MySQL Performance: Scalability & Systems Evaluation Overview 2025

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

- **Standalone** MySQL Performance & Scalability Overview @Linux
- Systems Evaluation by MySQL Benchmark Workloads
- Q & A

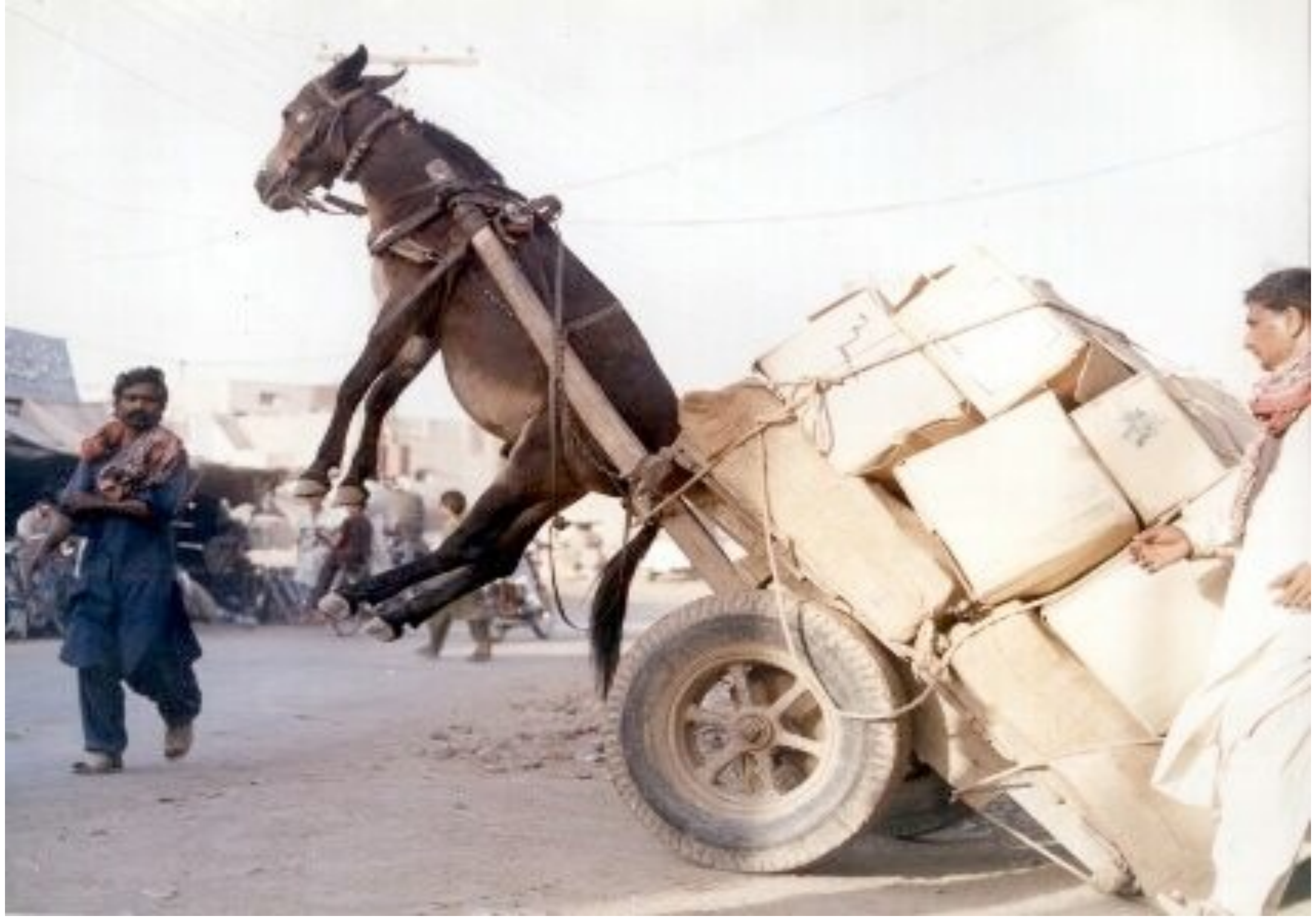
Why Scalability ?...

Why Scalability ?..

- Any solution may look “good enough”...

Why Scalability ?..

- Until it did not reach its limit..



Why Scalability ?..

- And even improved solution may not resist to increasing load..

Why Scalability ?..

- And reach a similar limit..

Why Scalability ?..

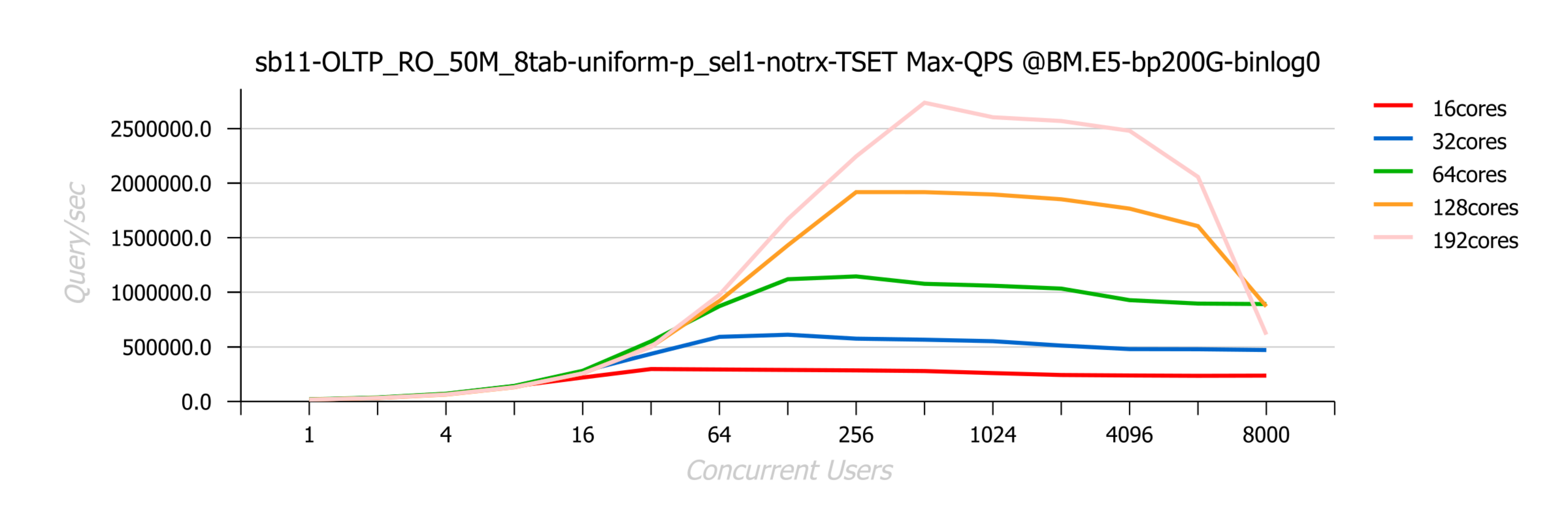
- Analyzing your workload performance and scalability by testing your limits may help you to understand ahead the resistance of your solution to incoming potential problems ;-)

Why Scalability ?..

- However :
  - Even a very powerful solution, but given in wrong hands may still be easily broken!... :-)

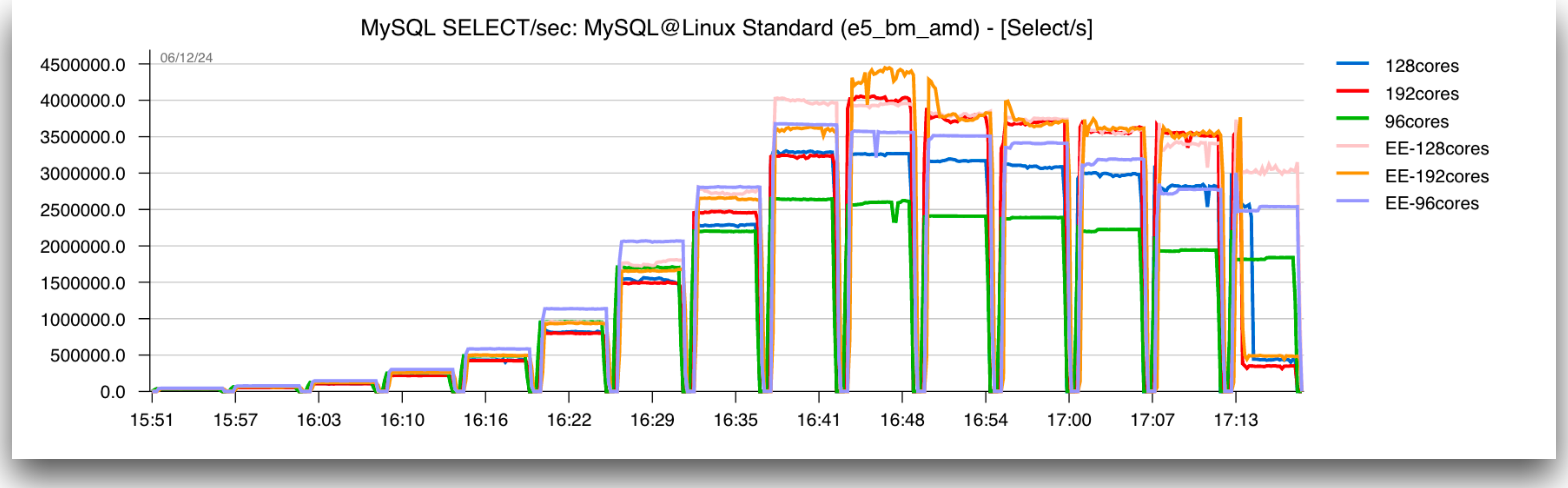
Scalability..

- what do we speak about ?
  - in short, scalability expectation : more resources => more work done
  - efficiency expectation : do more with less ;-))



Scalability.. (one more ;-))

- using prep.statements + UNIX socket :



MySQL Scalability Milestones over past 15 years..

- MySQL 5.5
  - delivered “already known” solutions (except BP instances and few other)..
- MySQL 5.6
  - first fundamental changes (kernel_mutex split, Adaptive Flushing, G5 patch, 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 (can be pretty big)
  - RW scalability.. & data security..
  - NOTE : Continuous Release Model ! => + new DBLWR, LOCK mgmt, REDO resize, etc..

MySQL Realities in 2025..

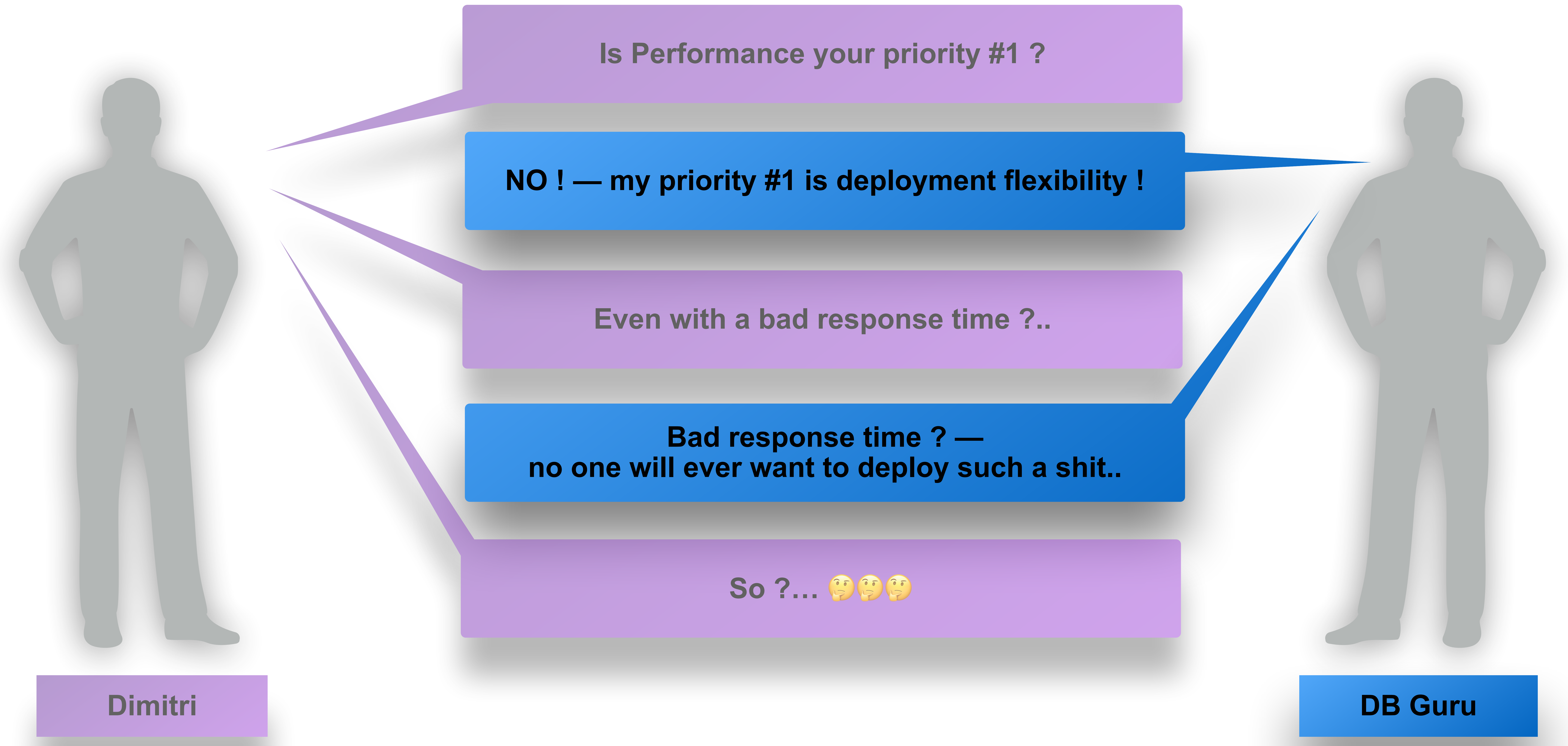
- Users before => “too long to wait for new features in new release !”
  - MySQL => here is MySQL 8.0 Continuous Release !

- Users => “too new many features every 3 months, we cannot follow !”
  - MySQL => OK, here is MySQL 8.4 LTS ;-))

- Users before => “I have 96cores systems, MySQL don’t scale !”
  - MySQL => OK, let’s work hard to improve overall scalability !

- Users today => “How I can speedup MySQL on my 2cores VM ?”..
  - MySQL => what ?.. ;-))
  - (hm.. maybe we should reconsider support for MySQL 5.1 ??.. ;-))

Why MySQL Performance ? => entry ticket



Is Performance your priority #1 ?

NO ! — my priority #1 is deployment flexibility !

Even with a bad response time ?..

Bad response time ? —
no one will ever want to deploy such a shit..

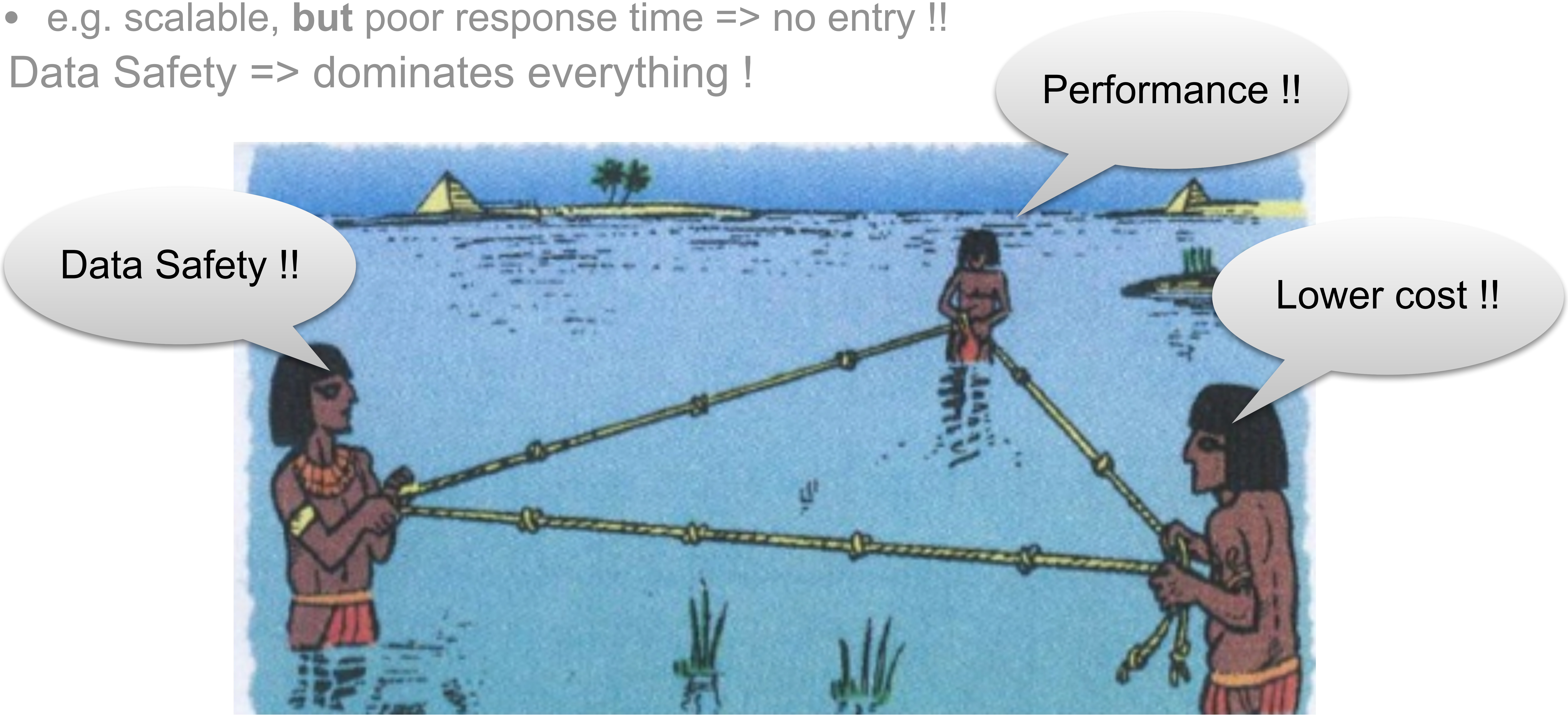
So ?… 🤔🤔🤔

Dimitri

DB Guru

About Priority Balance

- Performance -vs- Data Safety -vs- Lower Cost…
  - Performance => entry ticket
    - e.g. scalable, **but** poor response time => no entry !!
  - Data Safety => dominates everything !

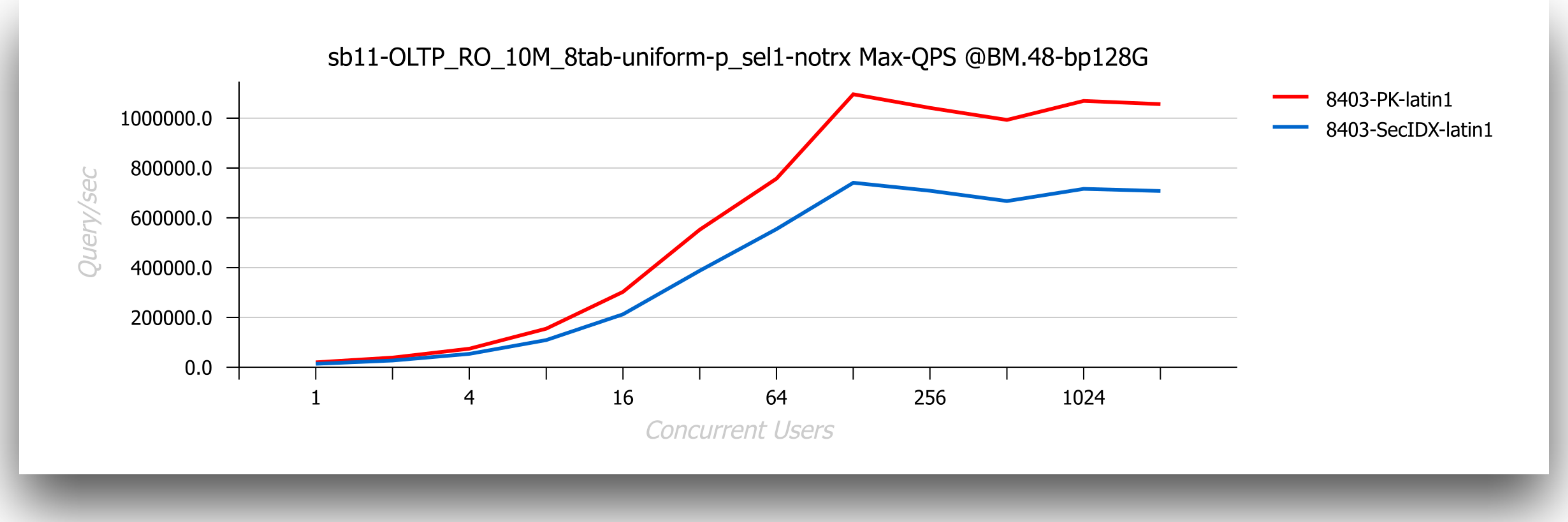


RO Scalability Limits

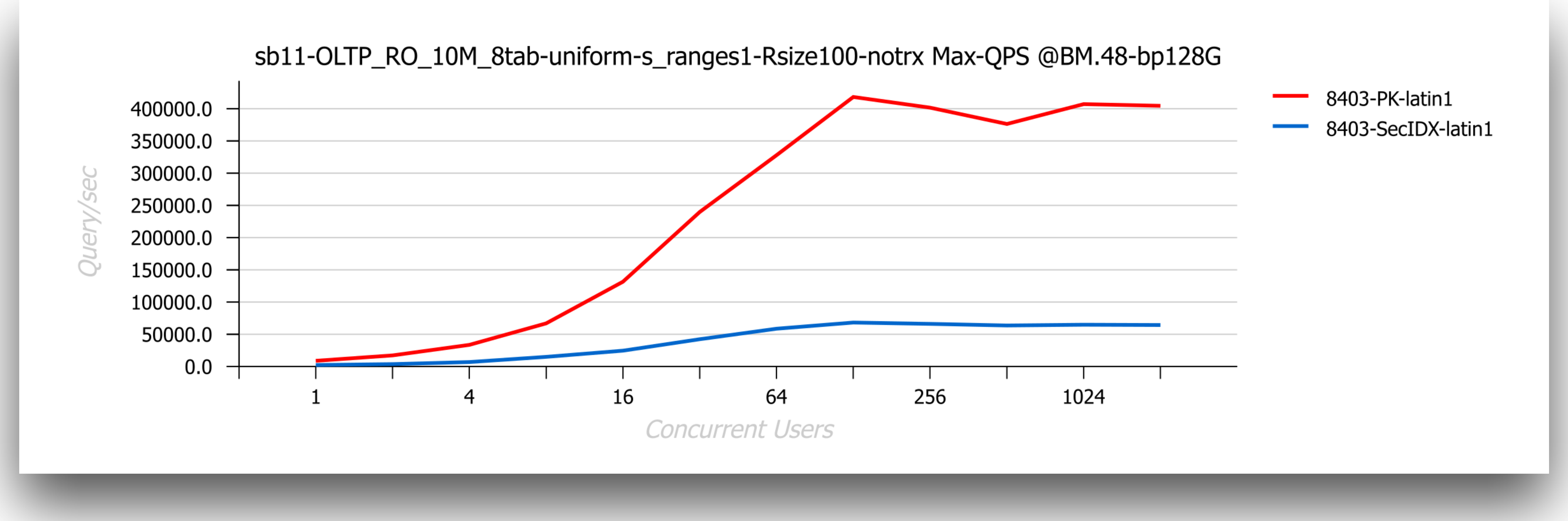
- simple & fast queries ? query execution plan ?
- malloc ! => consider tcmalloc or jemalloc
- re-connect or persistent connections ??
- SSL overhead
- AHI : on / off ? (since 8.4 def: off)
  - speed-up on low load
  - bottleneck on high load
- RW-lock contention on CPU cache
- PK in table is mandatory !
- PK vs SK lookups
- ...

RO Limits : PK -vs- SK lookups

- Point-SELECT
  - SELECT ..
    WHERE id = $v



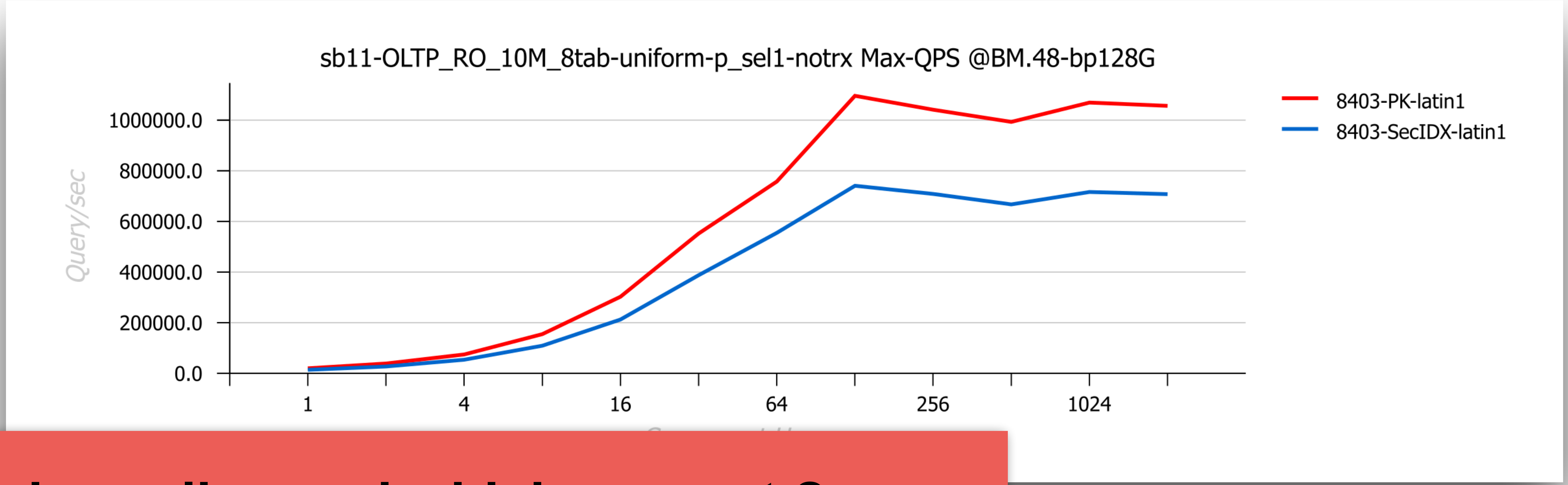
- Range-SELECT
  - SELECT ..
    WHERE id
    BETWEEN $v and $v + 100



ORACLE

RO Limits : PK -vs- SK lookups

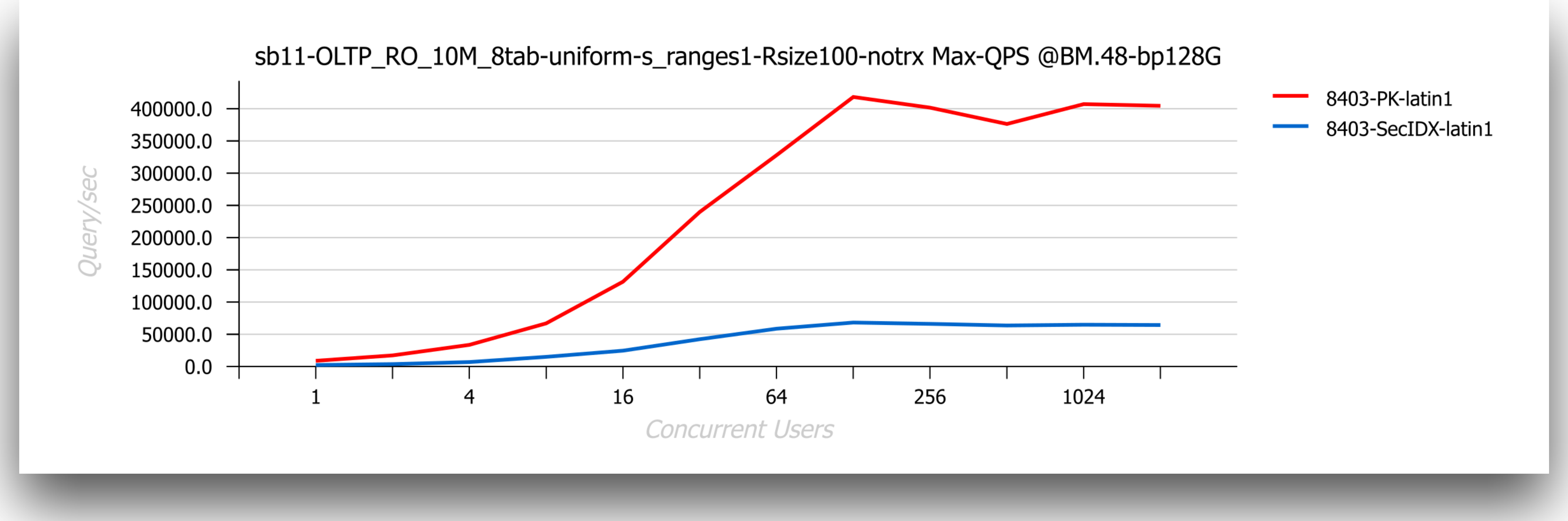
- Point-SELECT
  - SELECT ..
    WHERE id = $v



sb11-OLTP_RO_10M_8tab-uniform-p_sel1-notrx Max-QPS @BM.48-bp128G

Q : which one is scaling and which one not ?

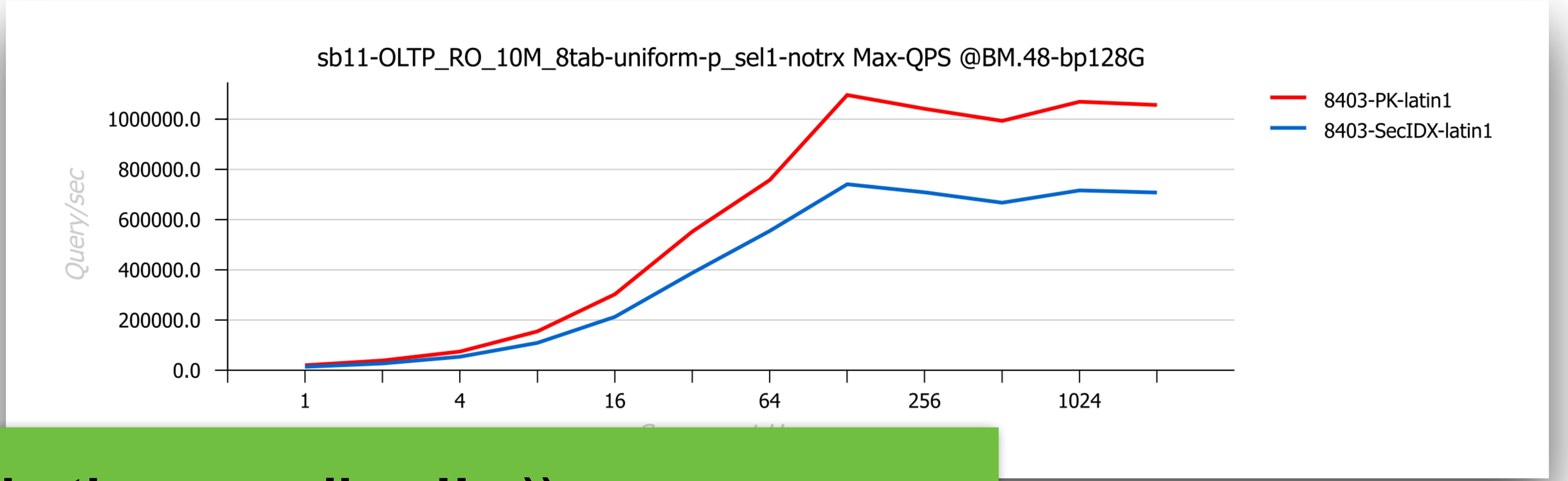
- Range-SELECT
  - SELECT ..
    WHERE id
    BETWEEN $v and $v + 100



sb11-OLTP_RO_10M_8tab-uniform-s_ranges1-Rsize100-notrx Max-QPS @BM.48-bp128G

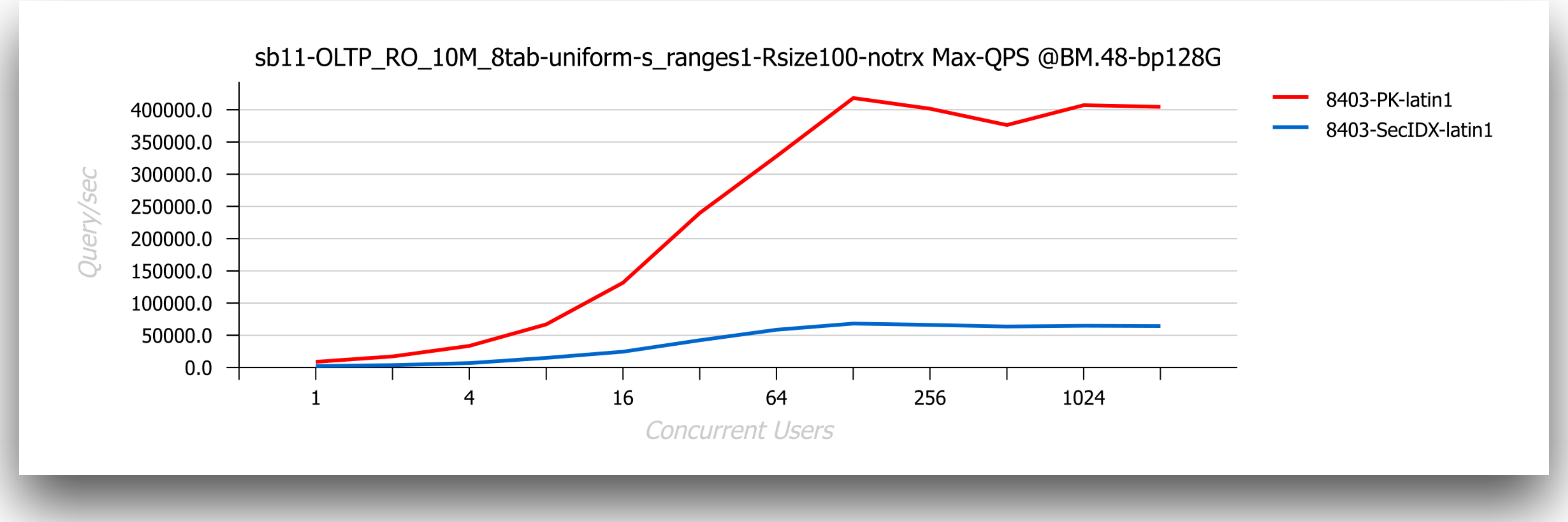
RO Limits : PK -vs- SK lookups

- Point-SELECT
  - SELECT ..
    WHERE id = $v



A : both are scaling !! ;-))

- Range-SELECT
  - SELECT ..
    WHERE id
    BETWEEN $v and $v + 100

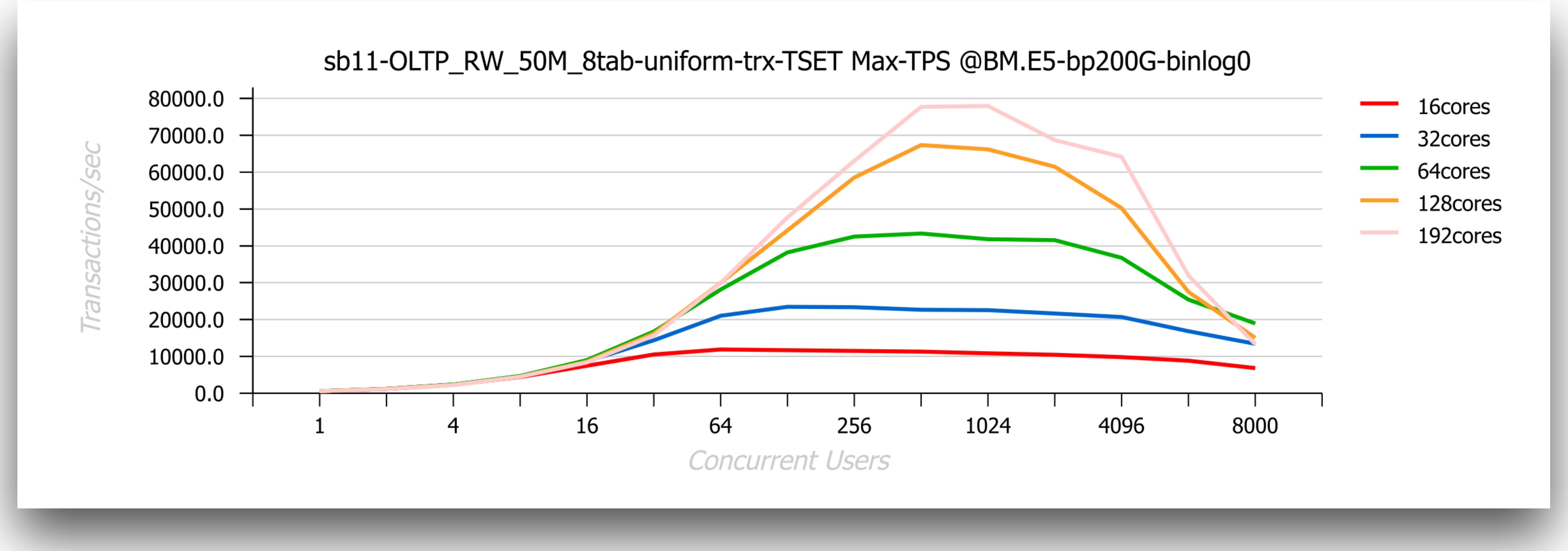


RW Scalability Limits (RO + more)

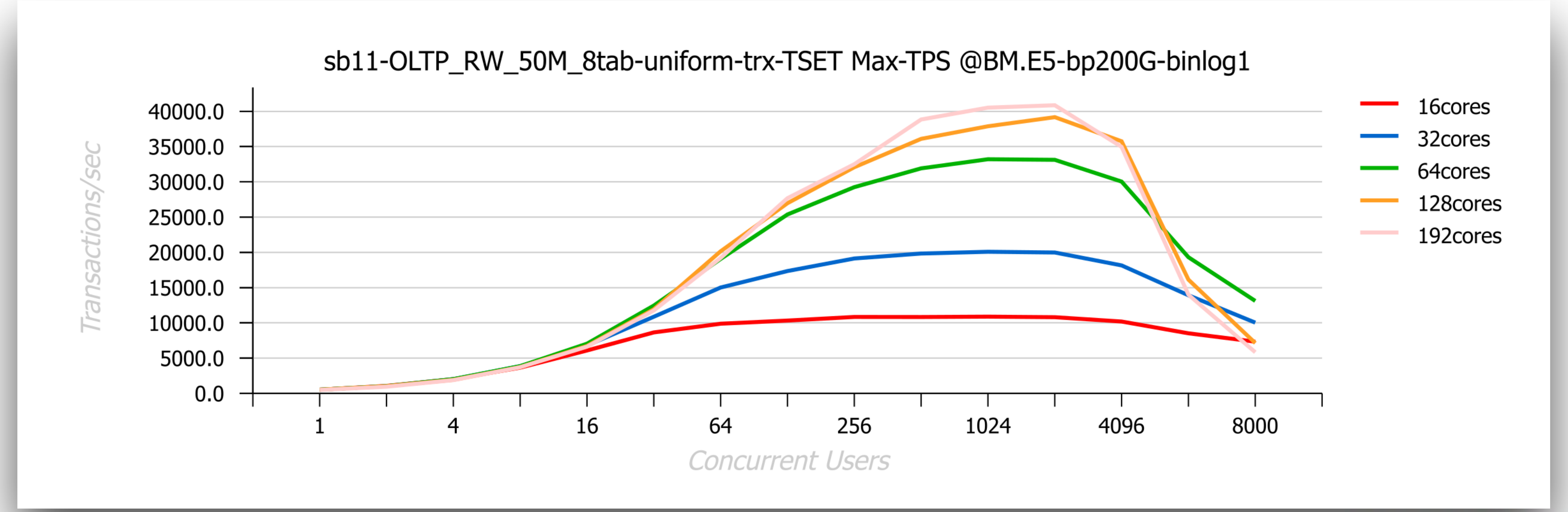
- REDO write+fsync latency
  - ex. 1ms = 1000 TPS max for single-thread
  - e.g. single-thread apps is not a right design ! ;-))
- REDO efficiency
  - small systems up to 8cores => REDO threads = OFF
- ReadView lock contentions (part of "trx_sys")
- LOCK mgmt contentions (“lock_sys”, largely improved since 8.0)
- data contention ! => new ThreadPool to solve this !
- Binlog impact
  - (post-REDO activity design)

Binlog Impact on Scalability | OLTP_RW

- Binlog = ON

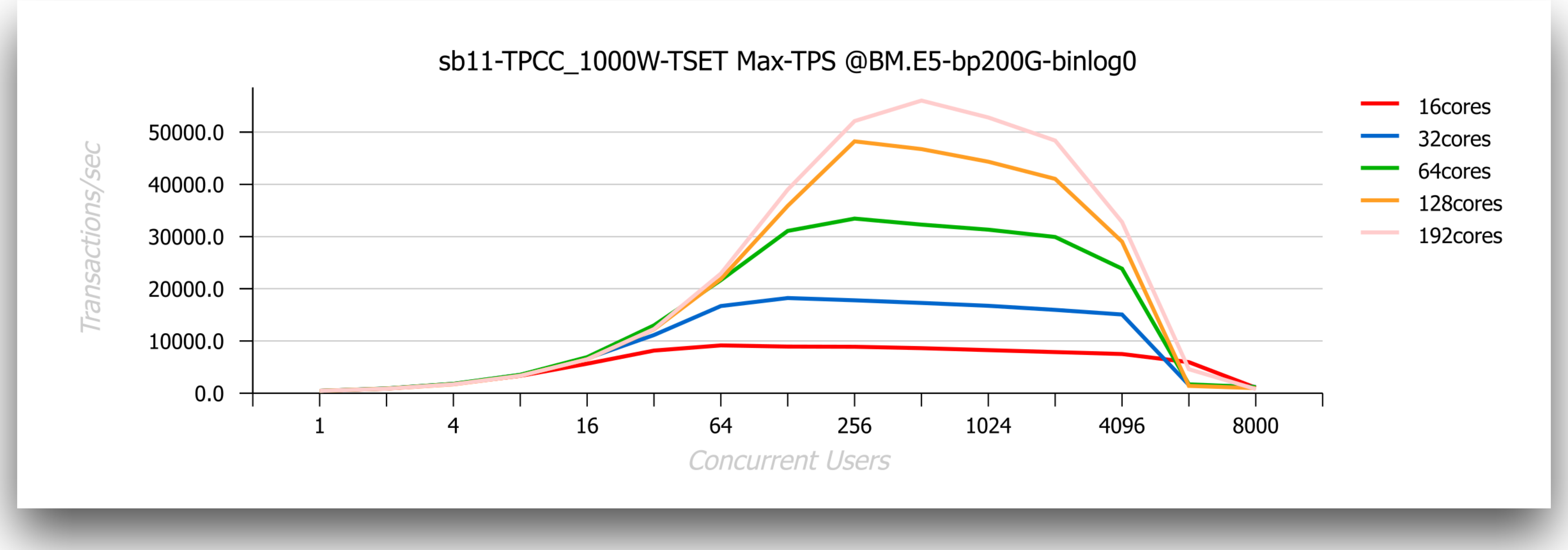


- Binlog = OFF

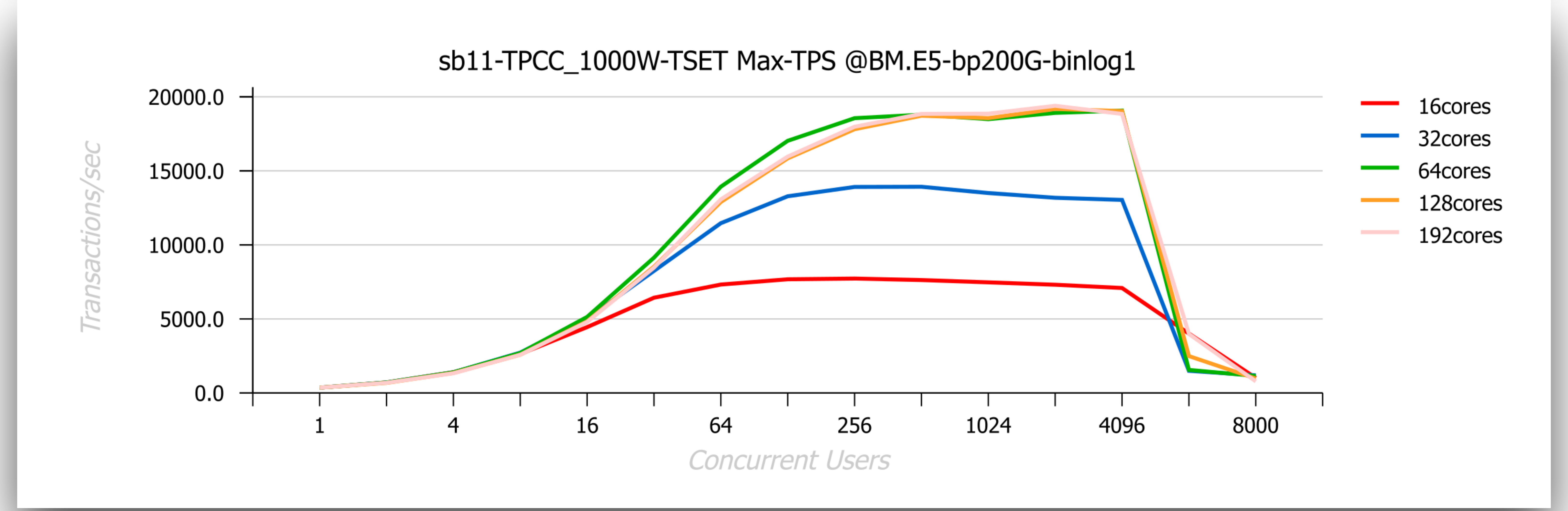


Binlog Impact on Scalability | TPCC-1000W

- Binlog = ON



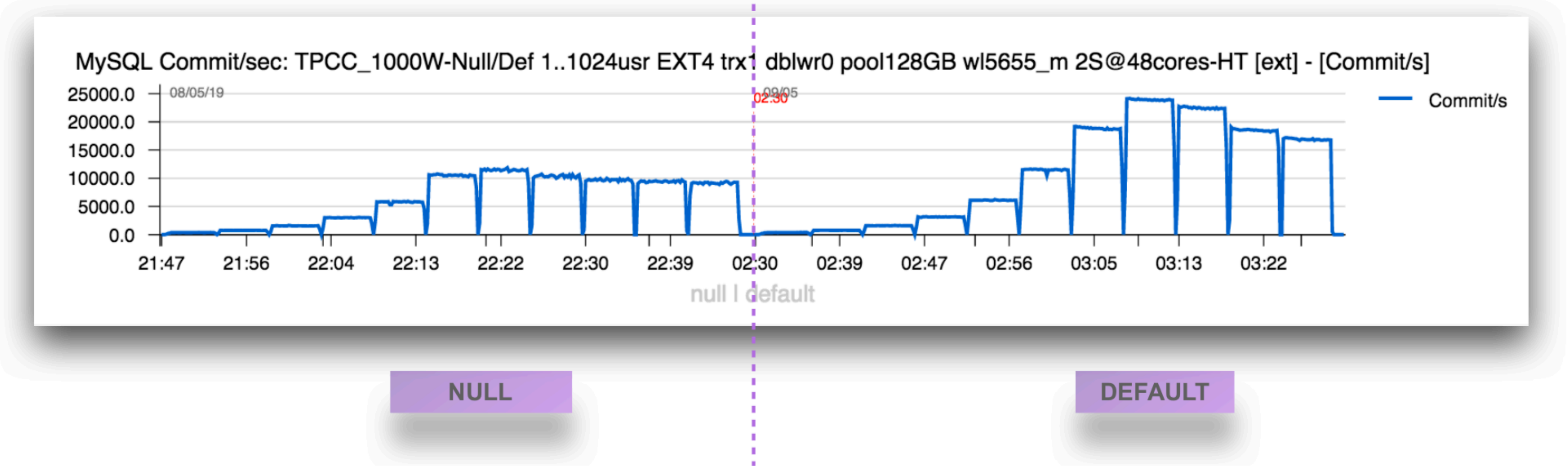
- Binlog = OFF



RW : In-Memory (e.g. no I/O Reads)

- REDO write latency & REDO space
- Page Flushing / IO capacity / REDO size
- Purge lagging / UNDO / History Length
- auto-truncate UNDO ?
- IBUF : no help, but merges may impact !
- NULL -vs- DEFAULT :
  - use DEFAULT to avoid massive page split / merges spikes !

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



RW : IO-bound (all previous + more)

- Storage IOPS capacity
- LRU Flushing driven
- BP instances / Page Cleaners
- Single Page Flushing
- DBLWR => if using RAID-n, then mind Storage stripe size !
- Oracle Linux Team => Atomic IO !
- IBUF : helps ! but beware side effects
  - ex. : on BP resize (reduce)
  - ex. : on recovery process (fixed since MySQL 9.2)
- fil_sys mutex contention (sharded since 8.0, but still problematic)
- partitions ?
- compression ?

Evaluating Systems for MySQL Performance

MySQL Config

NOTE to PeterZ :
please,
update your slides !!

- use 8.4+ version : **auto-adaptive** default config !!
- e.g. just use default config + InnoDB dedicated = ON
  - auto-adaptive : BP size, BP instances, Page Cleaners, REDO size, etc..
  - by default : AHI = off, IBUF = off, O_DIRECT
- data safety & security first !!!
  - trx_commit=1, sync_binlog=1 (don't expect HA will solve you)

```
[mysqld]
 innodb_dedicated_server = on
 max_connections = 4000
```

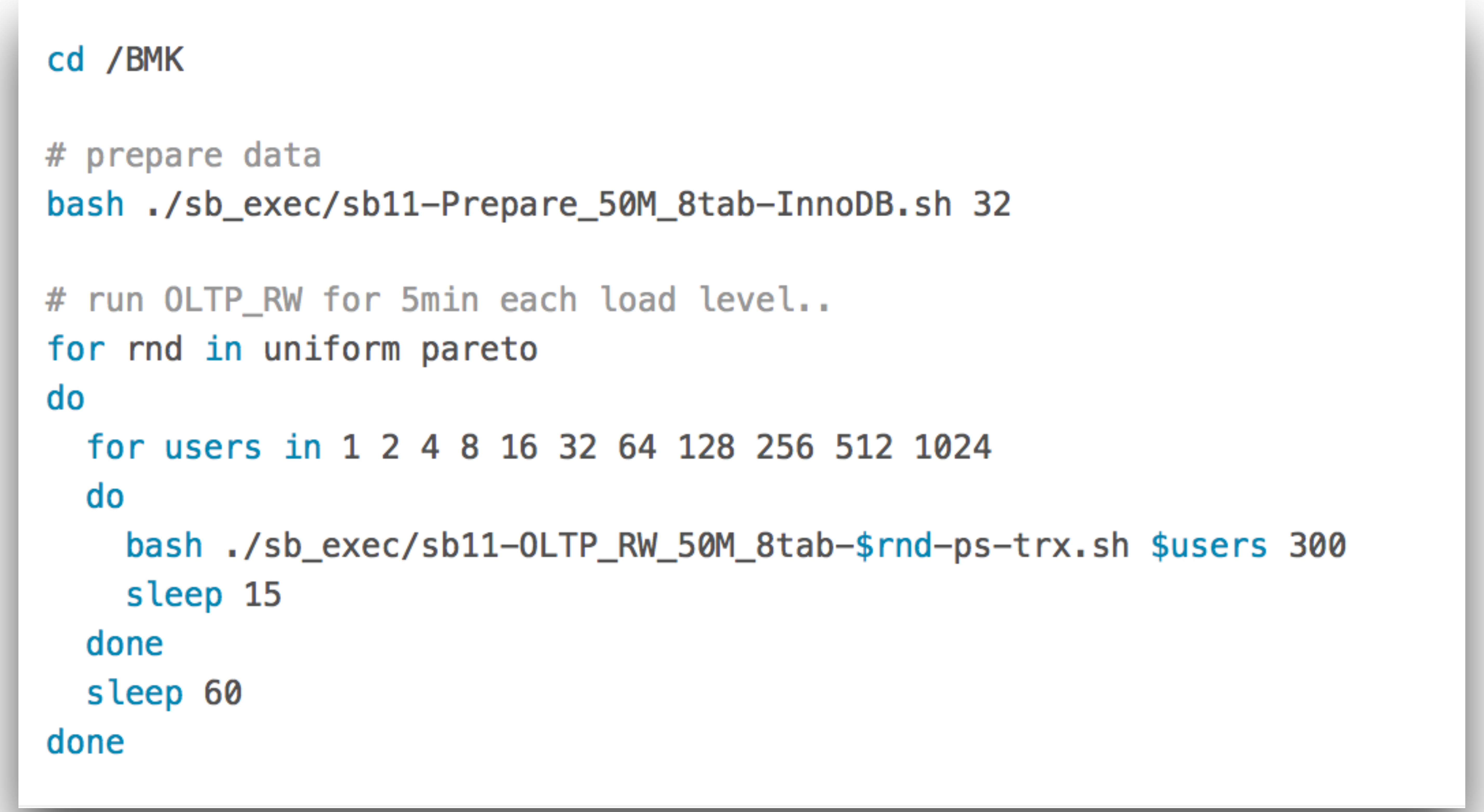
- NOTE : consider also to test with MySQL EE !!

Test Workloads

- start with generic workloads
  - more easy to compare with “expectations”
  - more easy to report problems and reproduce it by others
- understand what you're ~~doing~~ testing !!
  - go from the most simple to more complex
  - have a clear target from the beginning ! (except if you have unlimited time ;-))
  - when Network is poor => you’re not testing the System, but Network !
  - using Java-based load-generator ? => be sure you’re not just testing JVM itself !
  - using too small data set ? => you’re probably just testing CPU cache ;-))
  - using not-scaling test workload => you’re not testing the System, but bottleneck !
  - e.g. avoid to have useless results..
- finish with your's **Production** workload
  - if not possible, then with what is **the most representative** for your Production !
  - **Sysbench Lua** : simple & efficient **platform** to develop test workloads for MySQL !

To Simplify Your Life : Benchmark Kit (BMK-kit)

- Ready out-of-the-box :
  - shipped with Sysbench binaries for Linux x64 / arm64
  - pre-generated scripts for all workloads
  - script name = test & options
  - all standard Sysbench tests
  - many additional extensions
  - includes TPCC-like
  - includes dbSTRESS
  - frequent updates



```
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
```

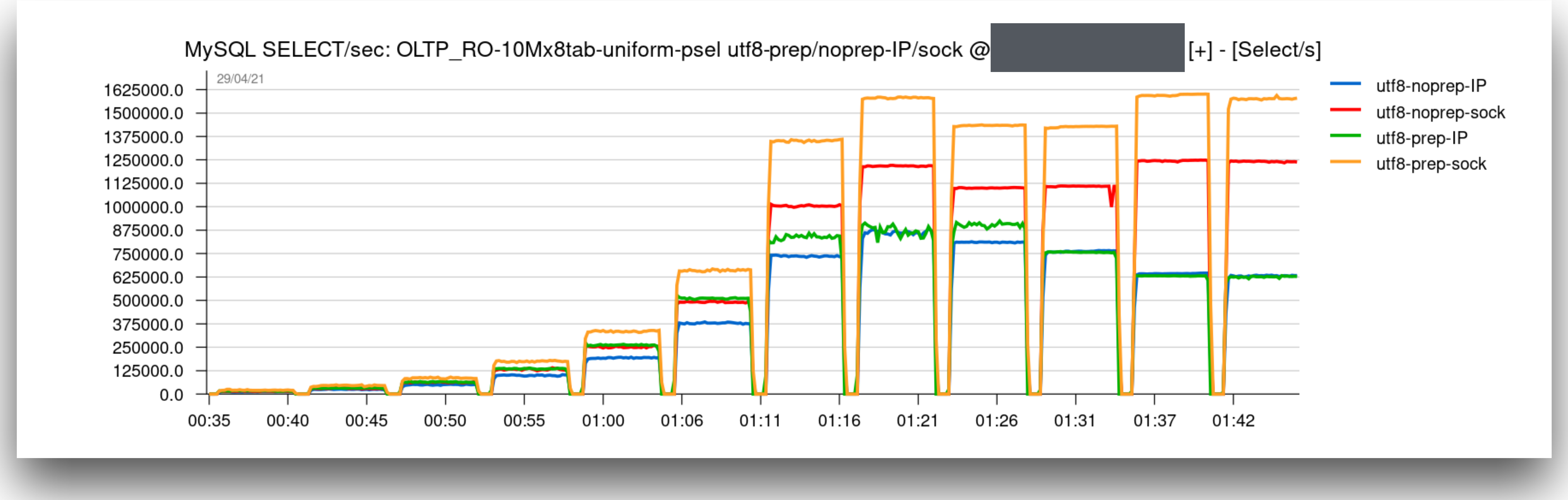
Complete detailed HOWTO : http://dimitrik.free.fr/blog/posts/mysql-perf-bmk-kit.html

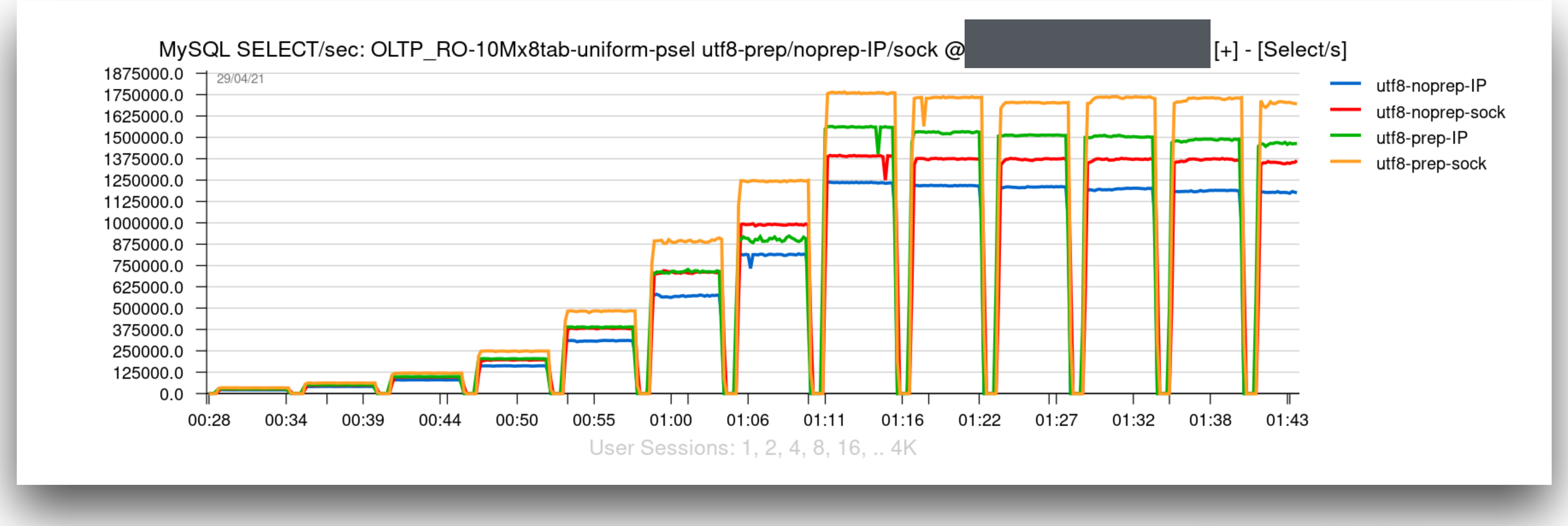
Starting Point : RO In-Memory

- start with Point-Selects
  - 10Mx8tab = 20GB, or 50Mx8tab = 100GB — according to available RAM
- BP size big enough to keep your data cached
- prepared statements = on / off ??
- first use connections via UNIX-socket to see the impact
- then IP-port
- then real Network
- finish with OLTP_RO

UNIX Socket -vs- IP port

- point-selects
- prep. stmt = on/off





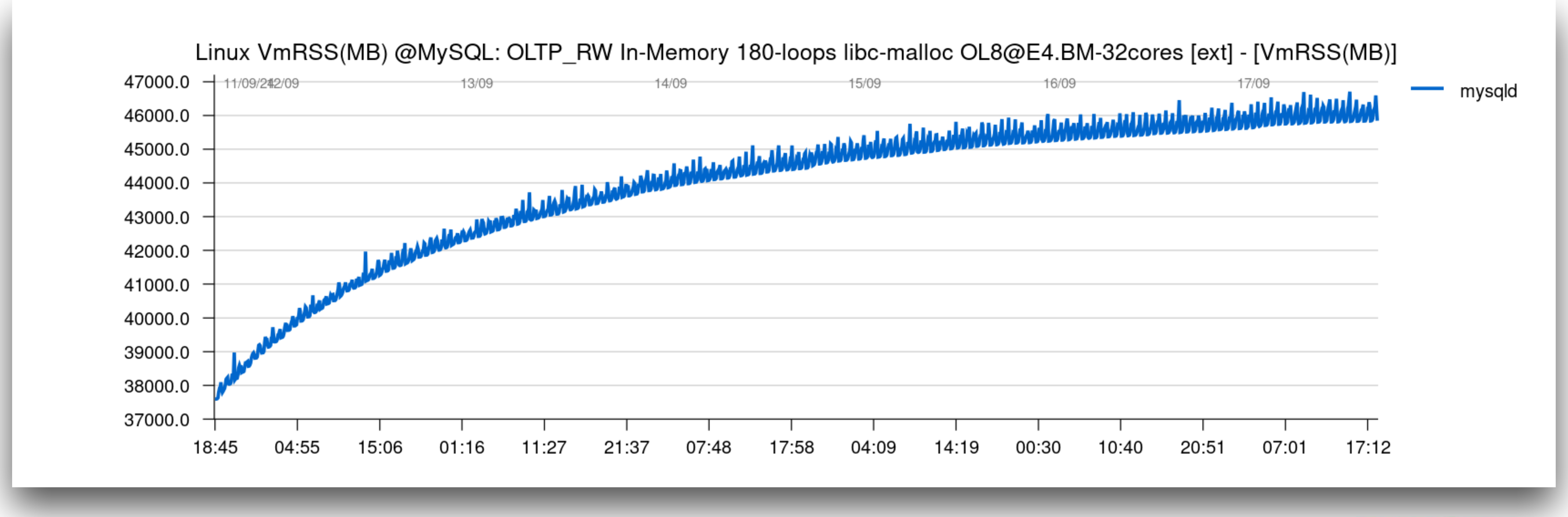
User Sessions: 1, 2, 4, 8, 16, .. 4K

RO In-Memory

- BM : NUMA & CPU affinity ??
- VM : always check CPU Steal% !!
- single-thread matters !!
  - base line for response time !
  - (e.g. can be only higher with growing load levels)
- malloc lib ! => critical !
  - glibc-malloc => **DANGER** !!
  - jemalloc => better, but may have allocation spikes
  - gpertools tcmalloc => currently the most efficient !

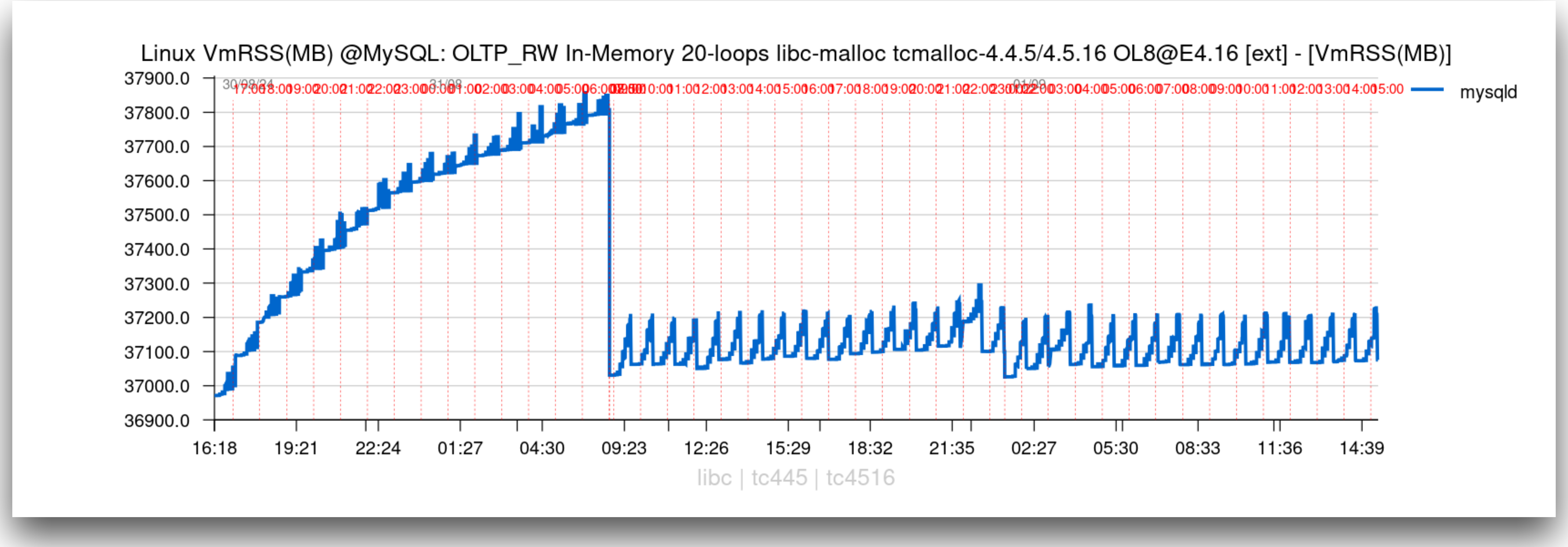
Malloc Libs Today

- glibc-malloc :
  - the latest glibc-malloc can be used with MT-apps today in most cases
  - but memory fragmentation !
- glibc-malloc : +10GB RSS usage over 1 week..



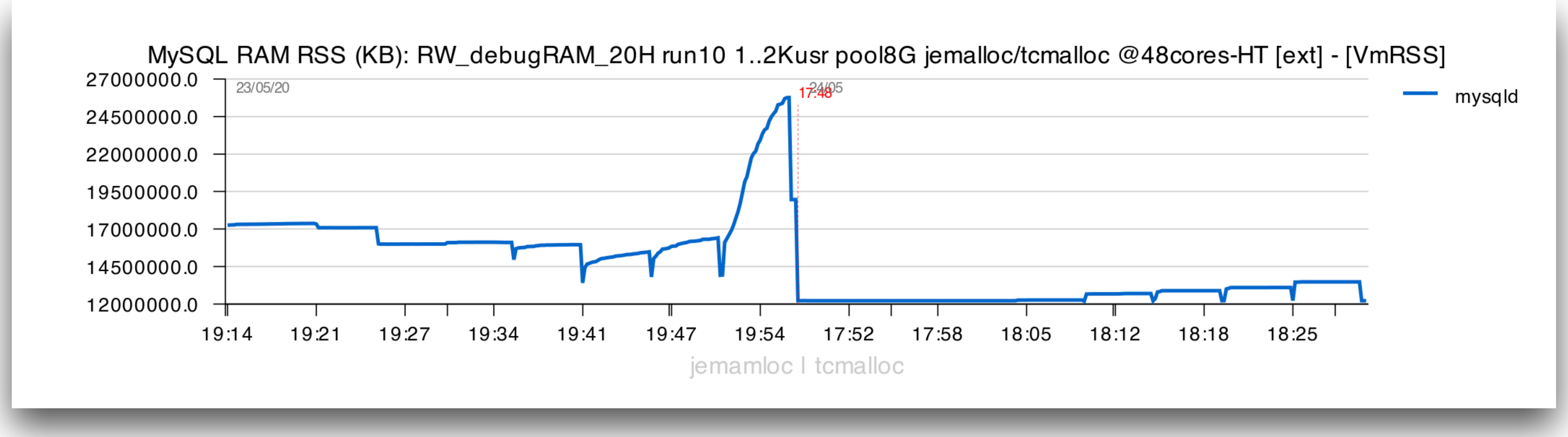
Malloc Libs Today

- glibc-malloc -vs- tcmalloc :



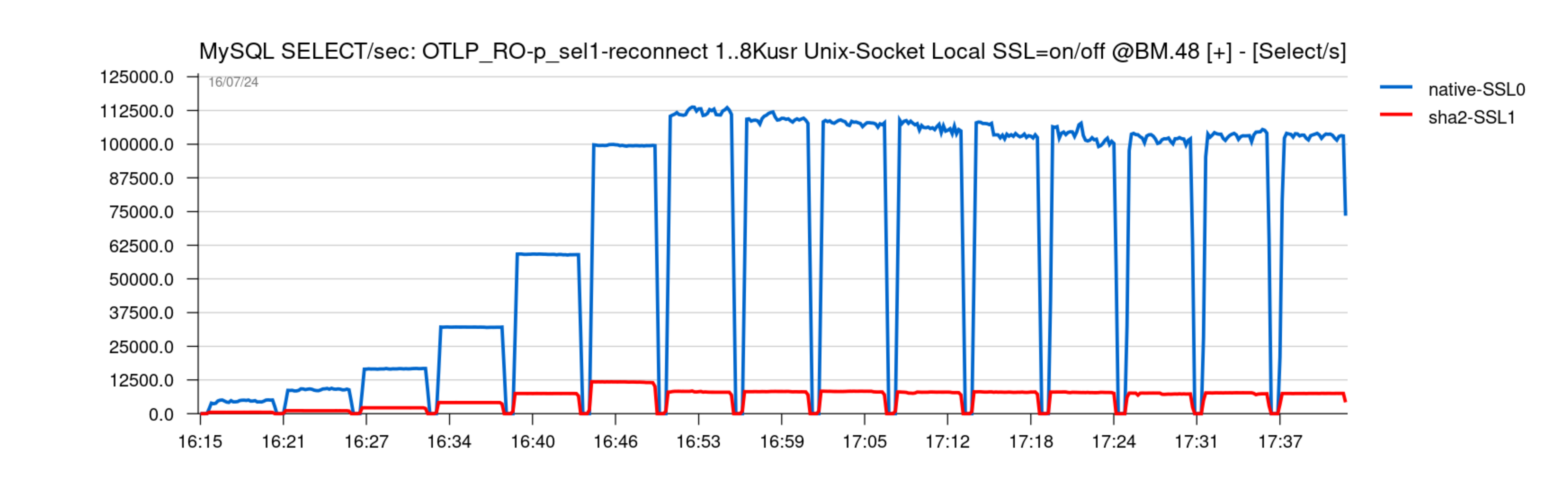
Malloc Libs Today

- jemalloc -vs- tcmalloc :



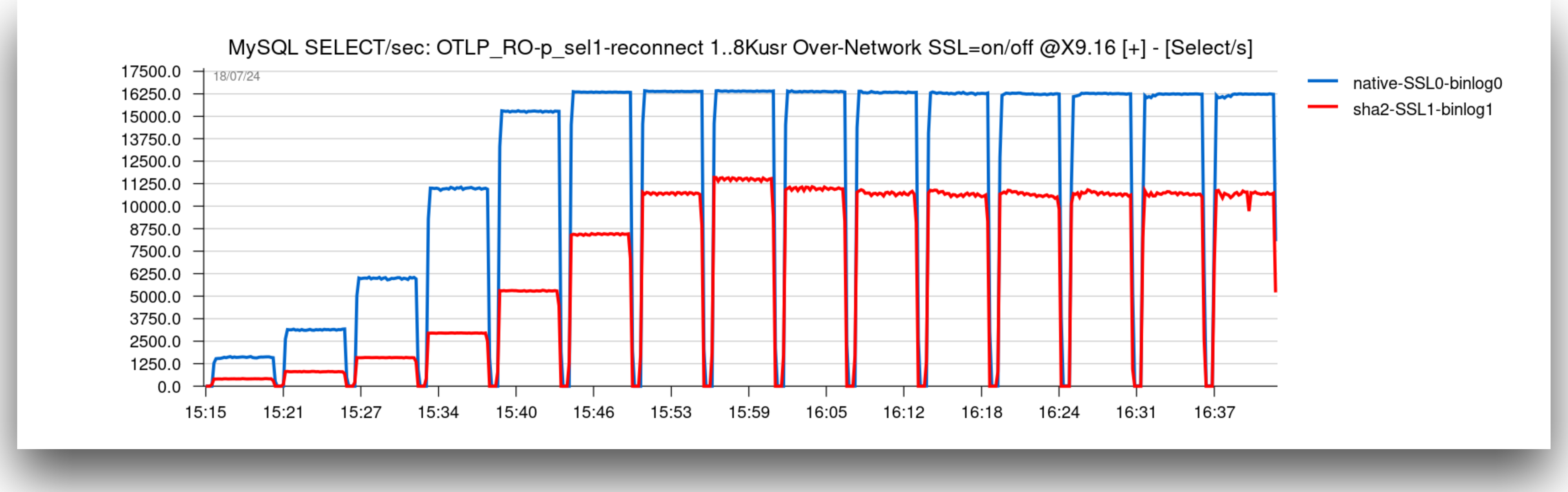
SSL Impact

- OpenSSL versions :
  - avoid OpenSSL-1.0 !! => bottleneck itself !
  - OpenSSL-1.1.1 => the most efficient as of today
  - OpenSSL-3.x => loosing 15-20% comparing to 1.1.1
- Re-Connect ?
  - SSL connection initialization has a very high cost !!

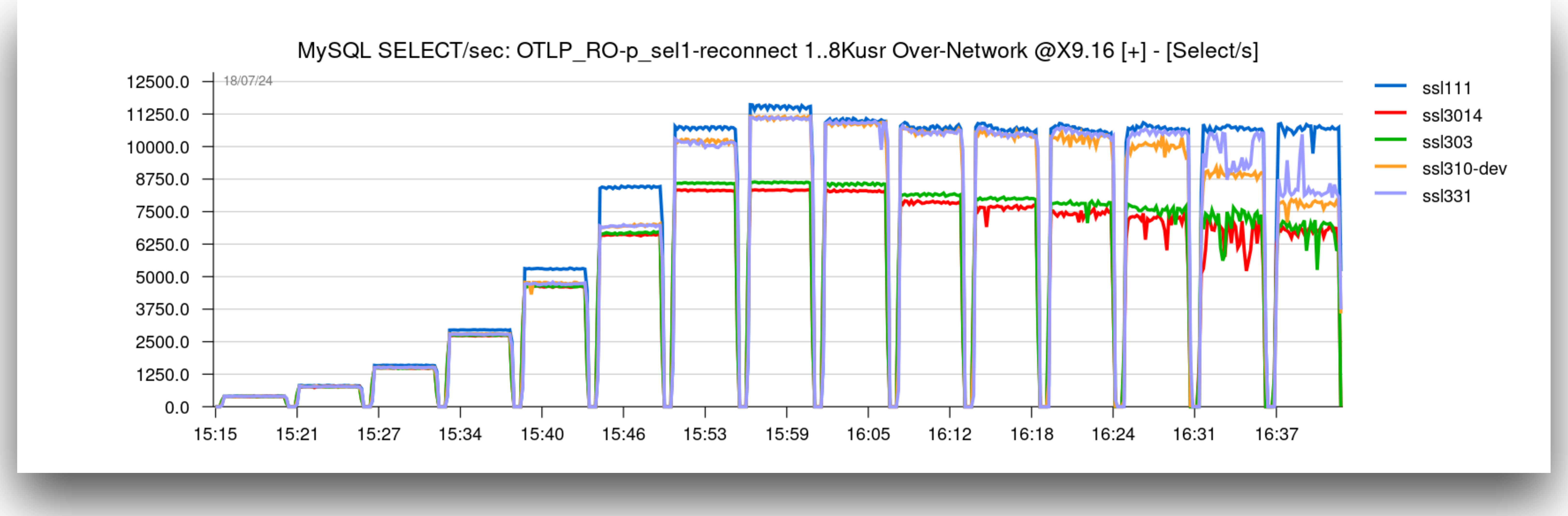


SSL Impact | Re-Connect Over Network

- SSL = on / off
  - x3 times on low/mid load
  - 30% on high load



- OpenSSL versions :
  - OpenSSL-1.1.1 => best
  - OpenSSL-3.0 => bad
  - OpenSSL-3.1+ => better



RW In-Memory

- XFS or EXT4 ?
- REDO latency : single thread write+fsync for single file !
  - NOTE : latency is not always directly related to IOPS !!

- simple test for REDO latency :

```
$ fio --ioengine=psync --blocksize=512 --fsync=1 -rw=write \
 --numjobs=1 --iodepth=1 --name=TEST --filename=/path/to/file \
 --size=256m --runtime=60
```

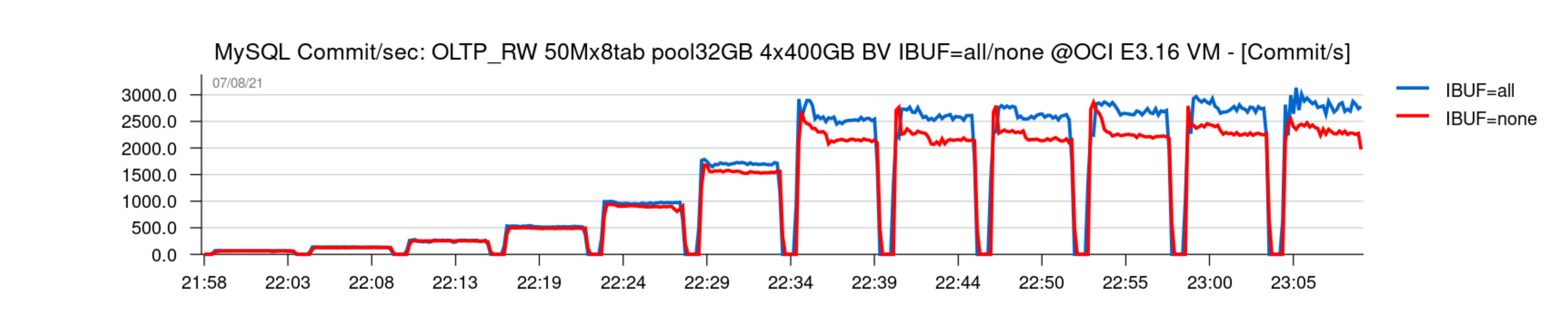
- Binlog = ON/OFF

RW IO-bound

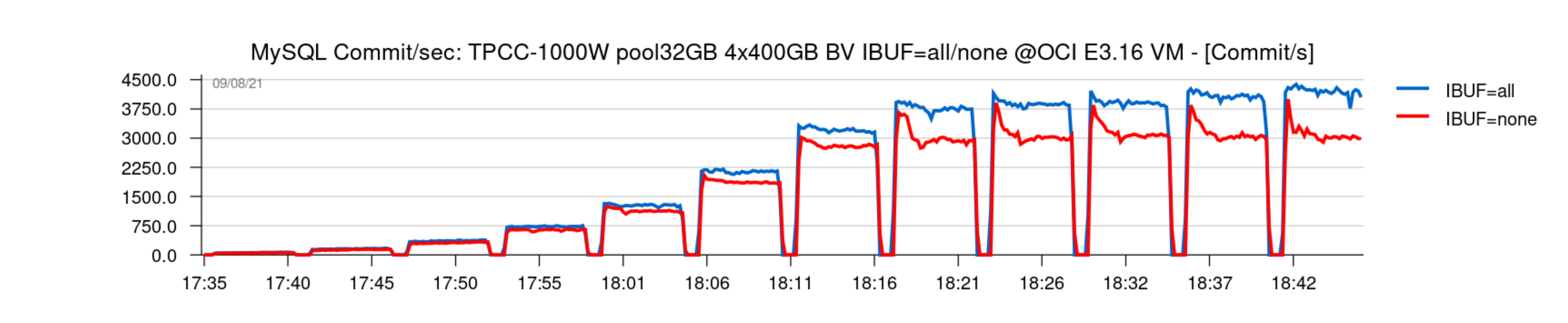
- use 1TB of data (or more) to make it more realistic !!
- IBUF helps ! but beware trade-offs !
- monitoring !
  - e.g. don't look only on final numbers !!
- beware of fake results !
- compression ?
- partitions ?

InnoDB IBUF (Change Buffer)

- IO-bound OLTP_RW :

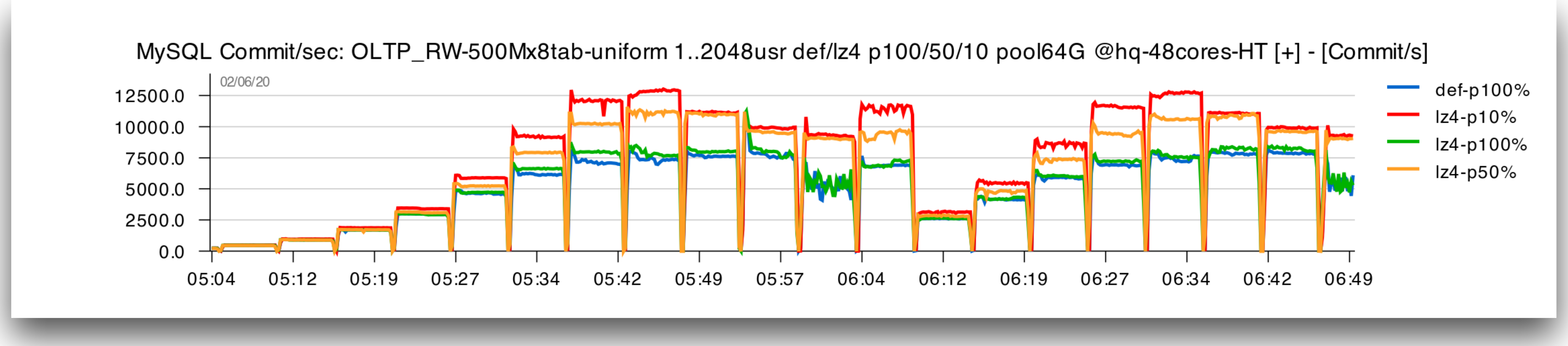


- IO-bound TPCC :



InnoDB Transparent Compression

- IO-bound OLTP_RW
- different level of data “randomness” : 10% / 50% / 100%
- transparent compression LZ4, data size = 1TB :
  - **up to 50% gain** when data are **favorable** for compression !!



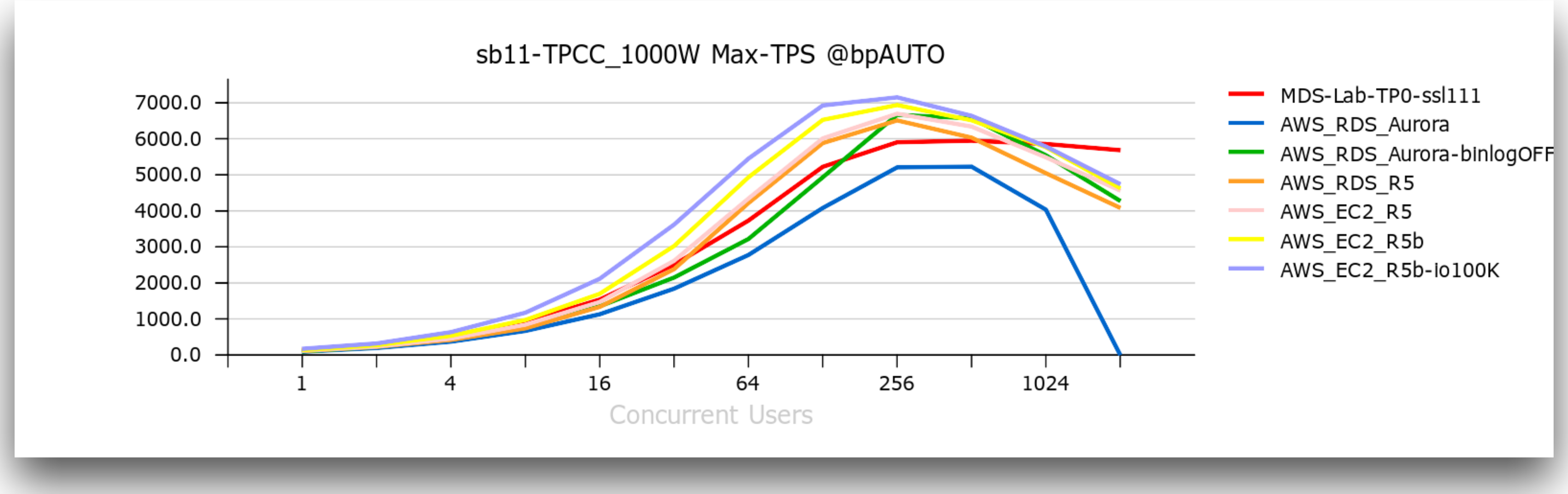
Shortlist Scenario “by Dimitri”

- In-Memory : 100GB
  - (50Mx8tab / 100W / 1000W)
- IO-bound : 1TB
  - (500Mx8tab / 10000W)
- OLTP_RO point-selects
- OLTP_RO
- OLTP_RW
- UPDATEs bombarding
- INSERTs bombarding
- TPCC-100W/ 1000W/ 10000W
- ... more tests if time permits ...

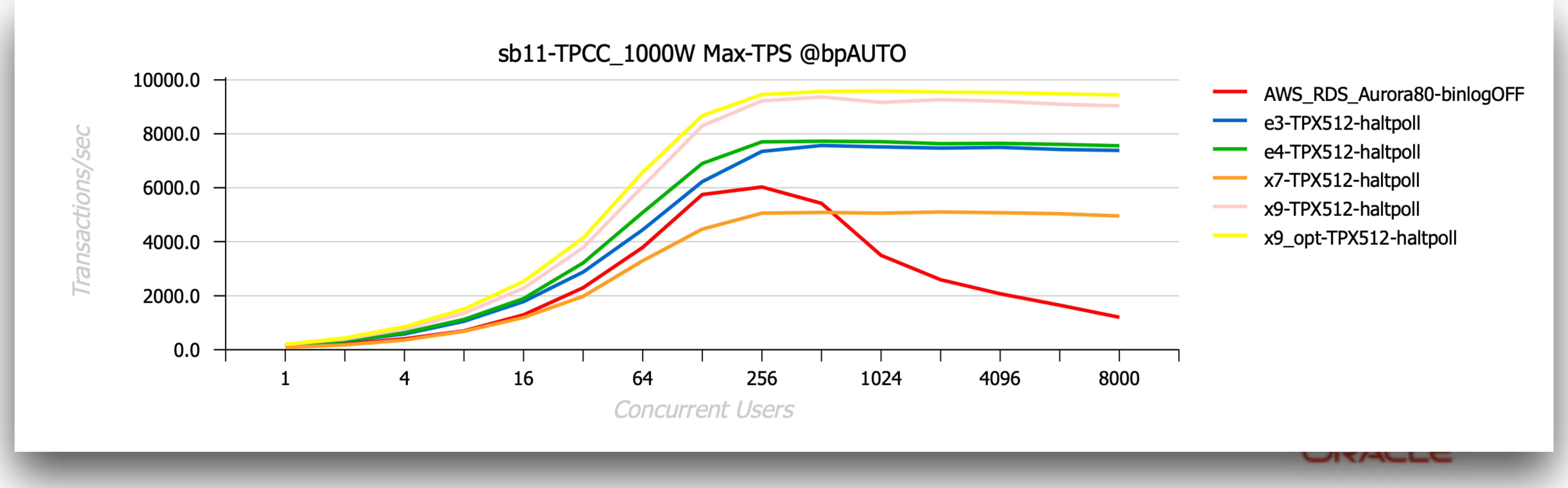
- **NOTE** : just to remind, the target is about "system evaluation" !!

Few TPCC Results | 1000W (100GB), RAM 256G, 16cores

- Amazon
  - RDS
  - EC2
  - Aurora

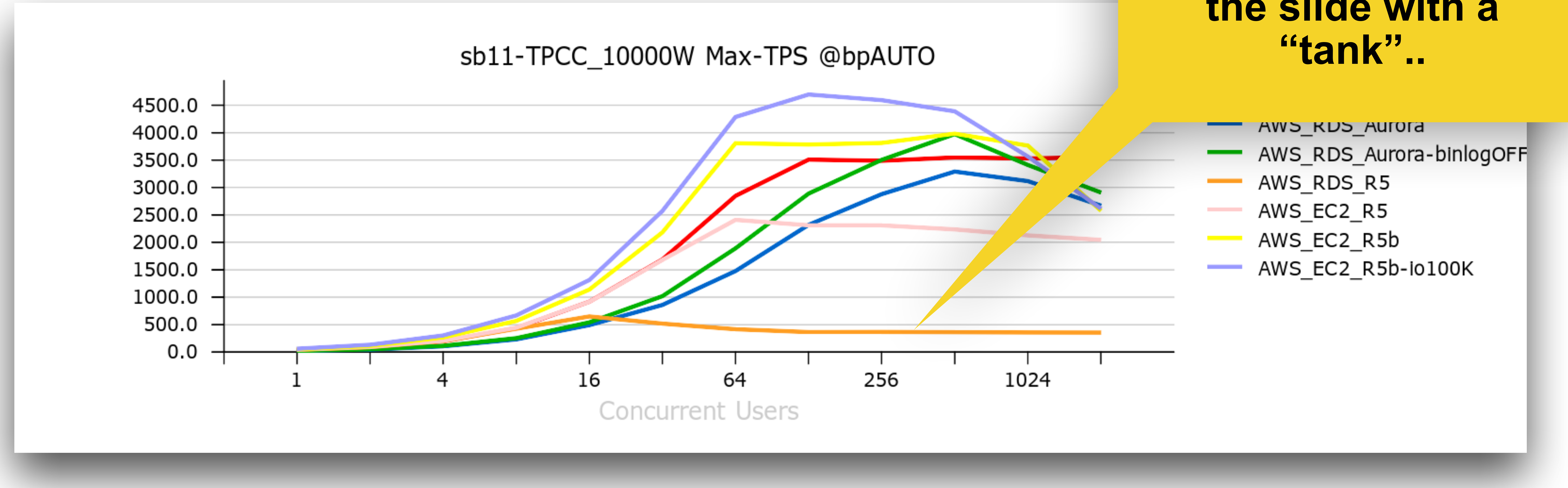


- MDS
  - E3
  - E4
  - X9



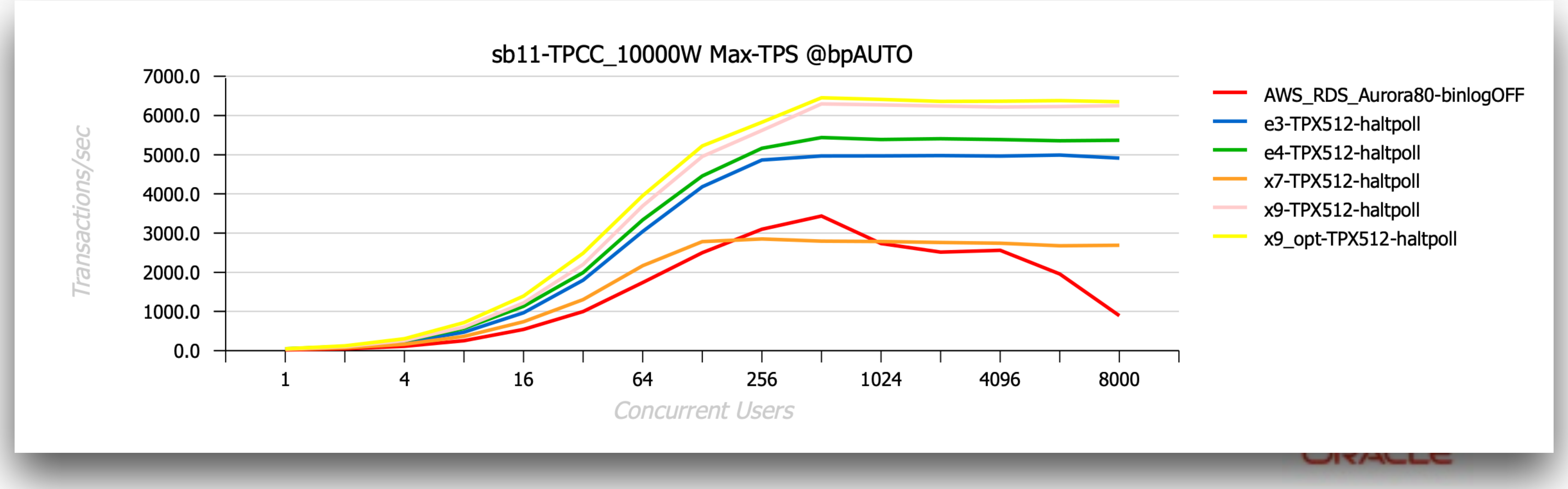
Few TPCC Results | 10000W (1TB), RAM 256G, 16cores

- Amazon
  - RDS
  - EC2
  - Aurora



If you recall the slide with a "tank"..

- MDS
  - E3
  - E4
  - X9



Hope everything is more clear now for you !

One more thing.. ;-))

- All graphs are built with ***dim_STAT*** (http://dimitrik.free.fr)
  - All System load stats (CPU, I/O, Network, RAM, Processes, etc..)
    - Mainly for Linux x64 / arm64 / MacOS (but can be any other UNIX too :-))
    - Add-Ons for MySQL, Oracle DB, PostgreSQL, Java, etc.
  - MySQL Add-Ons:
    - mysqlLOAD : compact stats data, multi-host monitoring oriented
    - mysqlWAITS : top wait events from Performance SCHEMA
    - InnodbSTAT : most important data from InnoDB status
    - innodbMUTEX : monitoring InnoDB mutex waits
    - innodbMETRICS : all counters from the METRICS table
    - Perf Profiling / DTrace call-stacks / etc..

- Links
  - http://dimitrik.free.fr - dim_STAT
  - http://dimitrik.free.fr/blog/posts/mysql-perf-bmk-kit.html - BMK-kit
  - http://dimitrik.free.fr/blog - Articles about MySQL Performance, etc.

Thank You !!!