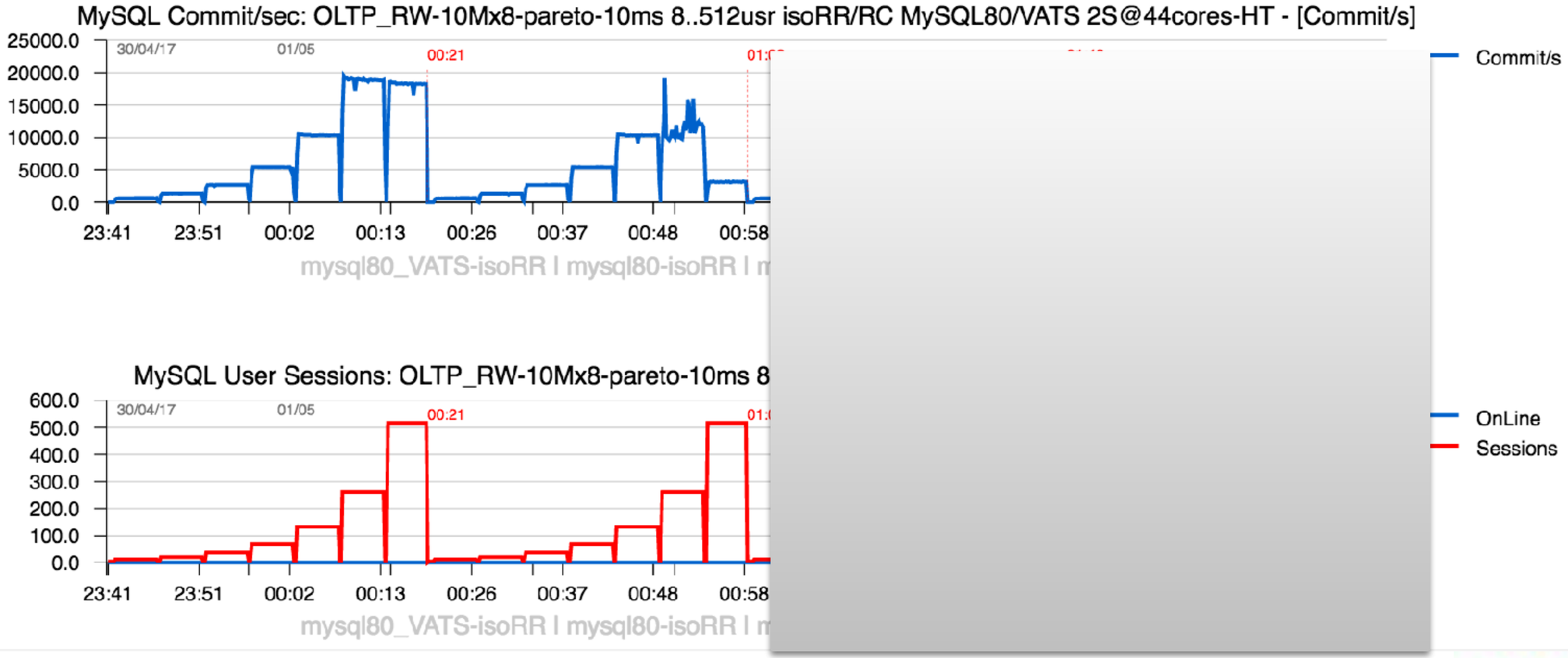
- CATS : Contention-Aware Transactions Scheduling
  - where it helps ? — workloads hitting row lock contentions
  - how to recognize ? — monitor your “show engine innodb mutex” !!





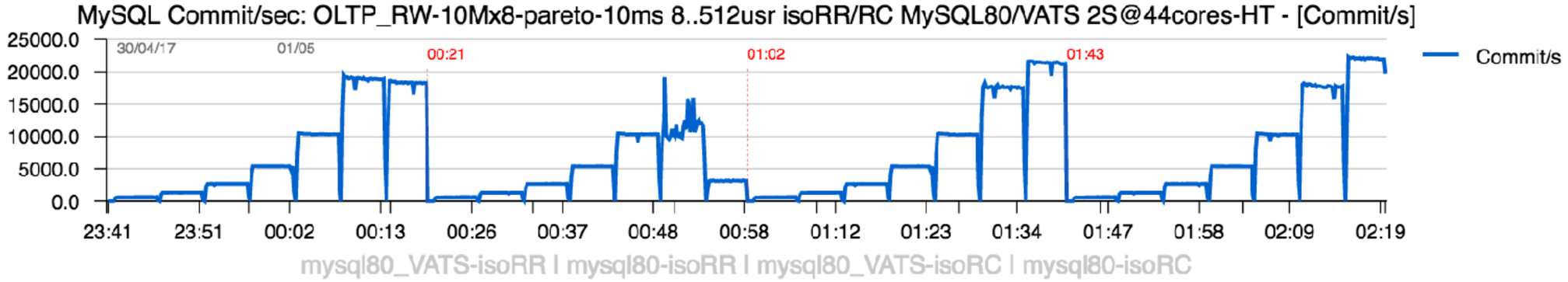
MySQL 8.0 : InnoDB CATS (VATS)

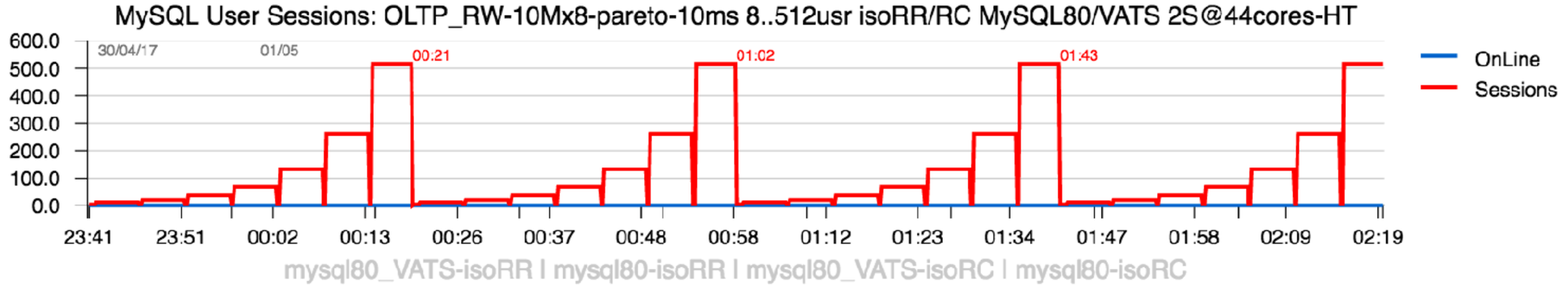
- CATS : Contention-Aware Transactions Scheduling
  - so, look in depth, understand your workload..
  - ex: RR -vs- RC transaction isolation on the same workload :



MySQL 8.0 : InnoDB CATS (VATS)

- CATS : Contention-Aware Transactions Scheduling
  - so, look in depth, understand your workload..
  - ex: RR -vs- RC transaction isolation on the same workload :





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

- Call To Action :
  - 2) download 8.0-labs / 8.0-rc
  - 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.