MySQL

ORACLE

MySQL Performance: Benchmarks & 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..
- HOWTO
- Q & A

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

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

USE YOUR BRAIN !!! ;-)

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

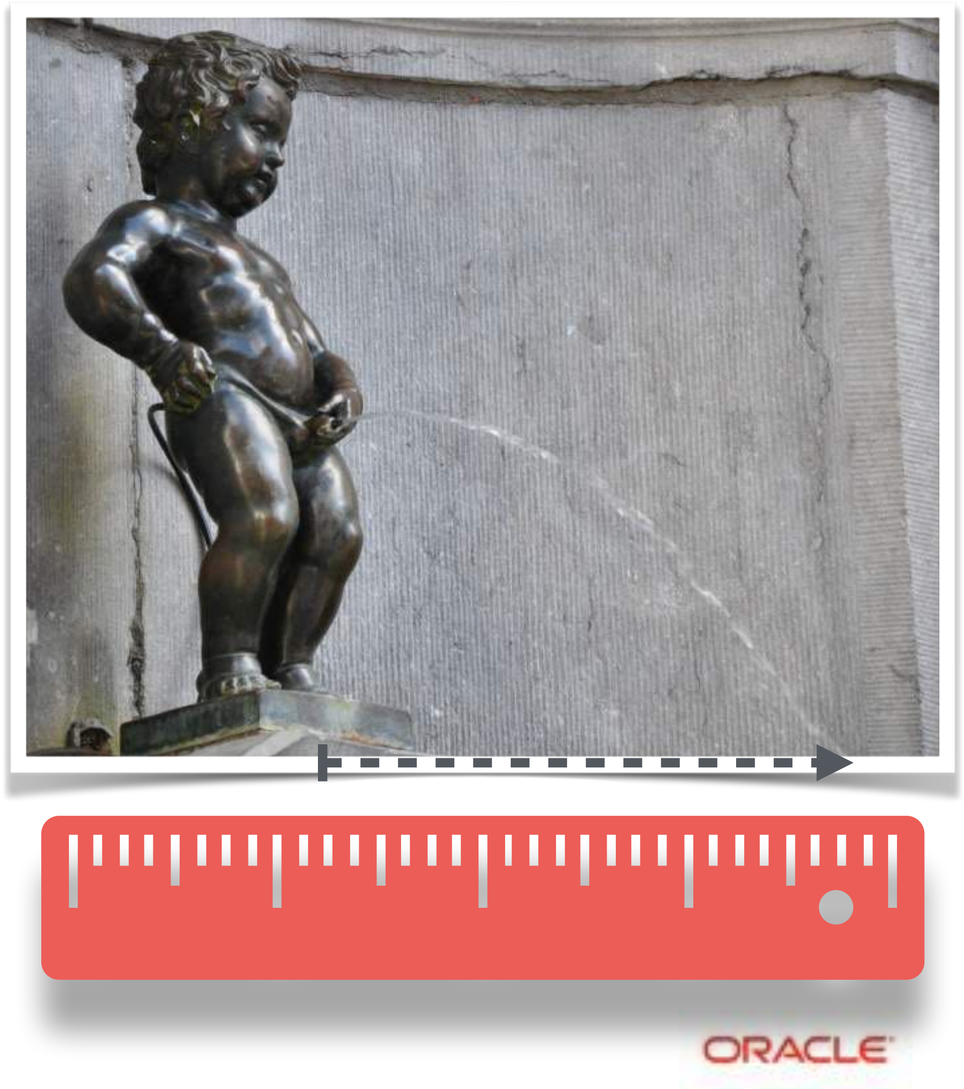
USE YOUR BRAIN !!! ;-)

#2 - Monitoring is THE MUST !

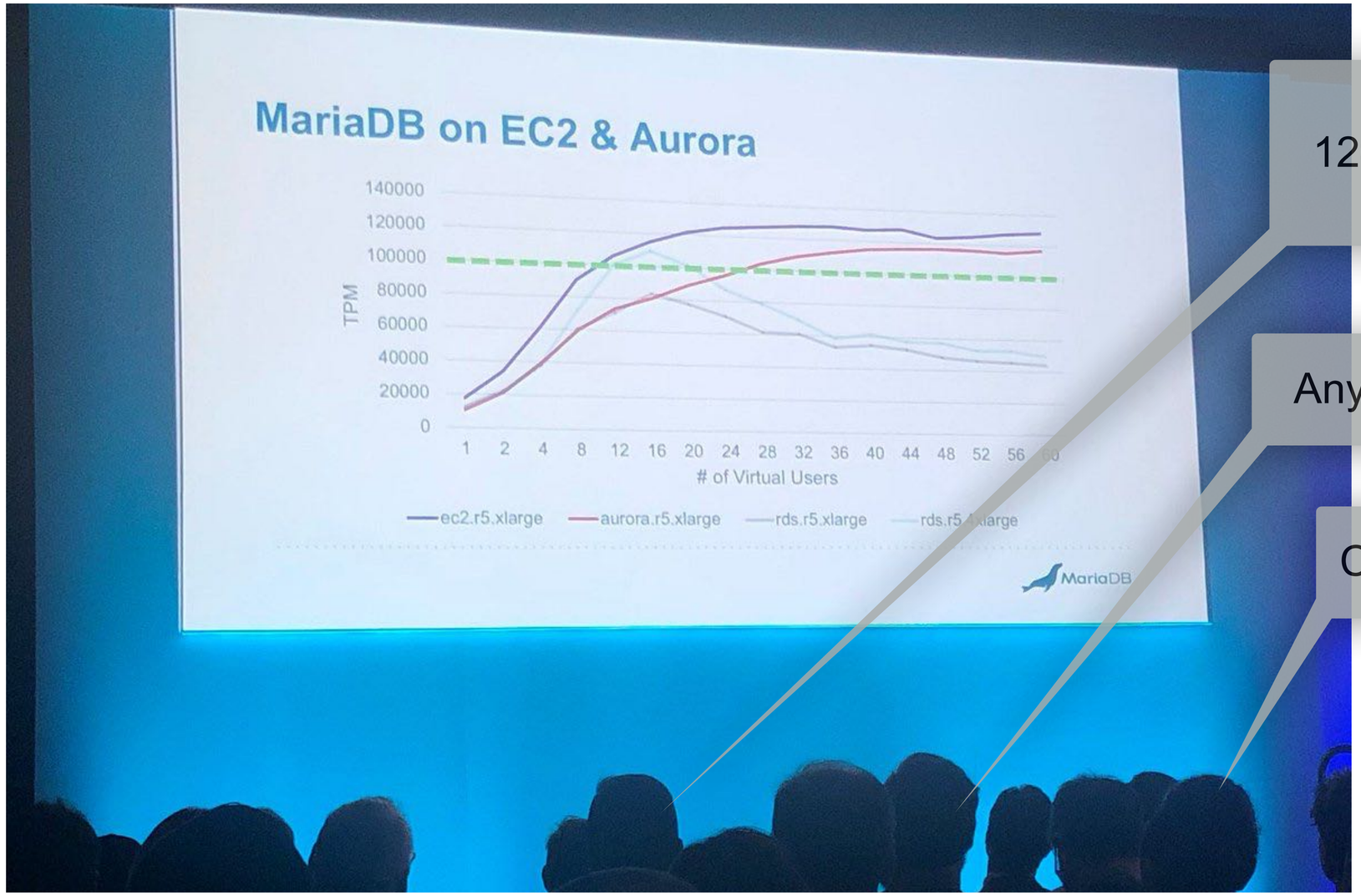
even **don’t** start to **touch** anything
without monitoring.. ;-)

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 ;-))



Indeed, There Are Some Bad Examples..



Hmm..
120K TPM = only **2K TPS**
why so slow ???

Any more **details** ???

Can I **reproduce** ???

- image credits : https://twitter.com/bytebot/status/1100410544480124928

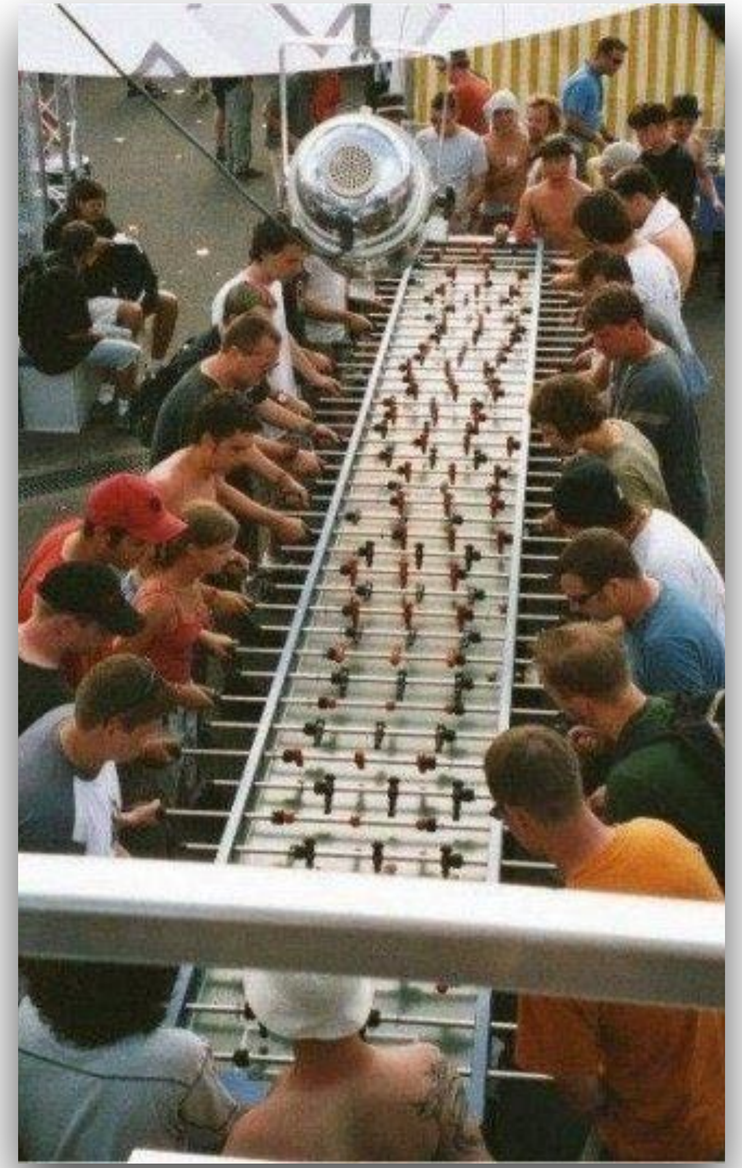
Benchmarks & MySQL

- 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..

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!
- 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..

Understand what you're testing !!!

- Recently many vendors started to propose their Test Workloads
- Example 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 !!!

- Recently many vendors started to propose their Test Workloads
- Example 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 represent your Production Workload ???
  - (e.g. mind "Best practice #1")

Popular “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 :
  - DBT-2 / TPCC-like / HammerDB / Sysbench-TPCC
- TPC-H / DBT-3
  - 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
  - FB workload

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 With 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
```

Using Sysbench ? => Mind The Details !!!

- Random (rand-type=…) Access :
  - special / uniform / pareto / gaussian / zipfian
  - for more details see: https://github.com/akopytov/sysbench/issues/329
  - how impacts on MySQL workloads ? => see in next slides..
- Note : by default “historically” => special (!!!)
  - **special** = mostly accessing only 1% of your data !!
  - e.g. 2GB or 2TB of data = nearly the same result with 32GB Buffer Pool

Using Sysbench ? => Mind The Details !!!

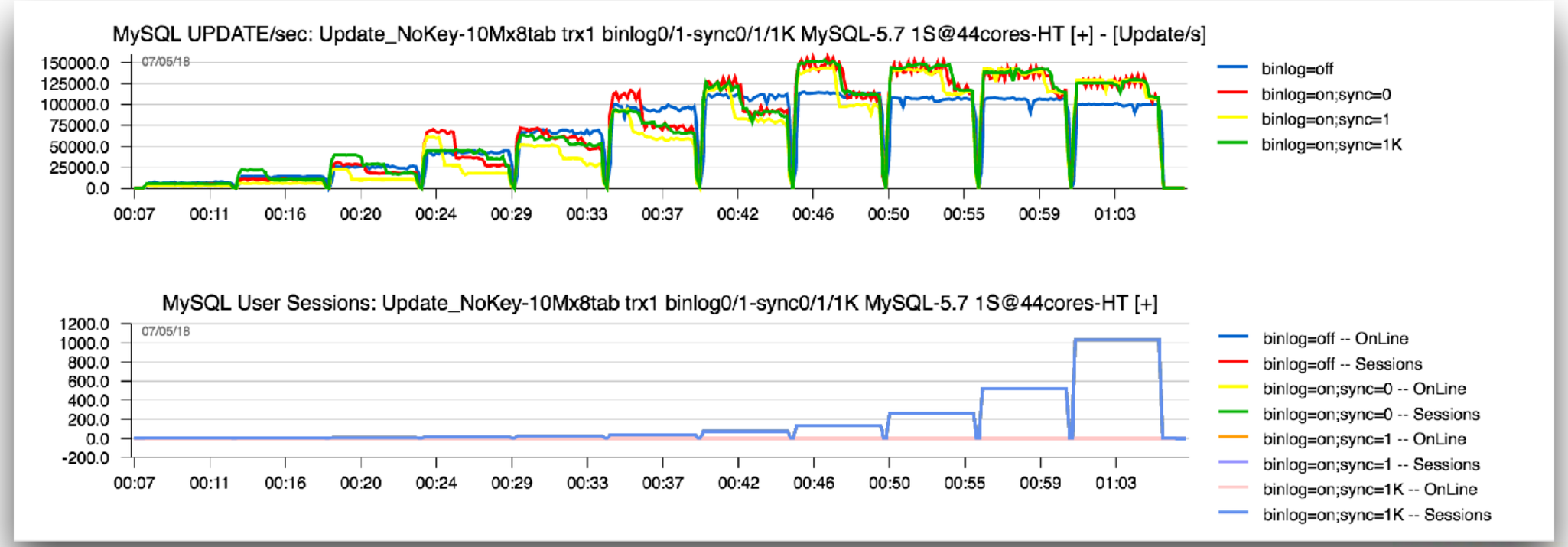
- Random (rand-type=…) Access :
  - special / uniform / pareto / gaussian / zipfian
  - for more details see: https://github.com/akopytov/sysbench/issues/329
  - how impacts on MySQL workloads ? => see in next slides..
- Note : by default “historically” => special (!!!)
  - **special** = mostly accessing only 1% of your data !!
  - e.g. 2GB or 2TB of data = nearly the same result with 32GB Buffer Pool
- Victims :
  - **PostgreSQL** : https://www.ongres.com/blog/benchmarking-do-it-with-transparency/
  - **ScaleGrid** : https://scalegrid.io/blog/how-to-improve-mysql-aws-performance-2x-over-amazon-rds-at-the-same-cost/

Using Sysbench ? => Mind The Details !!!

- Random (rand-type=…) Access :
  - special / uniform / pareto / gaussian / zipfian
  - for more details see: https://github.com/akopytov/sysbench/issues/329
  - how impacts on MySQL workloads ? => see in next slides..
- Note : by default “historically” => special (!!!)
  - **special** = mostly accessing only 1% of your data !!
  - e.g. 2GB or 2TB of data = nearly the same result with 32GB Buffer Pool
- Victims :
  - **PostgreSQL** : https://www.ongres.com/blog/benchmarking-do-it-with-transparency/
  - **ScaleGrid** : https://scalegrid.io/blog/how-to-improve-mysql-aws-performance-2x-over-amazon-rds-at-the-same-cost/
- But :
  - At least they **published** all the **details**, so the confusion in results could be explained, and error found..

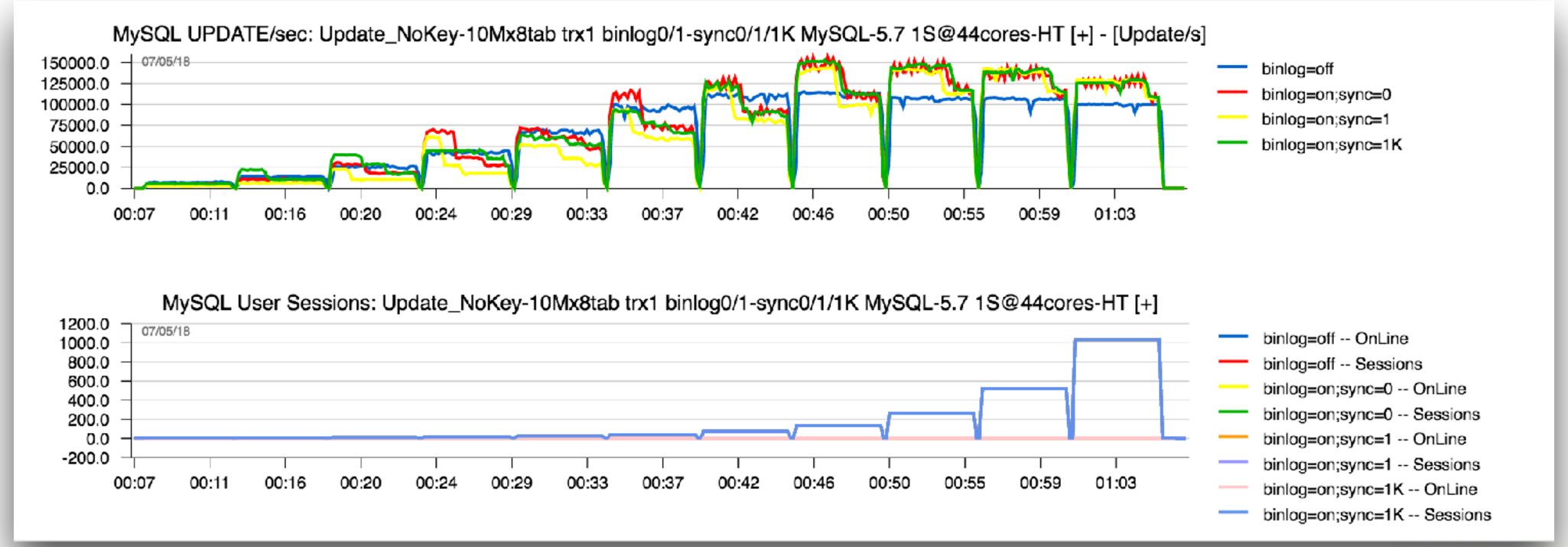
Using Sysbench ? => More About Random

- Observation :
  - Enabling Binlog improves performance ?? — Really ?? WTF ??
  - Or maybe there is a bug, and Binlog disabling REDO log ??..
  - Or maybe we hit some “short CPU boost” frequency feature ??
  - Or .. ??



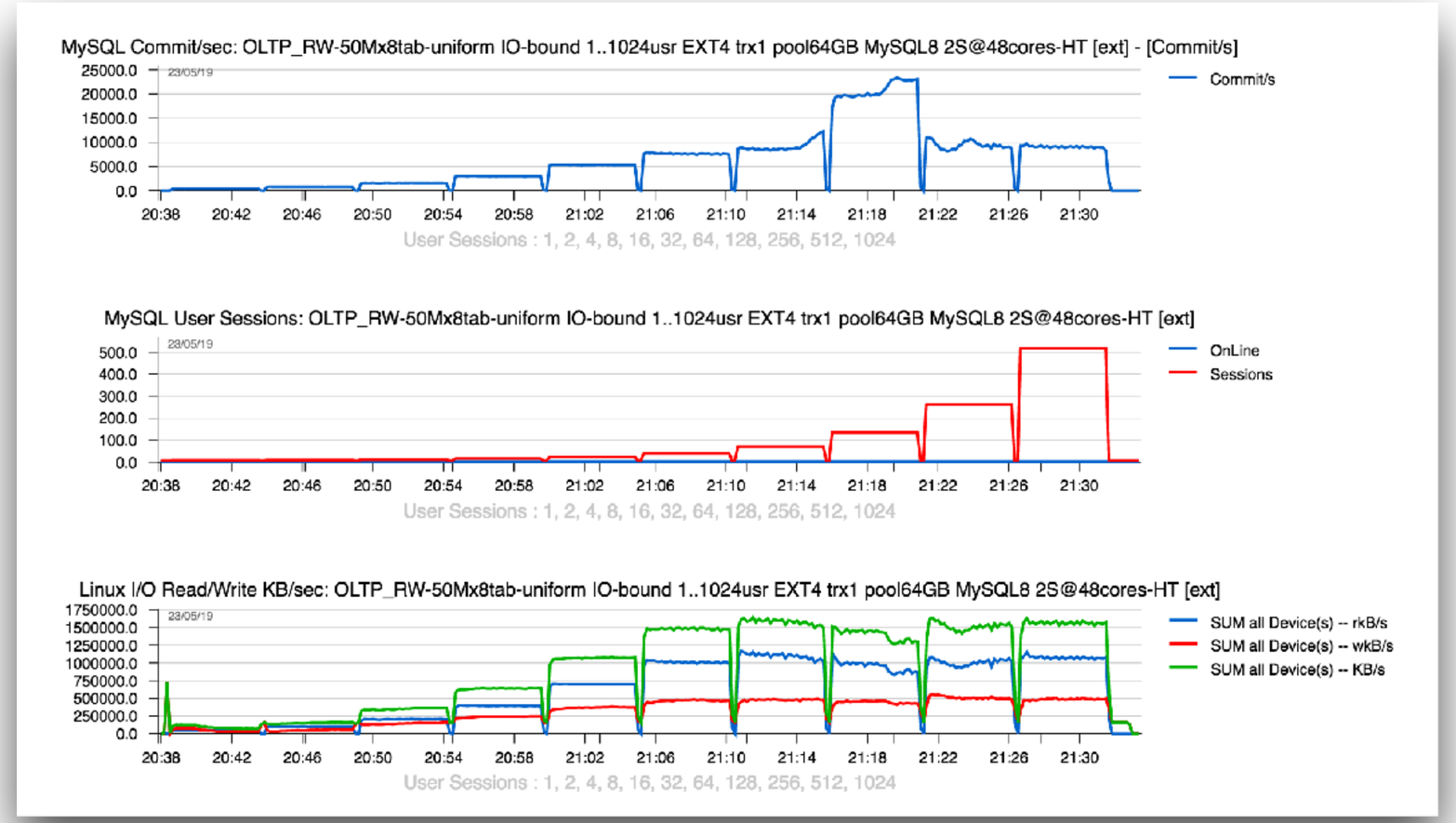
Using Sysbench ? => More About Random

- Explanation :
  - **rand-seed**=N : 0 = current time, not 0 = your choice (helps to debug)
  - here: rand-seed=**1** ; so on every new load level the same “random” values used
  - so, UPDATE with the same value = NO CHANGE ! => of course it goes faster ;-))
  - why only with Binlog ? => Good Question !!! ;-)) (but who cares?? — just use ZERO !!)



Don’t Look Only On Numbers !!!

- **IO-bound** Workload
  - but x2 times higher TPS on 128 users load level !! => “False Positive”
  - use Best Practice #2 (Monitoring) and then #1 ! ;-))



The 4 Main Sysbench “entry-ticket” Workloads

- OLTP_RO-point_select
  - the most aggressive, most simple (PK lookup), scaling well since MySQL 5.7
  - start with this to test your HW scalability, compare your results
  - meet any trouble ? => makes no sense to go any further.. (fix it first ;-))
- OLTP_RO (mixed RO)
  - more various RO operations (ranges, order by, distinct (group by), + 10 point-selects)
- OLTP_RW (mixed RO + writes)
  - mix of all, good enough “generic” OLTP
- OLTP_RW-update_no_index (aka Update_NoKEY)
  - bombarding with in-place UPDATE(s) => good for testing write bottlenecks
- Note:
  - our writes are NOT scaling (yet)

Minimal “Test Kit” Scenarios

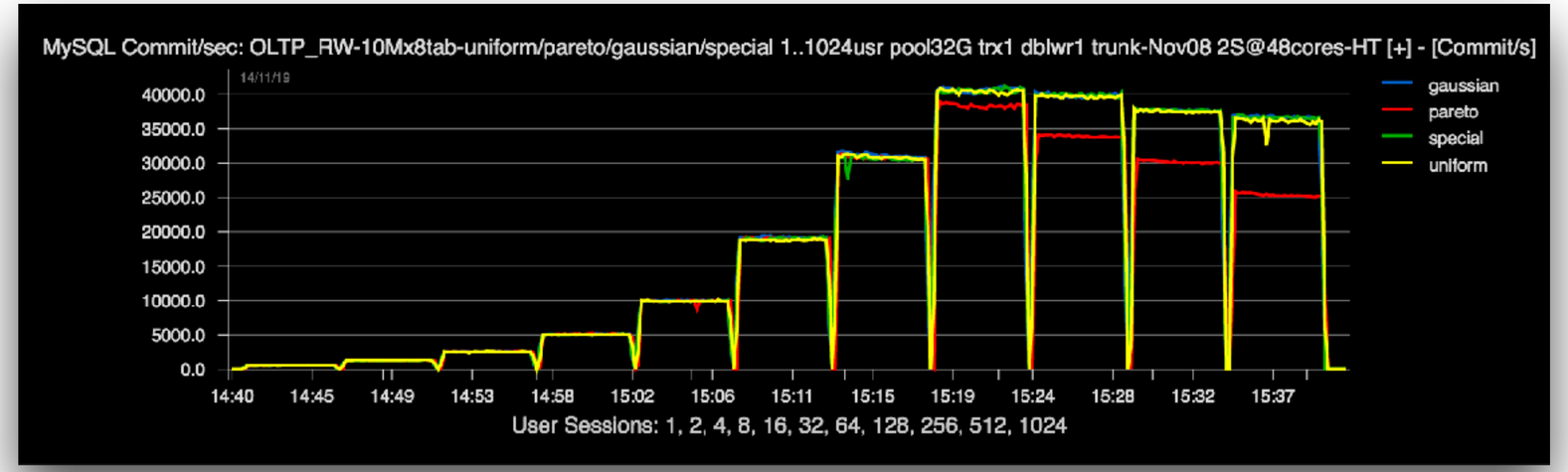
- Data Volume and Number of Tables :
  - 1 table : to simulate one “hot table” in your app/production
  - 8 tables : splits “hot table” issue(s) by 8 and let you see if it helps or you hit something else
- Data / Memory Ratio :
  - all in-memory
  - 1/2 of data can feet memory
  - 1/4 of data can feet memory
- Other :
  - random : uniform / pareto
  - prepared statements : ON / OFF

Minimal “Test Kit” Scenarios : Example

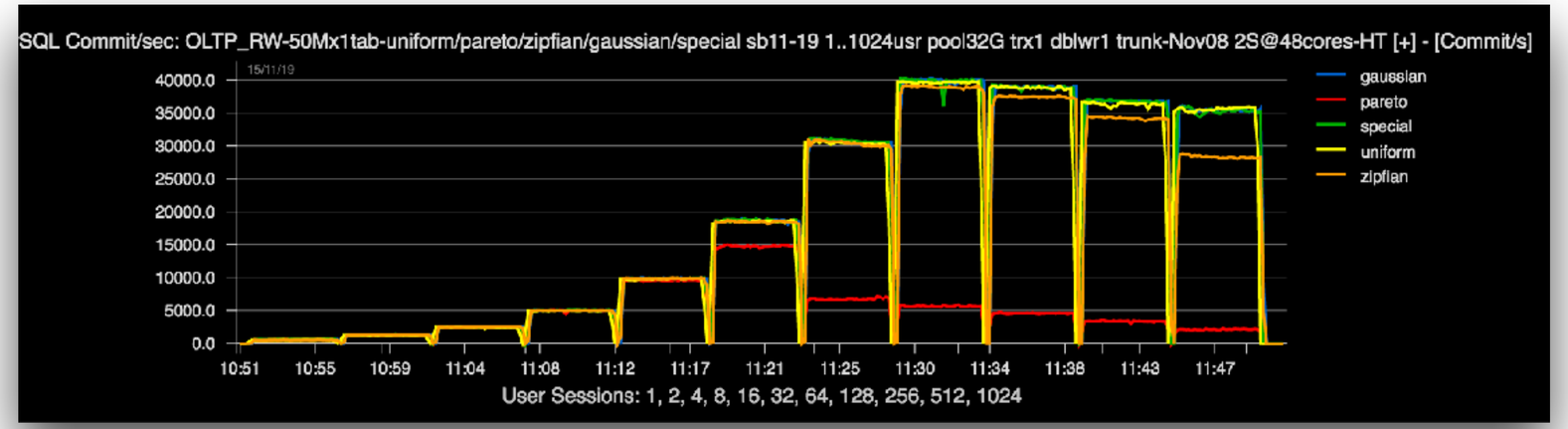
- Data Volume and Number of Tables :
  - 1 table : 400M rows (~120GB) - 400Mx1tab
  - 8 tables : 50M rows per table (same 400M in total) - 50Mx8tab
- Data / Memory Ratio :
  - all in-memory : BP = 128G
  - 1/2 of data can feet memory : BP = 64G
  - 1/4 of data can feet memory : BP = 32G
- Other :
  - random : uniform / pareto / special / gaussian / zipfian
  - prepared statements : ON
  - trx_commit=1, dblwr=1, checksum=1, PFS=on, UTF8, etc.

Sysbench OLTP_RW

- In-Memory, 8 tables :

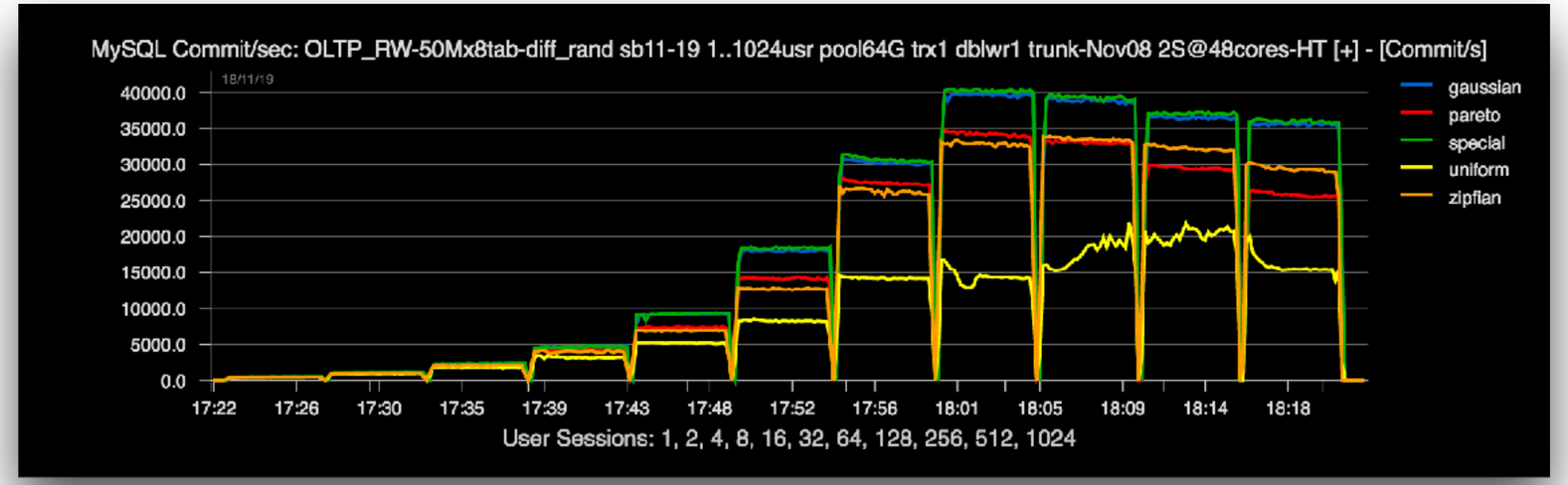


- In-Memory, 1 table :

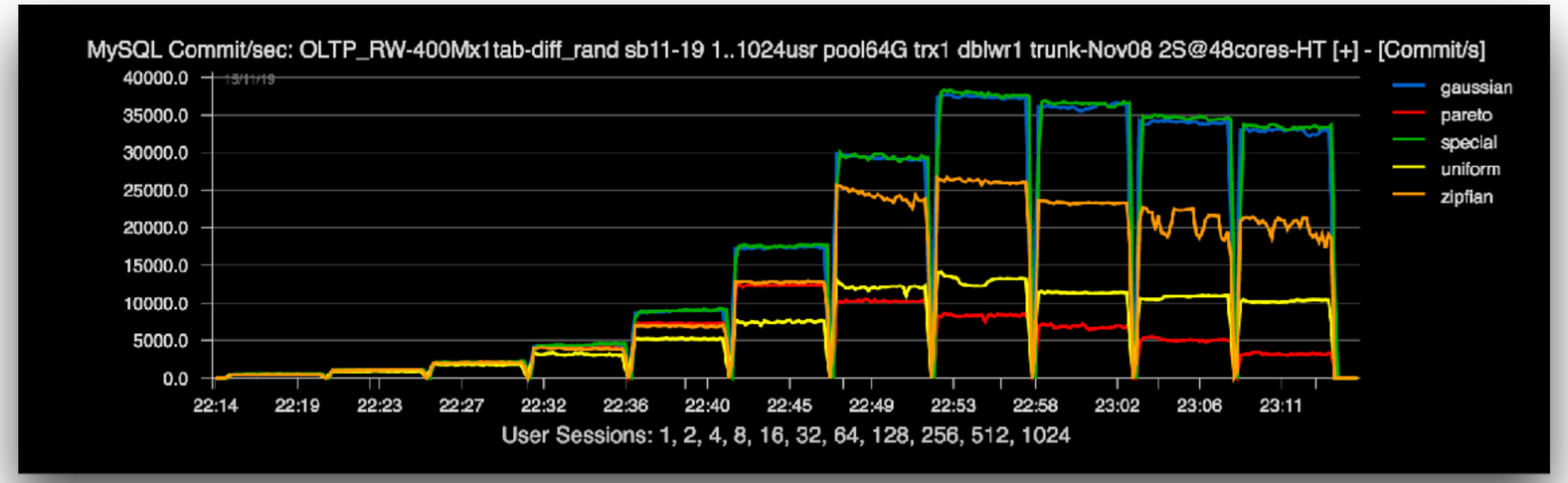


Sysbench OLTP_RW

- 1/2 Memory, 8 tables :

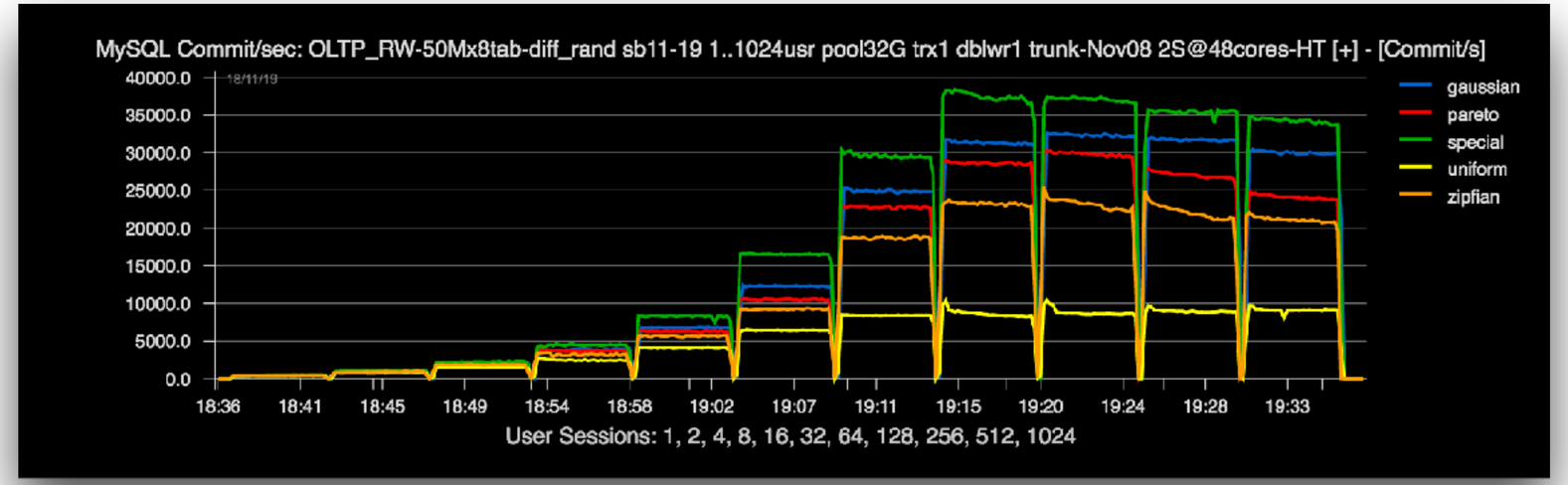


- 1/2 Memory, 1 table :

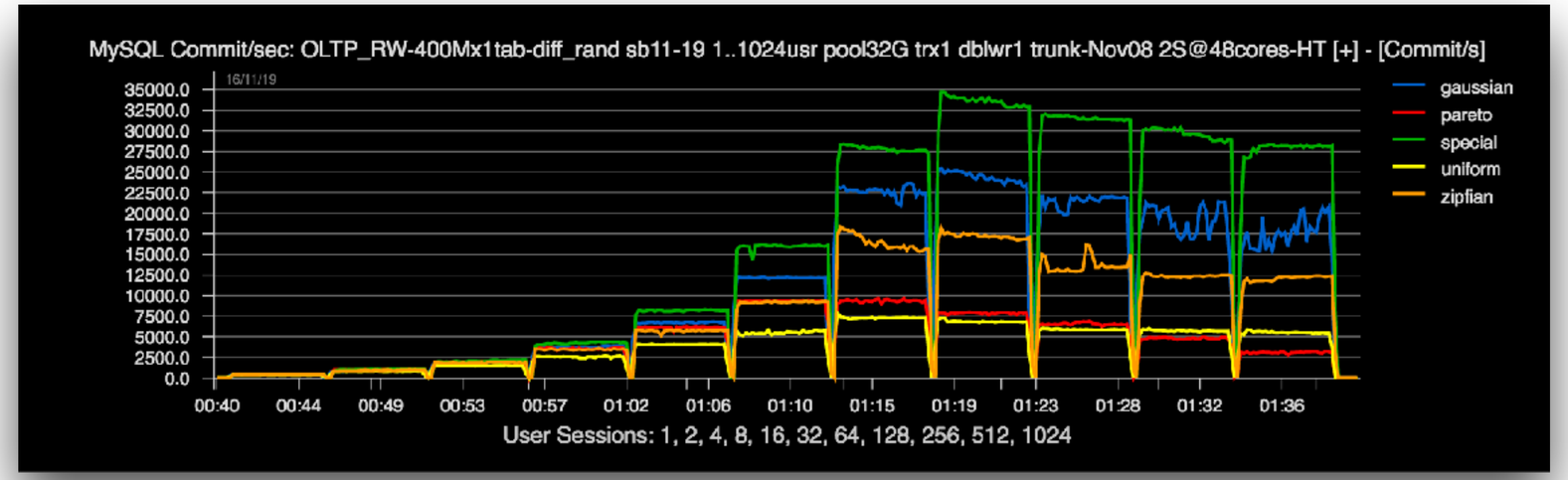


Sysbench OLTP_RW

- 1/4 Memory, 8 tables :



- 1/4 Memory, 1 table :



ORACLE

To Avoid Confusions : Benchmark Kit

- In The Box :
  - pre-generated scripts for all workloads
  - script name = test & options
  - all Sysbench tests
  - (but without “special” ;-))
  - includes TPCC
  - includes dbSTRESS
  - coming after FOSDEM ;-))

```
cd /BMK

# prepare data
bash ./sb_exec/sb11-Prepare_50M_8tab-InnoDB.sh 32

# run OLTP_RW for 5min each load level..
for rnd in uniform pareto
do
  for users in 1 2 4 8 16 32 64 128 256 512 1024
  do
    bash ./sb_exec/sb11-OLTP_RW_50M_8tab-$rnd-ps-trx.sh $users 300
    sleep 15
  done
  sleep 60
done
```

Yet More Things To Keep In Mind..

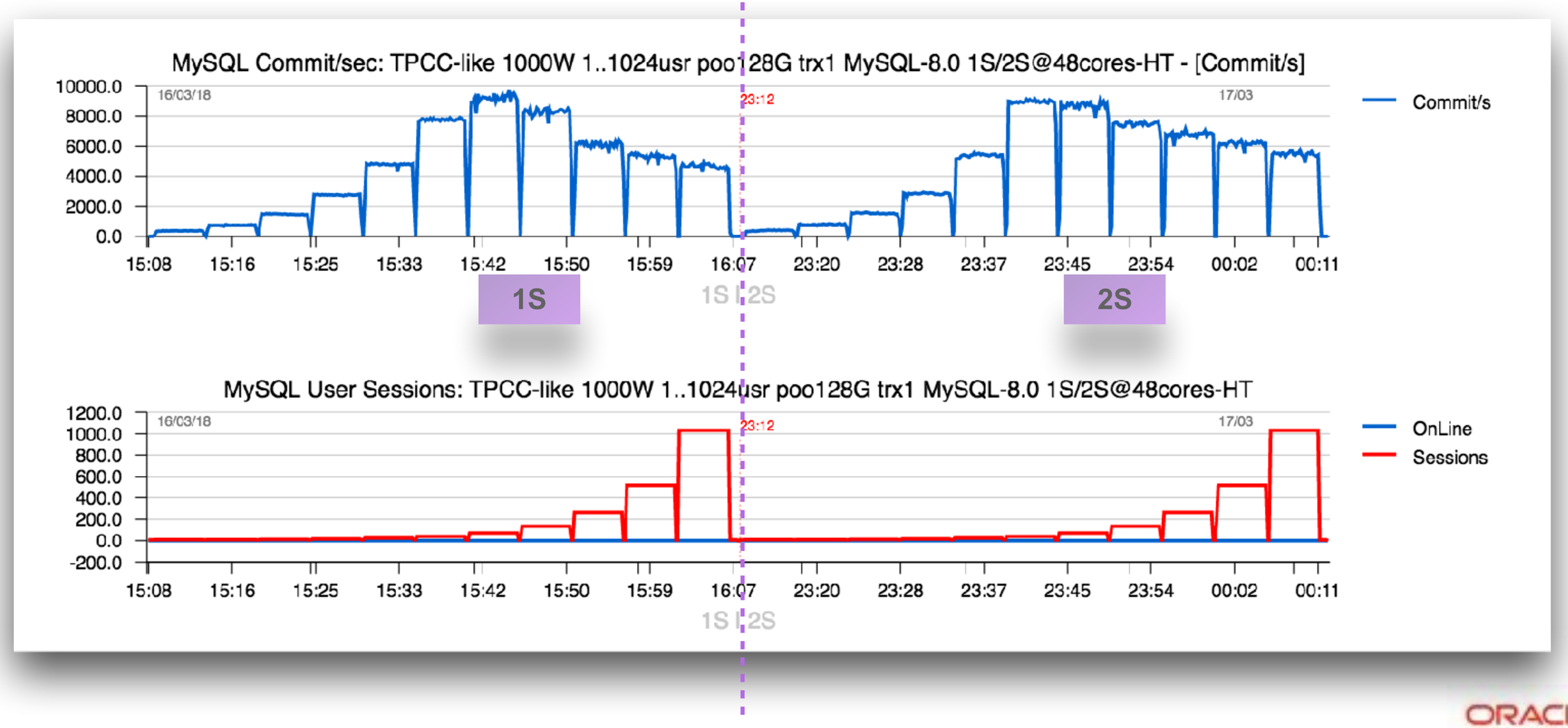
- Configurations :
  - jemalloc-5.2.1 => the MUST
  - same storage / diff drives
  - flash : (yes, please, LOVE your DATA !!) — *but mind latency driven, more drives != better perf*
  - EXT4 / XFS ?? — *massive EXT4 regression in the latest kernels..*
  - O_DIRECT !!!
  - dblwr = 1/0 (note: 1 by default)
  - Binlog = 1/0 (note: 1 by default)
  - trx_commit = N (note: 1 by default)
  - REDO size
  - IO capacity / capacity_max
  - flushing neighbors = OFF
  - UNIX socket / IP port (localhost) / xNET
  - ssl = 0/1 (note: 1 by default)
  - UTF8 -vs- latin1 (note: UTF8 by default)
  - UNDO auto-truncate (note: 1 by default)
  - MySQL 8.0 Resource Groups: you can split your activity..

TPCC Workloads

- DBT2
  - old stuff (but still loved by some ones (definitively not me ;-)))
- TPCC-like
  - more simple to use, but “hard-coded” ;-))
- HammerDB
  - fully based on Stored Procedures
- Sysbench TPCC
  - ported to Sysbench from TPCC-like by Percona
  - totally flexible (LUA scripts)
  - easily hackable
  - can use more than 1 TPCC “instance”

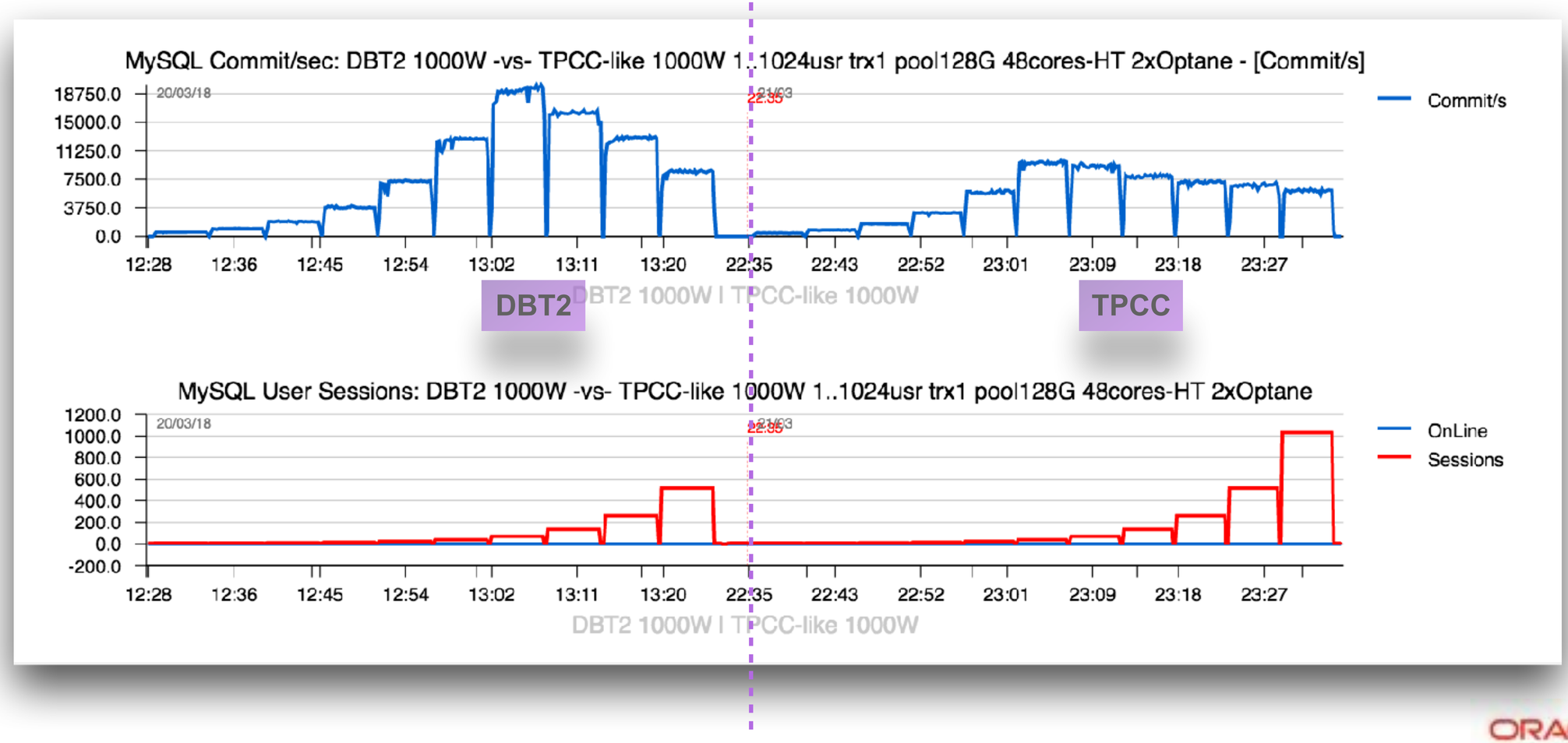
TPCC “mystery” : 1000W Workload @Sysbench-TPCC

- 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

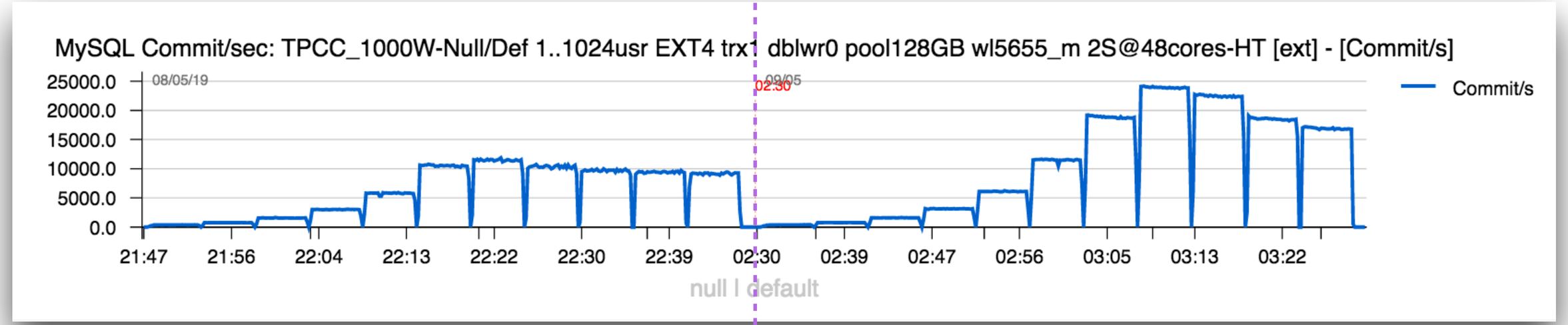
- Motivation
  - even if we could not “fix” index RW-lock contention in InnoDB on TPCC…
  - …but we can find a “workaround” on how to avoid it => **already a huge gain !!!**
- 1) Ranger (ex-MySQL, now Percona, DBT2 lover in the past ;-))
  - => who changed the DBT2 schema ???
  - => as the result : 1/2 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

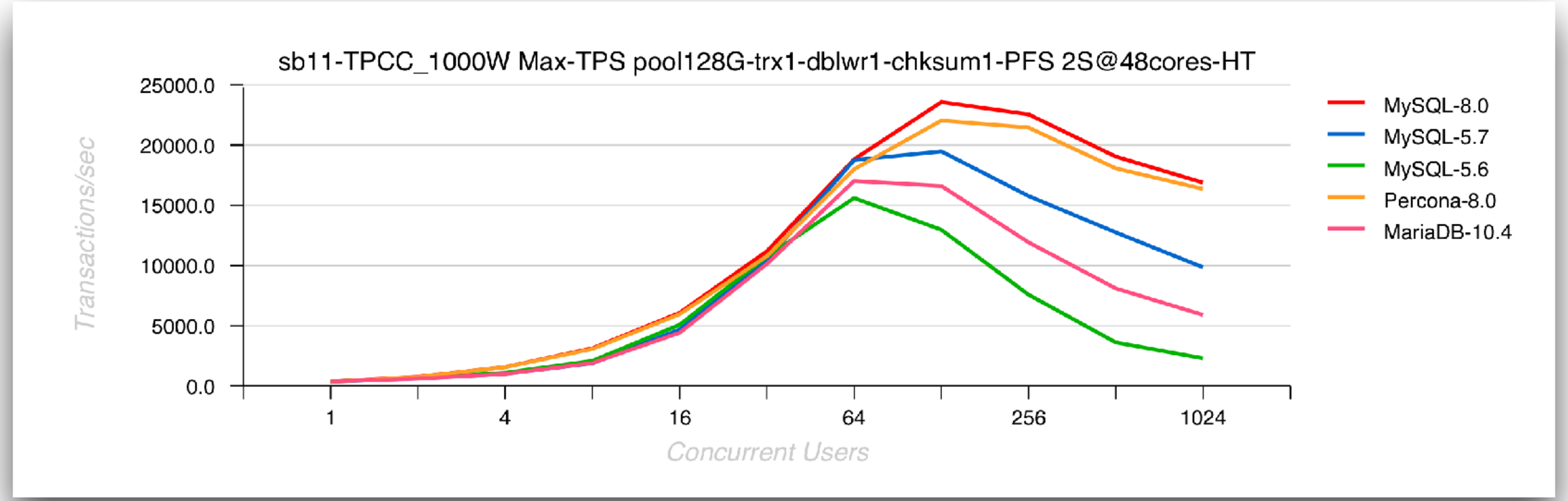


NULL

DEFAULT

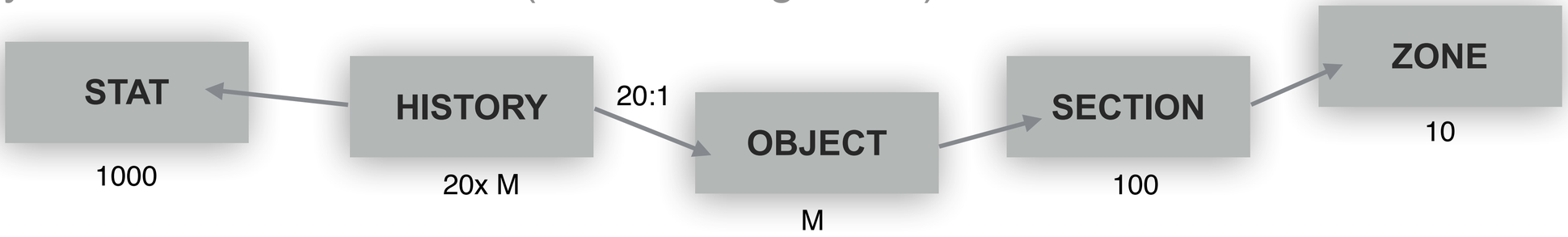
TPCC “mystery” : test results (Sep.2019)

- Sysbench-TPCC 1000W, using DEFAULT
  - **in-memory** (pool=128G)
  - 2S 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 :

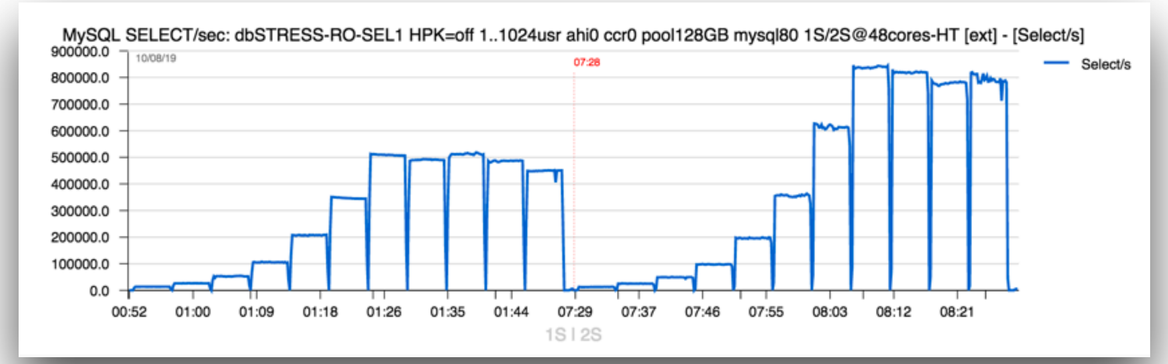


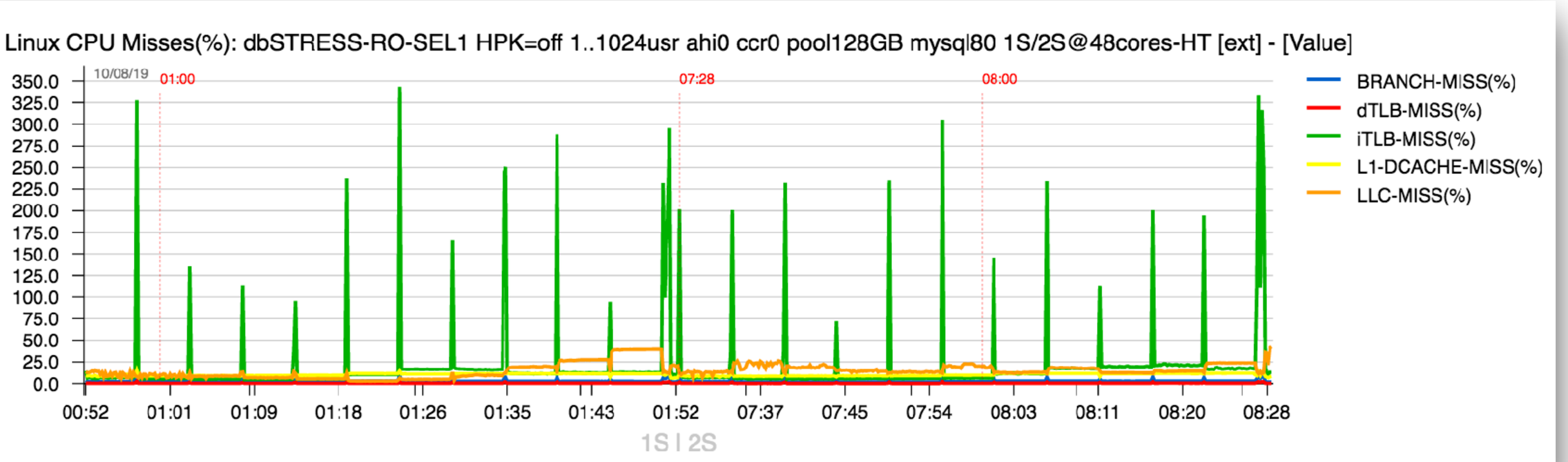
  - queries :
    - SEL1 : join of 3 tables by OBJECT REF (OBJECT => SECTION => ZONE)
    - SEL2 : join of 2 tables by OBJECT REF (OBJECT => HISTORY)
    - WRITE: DELETE/ INSERT/ UPDATE of single HISTORY record by OBJECT REF

- 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

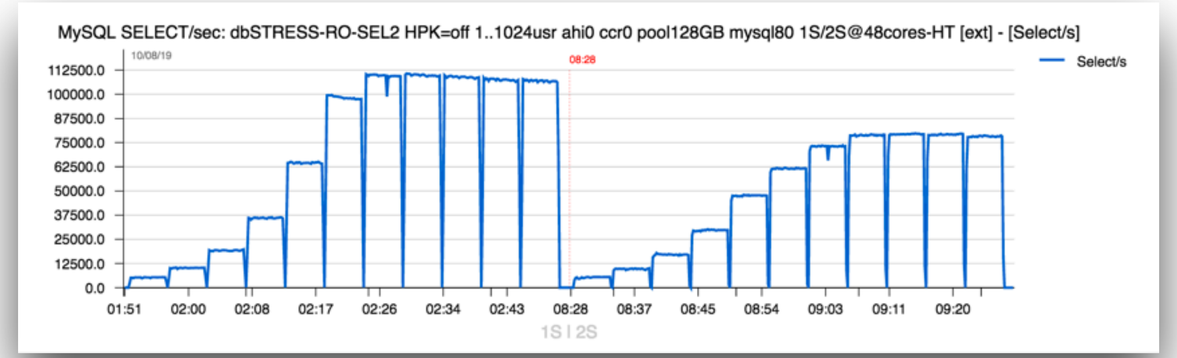
- SEL1 :
  - scaling from S1 to S2

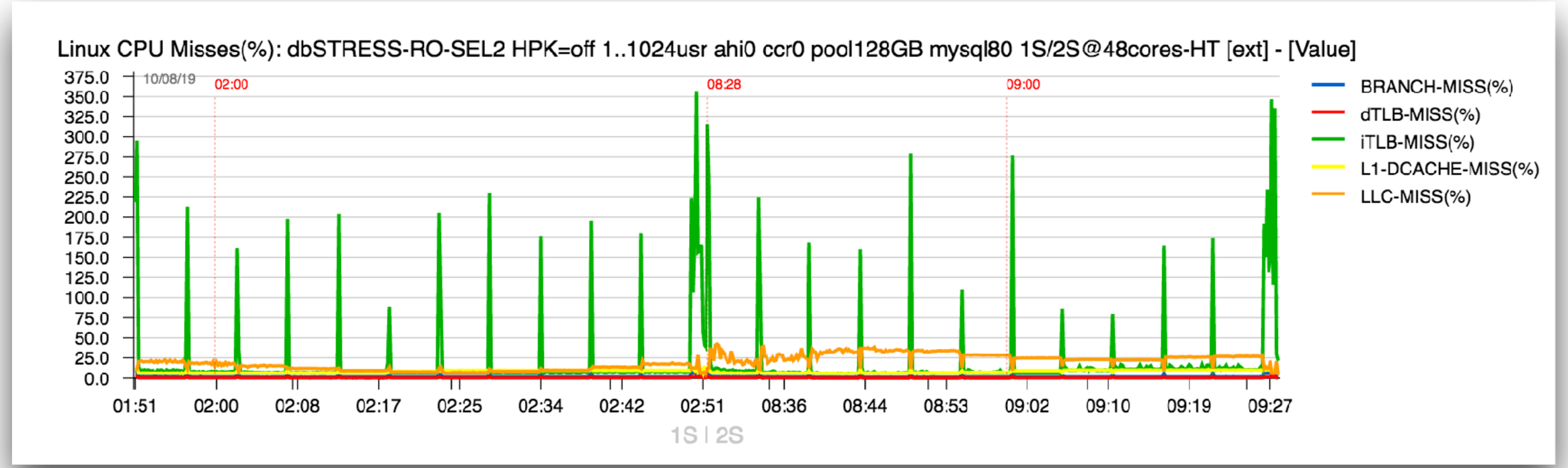




dbSTRESS-RO-SEL2

- SEL2 :
  - NOT scaling from S1 to S2

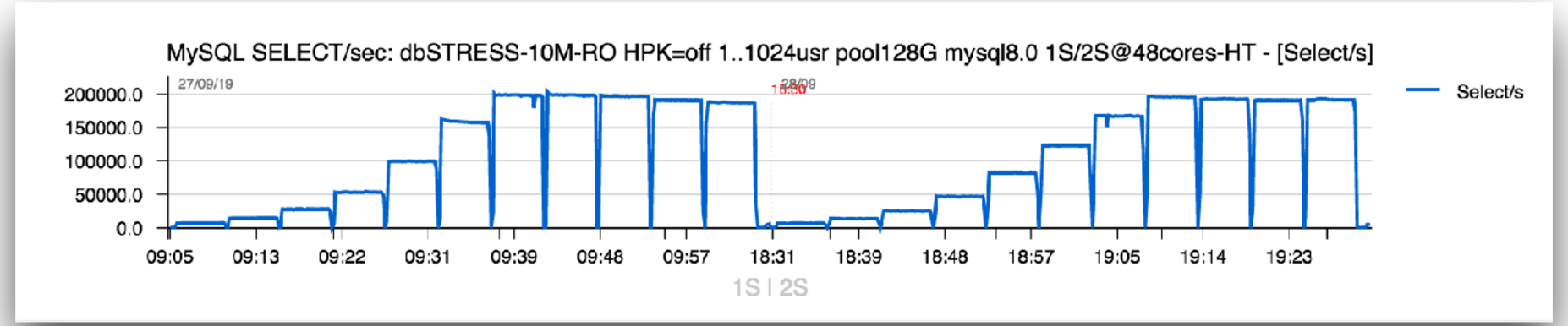




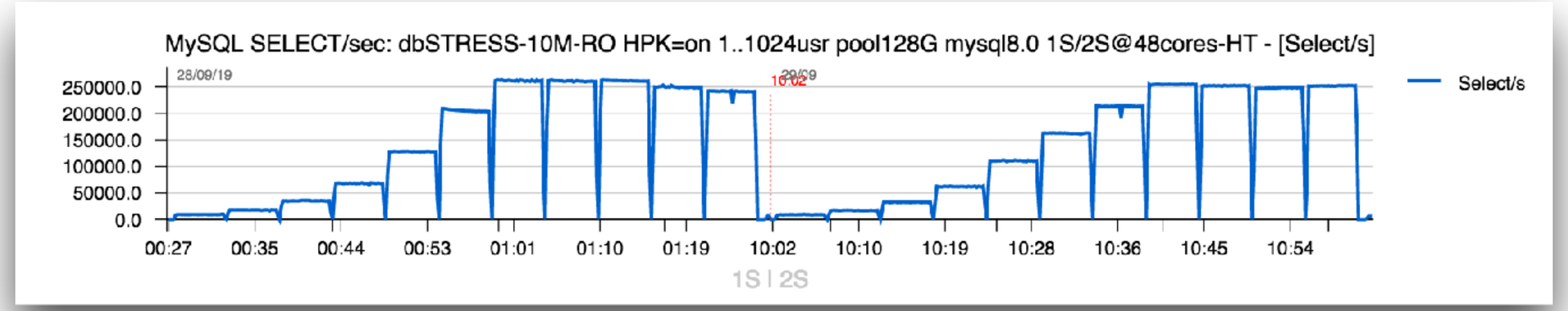
ORACLE

dbSTRESS-RO (SEL1 + SEL2)

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



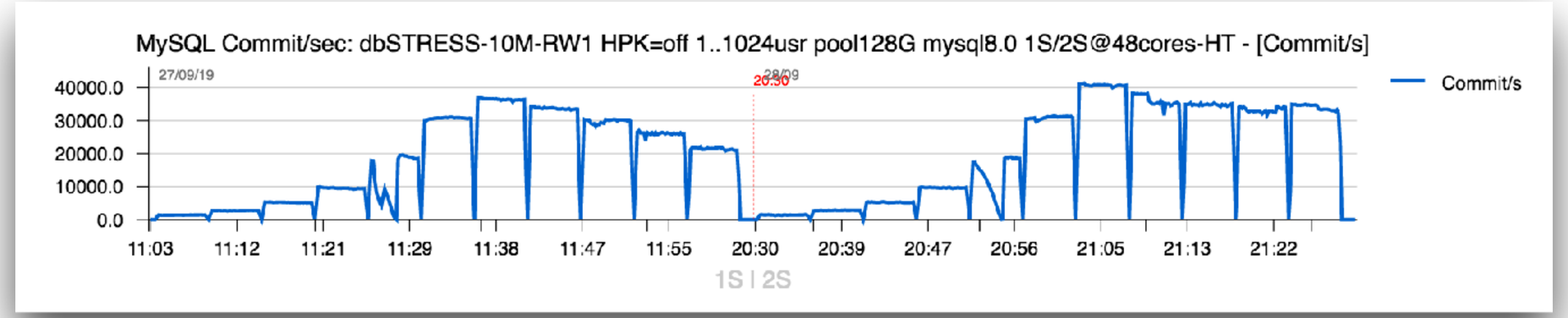
- HISTORY table using PK (HPK=on) :



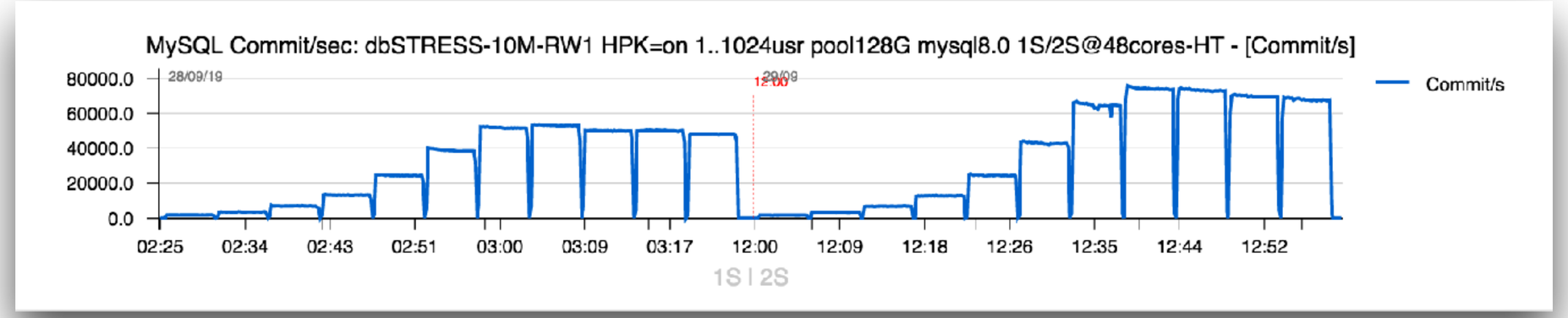
ORACLE

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

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



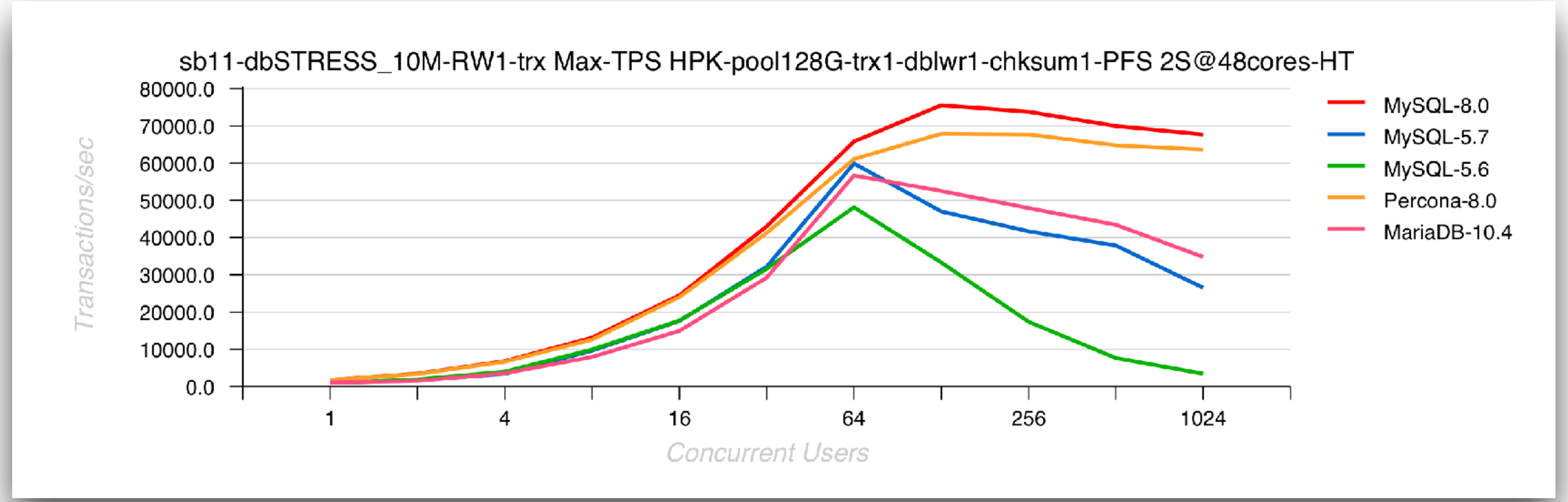
- HISTORY table using PK (HPK=on) :



ORACLE

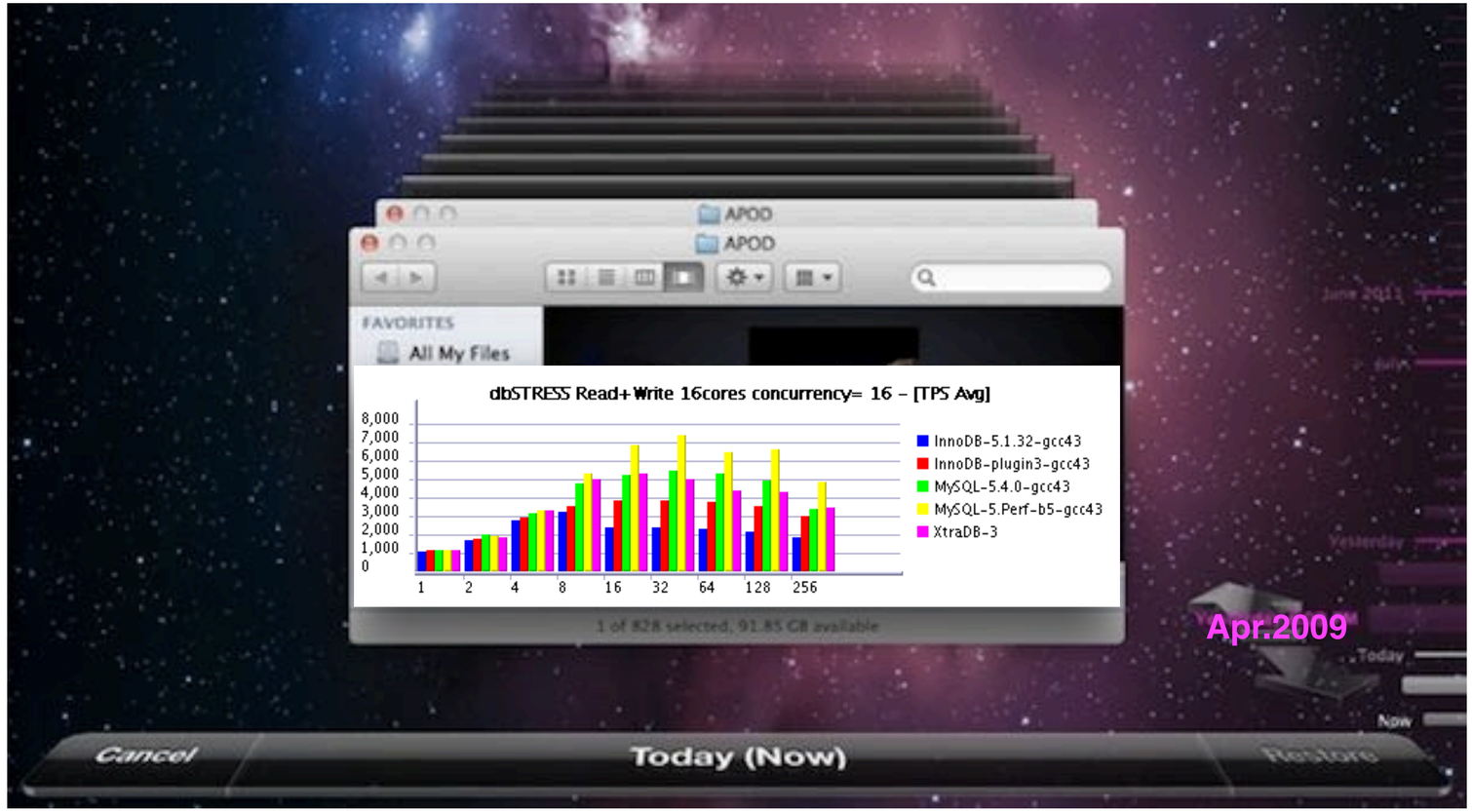
dbSTRESS 10M RW1 Test + HPK=on

- 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

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 profiler stats), 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.