




ORACLE

# MySQL Performance: Demystified Tuning and Best Practices...

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
- Workload oriented tuning and MySQL Internals
- Performance improvements in MySQL 5.7 & Benchmark results
- Let's investigate together...
- Q & A



# Why MySQL Performance ?...

# Why benchmarking MySQL?..

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



# Why benchmarking MySQL?..

- Until it did not reach its limit..





# Why benchmarking MySQL?..

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





# Why benchmarking MySQL?..

- And reach a similar limit..



# Why benchmarking MySQL?..

- A good benchmark testing may help you to understand ahead the resistance of your solution to incoming potential problems ;-)



# Why benchmarking MySQL?..

- But keep it in mind:
  - Even a very powerful solution but leaved in wrong hands may still be easily broken!... :-)





# **The Main MySQL Performance Tuning**

## **#1 Best Practice is... ???..**



**The Main MySQL Performance Tuning  
#1 Best Practice is... ???..**

**USE YOUR BRAIN !!!... ;-)**

# The Main MySQL Performance Tuning **#1** Best Practice is... ???..

**USE YOUR BRAIN !!!... ;-)**

A yellow starburst graphic with a black outline and a drop shadow, containing the text 'THE MAIN SLIDE! ;-))'.

**THE MAIN  
SLIDE! ;-))**

ORACLE®



# Think “Database Performance” from the beginning!

- Server:

- Having faster CPU is still better! 32 cores is good enough ;-)
- OS is important! - Linux, Solaris, etc.. (and Windows too!)
- Right malloc() lib!! (Linux: jemalloc, Solaris: libumem)

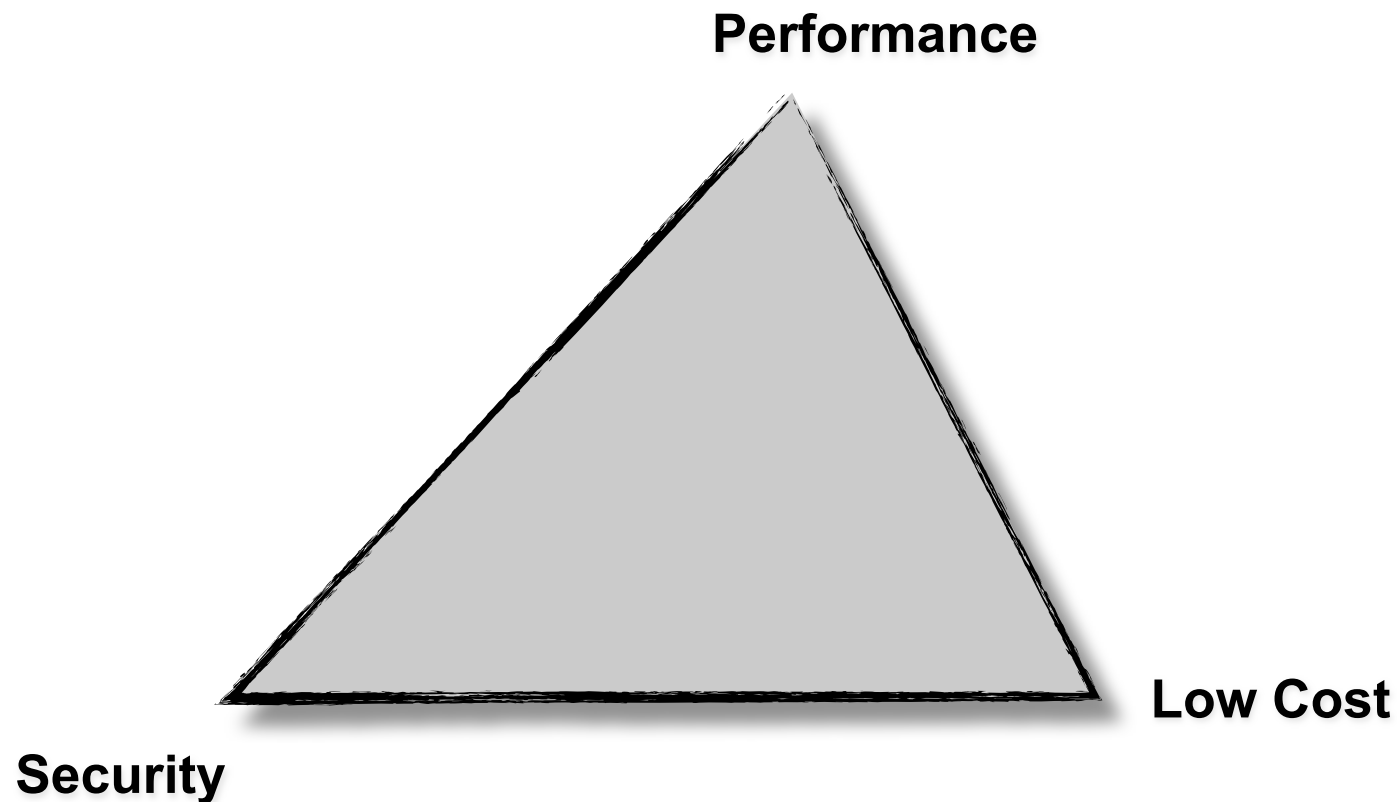
- Storage:

- Don't use slow disks! (except if this is a test validation goal :-))
- Flash helps when access is random! (reads are the most costly)
- FS is important! - ZFS, UFS, QFS, VxFS, EXT3, EXT4, XFS, etc..
- O\_DIRECT or not O\_DIRECT, AIO or not AIO, and be aware of bugs! ;-)
- **Do some generic I/O tests first !!** (Sysbench, IObench, iotop, etc.)

- Don't forget network !! :-) (faster is better, 10Gbit is great!)

# The Game of priorities & compromises...

- You'll always have a sacrifice of one from these 3 :



# Only a real test gives you a real answer...

- Avoid to tweak on production systems ;-)
  - Rather try to reproduce your load on a similar, but dedicated to test server
- Want to simulate your production workload?..
  - Then just simulate it! (many SW available, not always OSS/free)
  - Hard to simulate? - adapt some generic tests
- Want to know capacity limits of a given platform?
  - Still try to focus on the test which are most significant for you!
- Want just to validate config settings impacts?
  - Focus on tests which are potentially depending on these settings
  - Or any, if the goal is to prove there are not depending ;-)
- Well, just **keep thinking** about what you're doing ;-)

# “Generic” Test Workloads @MySQL

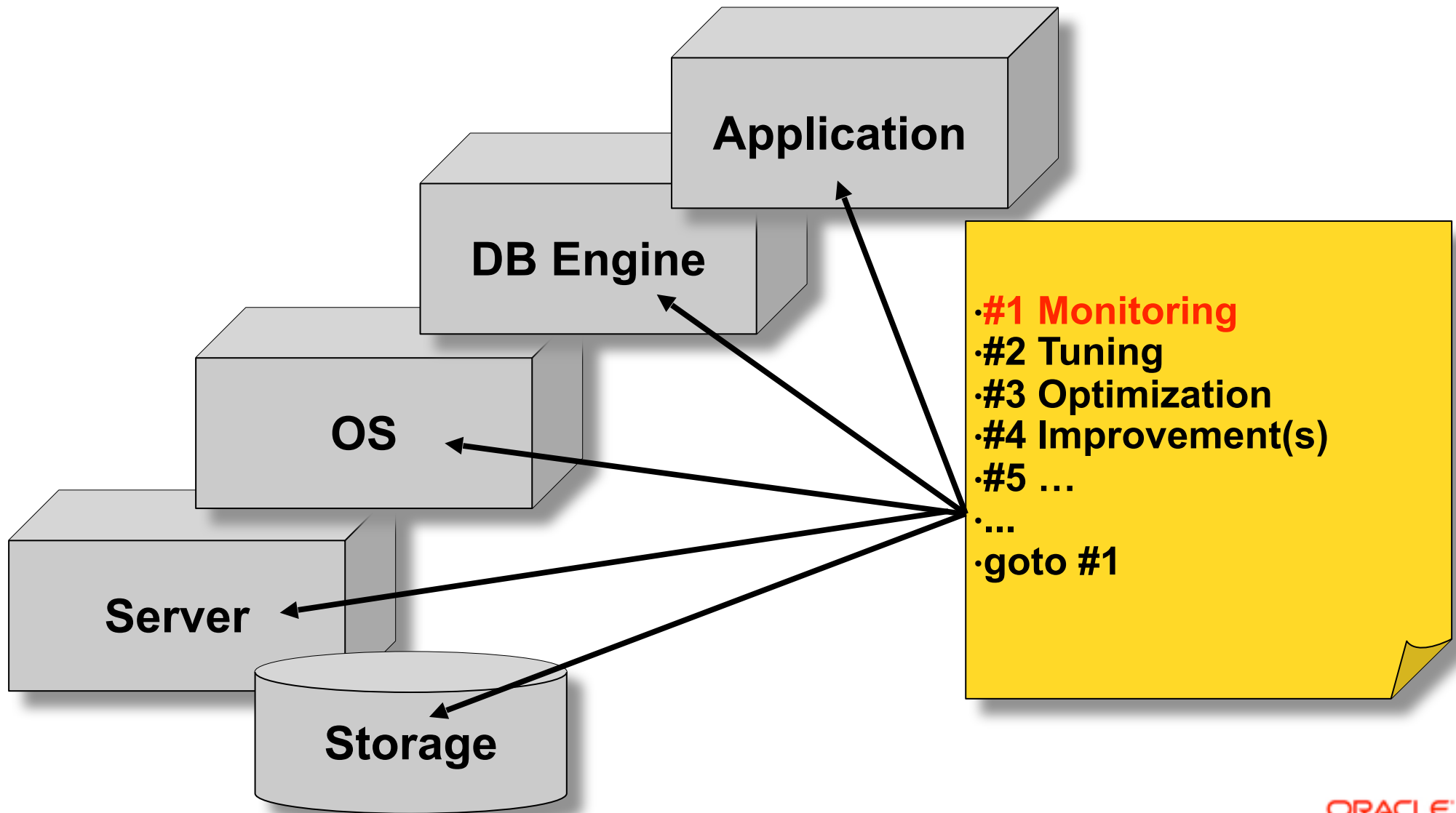
- Sysbench
  - OLTP, RO/RW, 1-table, since v0.5 N-table(s), lots load options, deadlocks
- DBT2 / TPCC-like
  - OLTP, RW, very complex, growing db, no options, deadlocks
  - In fact using mostly only 2 tables! (thanks Performance Schema ;-))
- dbSTRESS
  - OLTP, RO/RW, several tables, one most hot, configurable, no deadlocks
- LinkBench (Facebook)
  - OLTP, RW, very intensive, IO-hungry..
- DBT3
  - DWH, RO, complex heavy query, loved by Optimizer Team ;-)

# Test Workload / Analyze

- Before to jump to 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..” ;-))



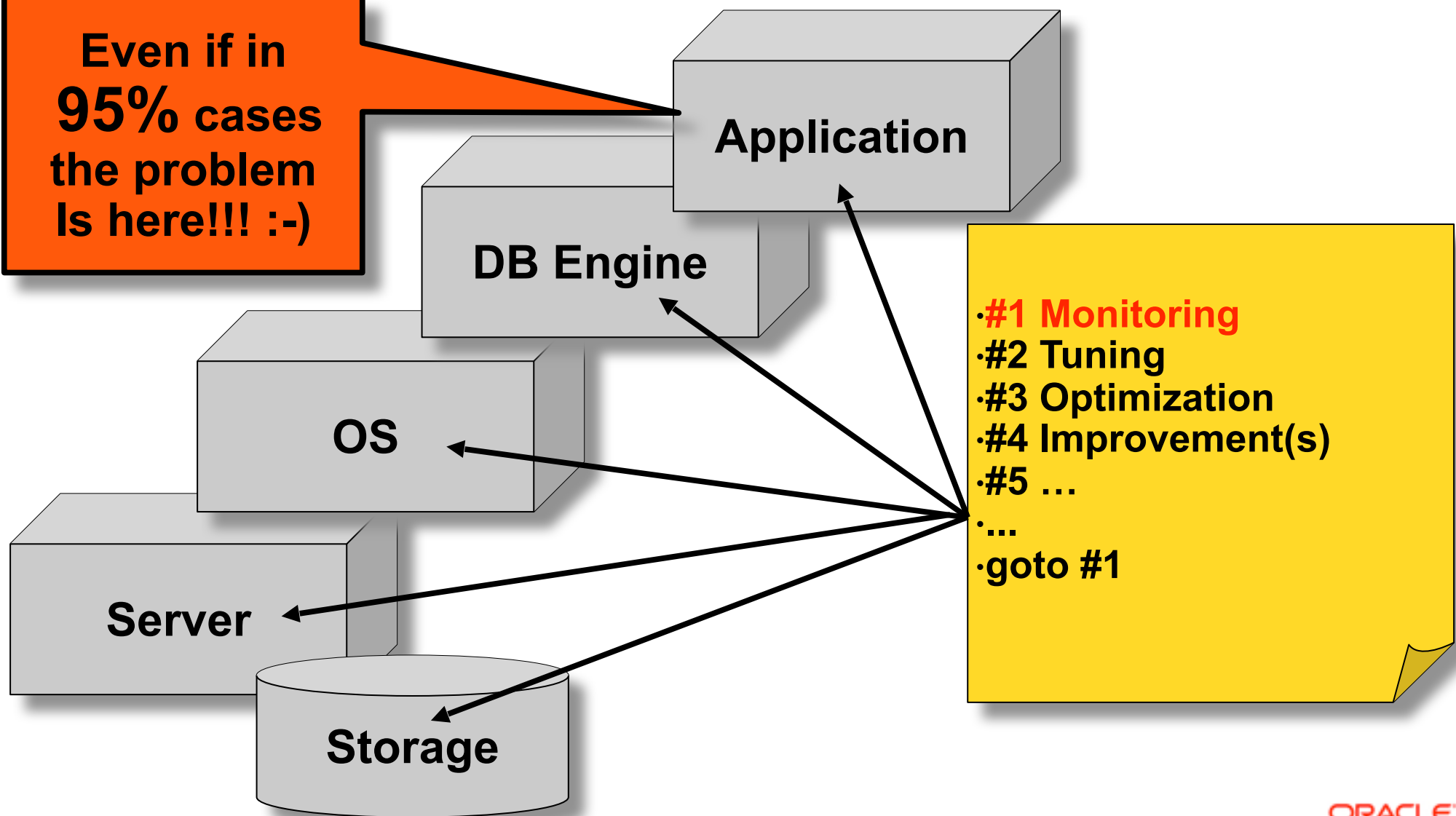
# The Infinite Loop of Database Tuning...





# The Infinite Loop of Database Tuning...

Even if in  
**95%** cases  
the problem  
is here!!! :-)



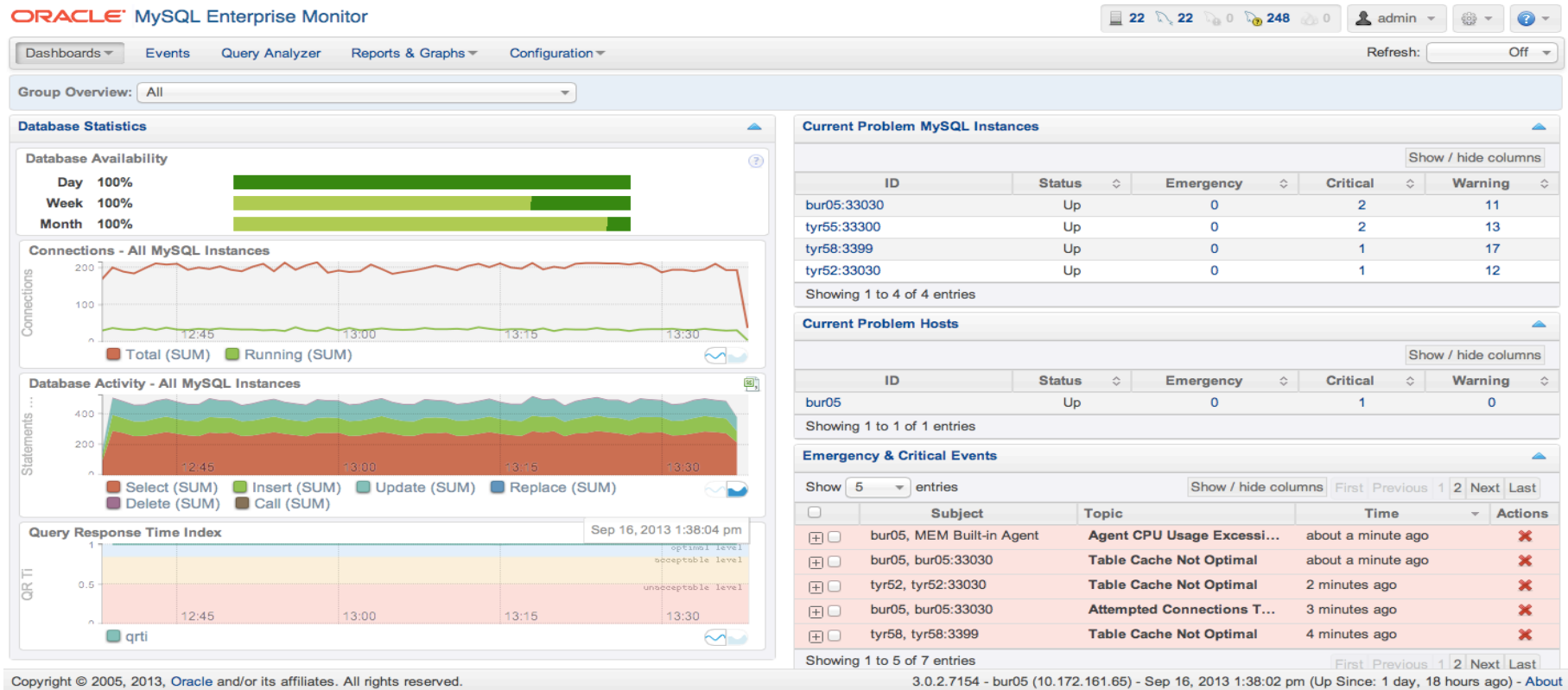


# **Monitoring is THE MUST !**

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

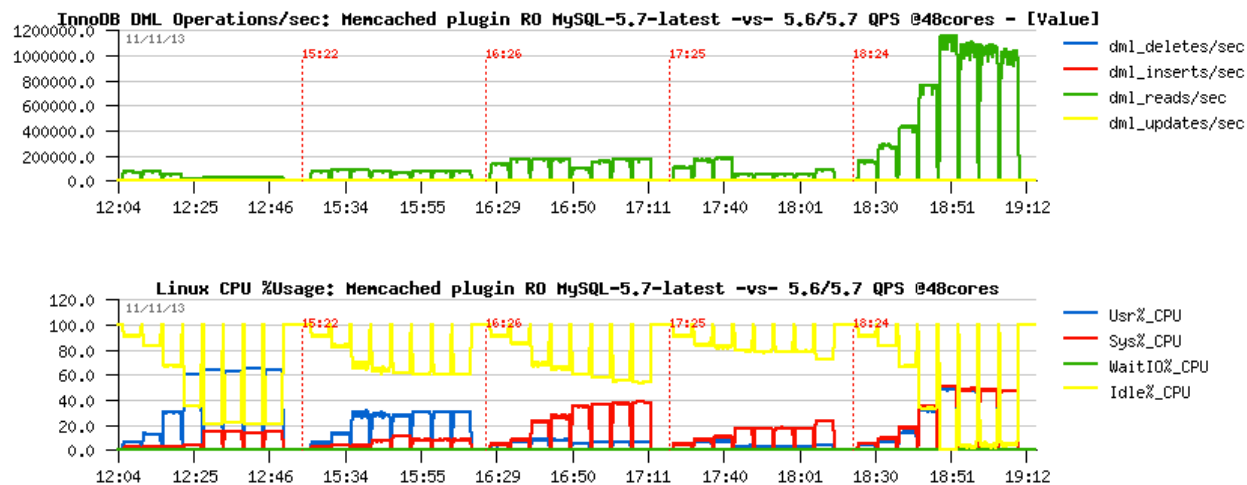
# MySQL Enterprise Monitor

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



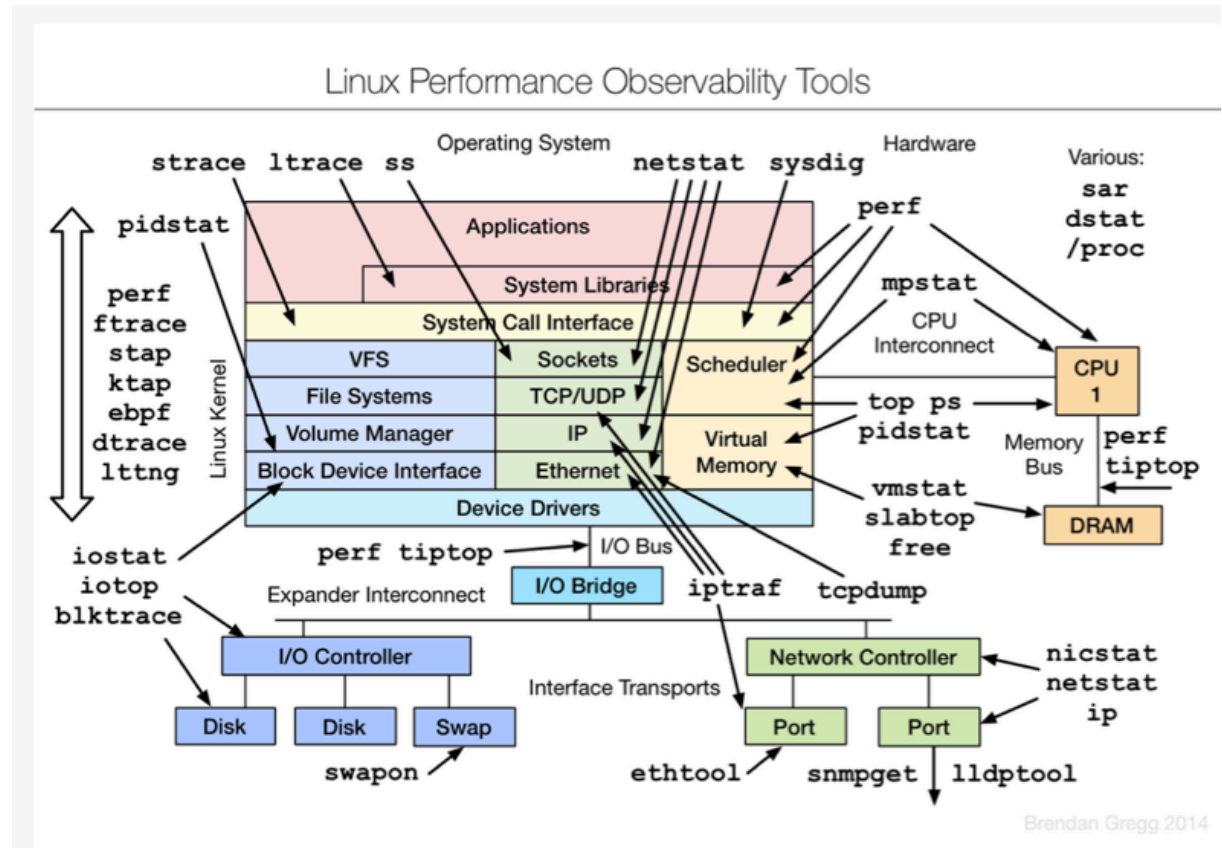
# Other Monitoring Tools

- Cacti, Zabbix, Nagios, Etc.....
- dim\_STAT
  - well, I'm using this one, sorry ;-)
  - all graphs within presentation were made with it
  - details are in the end of presentation..



# System Monitoring (Linux)

- Keep an eye on :
  - CPU Usage%
  - Run queue
  - RAM / swap
  - Top processes
  - I/O op/sec / MB/sec
  - Network traffic
  - etc..

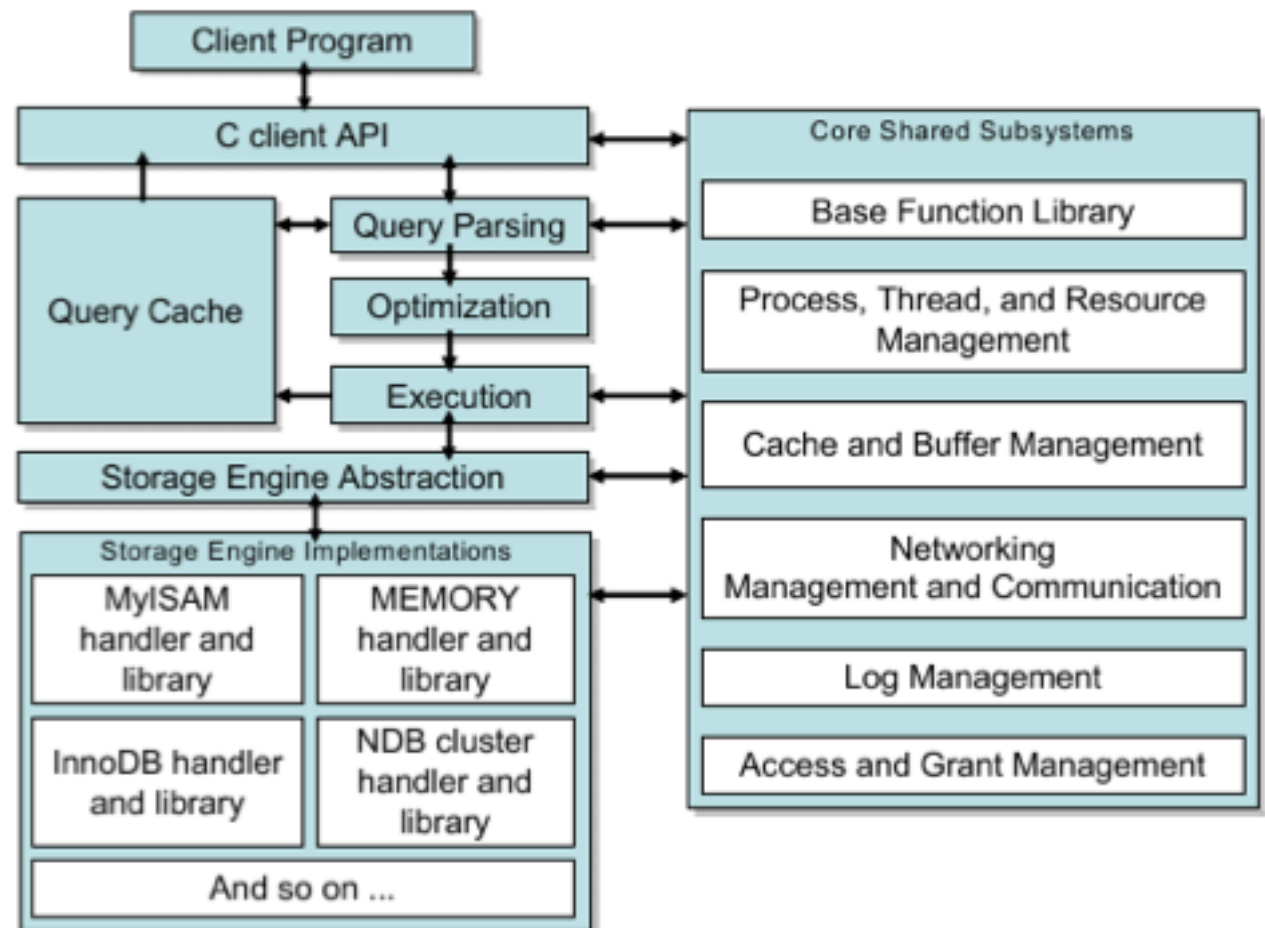


Credits : Brendan GREGG (visit his site!!)

# MySQL Internals Overview

- Multi-Threaded

- fast context switch
- all threads see all data
- so, data lock is needed
- design is very important
- MT malloc() !!!





# What are mutexes?...

# What are mutexes?...



# What are mutexes?...



# What are mutexes?...



?



# What are mutexes?...



# What are mutexes?...





# What are mutexes?...



# What are mutexes?...



Spin Wait Delay = zzz...;-)  
(PAUSE execution)

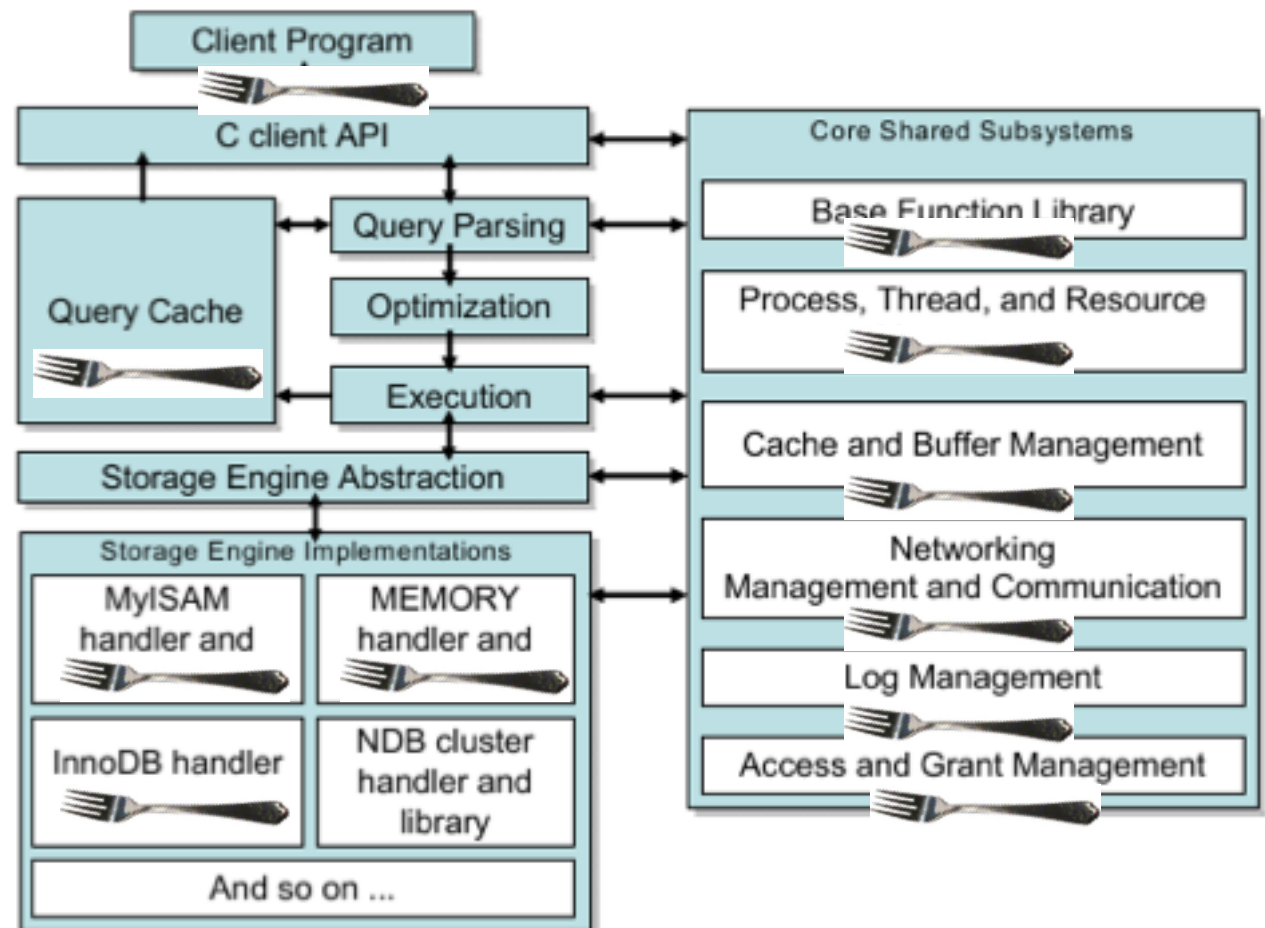
# What are mutexes?...



# MySQL Internals RE-Overview ;-)

- Multi-Threaded

- fast context switch
- all threads see all data
- so, data lock is needed
- design is very important
- MT malloc() !!!



# InnoDB Internals Overview

- Also Multi-Threaded ;-)
  - user threads
  - “background” threads :
    - Master thread
    - Cleaner thread(s)
    - Purge thread(s)
    - IO threads
  - mutexes and RW-locks
  - most famous in the past :
    - MySQL : LOCK\_open
    - InnoDB : kernel\_mutex

# Performance Schema: Gold Mine of Info!

- Just a point about how to analyze mutex lock **contentions**

```
mysql> select EVENT_NAME, max(SUM_TIMER_WAIT)/1000000000000 as WaitTM  
        from events_waits_summary_global_by_event_name group by 1 order by 2 desc limit 5;
```

EVENT_NAME	WaitTM
wait/io/file/innodb/innodb_data_file	24404.2548
idle	1830.1419
wait/synch/rwlock/innodb/hash_table_locks	25.2959
wait/synch/mutex/innodb/file_system_mutex	24.9102
wait/io/file/innodb/innodb_log_file	11.2126

5 rows in set (0.03 sec)



```
mysql> select EVENT_NAME, max(SUM_TIMER_WAIT)/1000000000000 as WaitTM  
        from events_waits_summary_by_instance group by 1 order by 2 desc limit 5;
```

EVENT_NAME	WaitTM
wait/io/file/innodb/innodb_data_file	791.3204
wait/synch/mutex/innodb/file_system_mutex	25.8183
wait/synch/rwlock/innodb/btr_search_latch	5.2865
wait/io/file/innodb/innodb_log_file	4.6977
wait/synch/rwlock/sql/LOCK_grant	4.4940

5 rows in set (0.06 sec)



# Visual explanation : MyISAM -vs- InnoDB ;-)

- MyISAM -vs- InnoDB
  - (table locking -vs- row locking)

MyISAM



InnoDB



# Basic Tuning

- Understanding HW platform **limits**
  - helps you to deploy your MySQL Server in the most optimal way..
- Understanding MySQL Server **internals**
  - helps you to configure your database settings in the most optimal way..
  - some limitations are here “by design”,
- Understanding of your **Workload**
  - helps you to tune the whole solution in the most optimal way ;-)
  - 20% of known issues covering 80% of most common problems..
  - So, adapt some best practices from the beginning..
- There is **NO** “Silver Bullet” !!!
  - Think about the #1 MySQL Performance Best Practice ;-))



# Analyzing Workloads...

- Read-Only (RO) :

- Nothing more simple when comparing DB Engines, HW configs, etc..
- RO In-Memory : data set fit in memory / BP / cache
- RO IO-bound : data set out-passing a given memory / BP / cache

- Read+Write (RW) :

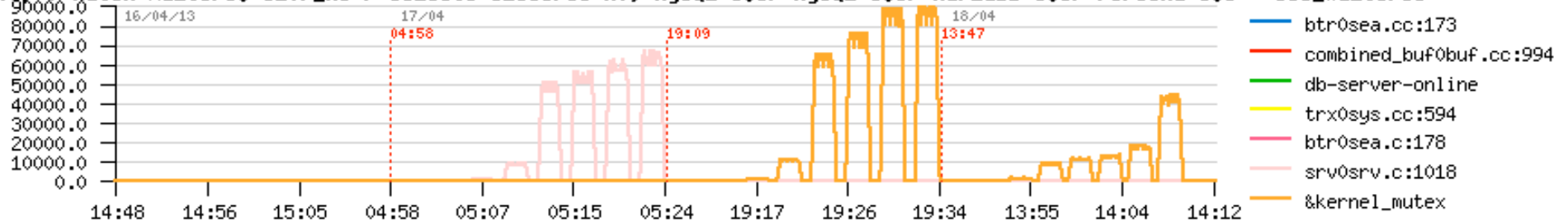
- I/O is **ALWAYS** present ! - storage performance matters a lot !
- may be considered as always IO-bound ;-)
- RW In-Memory : same as RO, data set fit in memory, but :
  - small data set => small writes
  - big dataset => big writes ;-)
- RW IO-bound : data set out-passing a memory
  - means there will be (a lot of?) reads !
  - don't forget that I/O random reads = I/O killer !

# Workloads : Read-Only In-Memory

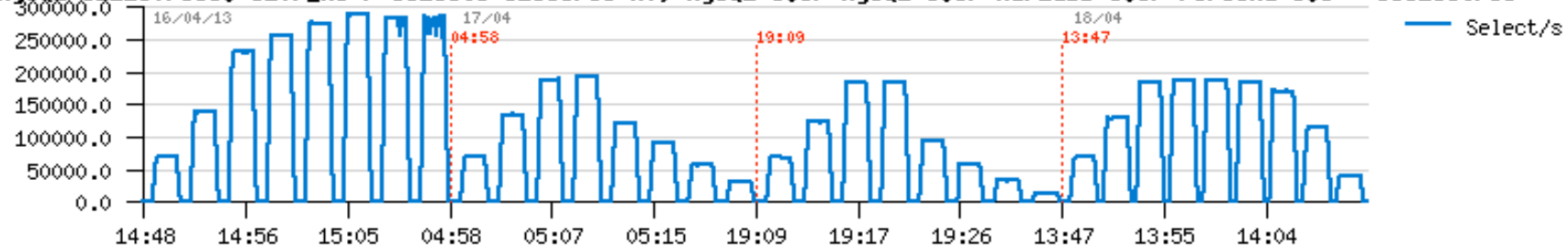
- Generally CPU / RAM bound + internal contentions ;-)
- 5.5 :
  - kernel\_mutex (main killer in InnoDB)
  - LOCK\_open
  - + many other remain hidden ;-)
- 5.6 :
  - kernel\_mutex => trx\_sys + lock\_sys
  - => hot trx\_sys : RO transactions, but can be impacted by RW
  - => MDL : hash lock instances
  - => LOCK\_open : table cache instances
  - => G5! (false cache sharing) ==> where Databases SW is hitting HPC ;-)

# InnoDB: Read-Only Transactions in 5.6 (Apr.2013)

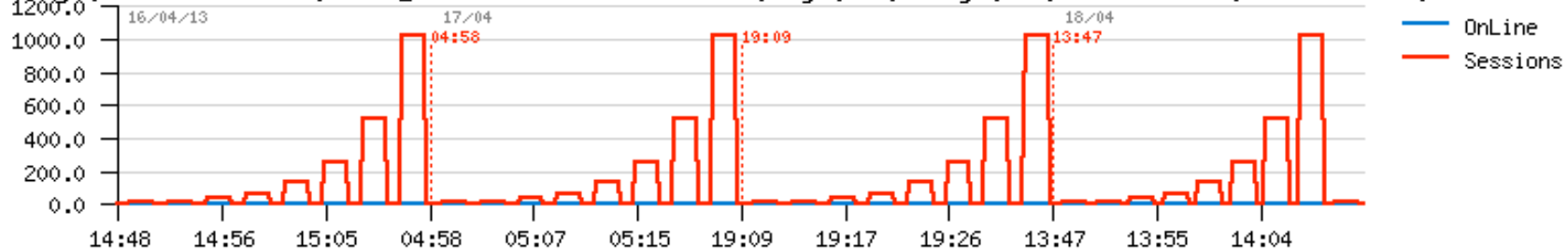
InnoDB Top-7 Mutex Waits/s: OLTP\_RO P-selects @16cores-HT, MySQL-5.6/ MySQL-5.5/ MariaDB-5.5/ Percona-5.5 - [os\_waits/s]



MySQL SELECT/sec: OLTP\_RO P-selects @16cores-HT, MySQL-5.6/ MySQL-5.5/ MariaDB-5.5/ Percona-5.5 - [Select/s]

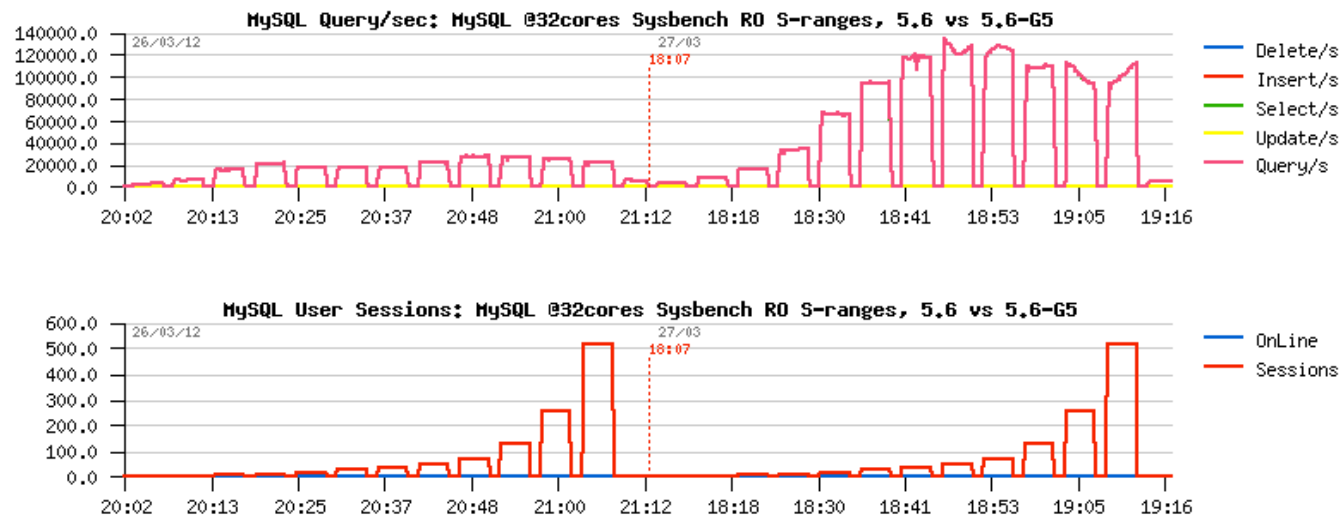


MySQL User Sessions: OLTP\_RO P-selects @16cores-HT, MySQL-5.6/ MySQL-5.5/ MariaDB-5.5/ Percona-5.5



# InnoDB : false sharing of cache-line fixed!

- RO or RW Workloads
  - “G5” patch! :-)
  - Over x2(!) times better on Sysbench OLTP\_RO,
  - x6(!) times better on SIMPLE-Ranges!
  - NOTE: the fix is not applicable on 5.5..

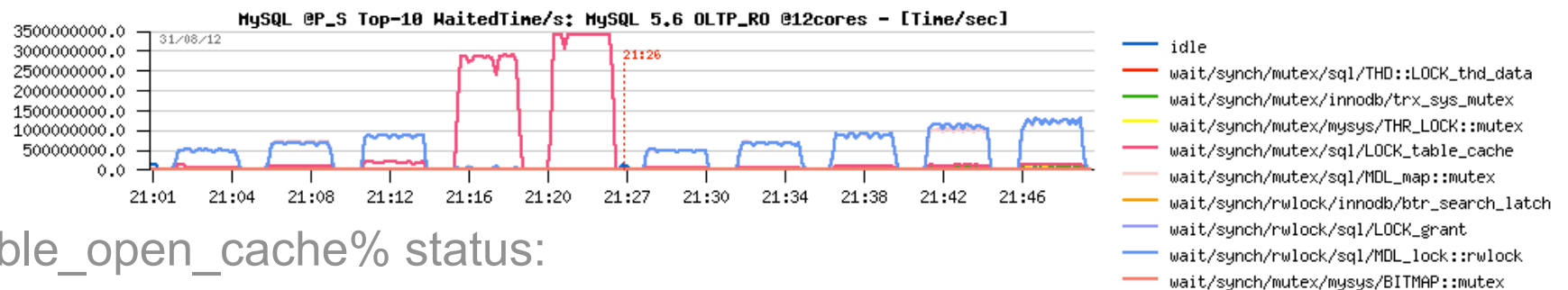


# MySQL Internals: “killer” LOCK\_open mutex

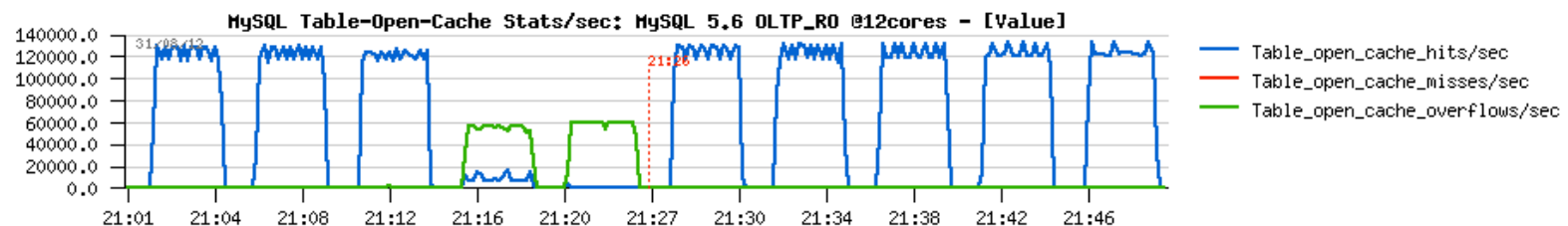
- MySQL 5.5 and before:
  - Keep “table\_open\_cache” setting big enough!
  - Monitor global status for '%opened%'
  - Once this contention become the most hot – well, time to upgrade to 5.6 ;-))
- Since MySQL 5.6:
  - Fixed: several table open cache instances
  - But it doesn't mean you can use a small “table\_open\_cache” either ;-)
  - Monitor PFS Waits!
  - Monitor “table\_open\_cache%” status variables!
  - Keep “table\_open\_cache\_instances” at least bigger than 1

# MySQL 5.6 Internals : low table\_open\_cache (2)

- MySQL 5.6 :
  - Not big enough “table\_open\_cache” setting
  - PFS Waits monitoring: LOCK\_table\_cache become the most hot:



- Table\_open\_cache% status:



# RO related starter configuration settings

- my.conf :

```
join_buffer_size=32K  
sort_buffer_size=32K
```

```
table_open_cache = 8000  
table_open_cache_instances = 16  
query_cache_type = 0
```

```
innodb_buffer_pool_size= 64000M (2/3 RAM ?)  
innodb_buffer_pool_instances=32  
innodb_thread_concurrency = 0 / 32 / 64  
innodb_spin_wait_delay= 6 / 48 / 96
```

```
innodb_stats_persistent = 1  
innodb_adaptive_hash_index= 0 / 1  
innodb_monitor_enable = '%'
```



# Workloads : Read-Only In-Memory @MySQL 5.7

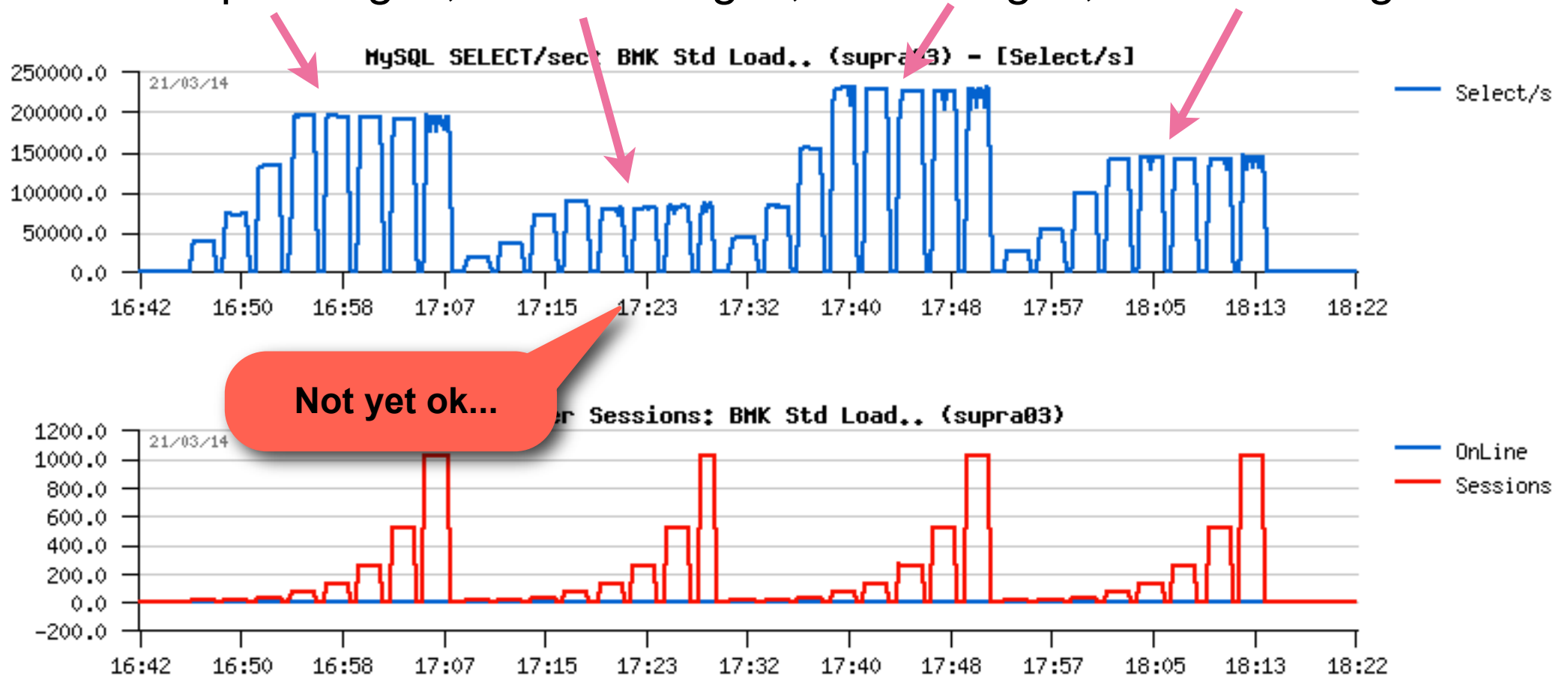
- 5.7 :

- trx\_sys : redesigned TRX list! (yet better than RO transactions)
  - made MDL very hot !! regression on single-hot-table workloads..
- MDL : lock free since DMR4 !!
  - made THR\_lock very hot !! fixed since DMR5 !!!
- Connect : remastered => 70K connect/disconnect/sec
- QPS :
  - SQL : **over 500K (!) QPS** (SQL) on point-selects
  - Memcached plugin : **rocks over 1M (!) QPS**
- InnoDB spin lock delay : still remains !
- Scalability: very good, but RO Dranges remains..
- AHI : remains



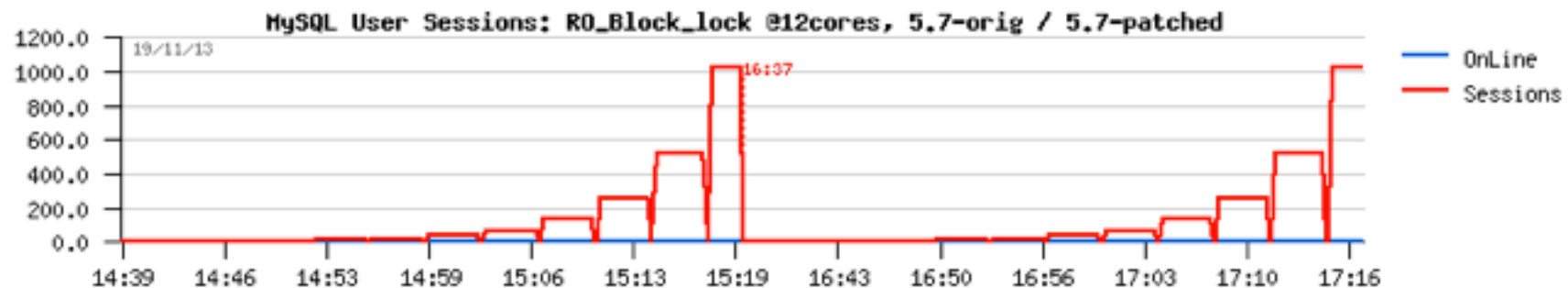
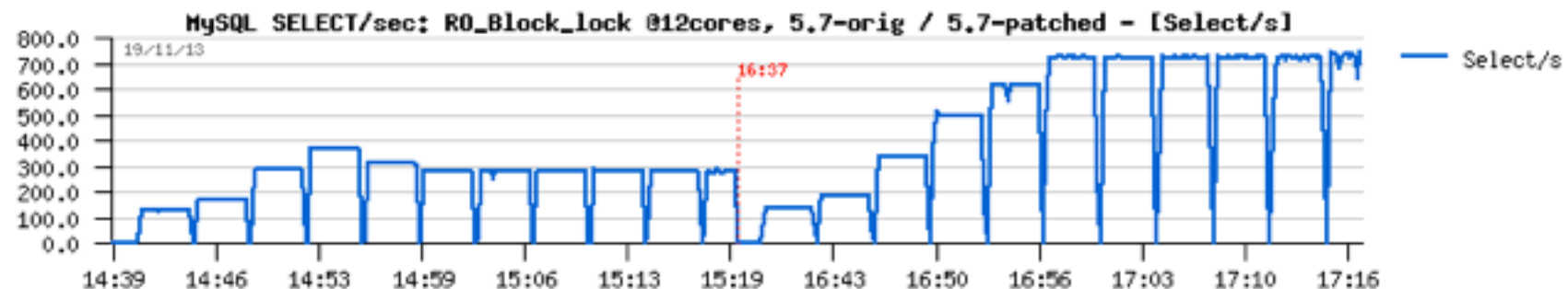
# Sysbench OLTP\_RO Workloads @MySQL 5.7

- Simple ranges, Distinct ranges, SUM ranges, Ordered ranges



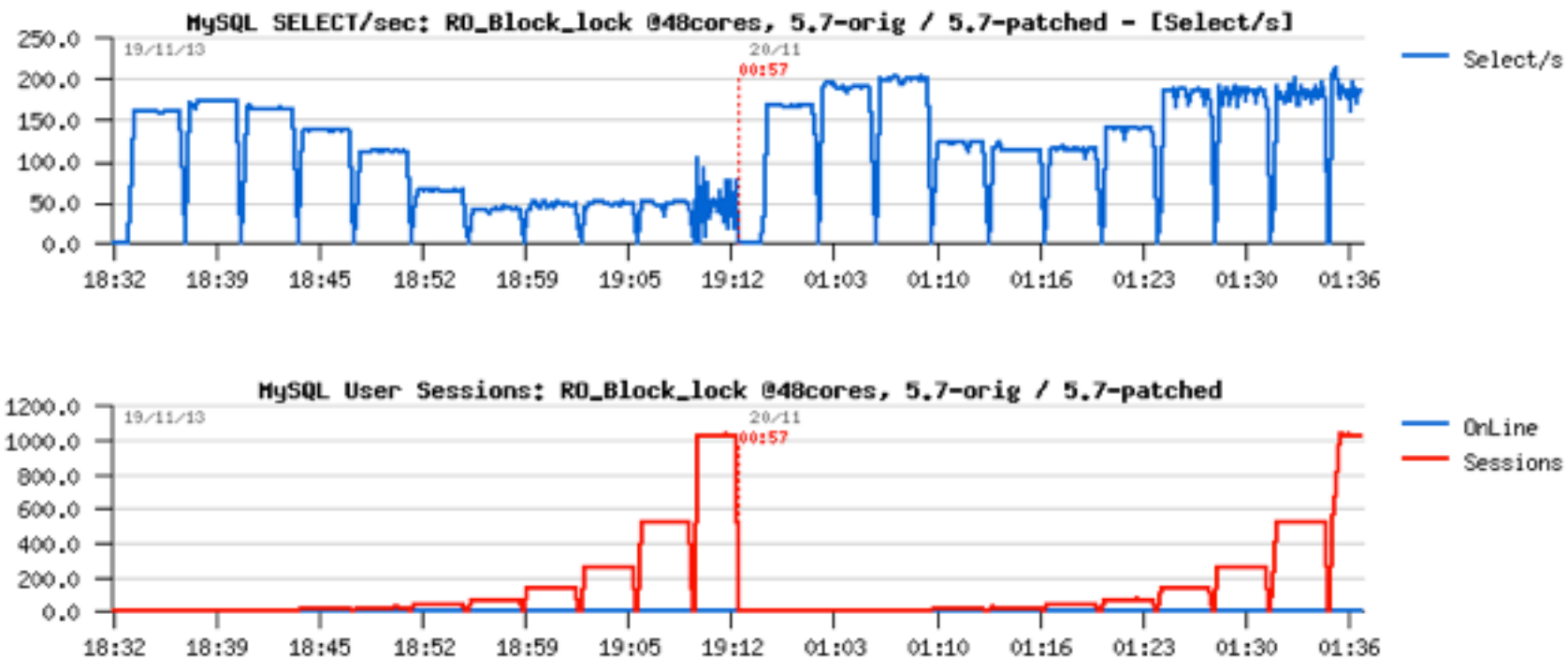
# InnoDB block lock contentions...

- Being here from a long long time (by design)..
- Improved in 2013, but not yet fully fixed..
- Can be seen as :



## InnoDB block lock contentions... (cont.)

- Being here from a long long time (by design)..
- Improved in 2013, but not yet fully fixed..
- But also as :



## InnoDB block lock contentions... (cont.)

- Being here from a long long time (by design)..
  - Improved in 2013, but not yet fully fixed..
  - A true fix requires a full redesign of block related internals..
  - in TODO, but not for tomorrow ;-)
- Workarounds :
  - QueryCache ;-) well, any kind of cache ;-)
  - BTW, because of a widely used caching solutions around of MySQL servers in production made this issue “invisible” for so long time.. (that’s why)..
    - [MySQL Query Cache](#)

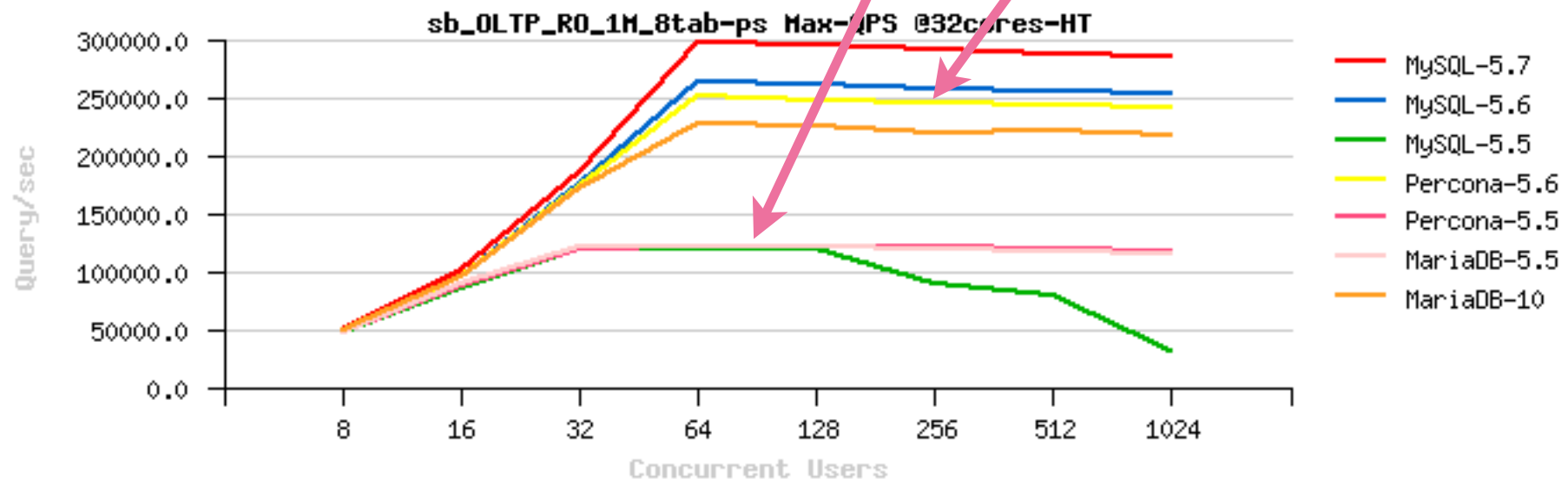
# When hitting “by design” issues..

- Could we consider it as a bug?..
  - not really, as it’s “by design” ;-)
  - regression? - nor either, as it was always like this ;-)
  - So? what to do? - Continue to complain and then you’ll see it fixed ;-)



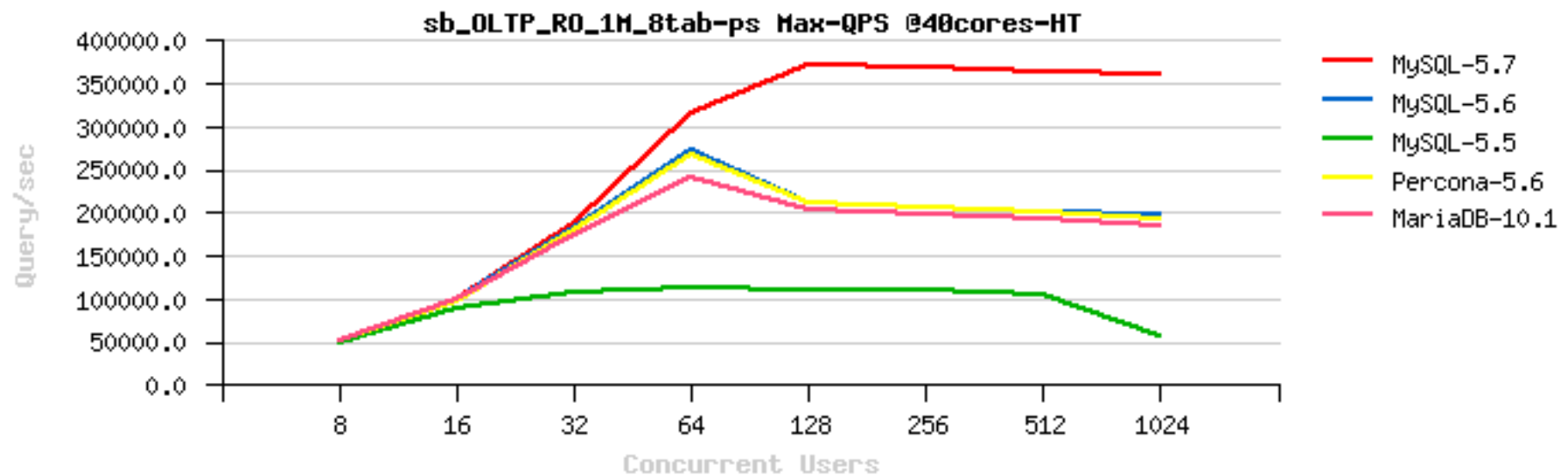
# RO In-Memory @MySQL 5.7

- Sysbench OLTP\_RO 8-tables, 32cores-HT :
  - don't forget that Percona 5.6 & MariaDB-10 are **here** thanks to MySQL 5.6
  - otherwise the level of all “5.5” series is **here**



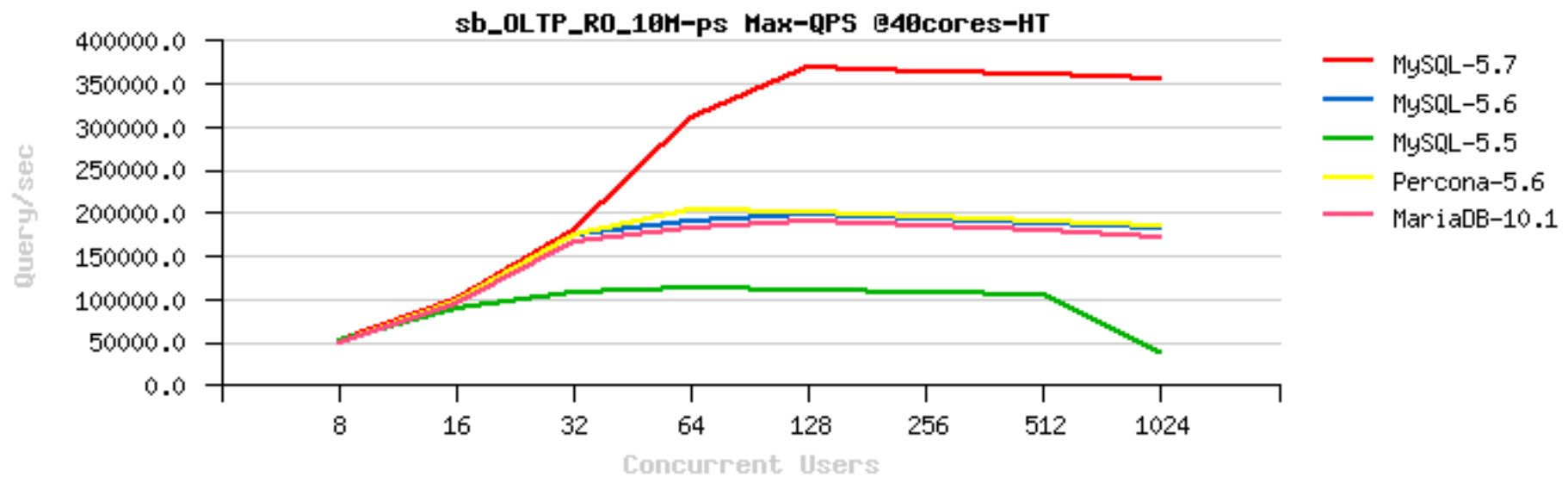
# RO In-Memory @MySQL 5.7

- Sysbench OLTP\_RO 8-tables, 40cores-HT :



# Sysbench OLTP\_RO Single-table

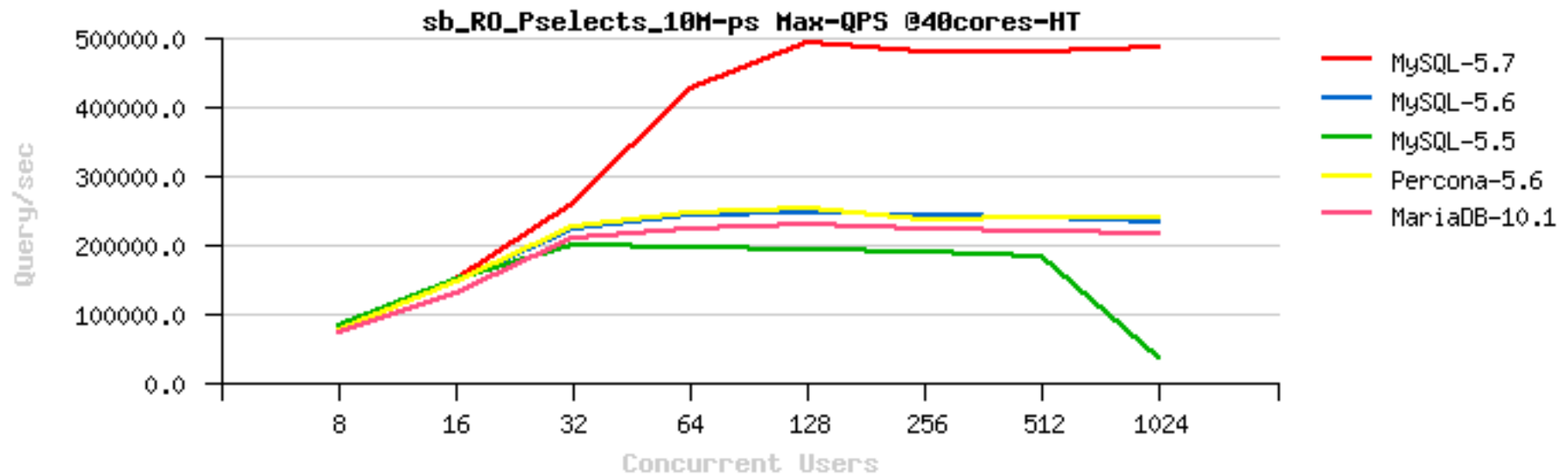
- Max QPS @40cores-HT :





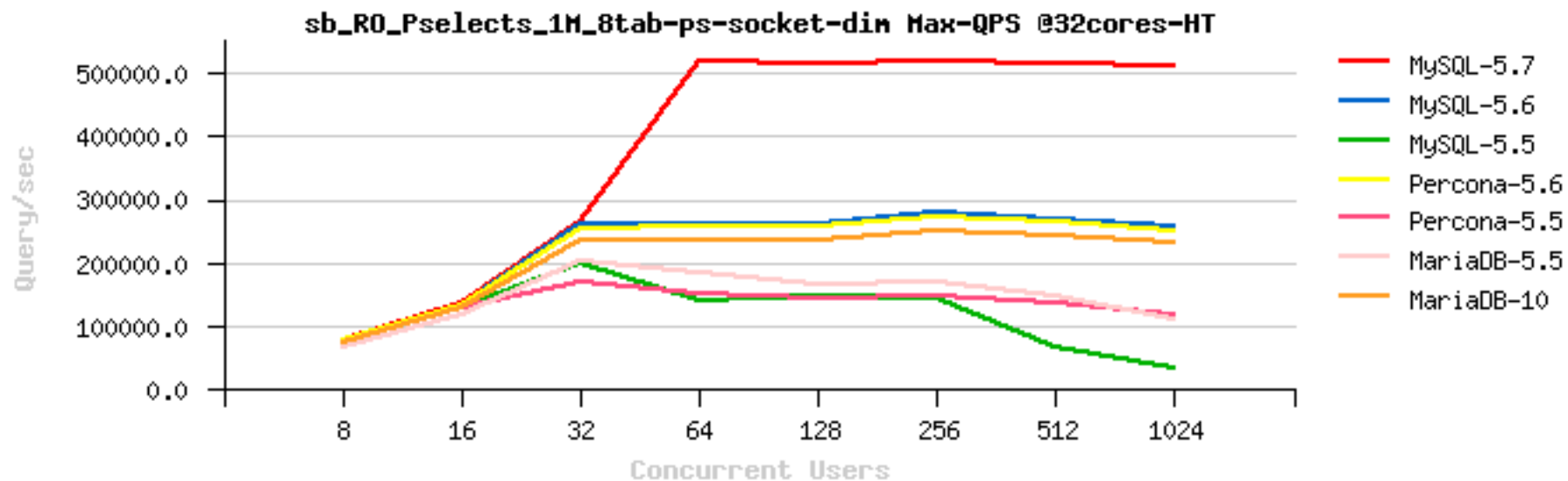
# Sysbench OLTP\_RO Point-Selects single-table

- Max QPS @40cores-HT :



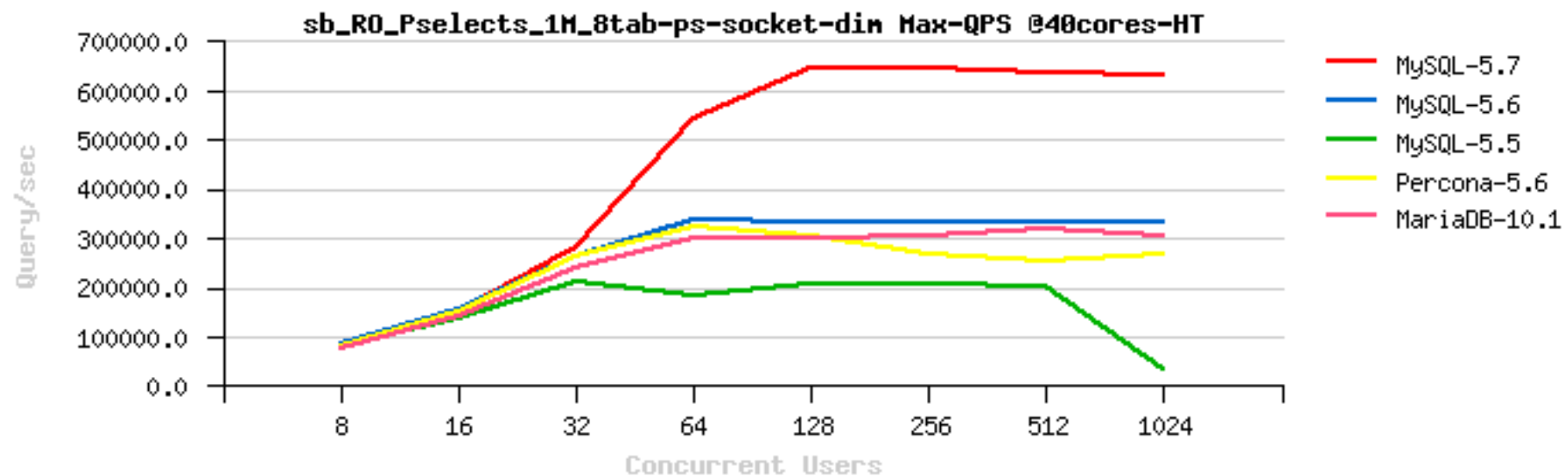
# RO In-Memory @MySQL 5.7

- **500K QPS** Sysbench Point-Selects 8-tab, 32cores-HT :



# RO In-Memory @MySQL 5.7

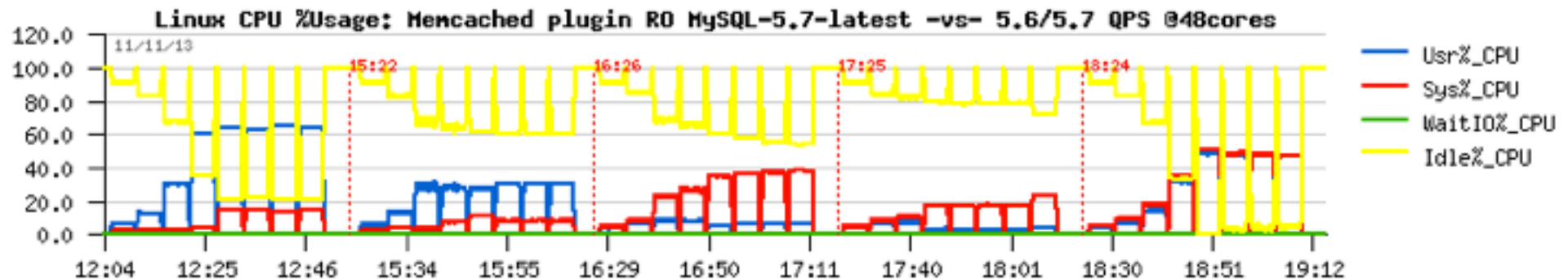
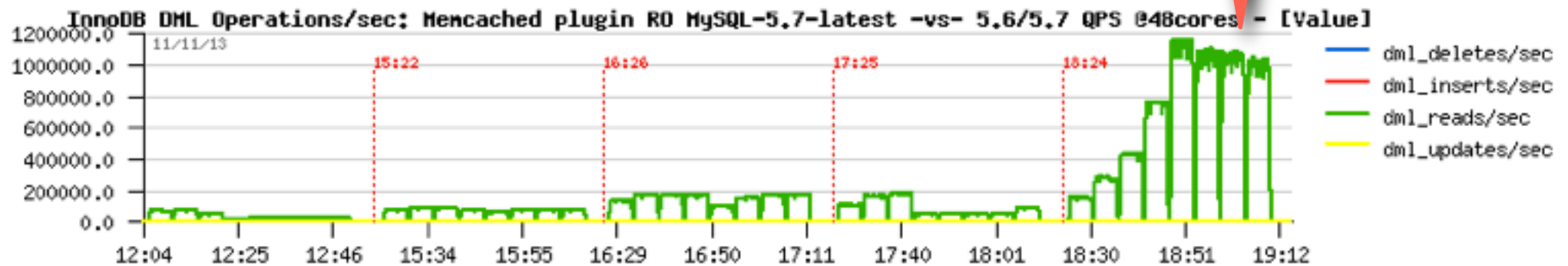
- **645K QPS** Sysbench Point-Selects 8-tab, 40cores-HT :



# InnoDB Memcached @MySQL 5.7

- **Over 1M (!) QPS** on 48cores-HT :

That's it ;-)



## Read-Only : IO-bound

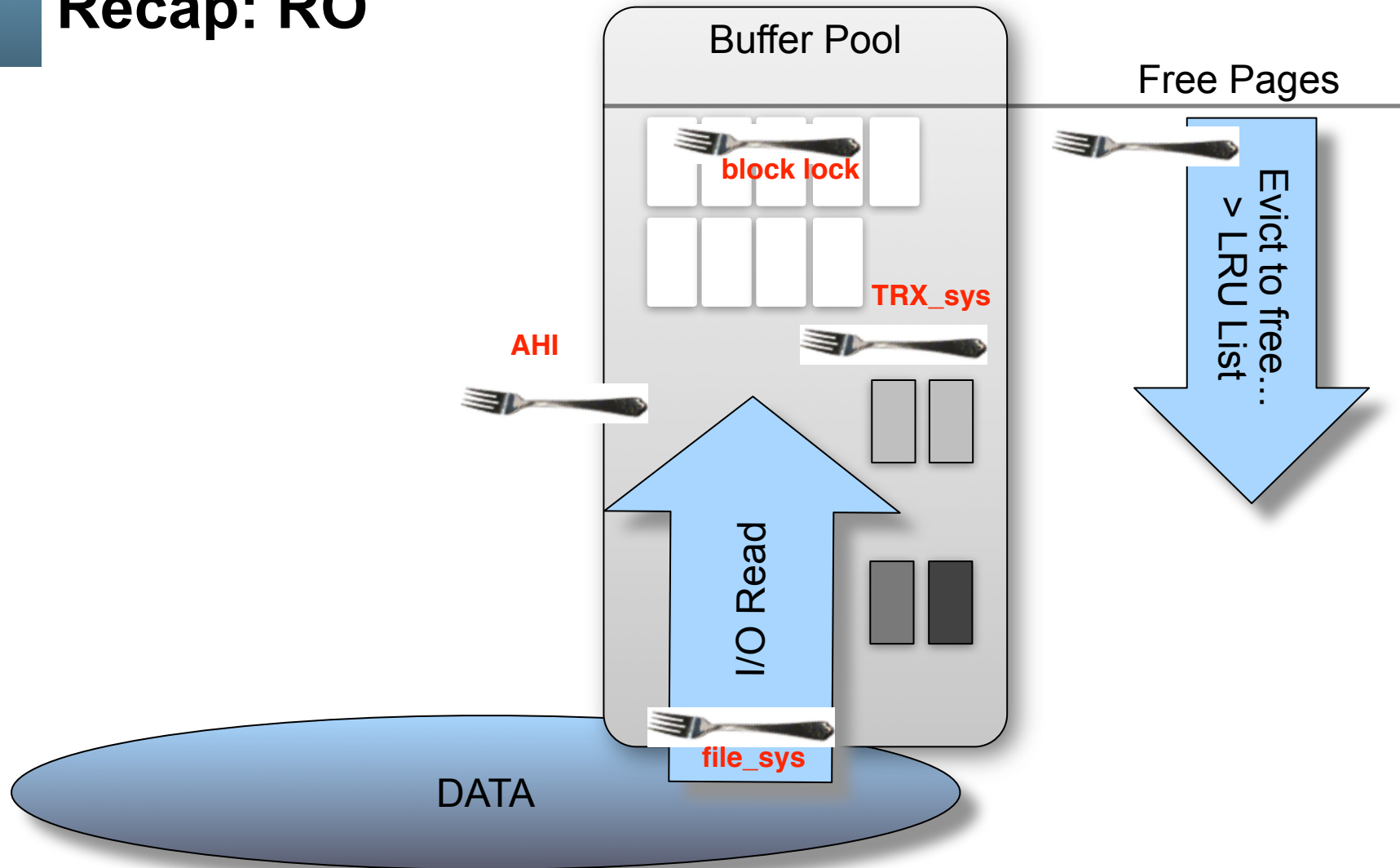
- 5.5 : hmm..
- 5.6 / 5.7 :
  - LRU driven : just page eviction, see METRICS stats
  - HDD : limited by your I/O layer..
  - SSD : limited by your I/O layer..
  - Really Fast Flash (LSI, Fusion-io, etc.) :
    - avg load : follow I/O performance
    - high load: file\_sys mutex contention...
  - also consider : innodb\_old\_blocks\_time & innodb\_old\_blocks\_pct
- 5.7 :
  - excessive page scan is fixed

## Hope it was not boring ?.. ;-)

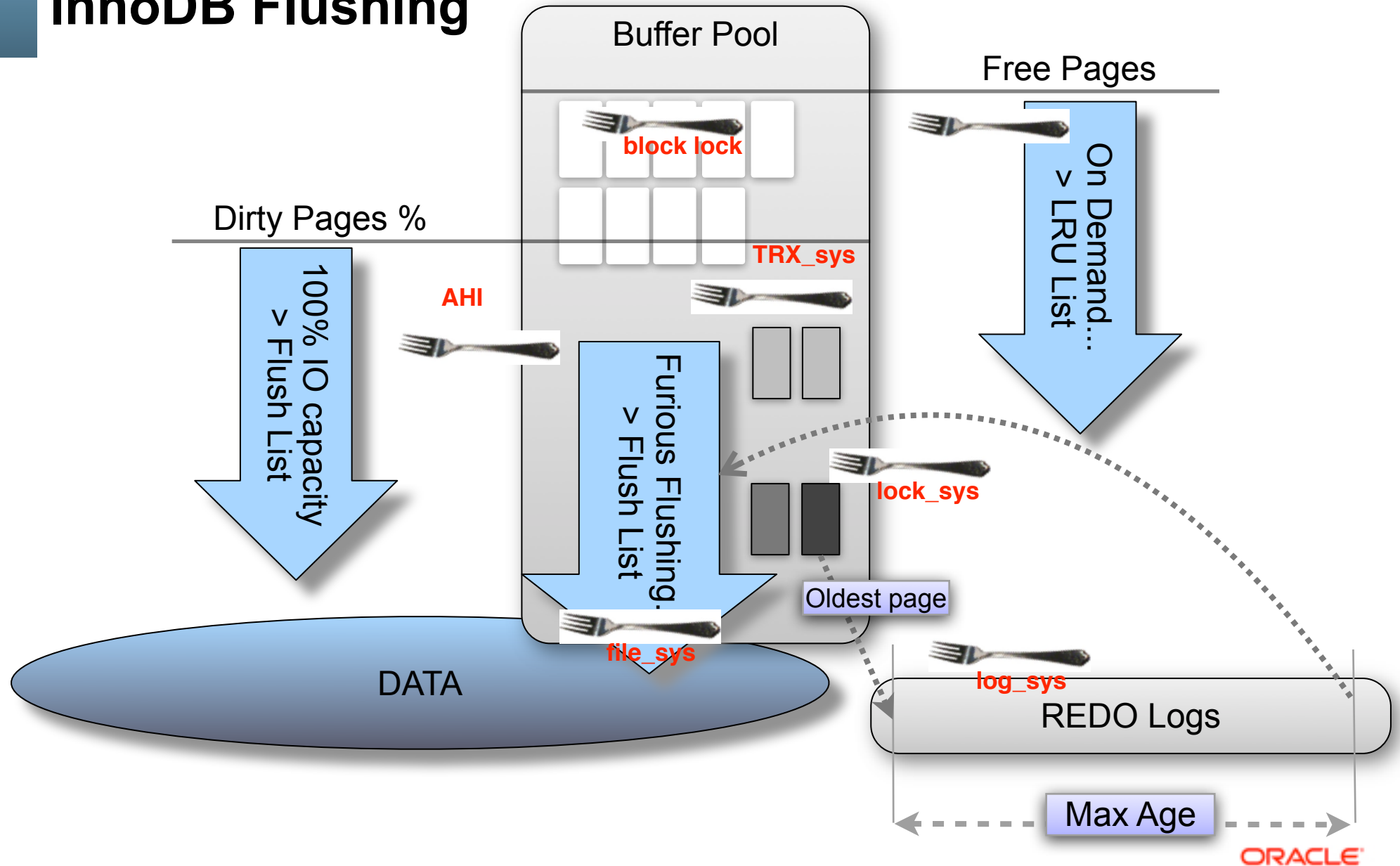
- Because since now I'll need your full attention ;-)



# Recap: RO



# InnoDB Flushing





# Read+Write Workloads

- Main points :
  - Processing itself / Data Safety
  - Internal contentions / Design limitations
  - Flushing / Checkpoint
  - Purge
  - Concurrency tuning

# Read+Write Performance @MySQL / InnoDB

- Transactional processing

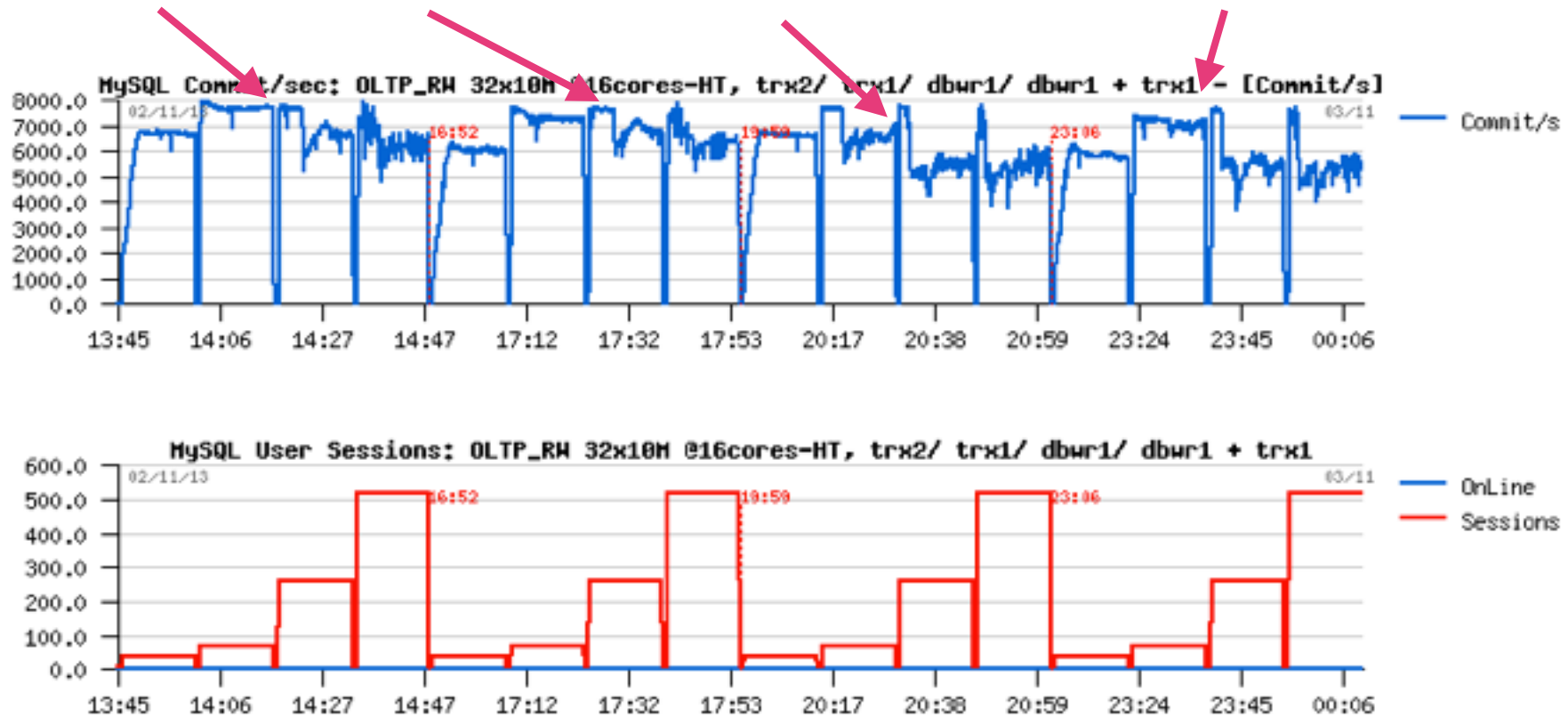
- your CPU-bound transactional processing defines your Max possible TPS
- with a bigger volume / more IO / etc. => Max TPS will not increase ;-)

- Data Safety

- binlog : overhead + bottleneck (be sure you have binlog group commit)
- InnoDB checksums : overhead (reasonable since crc32 is used)
- innodb\_flush\_log\_at\_trx\_commit = 1 : overhead + bottleneck
- InnoDB double write buffer : **KILLER** ! overhead + huge bottleneck..
  - need a fix / re-design / etc. in urgency ;-)
  - Fusion-io atomic writes is one of (**true** support in MySQL 5.7)
  - Using EXT4 with data journal is another one
  - but a true re-design is still preferable ;-)

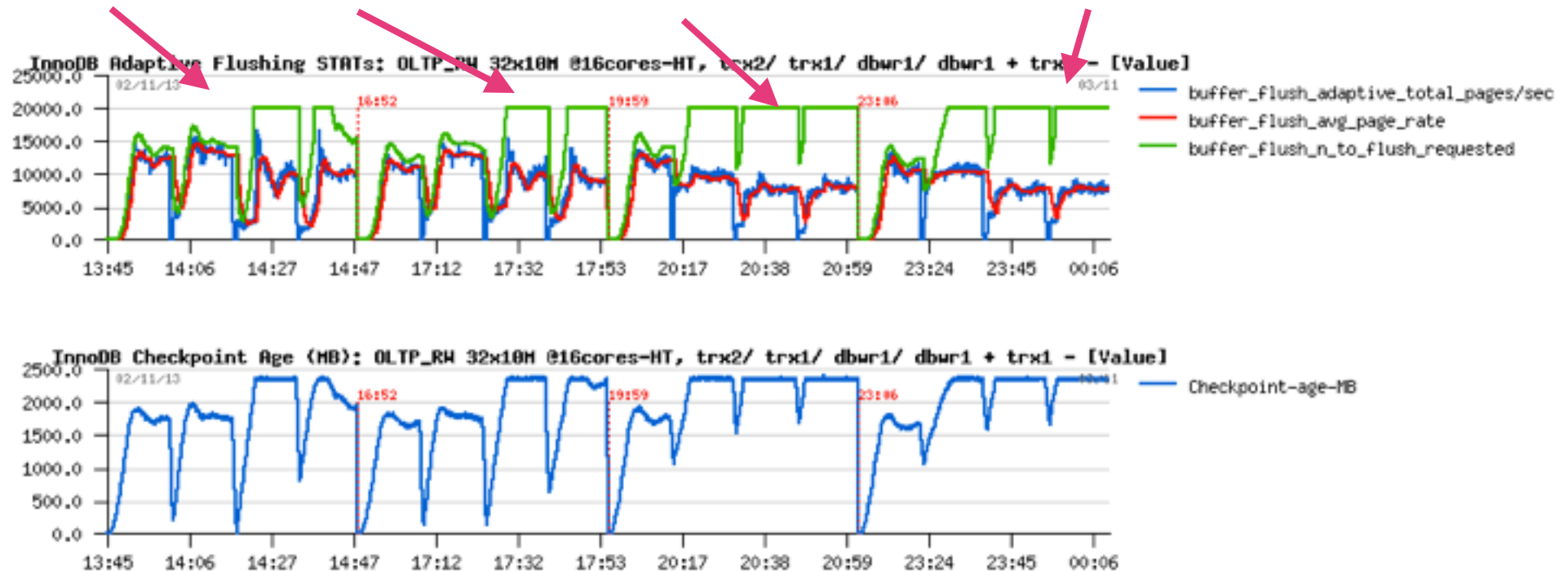
# Impact of “safety” options..

- OLTP\_RW 32x10M-tables @Percona-5.6
  - ( trx=2 )( trx=1 + chksum=1 )( dblwr=1 )( trx=1 + chksum=1 + dblwr=1 )



# Impact of “safety” options..

- OLTP\_RW 32x10M-tables @Percona-5.6
  - ( trx=2 )( trx=1 + chksum=1 )( dblwr=1 )( trx=1 + chksum=1 + dblwr=1 )



# Read+Write Workloads : In-Memory

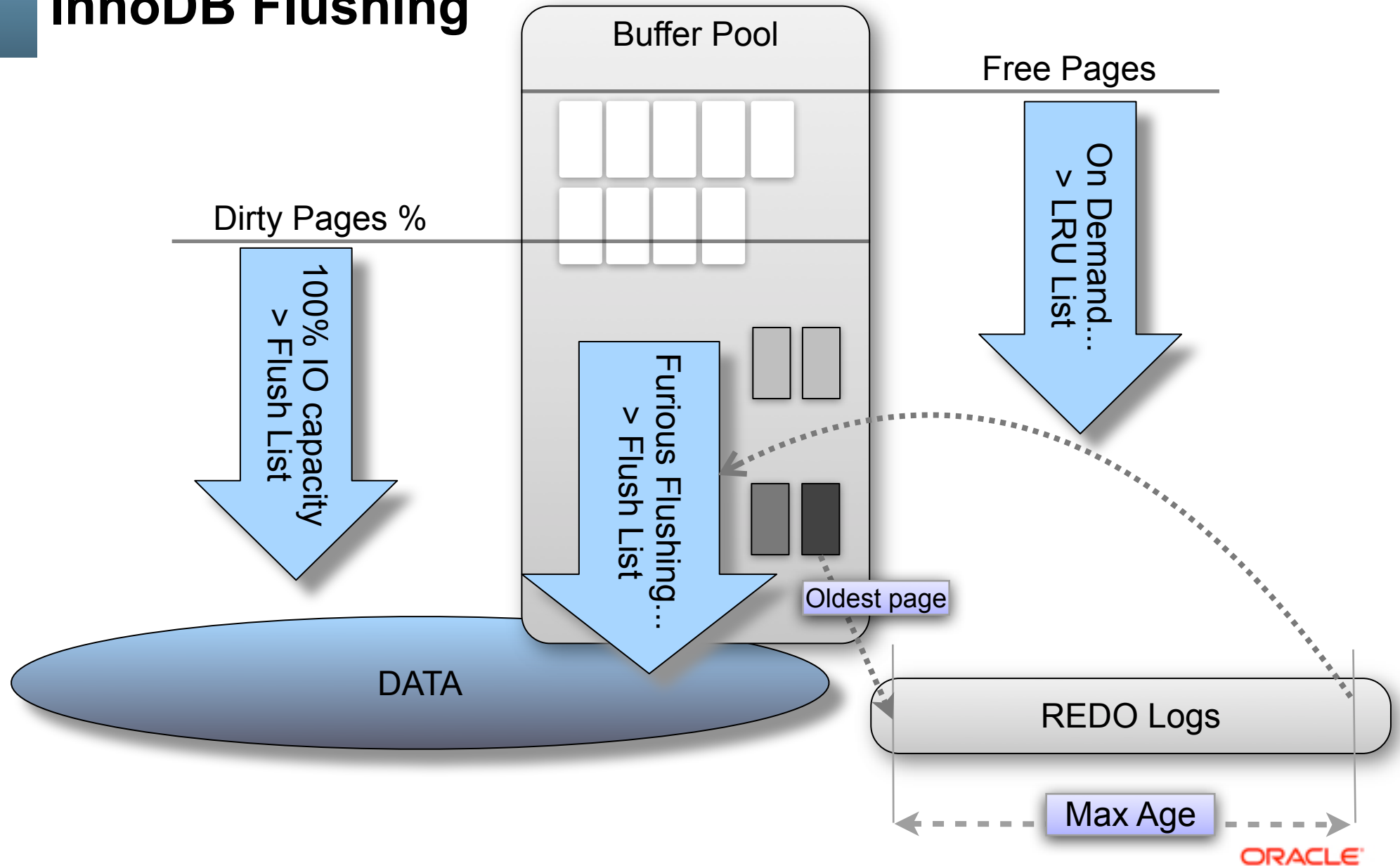
- Internal contentions / Design limitations
  - 5.5 : BP instances, RBS, etc..
  - 5.6 :
    - kernel\_mutex => trx\_sys & lock\_sys
    - all already mentioned on RO + still many remaining ;-)
    - up to 2TB REDO, etc..
  - 5.7 :
    - lock free MDL + THR\_lock
    - index lock : fixed !
    - lock\_sys : lowered
    - trx\_sys : lowered + TRX list related re-design
    - **log\_sys** : remains and killing ;-)
    - **fil\_sys** : killing too, but on a high level storage only ;-)

# Read+Write Workloads : InnoDB Flushing

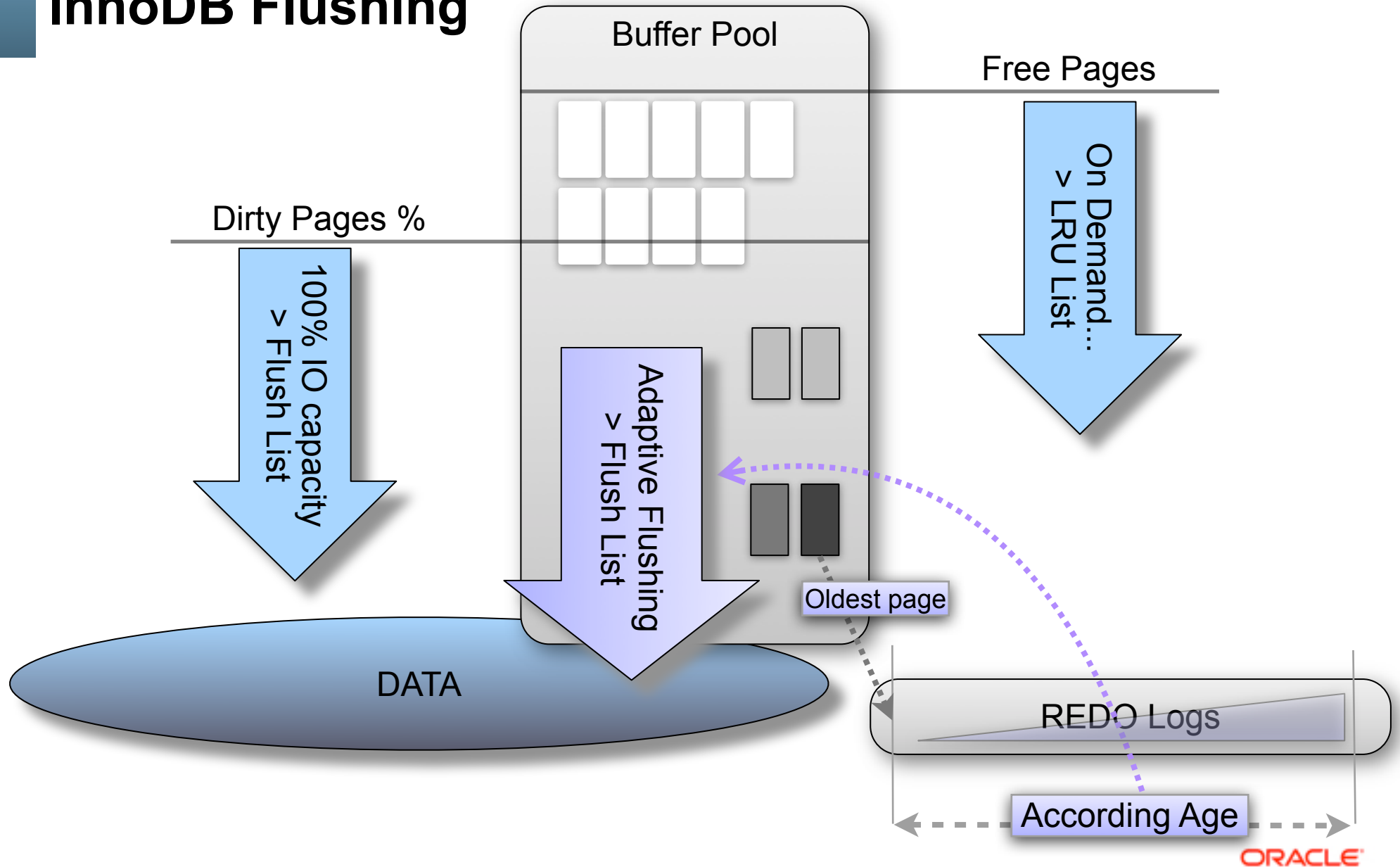
- InnoDB Flushing...

- 5.5 : no comments.. ;-)
- 5.6 :
  - Improved Adaptive Flushing (step 1)
  - Cleaner Thread
- 5.7 :
  - multiple Cleaner Threads
  - improved LRU flushing
  - improved Adaptive Flushing Design (step 2)

# InnoDB Flushing

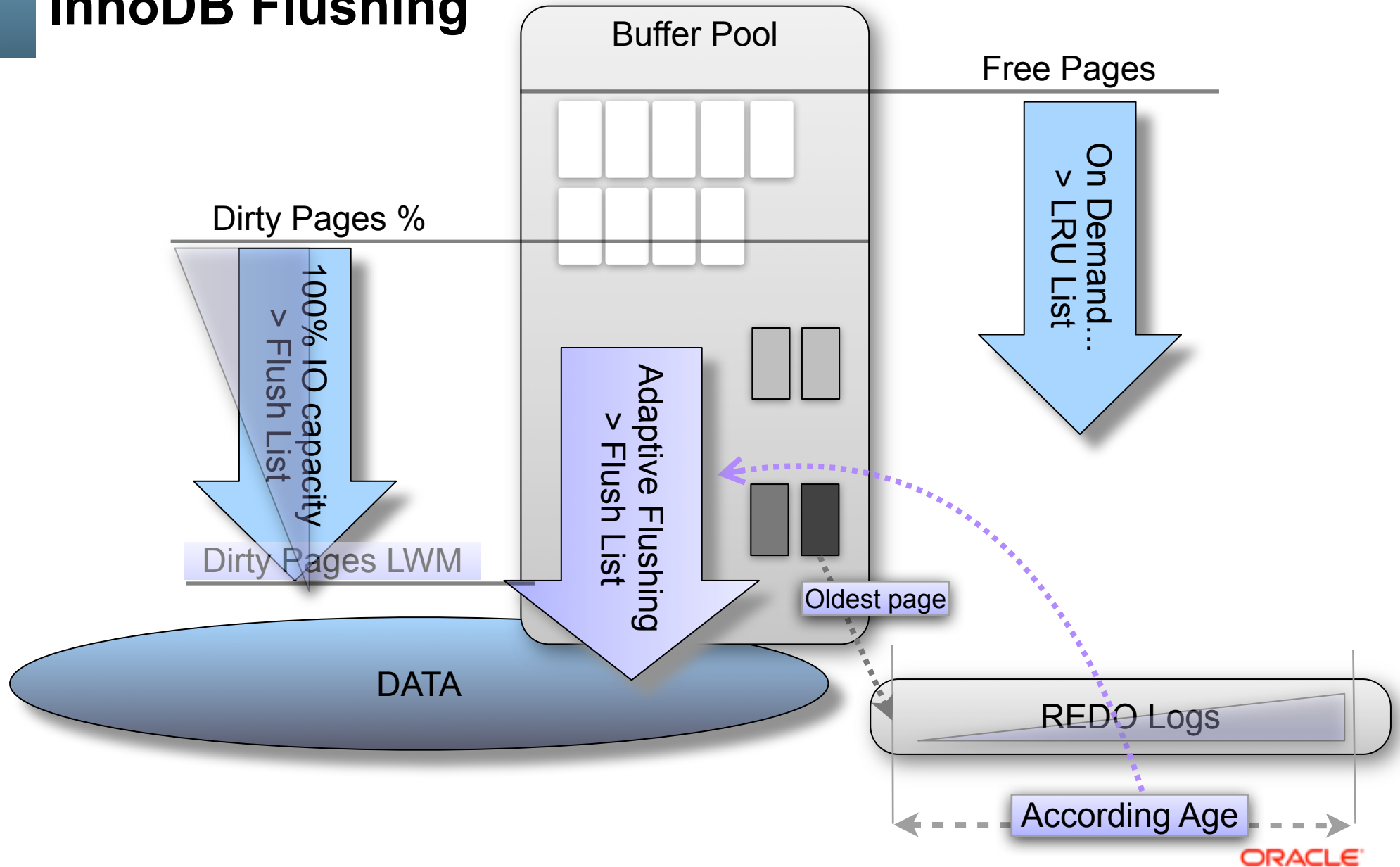


# InnoDB Flushing

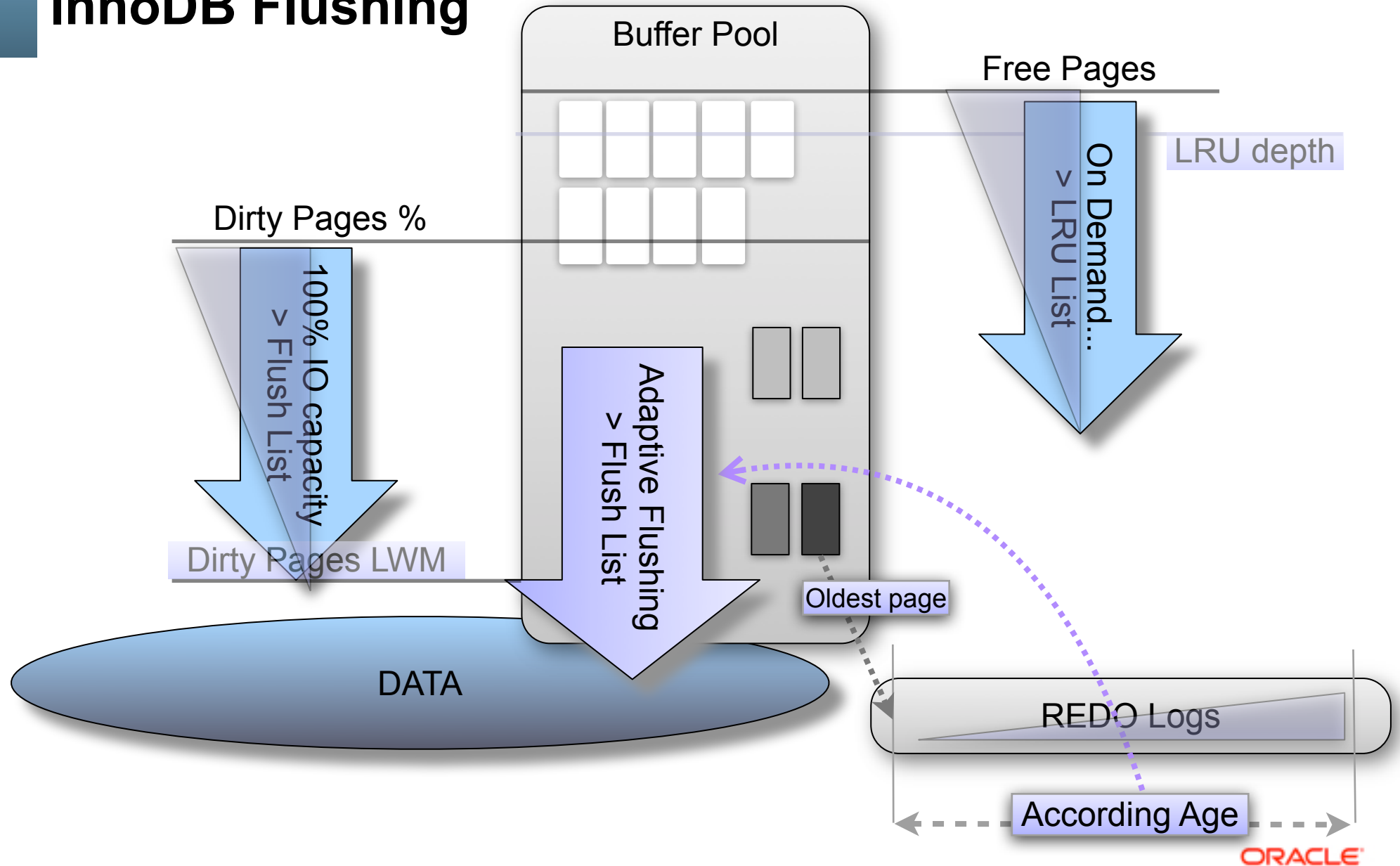




# InnoDB Flushing

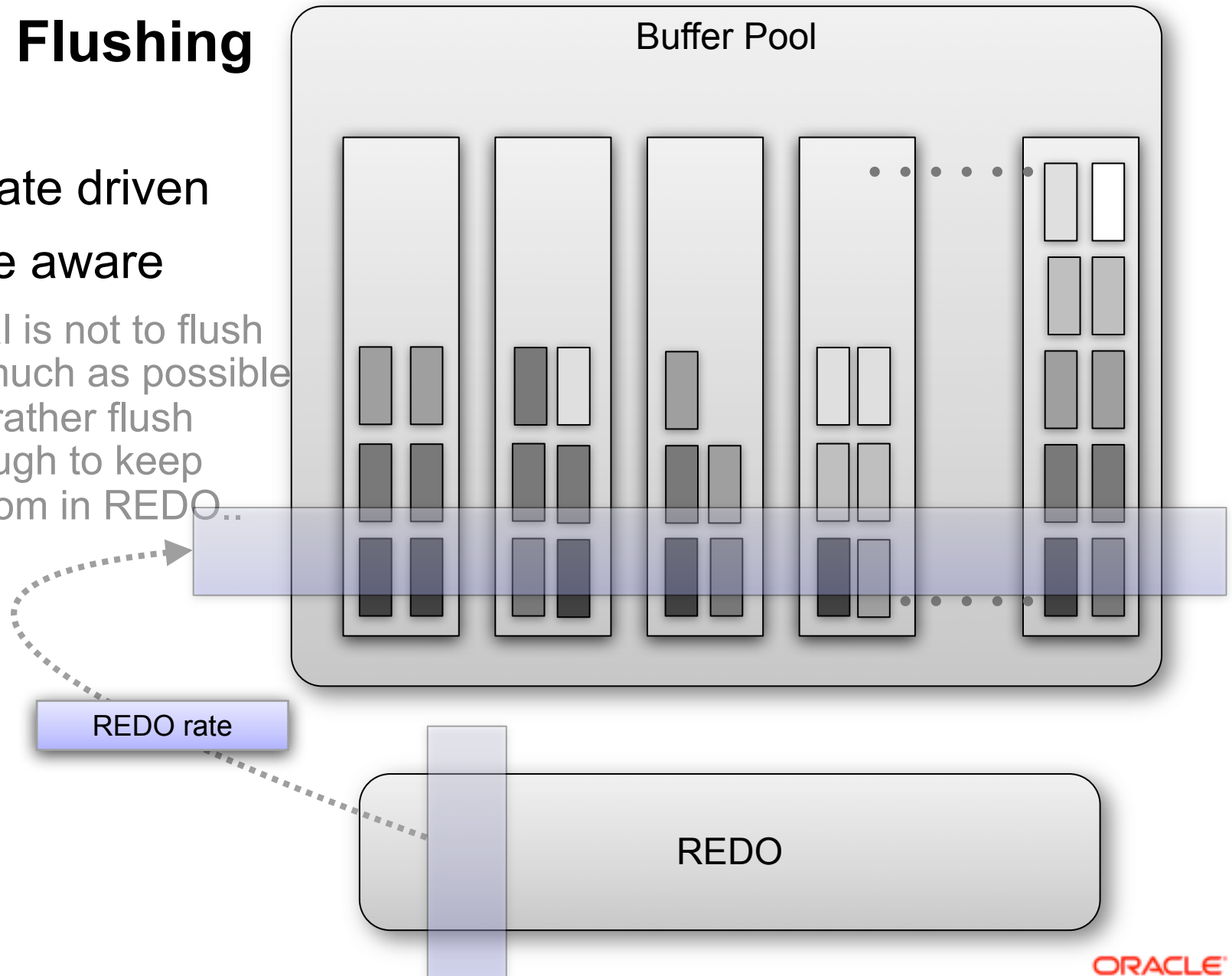


# InnoDB Flushing



# InnoDB Flushing

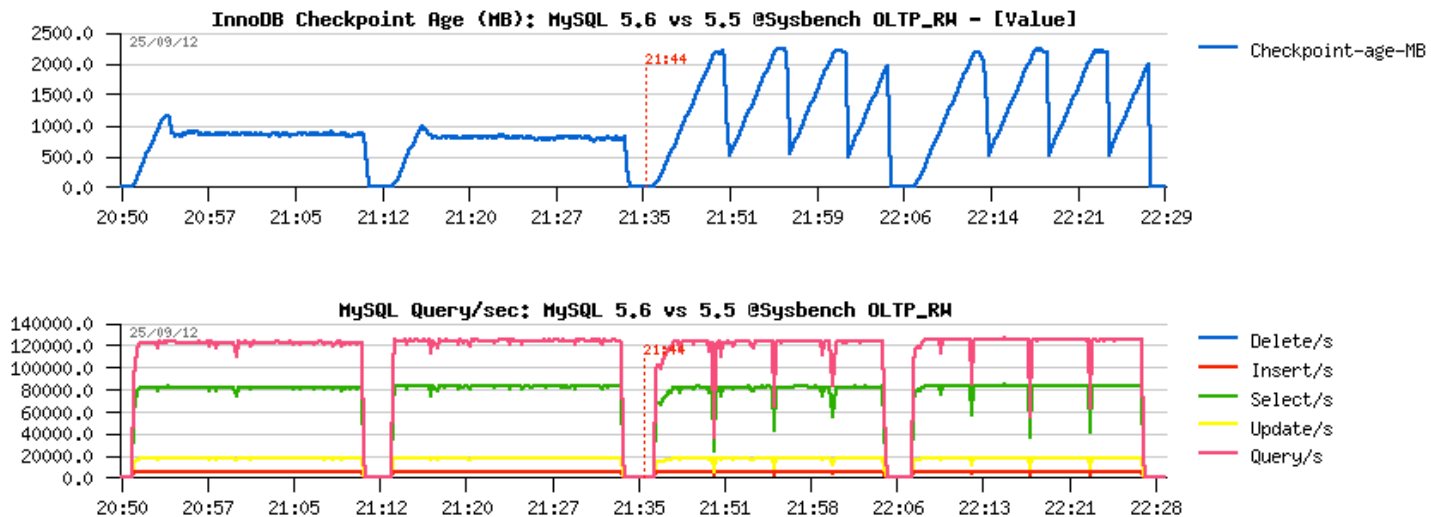
- REDO rate driven
- LSN Age aware
  - the goal is not to flush as much as possible but rather flush enough to keep a room in REDO..



# Adaptive Flushing: MySQL 5.6 vs 5.5

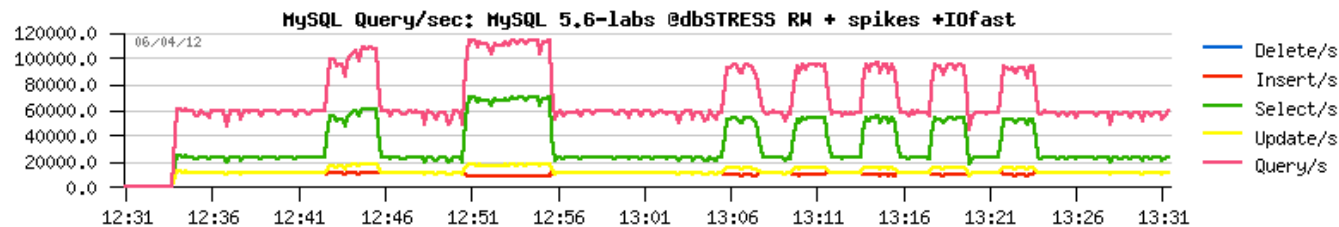
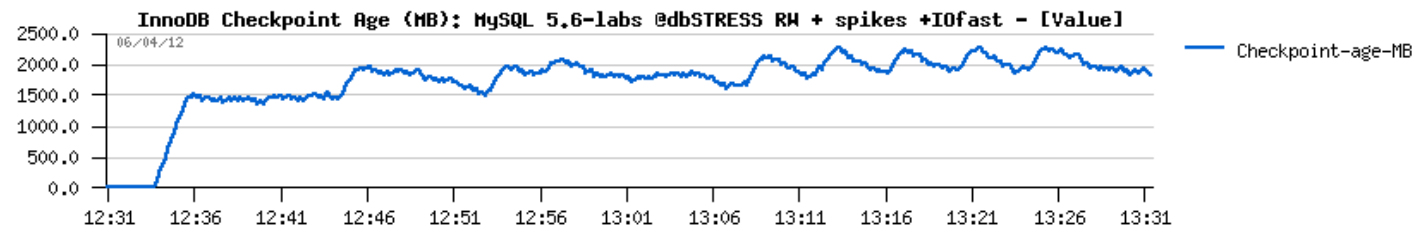
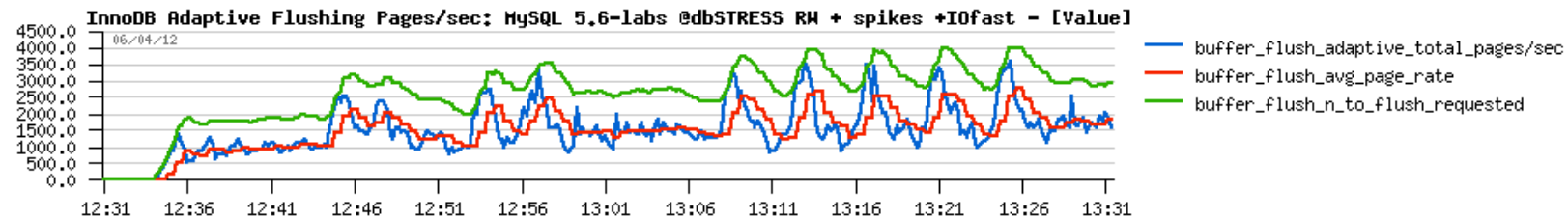
- OLTP\_RW Workload:

- Same IO capacity
- Different logic..



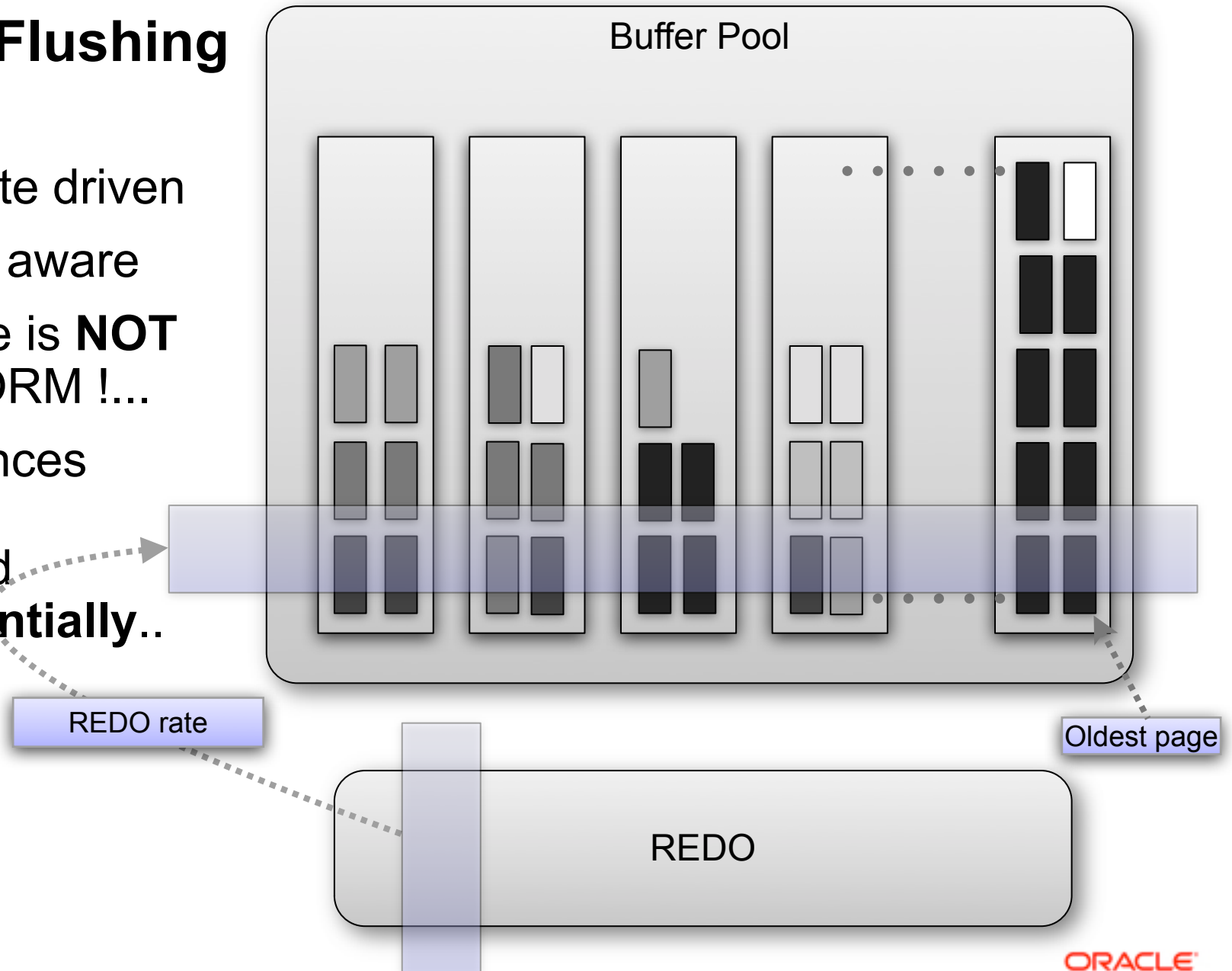
# InnoDB : Resisting to activity spikes in 5.6

- dbSTRESS R+W with spikes



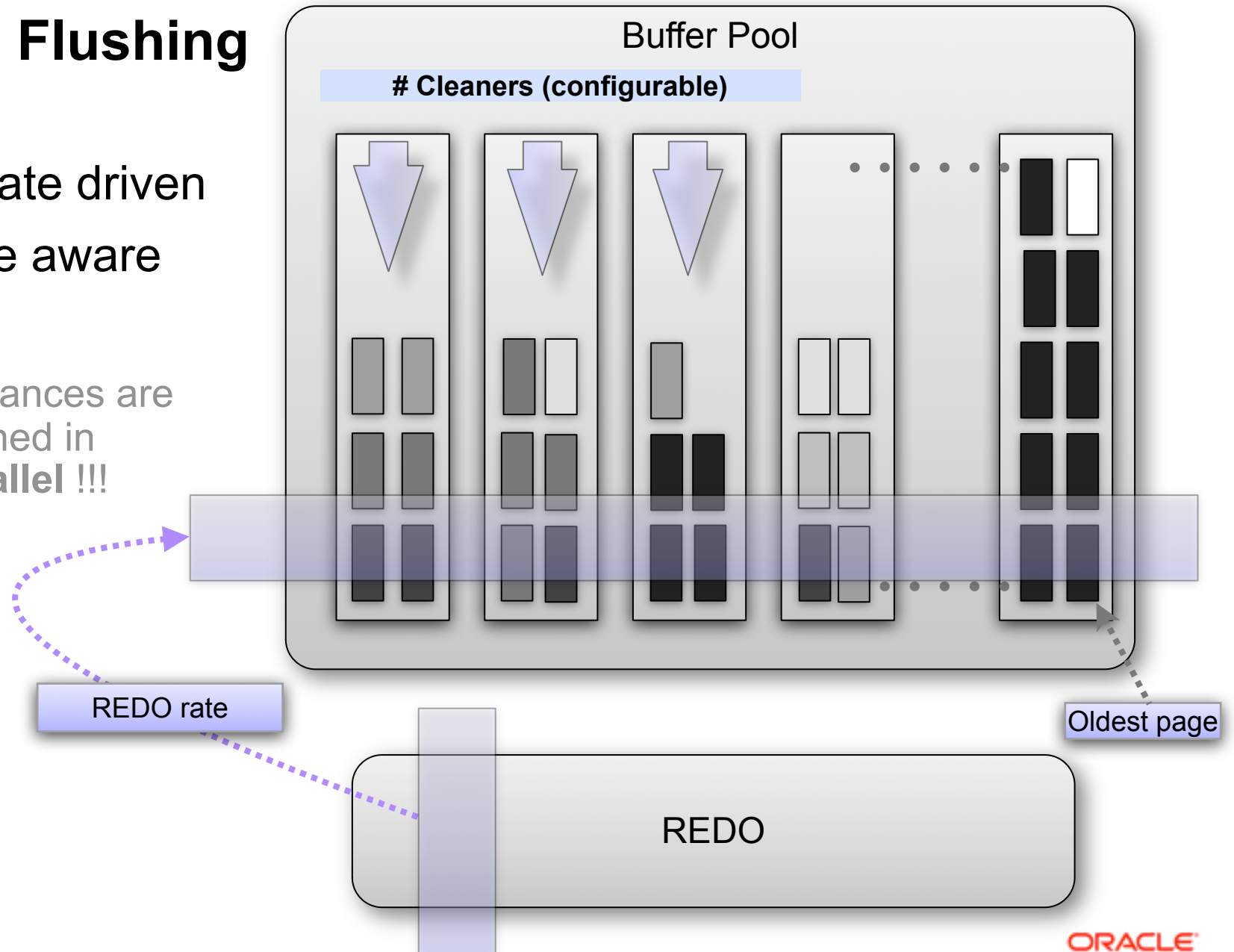
# InnoDB Flushing

- REDO rate driven
- LSN Age aware
- Page Age is **NOT** UNIFORM !...
- BP Instances are flushed **sequentially**..



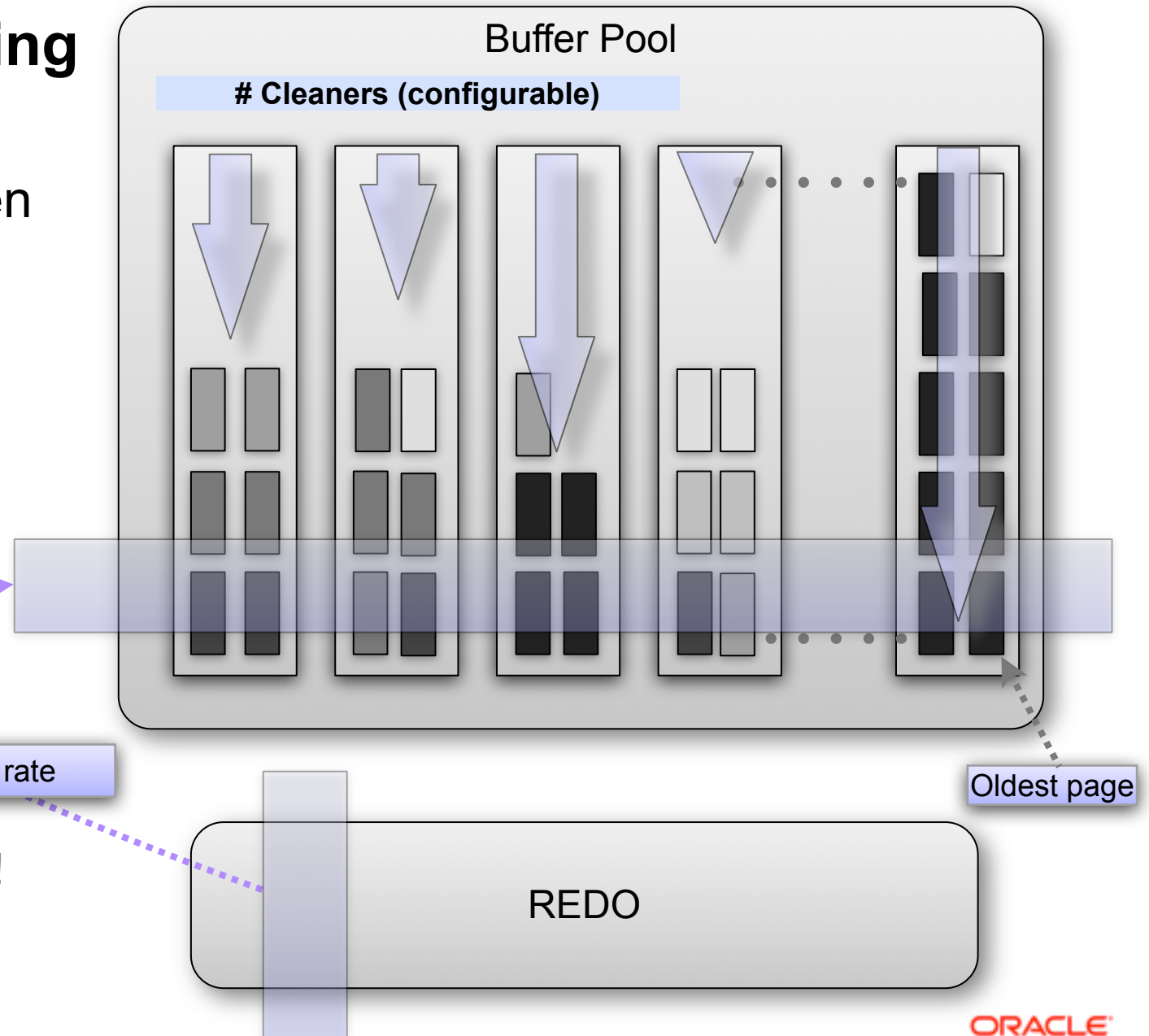
# InnoDB Flushing

- REDO rate driven
- LSN Age aware
- 5.7 :
  - BP Instances are flushed in **parallel !!!**



# InnoDB Flushing

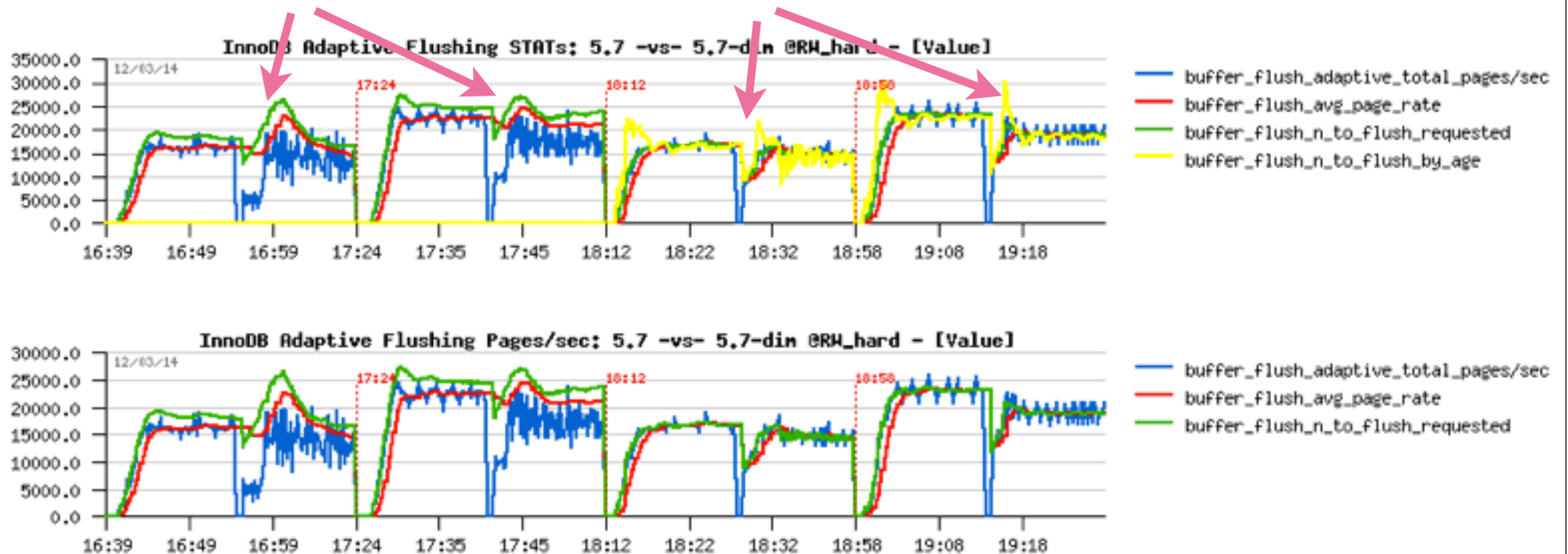
- REDO rate driven
- LSN Age aware
- 5.7 :
  - BP Instances are flushed in **parallel !!!**
  - Flushing rate is **adapted to Age distribution** within each BP instance !!!





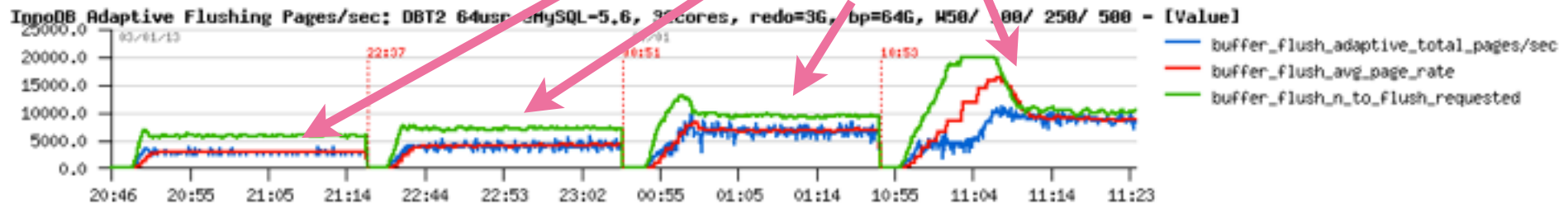
# InnoDB Flushing in 5.7

- Considering Age distribution :
  - Parallel Only -vs- Parallel + Age aware



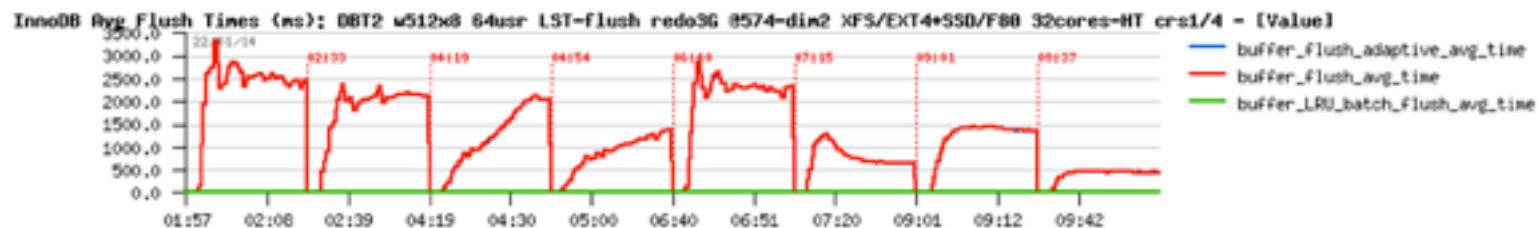
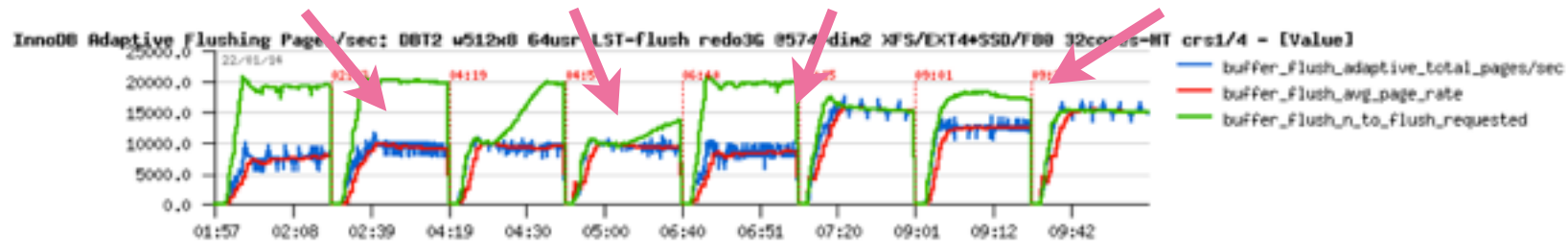
# RW IO-bound “In-Memory”

- Impact of the database size
  - with a growing db size the TPS rate may be only the same or worse ;-)
  - and required Flushing rate may only increase..
- DBT2 workload :
  - 64 users, db volume: 50W, 100W, 250W, 500W



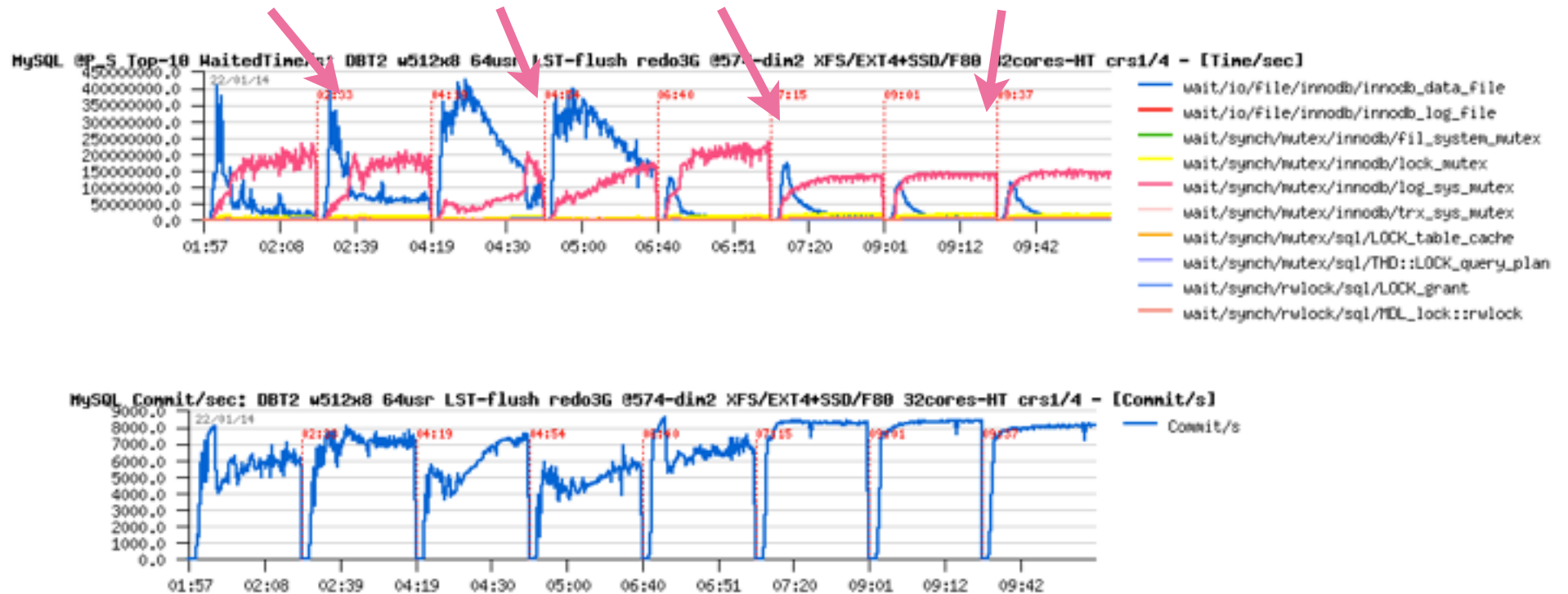
# InnoDB Flushing in 5.7

- Considering fast storage :
  - DBT2 512Wx8, 64usr, each test first with 1 then with 4 cleaners
  - XFS@SSD | EXT4@SSD | XFS@LSI-F80 | EXT4@LSI-F80



# InnoDB Flushing in 5.7

- Considering fast storage :
  - DBT2 512Wx8, 64usr, each test first with 1 then with 4 cleaners
  - XFS@SSD | EXT4@SSD | XFS@LSI-F80 | EXT4@LSI-F80



## RW IO-bound

- Still data In-Memory, but much bigger volume :
  - more pages to flush for the **same** TPS rate
- Data bigger or much bigger than Memory / cache / BP :
  - the amount of free pages becomes short very quickly..
  - and instead of mostly IO writes only you're starting to have IO reads too
  - these reads usually mostly random reads
  - if your storage is slow - reads will simply kill your TPS ;-)
  - if your storage can follow - then things become much more interesting
  - ..until you're hitting fil\_sys mutex contention and reach your Max TPS within a given conditions...
- NOTE:
  - using **AIO + O\_DIRECT** is the must for RW IO-bound !..

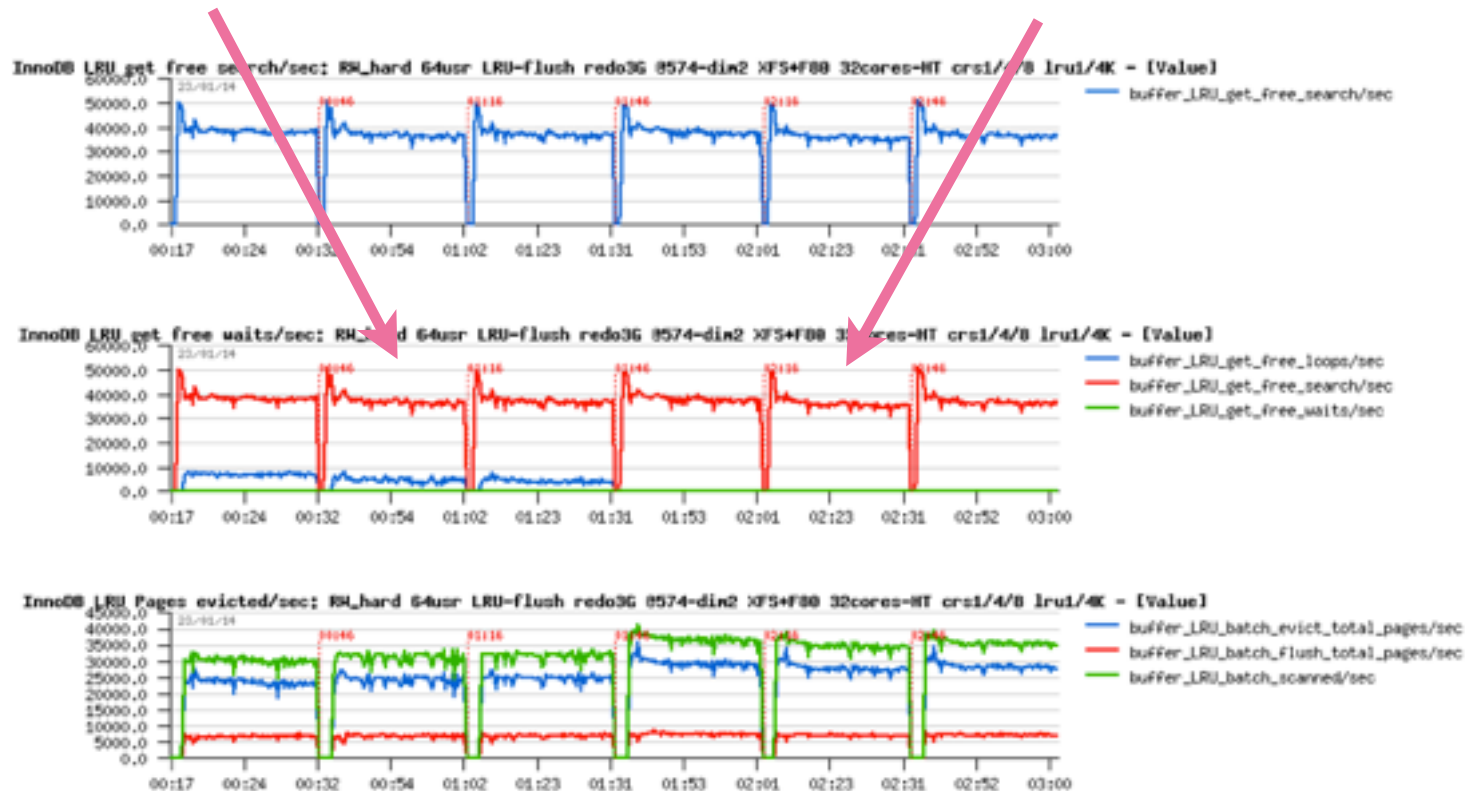
# RW IO-bound “Out-of-Memory”

Monitoring !!!!!

- LRU Flushing in 5.6 (broadly speaking) :
  - Cleaner thread for each BP instance :
    - check if free list contains at least N (LRU depth) pages : yes => return();
    - scan BP instance LRU list up to N (LRU depth) pages :
      - page is “dirty” : place it on flush, then clear & move to a free list
      - page is “not dirty” : clear & move it to a free list
      - free list reached N (LRU depth) pages: return()
  - User thread :
    - want a free page : get a one ? yes => return();
    - scan LRU list to see if can find one “not dirty” quickly..
      - found : clear & move it to a free list; goto begin..
      - not found : try to flush one; signal “flush event”; goto begin..
    - doing a second loop and there is still no free pages : sleep; then goto begin..
- Optimal : Cleaner is **always** keeping free lists non-empty ;-))

# RW IO-bound “Out-of-Memory”

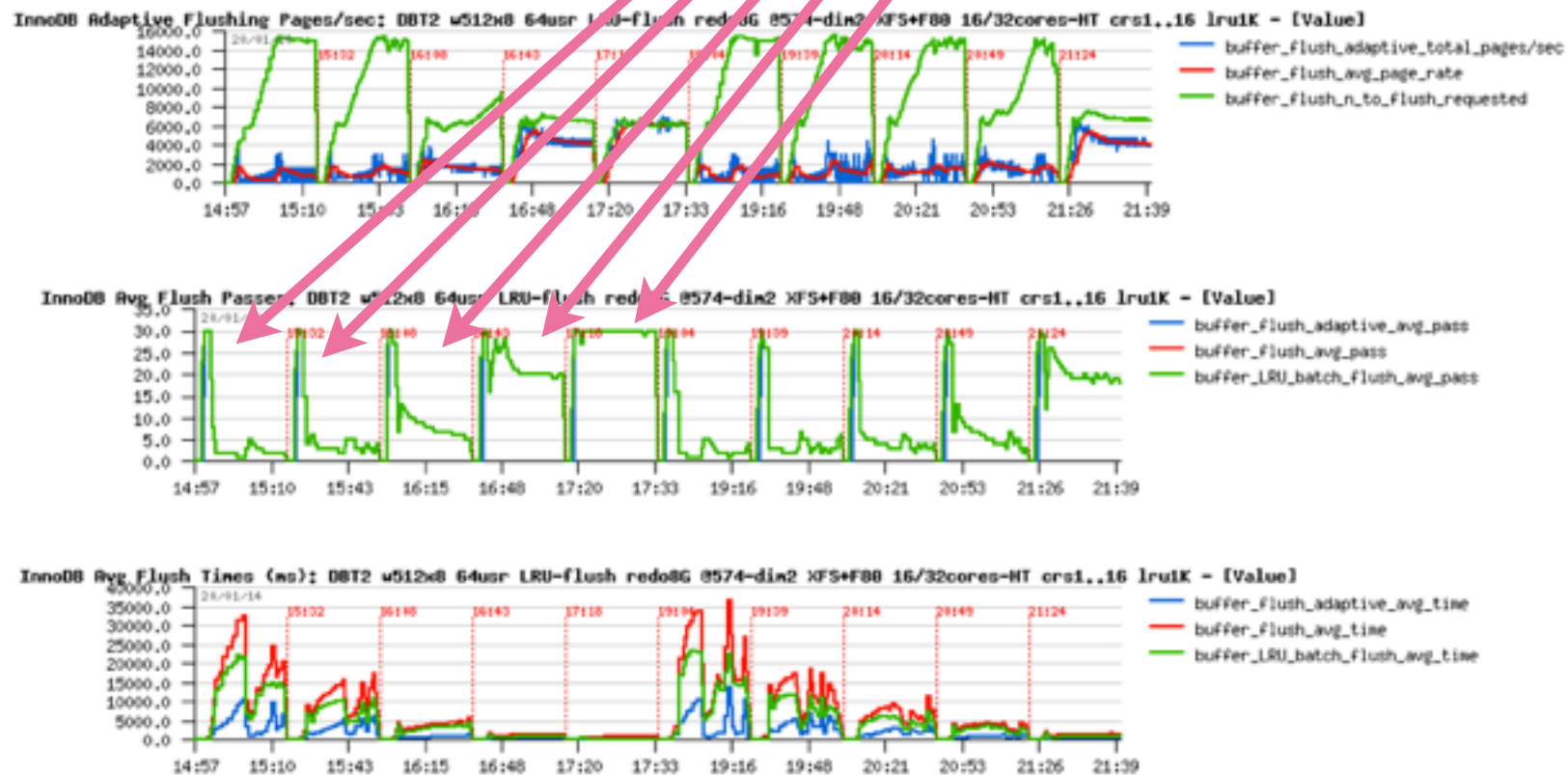
- LRU Flushing in 5.7 (broadly speaking) :
  - similar to 5.6 but with parallel Cleaners (but this is not always important ;-))
  - look: LRU depth=1K, cleaners=1/4/8 | LRU depth=4K, cleaners=1/4/8





# RW LRU-bound : FS impact..

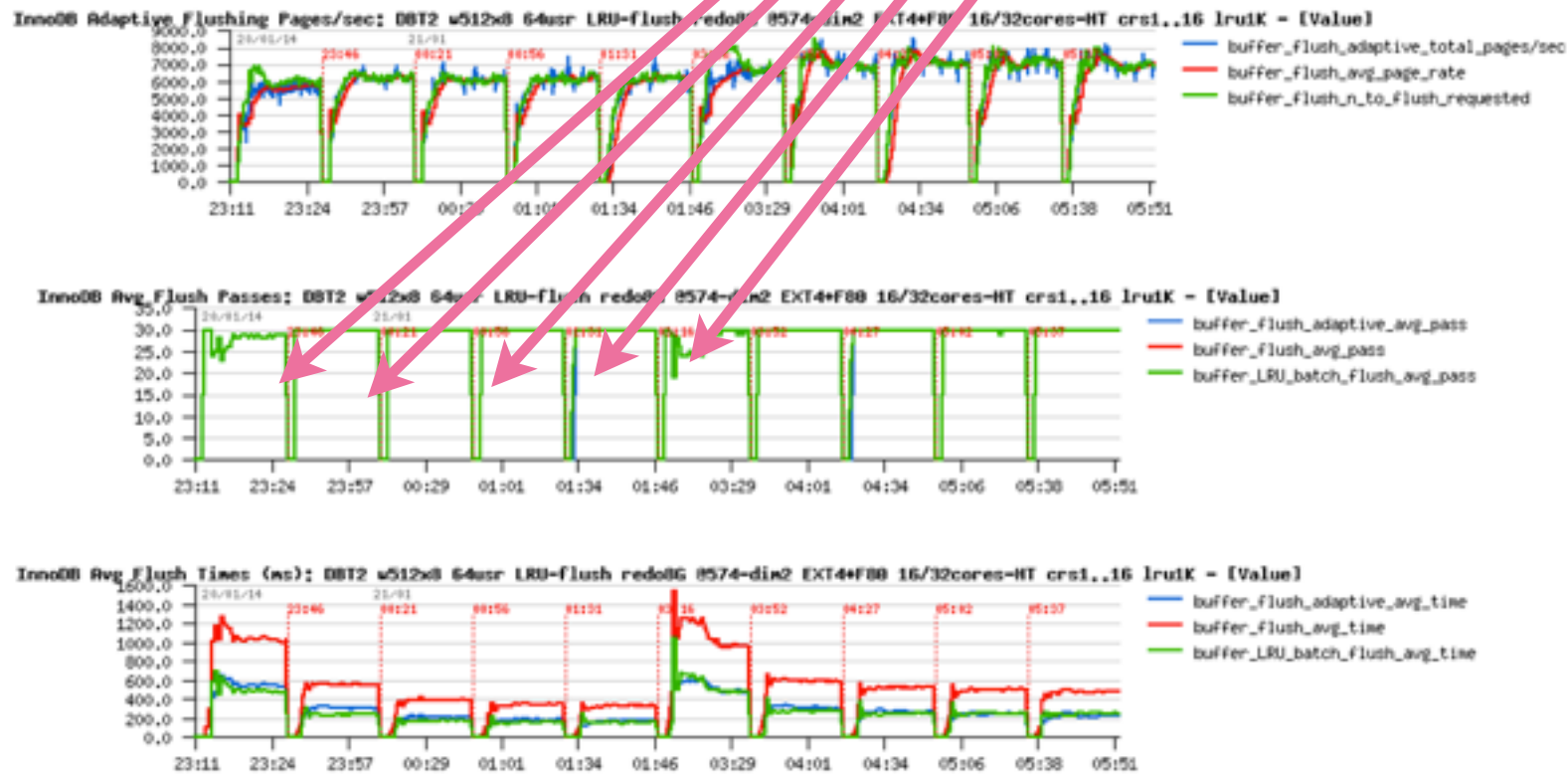
- DBT2 Workload, 64 users, **XFS**
  - LRU depth=1K, cleaners= 1, 2, 4, 8, 16 16cores-HT / 32cores-HT





# RW LRU-bound : FS impact..

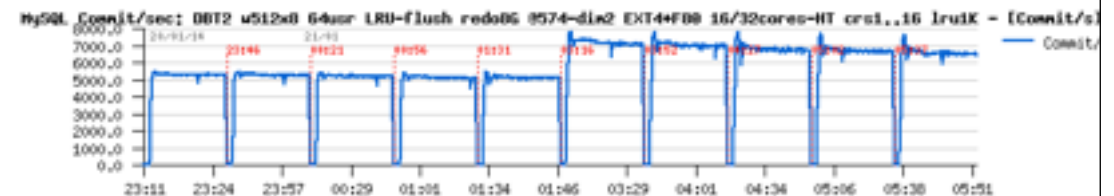
- DBT2 Workload, 64 users, **EXT4**
  - LRU depth=1K, cleaners= 1, 2, 4, 8, 16 16cores-HT / 32cores-HT



# RW LRU-bound : FS impact..

- DBT2 Workload, 64 users, **XFS -vs- EXT4**

- LRU depth=1K, cleaners= 1, 2, 4, 8, 16 16cores-HT / 32cores-HT
- More IO data wait on XFS...



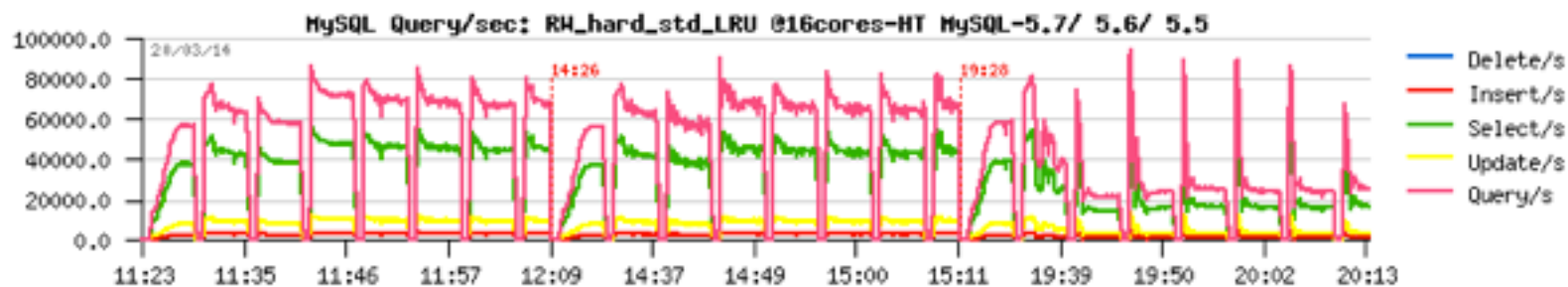
# RW LRU-bound : 5.5 is out of the game..

- Sysbench OLTP\_RW 10M x32-tables

- Users: 8, 16, 32 .. 1024

- MySQL : 5.7 / 5.6 / 5.5

**Please, upgrade me to 5.6 !!!**



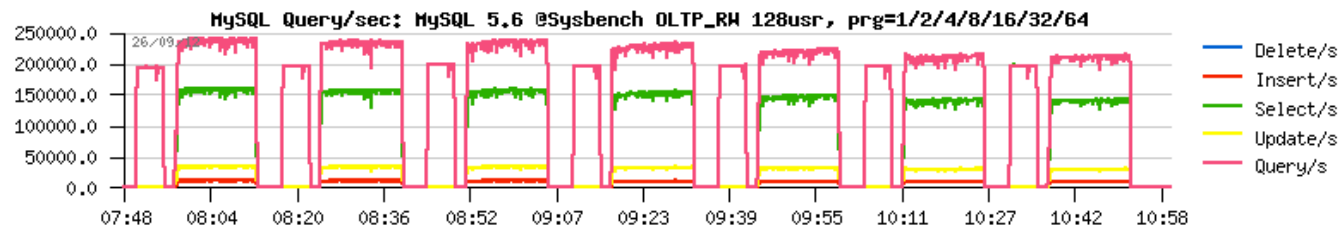
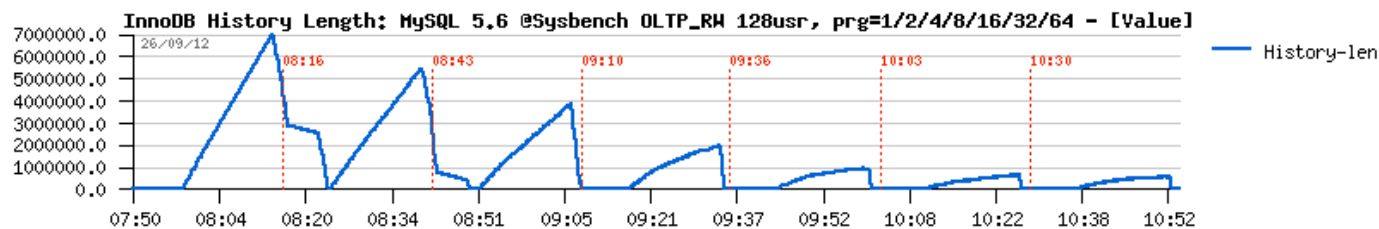
# Read+Write Workloads : InnoDB Purge

- InnoDB Purge...

- 5.5 : Purge Thread !!! ;-)
- 5.6 :
  - Multi-Threaded Purge !
  - fix for purge lag code !
- 5.7 :
  - monitor InnoDB History Length **ALWAYS** ! ;-)
  - if NO purge lagging : excellent! (& be happy! ;-))
  - if purge is lagging : use a purge lag config setting.. (& wait for fix)
- example of config for 5.6 and 5.7 to avoid purge lagging:
  - innodb\_max\_purge\_lag = 1000000 (1M max, ex.)
  - innodb\_max\_purge\_lag\_delay = 30000000
  - innodb\_purge\_threads = 4

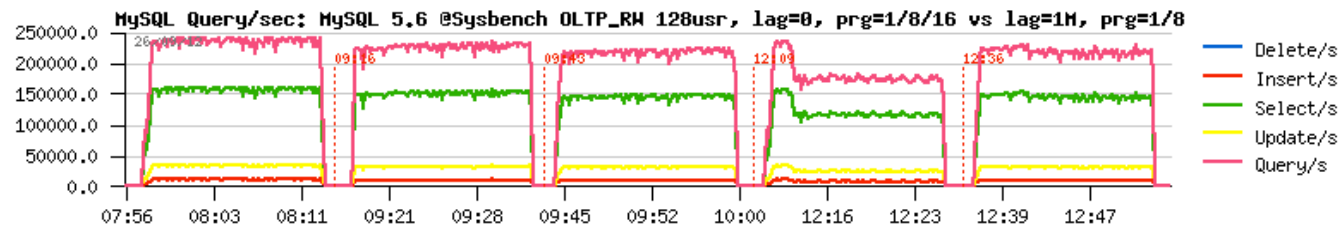
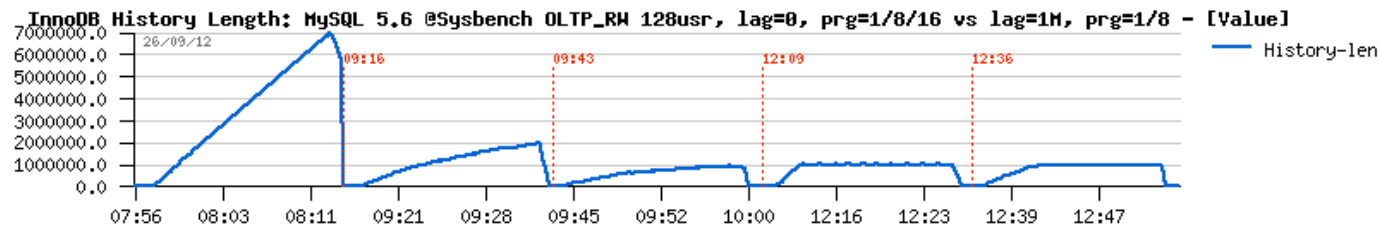
# InnoDB : Purge improvement since 5.6

- Several Purge Threads :
  - NOTE #1 : activation is auto-magical (I'm serious ;-))
  - NOTE #2 : look well on the graphs - purge is not free !!!



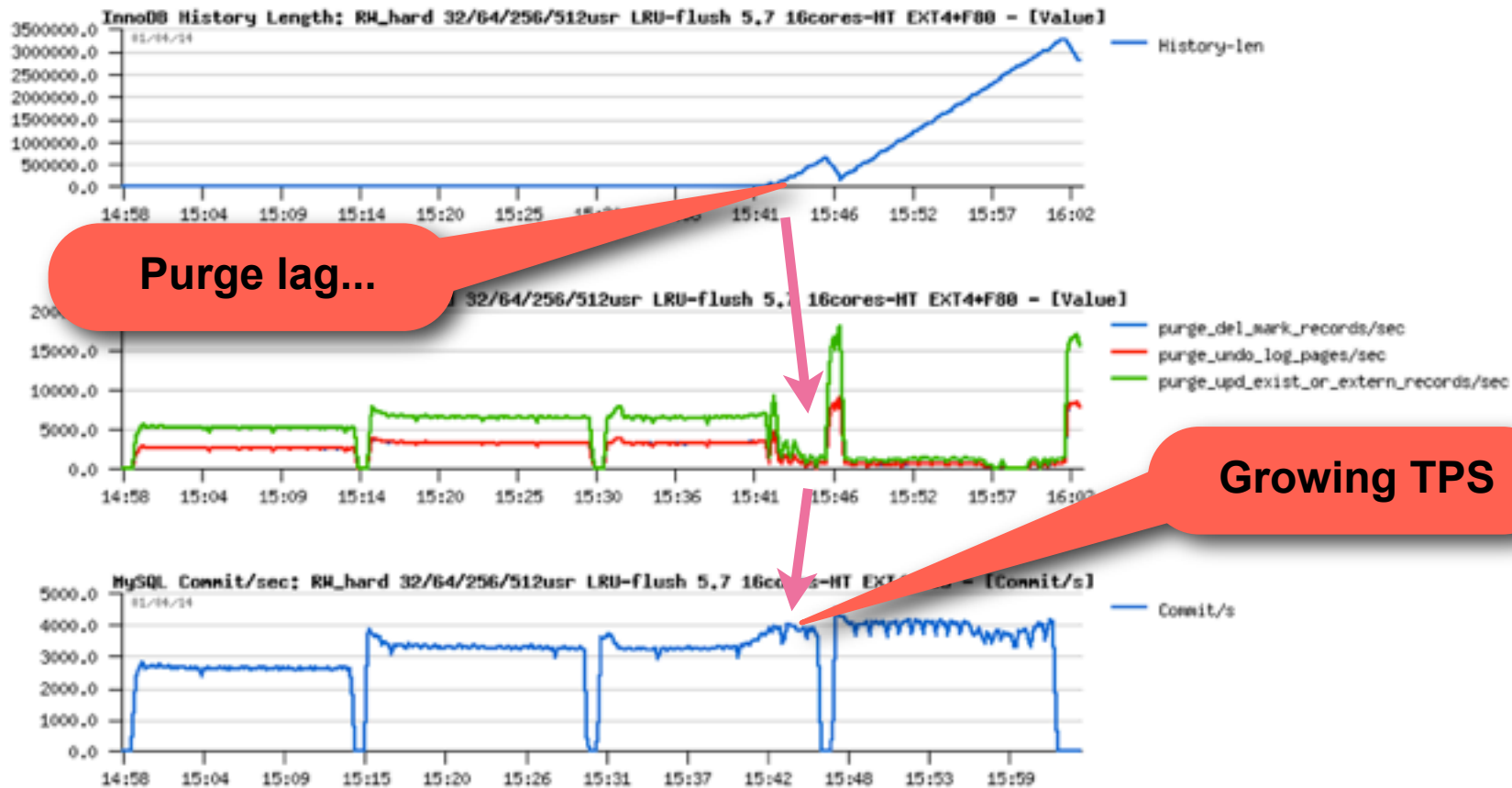
# InnoDB : Purge improvement since 5.6

- Fixed max purge lag code!
  - innodb\_max\_purge\_lag
  - innodb\_max\_purge\_lag\_delay <= configurable!
- Setting innodb\_max\_purge\_lag=1M:



# InnoDB : be sure your TPS is fair ;-)

- Purge lagging impact on IO-bound OLTP\_RW 10Mx32-tab:
  - moving from 3200 to 4000 TPS... - cool, right? ;-)



# RW IO/LRU-bound : “tuning” by elimination

- Filesystem : let's go with EXT4 ;-)
  - TODO : understand what is wrong with XFS...
- # Cleaner threads :
  - 2 or 4 should be enough.. - let's go with 4
- LRU depth :
  - dumb rule: the SUM setting should be bigger than a free page/sec demand
  - so for 40K get free page/sec setting LRU depth=2K with 32 BP instances should be more than enough..
  - but a free page demand may grow.. - let's go with LRU depth=4K and see ;-)
- Purge :
  - innodb\_max\_purge\_lag = 1000000
  - innodb\_max\_purge\_lag\_delay = 30000000
  - innodb\_purge\_threads = 4



# RW related starter configuration settings

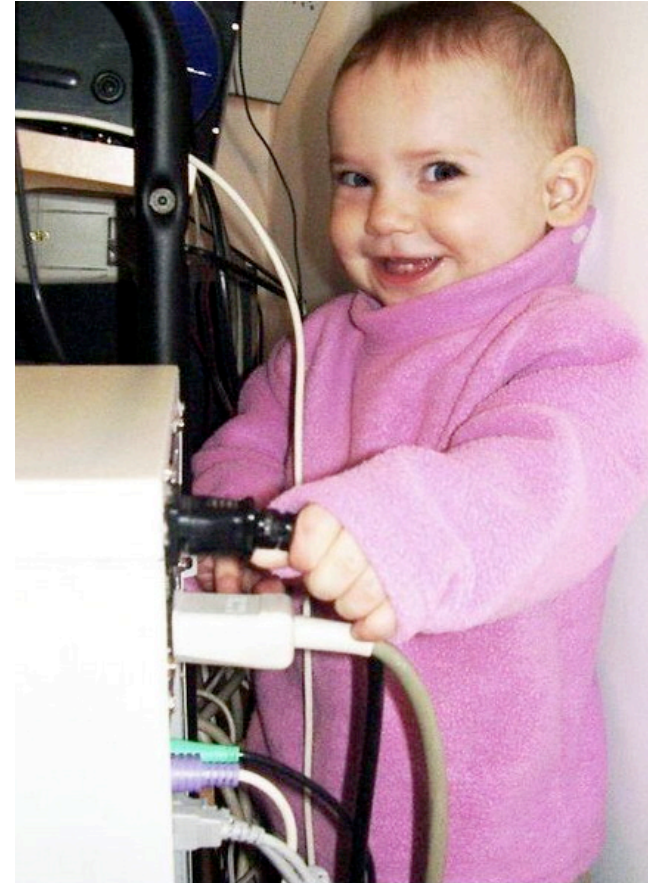
- my.conf :

```
innodb_file_per_table
innodb_log_file_size=1024M
innodb_log_files_in_group=3 / 12 / ...
innodb_checksum_algorithm= none / crc32
innodb_doublewrite= 0 / 1
innodb_flush_log_at_trx_commit= 2 / 1
innodb_flush_method=O_DIRECT
innodb_use_native_aio=1
innodb_adaptive_hash_index=0
```

```
innodb_adaptive_flushing = 1
innodb_flush_neighbors = 0
innodb_read_io_threads = 16
innodb_write_io_threads = 16
innodb_io_capacity=15000
innodb_max_dirty_pages_pct=90
innodb_max_dirty_pages_pct_lwm=10
innodb_lru_scan_depth=4000
innodb_page_cleaners=4
```

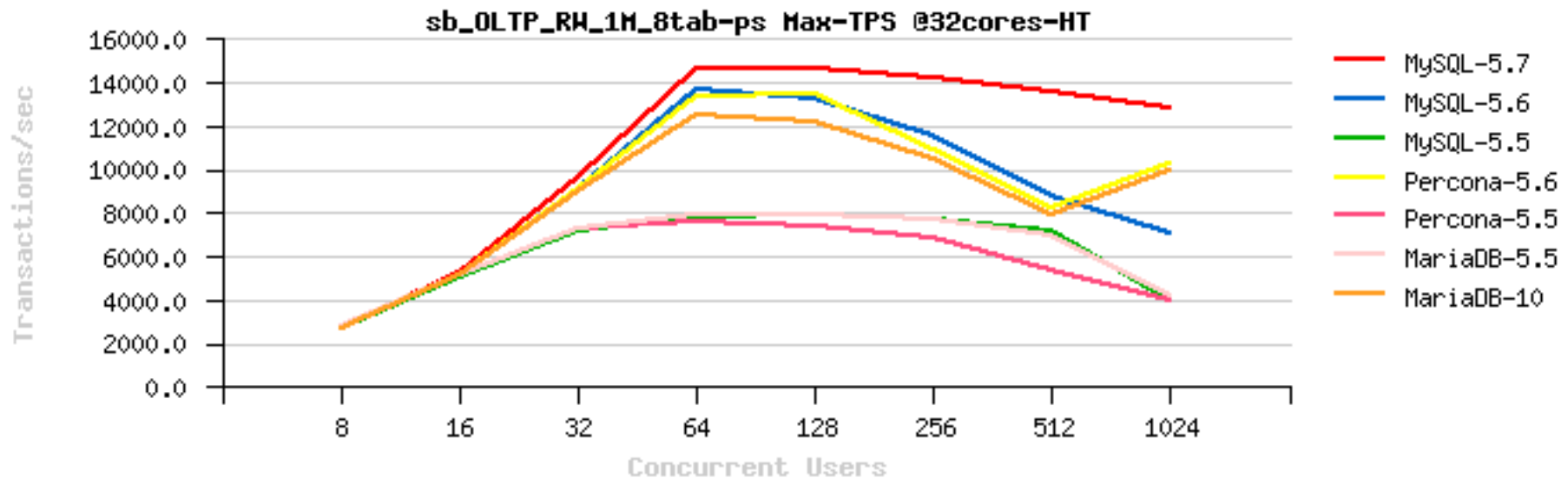
```
innodb_purge_threads=4
innodb_max_purge_lag_delay=30000000
innodb_max_purge_lag=1000000
```

```
binlog ??
```



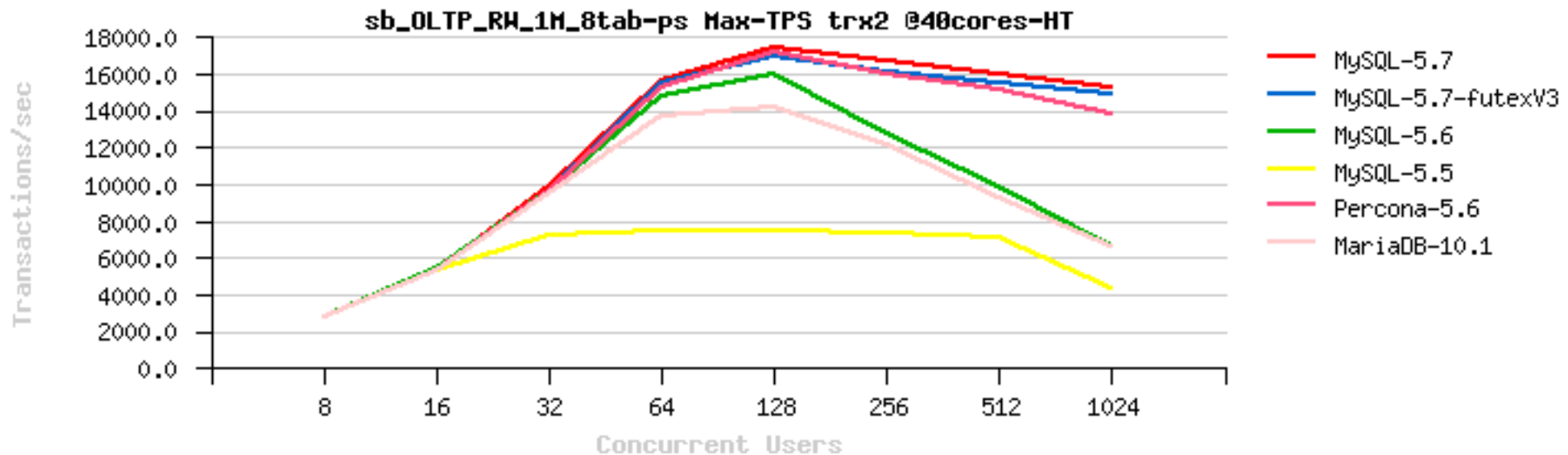
# RW In-Memory @MySQL 5.7

- Sysbench OLTP\_RW 8-tables 32cores-HT :
  - note again where are “5.5” series, and where are “based on 5.6”..
  - so, thank you MySQL 5.6 !! ;-))



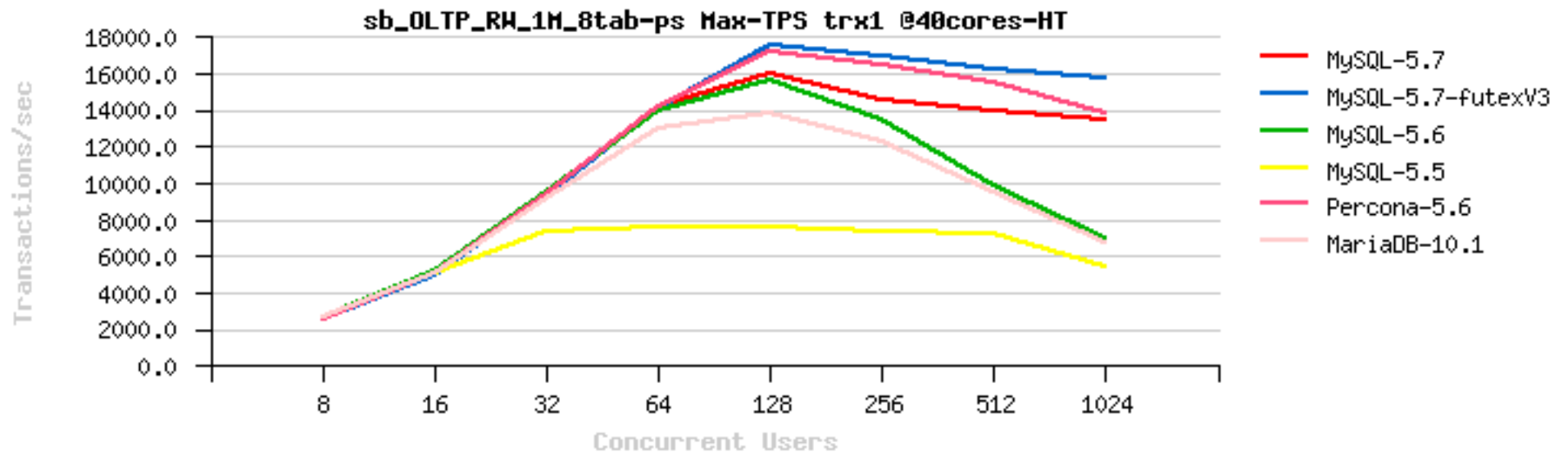
# Sysbench OLTP\_RW In-Memory

- Sysbench OLTP\_RW **8-tables** TRX2 @40cores-HT :
  - TRX2 : innodb\_flush\_log\_at\_trx\_commit = 2



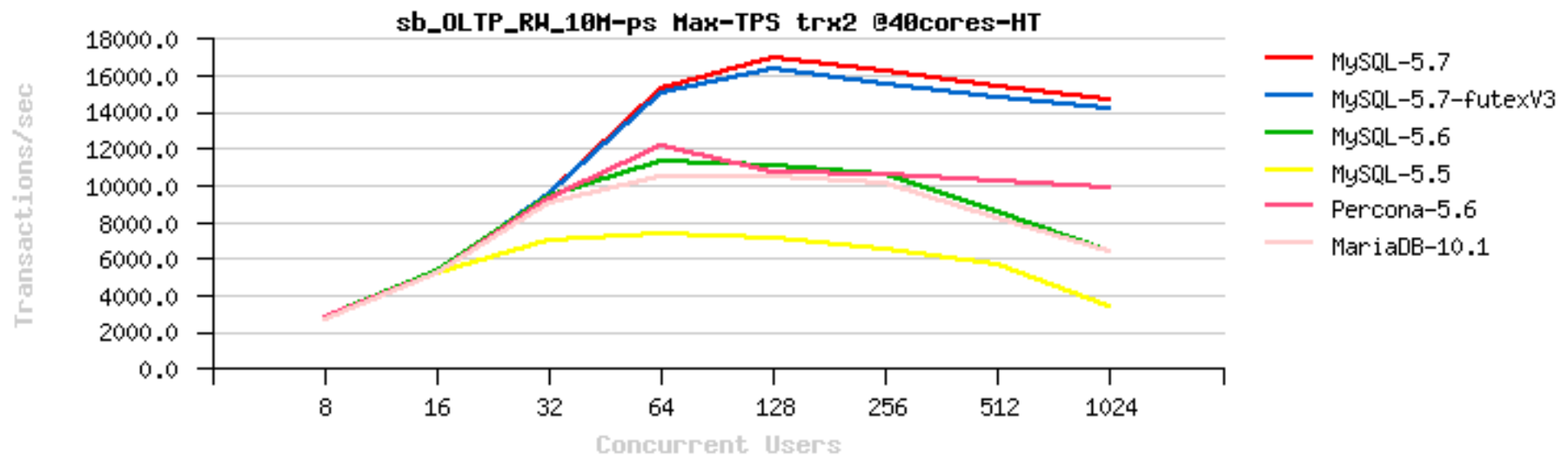
# Sysbench OLTP\_RW In-Memory

- Sysbench OLTP\_RW **8-tables** TRX1 @40cores-HT :
  - TRX1 : innodb\_flush\_log\_at\_trx\_commit = 1



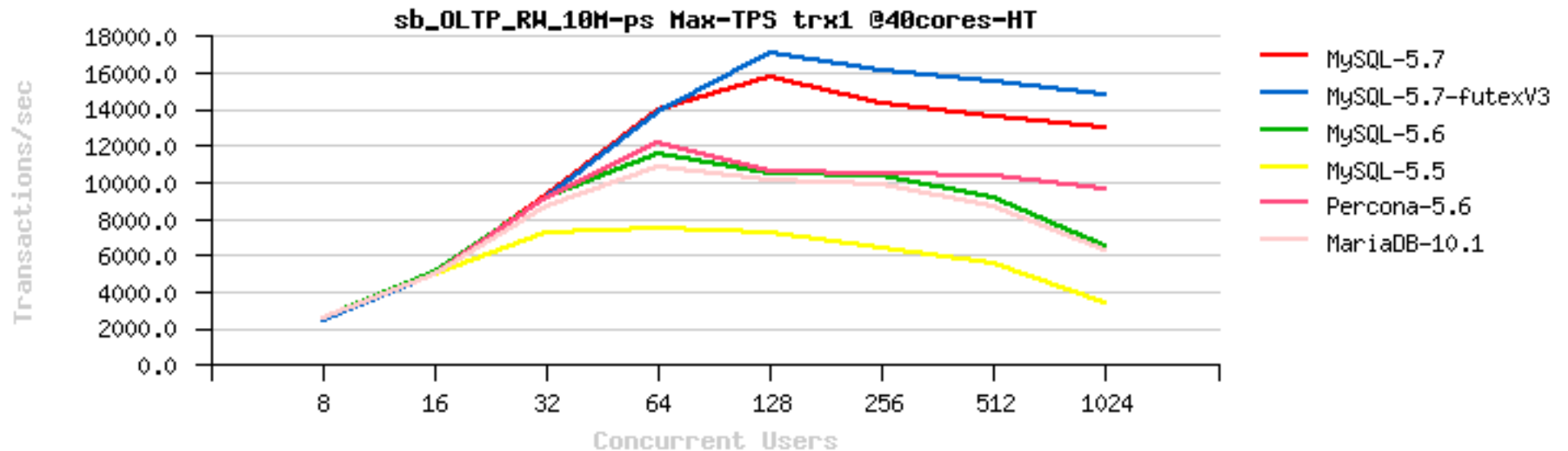
# Sysbench OLTP\_RW In-Memory

- Sysbench OLTP\_RW **1-table** TRX2 @40cores-HT :
  - TRX2 : innodb\_flush\_log\_at\_trx\_commit = 2



# Sysbench OLTP\_RW In-Memory

- Sysbench OLTP\_RW **1-table** TRX1 @40cores-HT :
  - TRX1 : innodb\_flush\_log\_at\_trx\_commit = 1



# Sysbench OLTP\_RW In-Memory

- Max TPS configs :

```
mysql> select max(tps), t_engine, t_ccr, spin_delay, trx_commit
        from Bench where t_name = 'sb_OLTP_RW_1M_8tab-ps'
        and t_tag= '575_DMR-RW' and t_cpu like '40cores%'
        group by 2,3,4,5 order by 1 desc limit 10;
```

max(tps)	t_engine	t_ccr	spin_delay	trx_commit
<b>17617</b>	<b>mysql576_futex_V3</b>	<b>64</b>	<b>96</b>	<b>1</b>
17497	mysql576_futex_V3	0	96	1
17438	mysql575	64	96	2
17307	mysql575	0	96	2
17231	percona5620	64	96	1
17197	percona5620	0	96	1
17168	percona5620	0	96	2
17113	percona5620	64	96	2
16963	mysql576_futex_V3	64	96	2
16780	mysql576_futex_V3	0	96	2

10 rows in set (0.03 sec)

```
mysql>
```

# High Concurrency Tuning

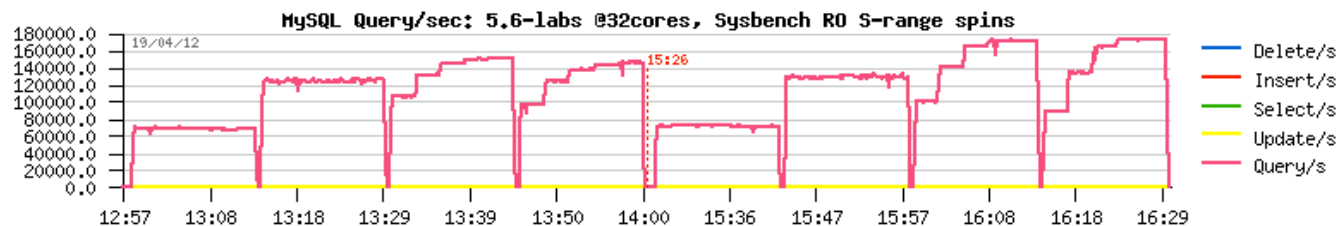
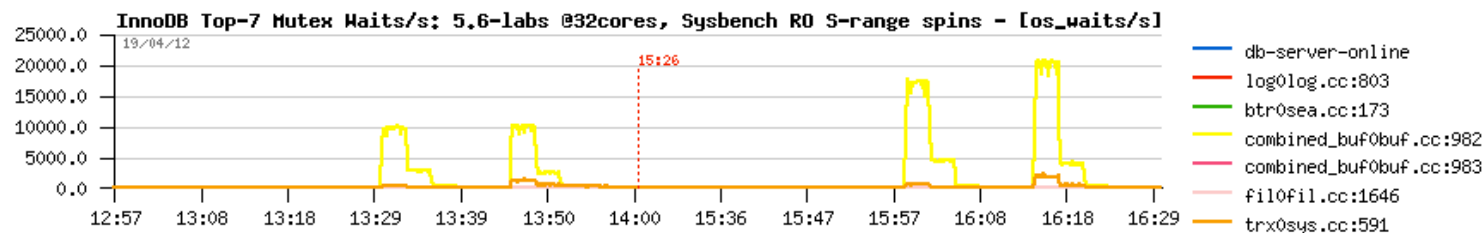
- If bottleneck is due a concurrent access on the same data (due application design) – ask your dev team to re-design ;-)
- If bottleneck is due MySQL/InnoDB internal contentions, then:
  - If you cannot avoid it, then at least don't let them grow ;-)
  - tune InnoDB spin wait delay to improve your Max QPS (dynamic)
  - innodb\_thread\_concurrency=N to avoid QPS drop on usr++ (dynamic)
  - CPU taskset / prcset (Linux / Solaris, both dynamic)
  - Thread Pool
  - NOTE:
    - things with contentions may radically change since 5.7, so stay tuned ;-)
    - InnoDB thread concurrency feature was **improved** in 5.6 and 5.7
    - the best working in 5.7, and using innodb\_thread\_concurrency=64 by default now makes sense..



# InnoDB Spin Wait Delay

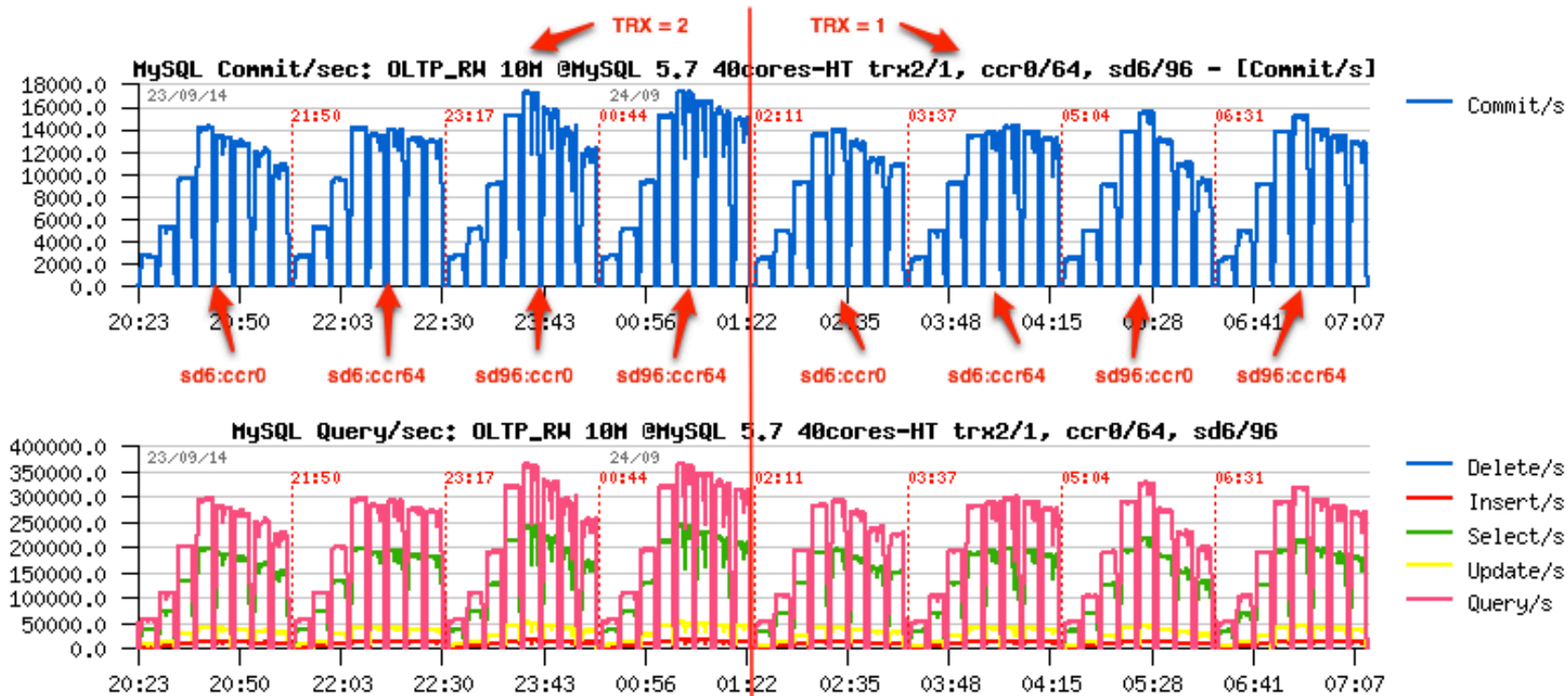
- RO / RW Workloads:

- With more CPU cores internal InnoDB contentions become more hot..
- Bind mysqld to less cores helps, but the goal is to use **more** cores ;-)
- Using innodb\_thread\_concurrency may not help anymore..
- So, innodb\_spin\_wait\_delay is entering in the game:



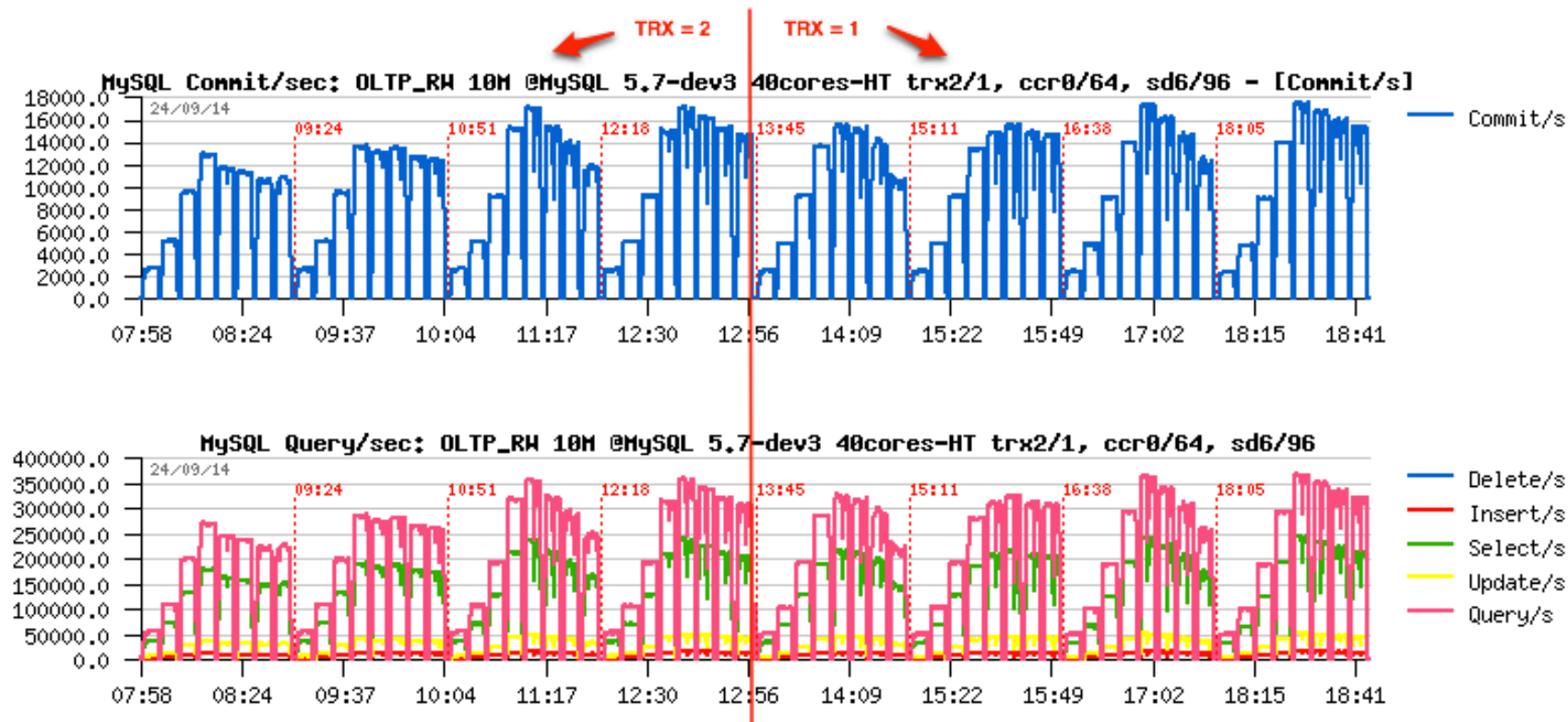
# Concurrency tuning on OLTP\_RW @MySQL 5.7

- OLTP\_RW 10M @MySQL 5.7 40cores-HT :
  - load conditions: TRX = 2 -vs- TRX = 1
  - cooking receipt : concurrency (ccr) & spin wait delay (sd)



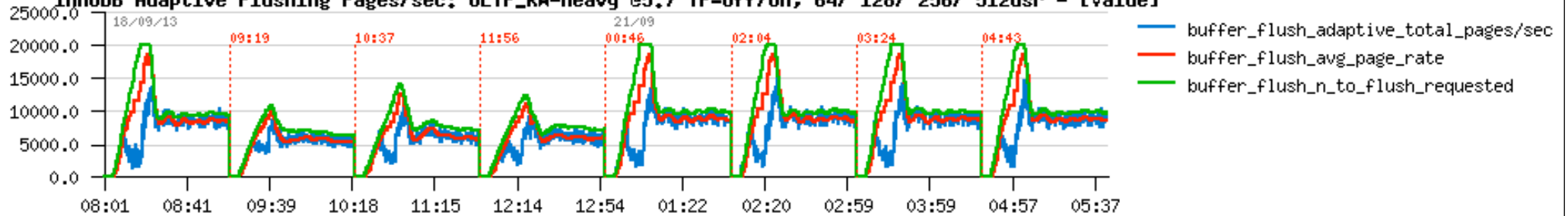
# Concurrency tuning on OLTP\_RW @MySQL 5.7-dev3

- OLTP\_RW 10M @MySQL 5.7-dev3 40cores-HT :
  - load conditions: TRX = 2 -vs- TRX = 1
  - cooking receipt : concurrency (ccr) & spin wait delay (sd)

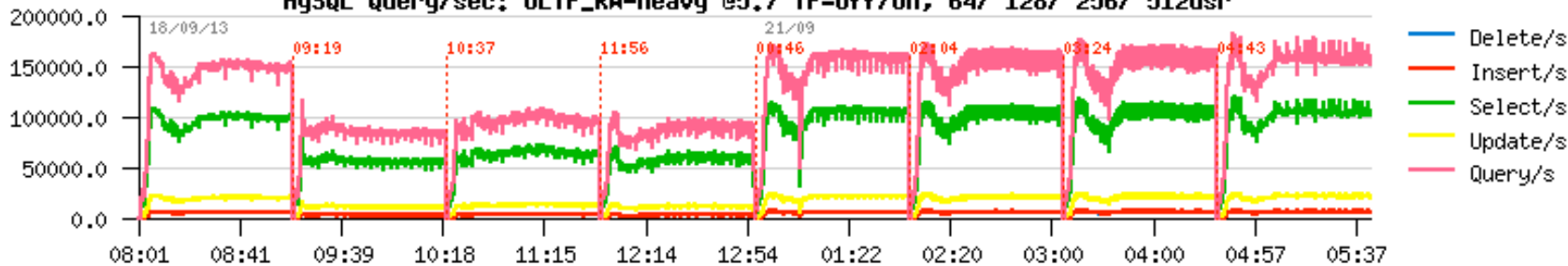


# Thread Pool in old MySQL 5.7 @Heavy OLTP\_RW

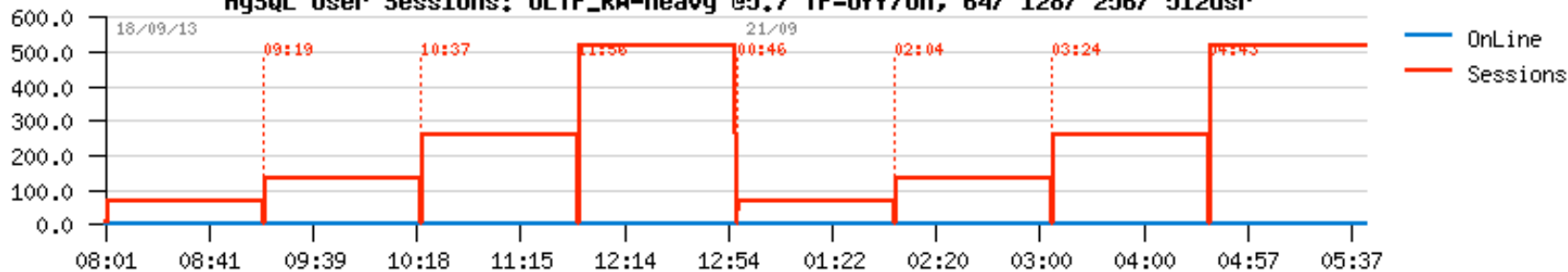
InnoDB Adaptive Flushing Pages/sec: OLTP\_RW-heavy @5.7 TP=off/on, 64/ 128/ 256/ 512usr - [Value]



MySQL Query/sec: OLTP\_RW-heavy @5.7 TP=off/on, 64/ 128/ 256/ 512usr



MySQL User Sessions: OLTP\_RW-heavy @5.7 TP=off/on, 64/ 128/ 256/ 512usr



## Tuning recap..

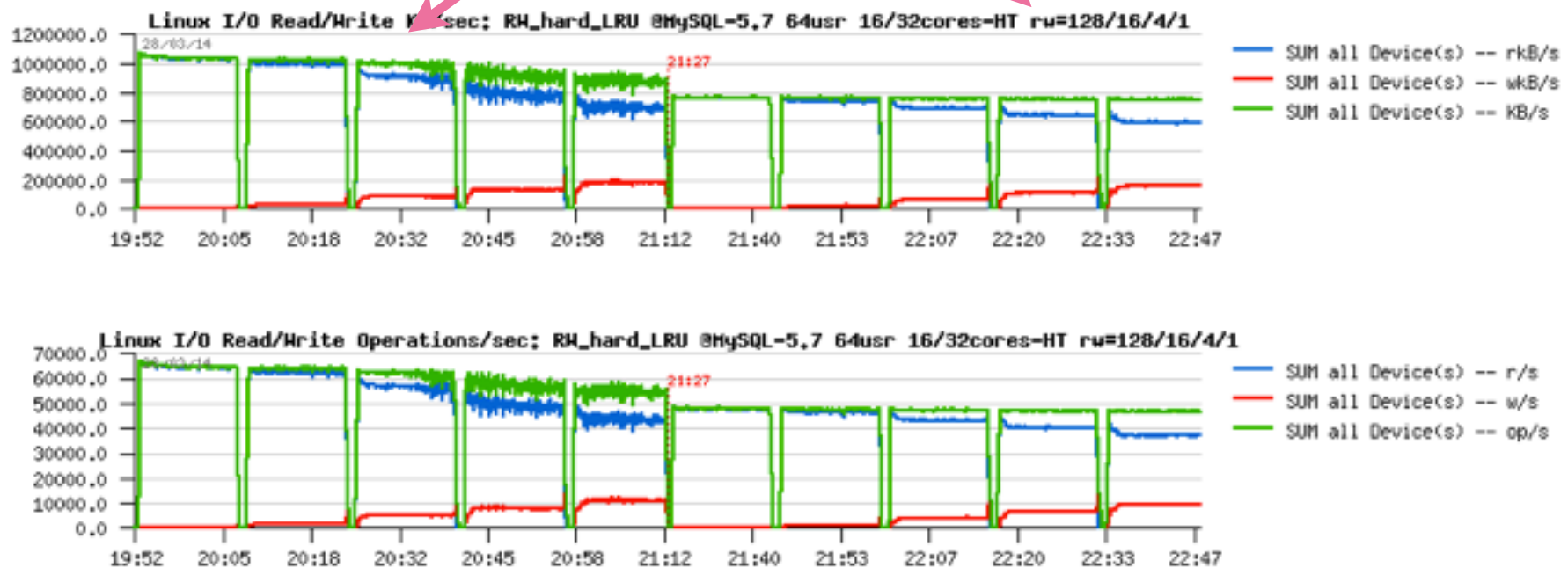
- Many things are limited “by design”.. - so, upgrade! ;-)
- Read-Only :
  - more and more better, BP size++, fil\_sys contention is limiting IO
  - block contention, AHI, secondary index issues -- by design ;-)
- Read+Write :
  - all from RO + log\_sys, lock\_sys, trx\_sys -- work in progress..
  - trx commit, dblwrite, purge lag, lru depth
  - io capacity, dirty pages lwm, REDO size..
  - O\_DIRECT + AIO
- All :
  - innodb thread concurrency, spin wait delay
  - open table cache instances, sort size, etc..

# Test Case : The RW IO-bound “mystery”..

- Test Case :
  - Workload: OLTP\_RW 10Mx32-tab Uniform
  - CPU config : 16cores-HT / 32cores-HT
  - IO subsystem : EXT4 on F80
  - Users : 64
  - R/W ratio : 128, 16, 4, 1

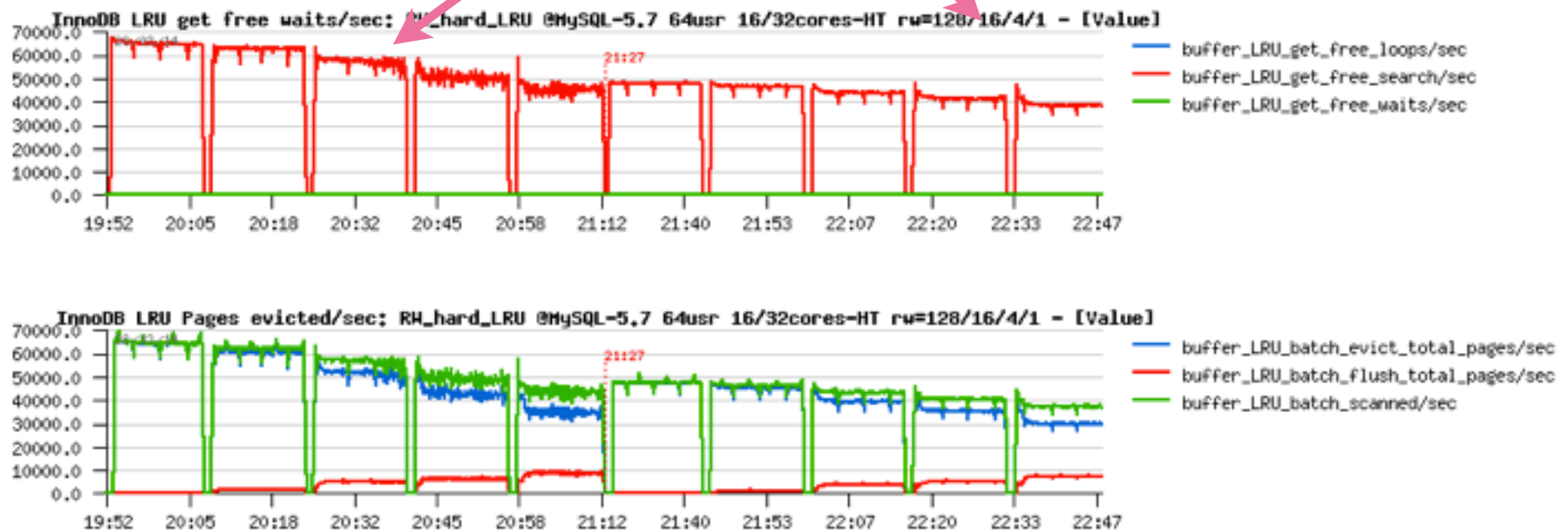
# The RW IO-bound “mystery”

- Focus on : I/O stats
- Engines: 5.7 latest, 16cores-HT / 32cores-HT



# The RW IO-bound “mystery”

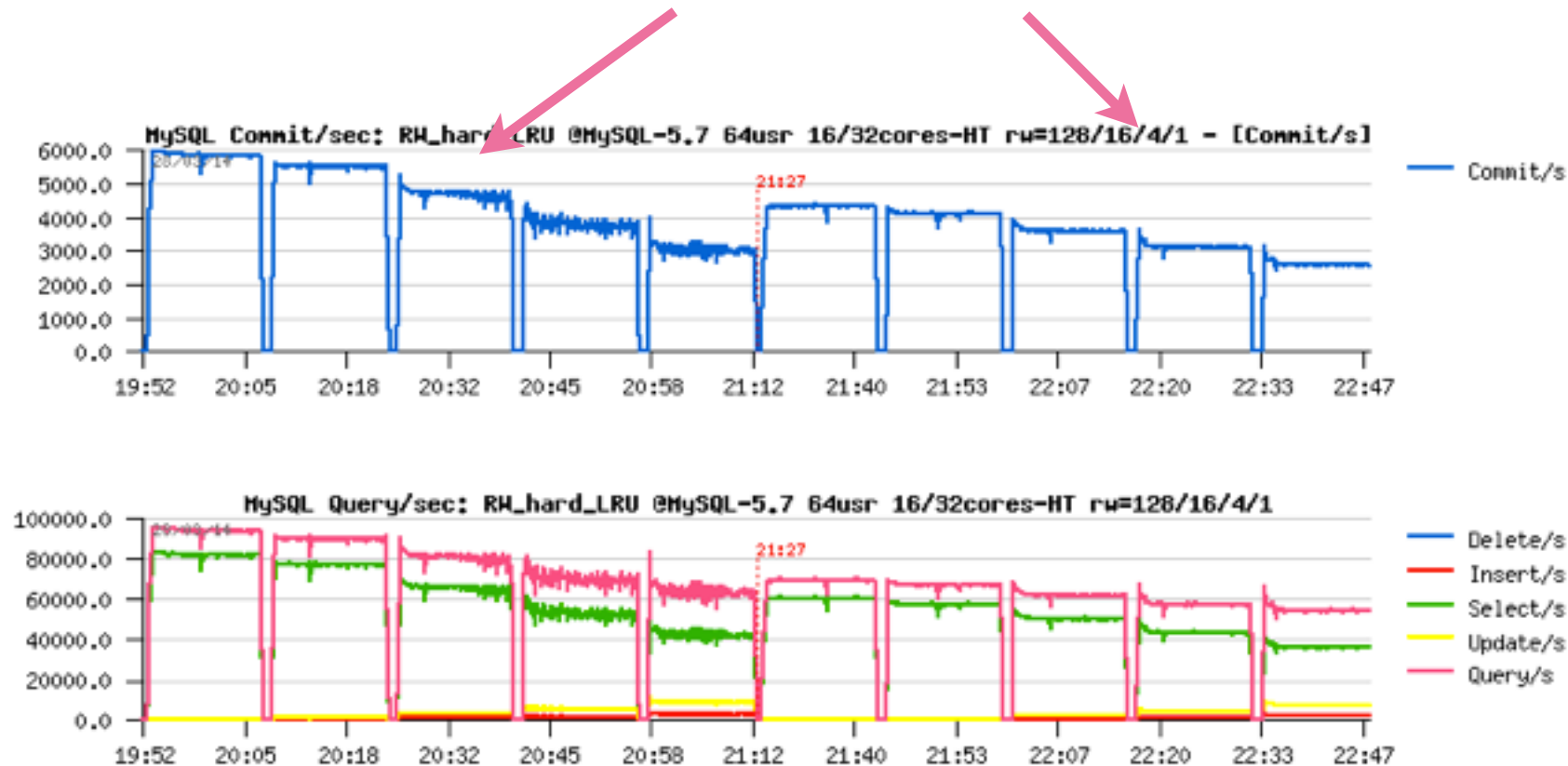
- Focus on : LRU stats
- Engines: 5.7 latest, 16cores-HT / 32cores-HT





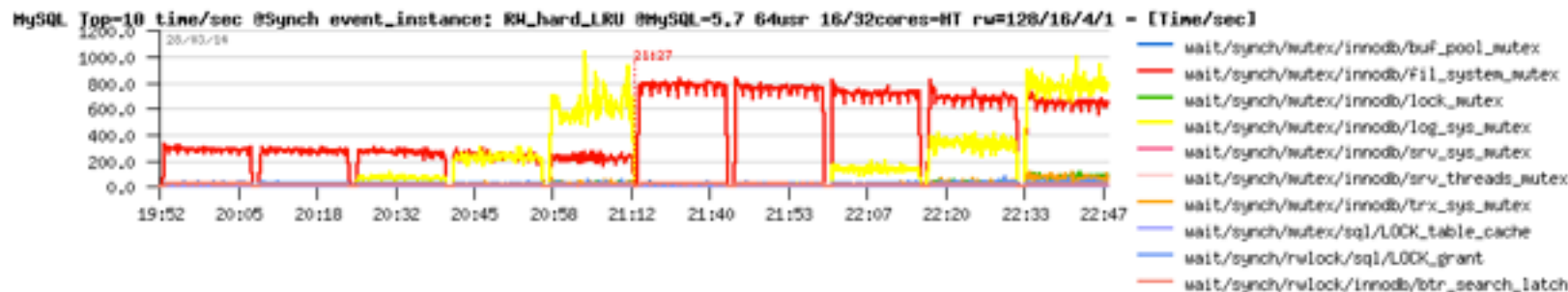
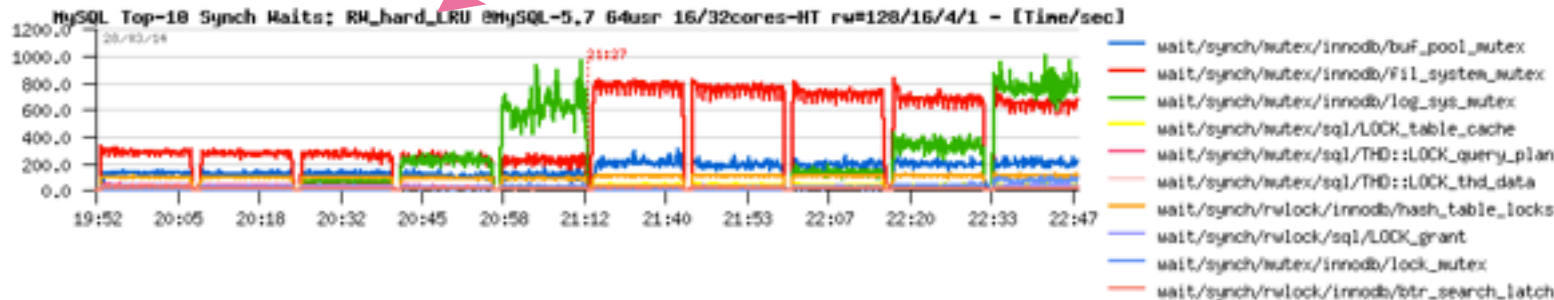
# The RW IO-bound “mystery”

- Focus on : TPS / QPS (note: x2 times worse at the end!!)
- Engines: 5.7 latest, 16cores-HT / 32cores-HT



# The RW IO-bound “mystery”

- Focus on : Lock contentions... (note: killing fil\_sys + log\_sys)
- Engines: 5.7 latest, 16cores-HT / 32cores-HT



# The RW IO-bound “mystery”

- Why not scaling?
  - InnoDB : killing fil\_sys + log\_sys
  - I/O : kernel contention !!!



```

17.80%      mysqld [kernel.kallsyms]      [k] _spin_lock_irq
|
|--- _spin_lock_irq
|
|---51.26%-- scsi_request_fn
|
|      |--89.93%-- __generic_unplug_device
|      |
|      |      |--65.42%-- __make_request
|      |      |generic_make_request
|      |      |submit_bio
|      |      |
|      |      |--99.83%-- dio_bio_submit
|      |      |__blockdev_direct_IO
|      |      |
|      |      |--97.28%-- ext4_ind_direct_IO
|      |      |ext4_direct_IO
|      |      |generic_file_aio_read
    
```

**So, work continues..  
stay tuned... ;-)**

## Few words about dim\_STAT (if you're asking ;-))

- All graphs are built with dim\_STAT (<http://dimitrik.free.fr>)
  - All System load stats (CPU, I/O, Network, RAM, Processes,...)
  - Manly for Solaris & Linux, but any other UNIX too :-)
  - Add-Ons for Oracle, MySQL, PostgreSQL, Java, etc.
  - 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! :-)

# THANK YOU !!!

- All details about presented materials you may find on:
  - <http://dimitrik.free.fr> - dim\_STAT, dbSTRESS, Benchmark Reports, etc.
  - <http://dimitrik.free.fr/blog> - Articles about MySQL Performance, etc.