MySQL 5.7 Performance: Scalability & Benchmarks

Dimitri KRAVTCHUK
MySQL Performance Architect @Oracle

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.

Are you Dimitri?.. ;-)

- Yes, it's me :-)
- Hello from Paris! ;-)
- Passionated by Systems and Databases Performance
- Previous 15 years @Sun Benchmark Center
- Started working on MySQL Performance since v3.23
- But during all that time just for “fun” only ;-)
- Since 2011 “officially” @MySQL Performance full time now
- http://dimitrik.free.fr/blog / @dimitrik_fr

Agenda

- Overview of MySQL Performance
- Workload oriented tuning and MySQL Internals
- Performance improvements in MySQL 5.7 & Benchmark results
- Pending issues..
- Q & A

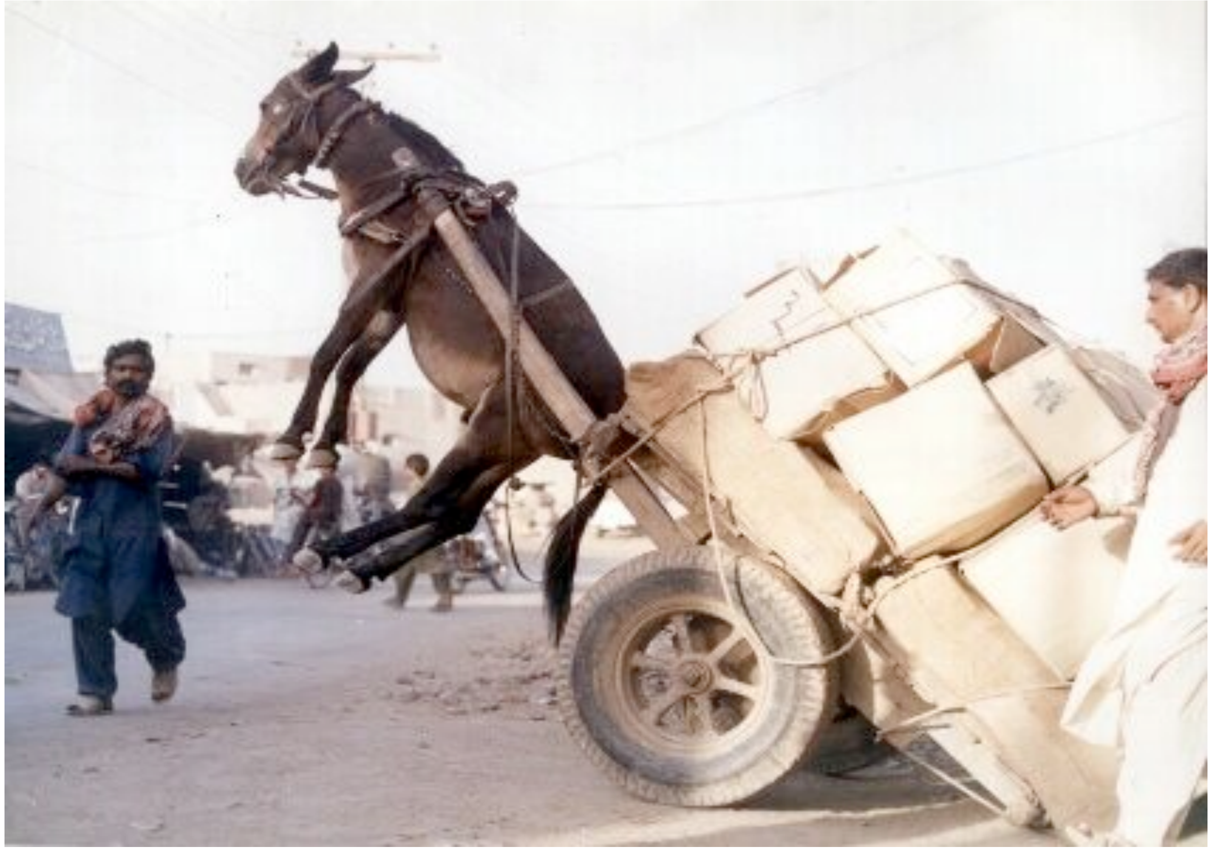
Why MySQL Performance ?...

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!... :-)

ORACLE

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... ???..

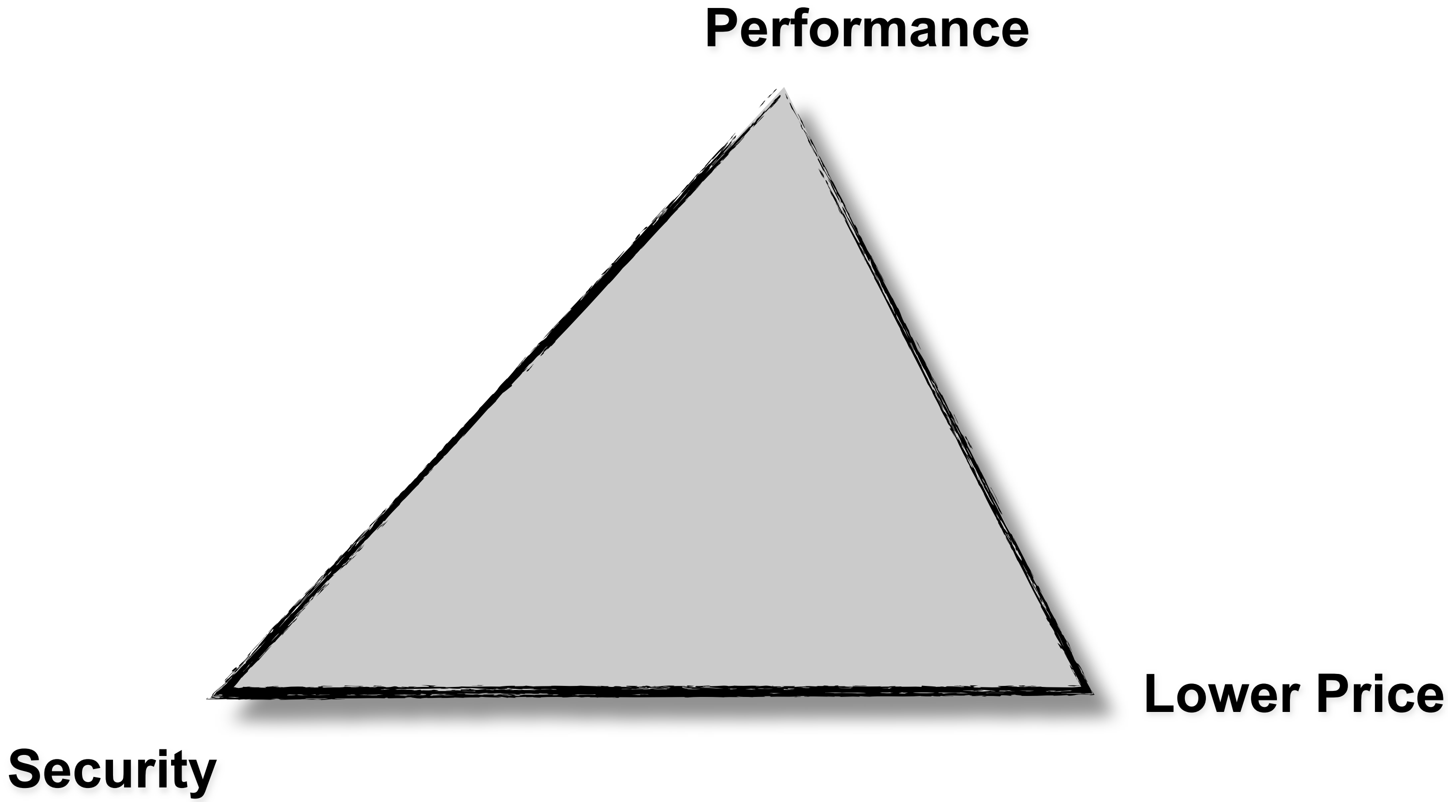
USE YOUR BRAIN !!!... ;-)

THE MAIN SLIDE! ;-))

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

Seek for your best option..



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

- So, benchmark! ;-) -- And start with a clear goal!
  - Otherwise: I've obtained all these results, and now... so what?..
- 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 ;-)

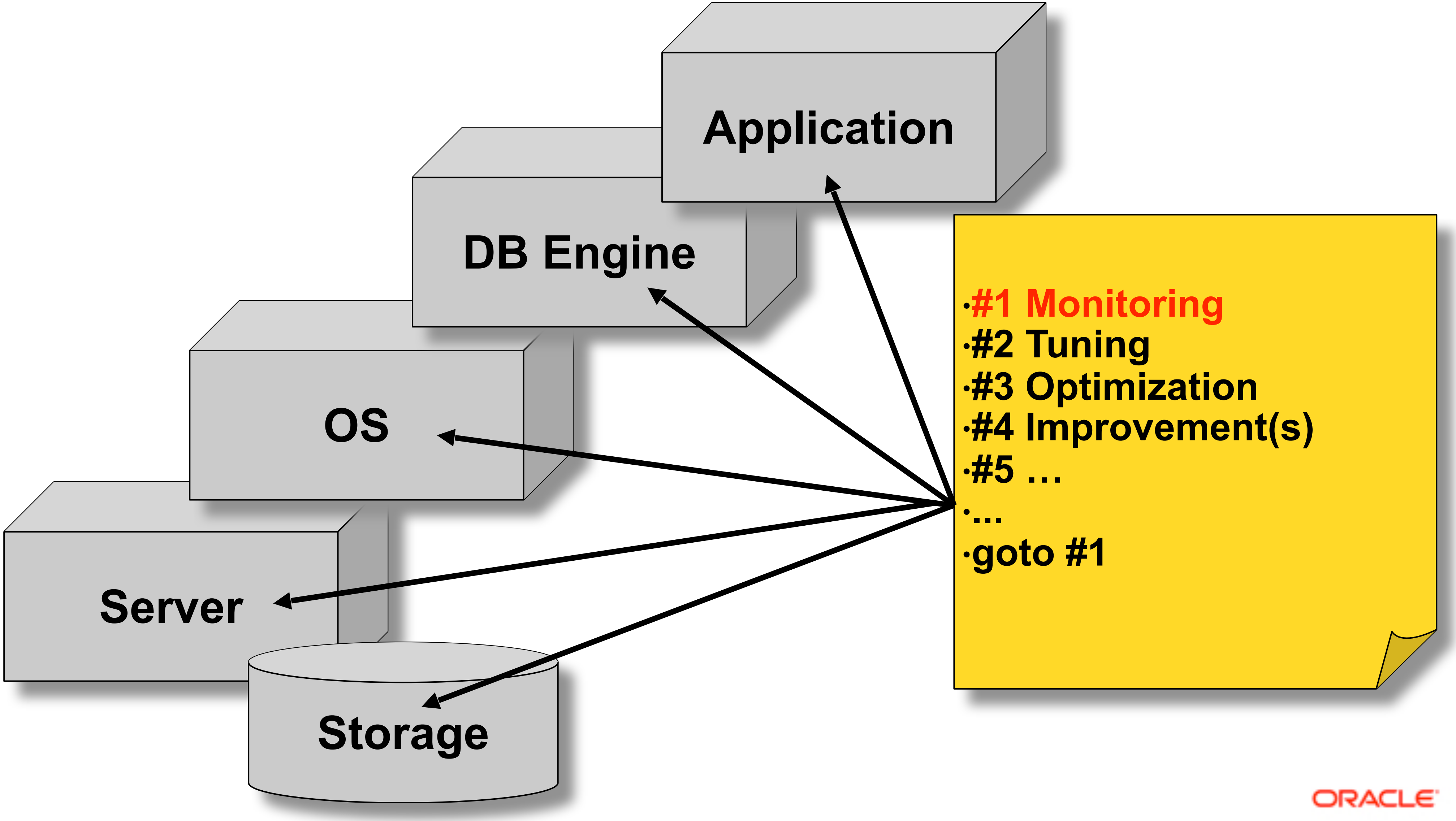
Test Workload

- Before to do 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..

“Generic” Test Workloads @MySQL

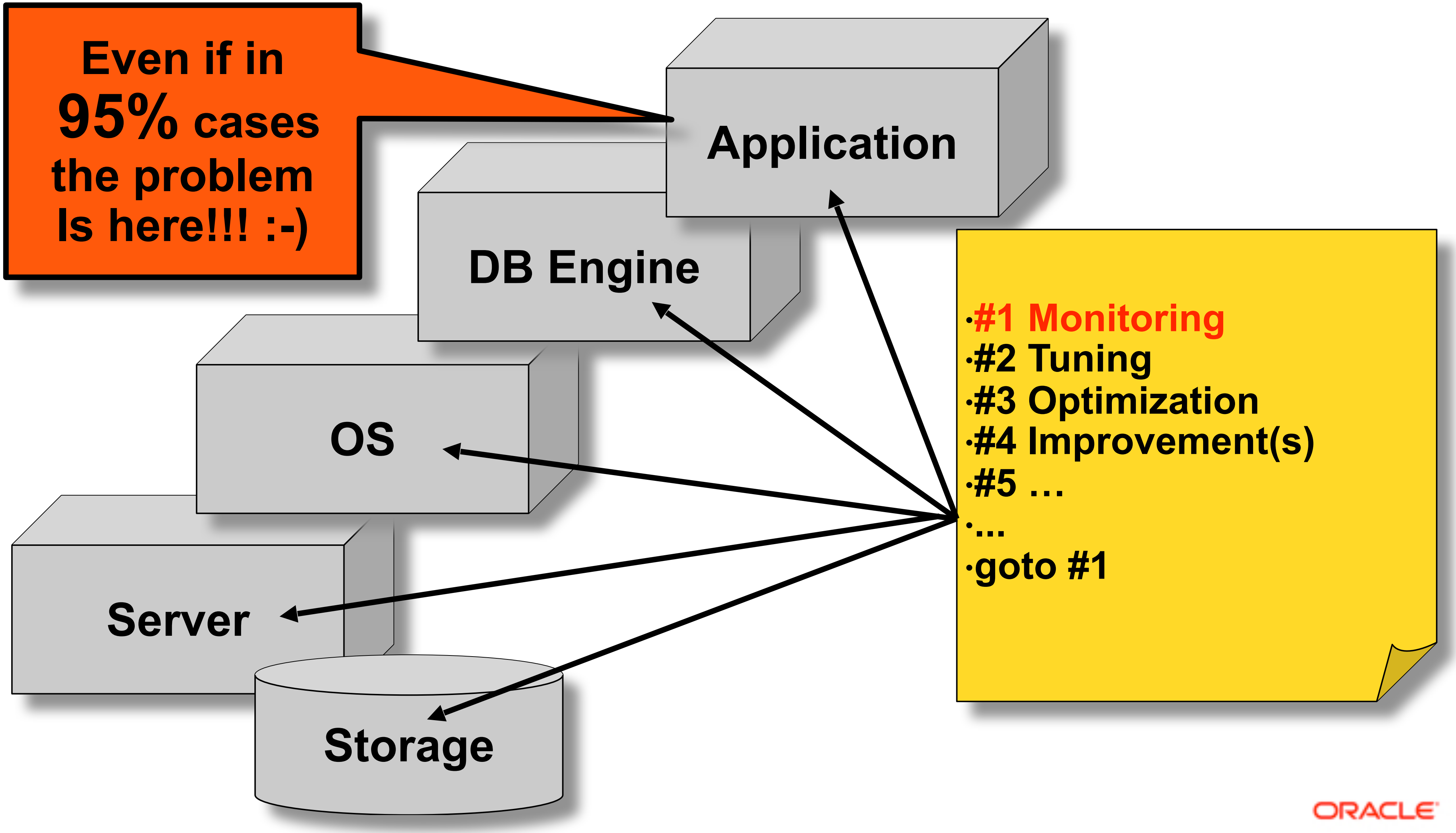
- Sysbench
  - OLTP, RO/RW, 1-table, since v0.5 N-table(s), lots load options, deadlocks
- DBT2 / TPCC-like
  - OLTP, RW, very complex, growing db, no options, deadlocks
  - In fact using mostly only 2 tables! (thanks Performance Schema ;-))
- dbSTRESS
  - OLTP, RO/RW, several tables, one most hot, configurable,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

The Infinitive Loop of Database Tuning...



- #1 Monitoring
- #2 Tuning
- #3 Optimization
- #4 Improvement(s)
- #5 …
- …
- goto #1

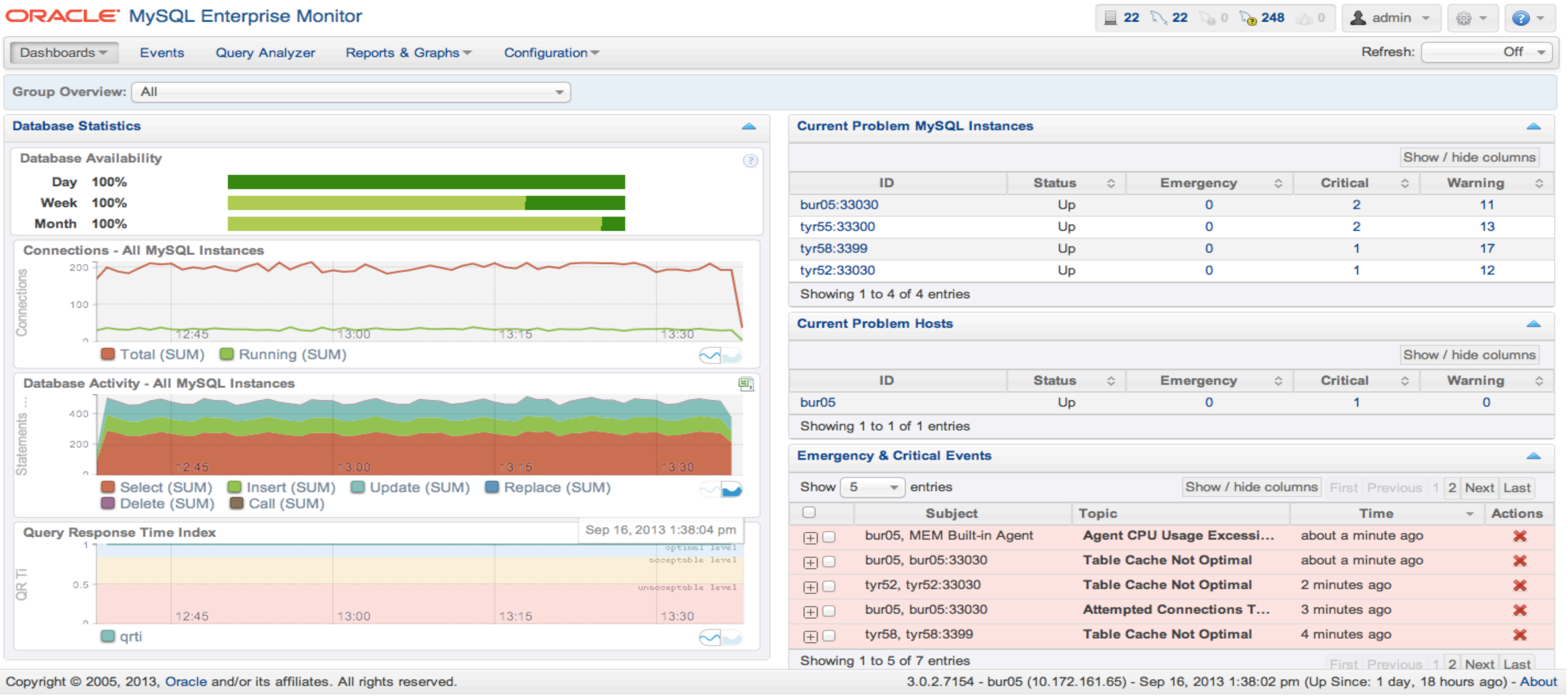
The Infinitive Loop of Database Tuning...



- #1 Monitoring
- #2 Tuning
- #3 Optimization
- #4 Improvement(s)
- #5 …
- …
- goto #1

MySQL Enterprise Monitor

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



ORACLE MySQL Enterprise Monitor

22 22 0 248 0 admin

Dashboards Events Query Analyzer Reports & Graphs Configuration

Refresh: Off

Group Overview: All

Database Statistics

Database Availability

Day	100%	
Week	100%	
Month	100%	

Connections - All MySQL Instances

Database Activity - All MySQL Instances

Query Response Time Index

Sep 16, 2013 1:38:04 pm

Current Problem MySQL Instances

Show / hide columns

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

Showing 1 to 4 of 4 entries

Current Problem Hosts

Show / hide columns

ID	Status	Emergency	Critical	Warning
bur05	Up	0	1	0

Showing 1 to 1 of 1 entries

Emergency & Critical Events

Show 5 entries

Show / hide columns First Previous 1 2 Next Last

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

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

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

3.0.2.7154 - bur05 (10.172.161.65) - Sep 16, 2013 1:38:02 pm (Up Since: 1 day, 18 hours ago) - About

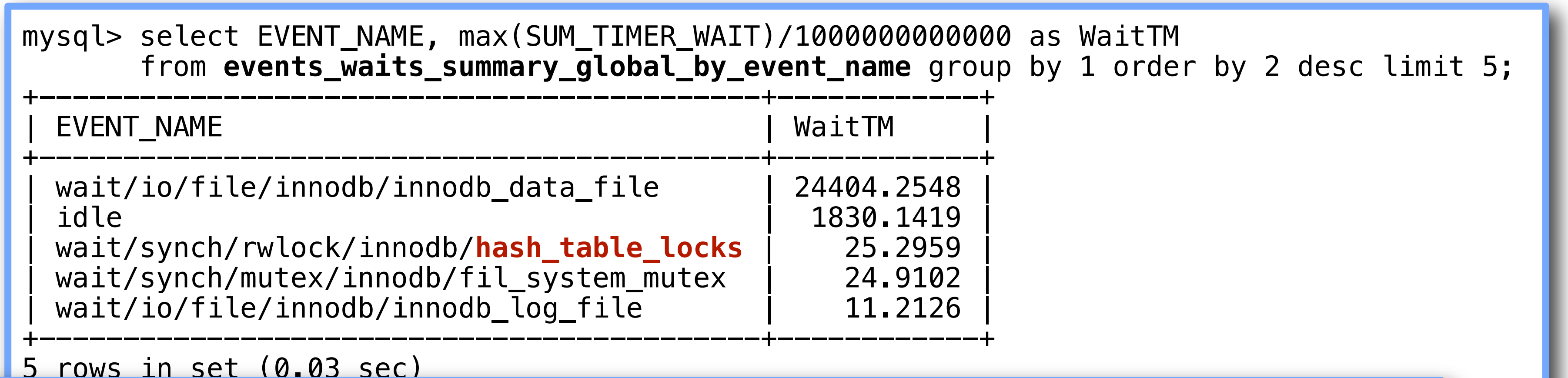
Other Monitoring Tools

- Cacti
- Zabbix
- Nagios
- Etc...

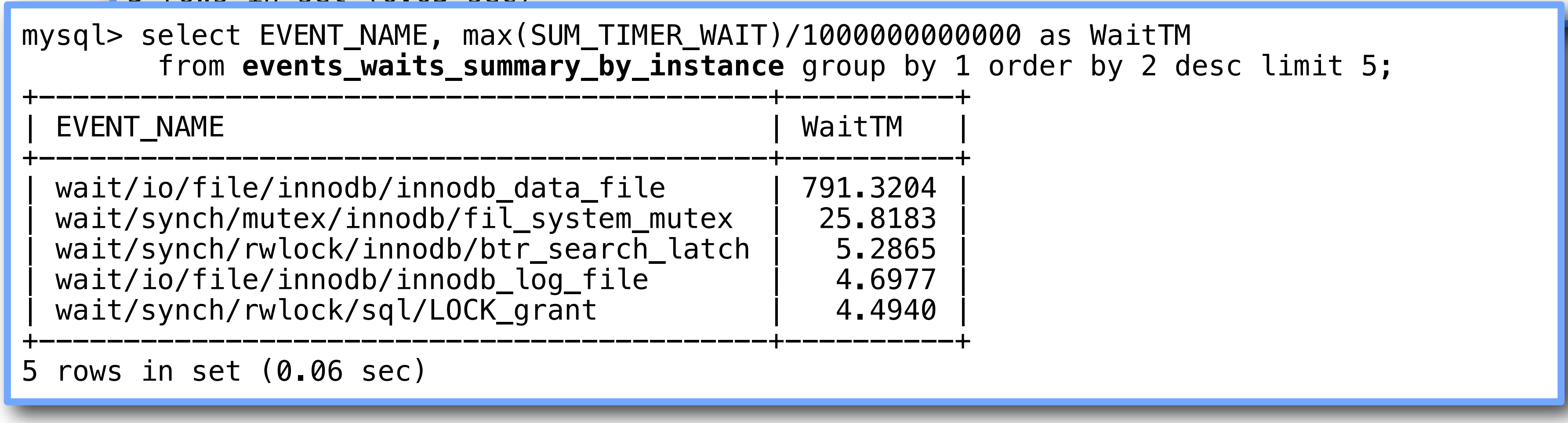
- dim_STAT
  - well, I'm using this one, sorry ;-)
  - all graphs within presentation were made with it
  - details are in the end of presentation..

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/fil_system_mutex  |    24.9102 |
| wait/io/file/innodb/innodb_log_file       |    11.2126 |
+-------------------------------------------+------------+
5 rows in set (0.03 sec)
```



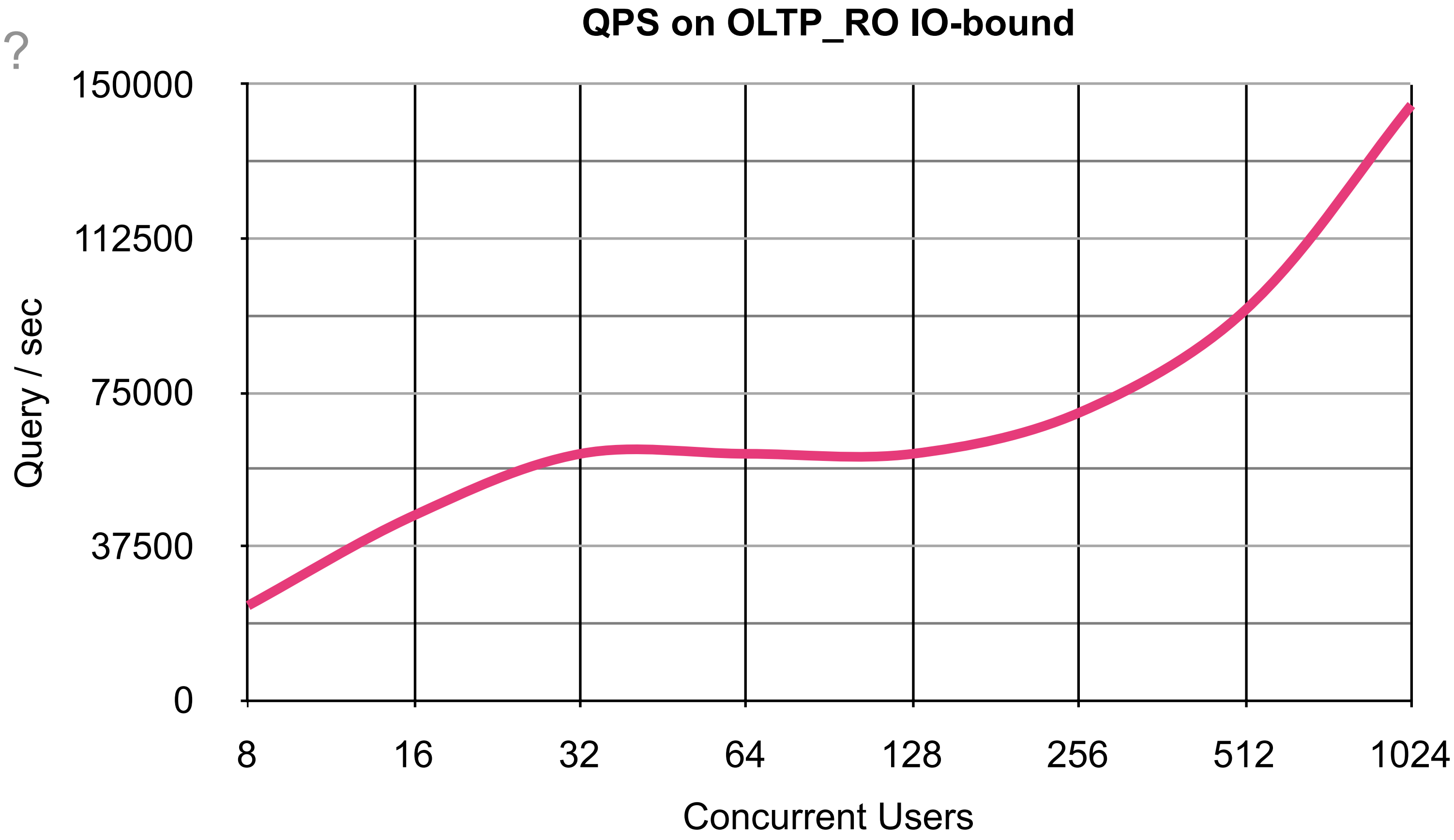
```
mysql> select EVENT_NAME, max(SUM_TIMER_WAIT)/1000000000000 as WaitTM
        from events_waits_summary_by_instance group by 1 order by 2 desc limit 5;
+-------------------------------------------+----------+
| EVENT_NAME                                | WaitTM   |
+-------------------------------------------+----------+
| wait/io/file/innodb/innodb_data_file      | 791.3204 |
| wait/synch/mutex/innodb/fil_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)
```

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..
  - use the best adapted Storage Engine
- 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 ;-))

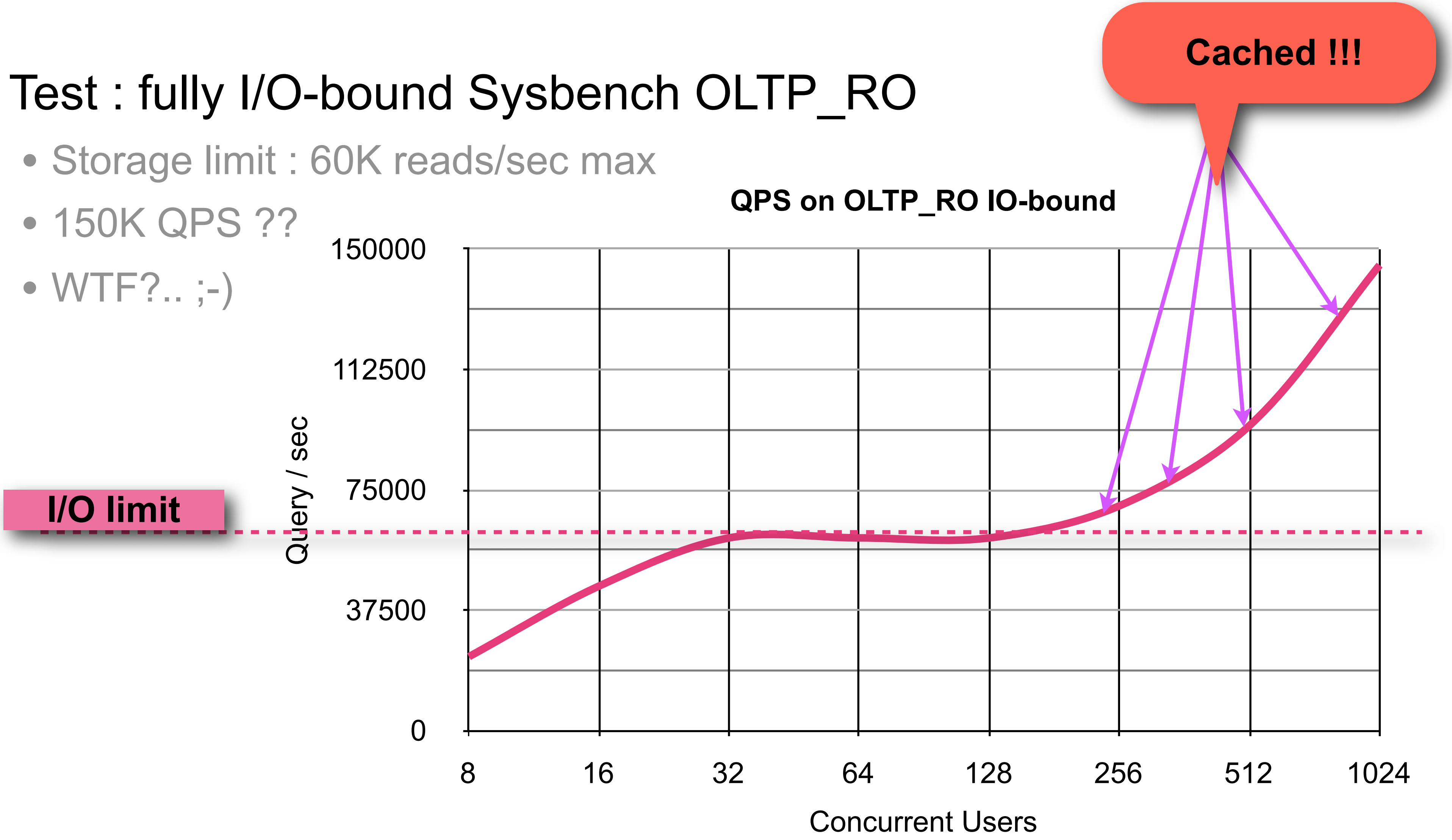
Let’s analyze the following benchmark result..

- Test : fully IO-bound OLTP_RO
  - Storage limit : 60K reads/sec max
  - 150K QPS ??
  - WTF?.. ;-)

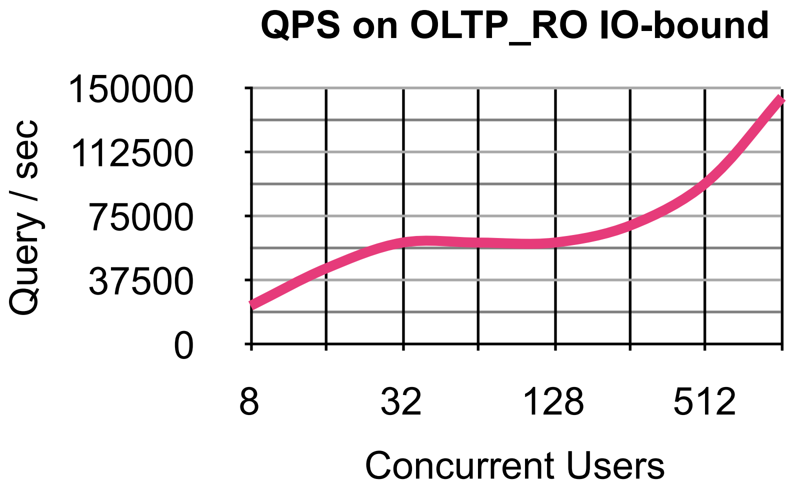


Let’s analyze the following benchmark result..

- Test : fully I/O-bound Sysbench OLTP_RO
  - Storage limit : 60K reads/sec max
  - 150K QPS ??
  - WTF?.. ;-)



Let’s analyze the following benchmark result..



- Test : fully IO-bound OLTP_RO
  - Storage limit : 60K reads/sec max
  - 150K QPS ??
  - WTF?.. ;-)
- The issue:
  - the random ID for a row acces is not that random as expected..
  - and with a higher workload the probability to get the same “random” row ID on the same time and by different threads only increasing..
  - workaround : for some of the tests started to use as many Sysbench processes as user threads (1 connection = 1 sysbench process)..

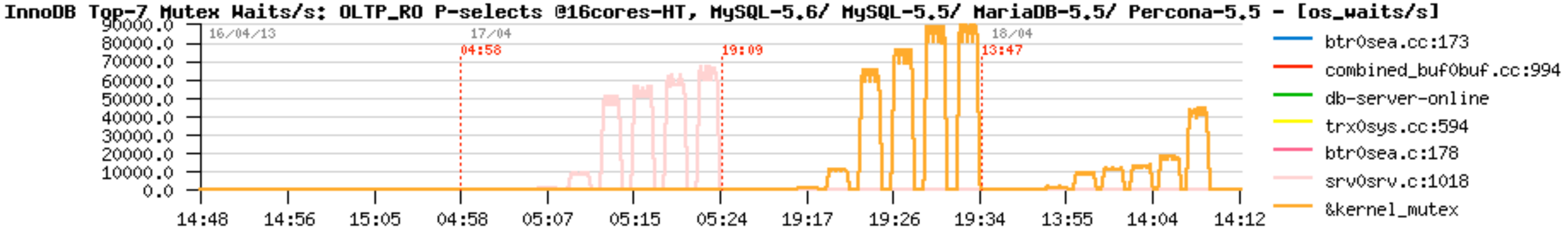
Analyzing Workloads...

- 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 !

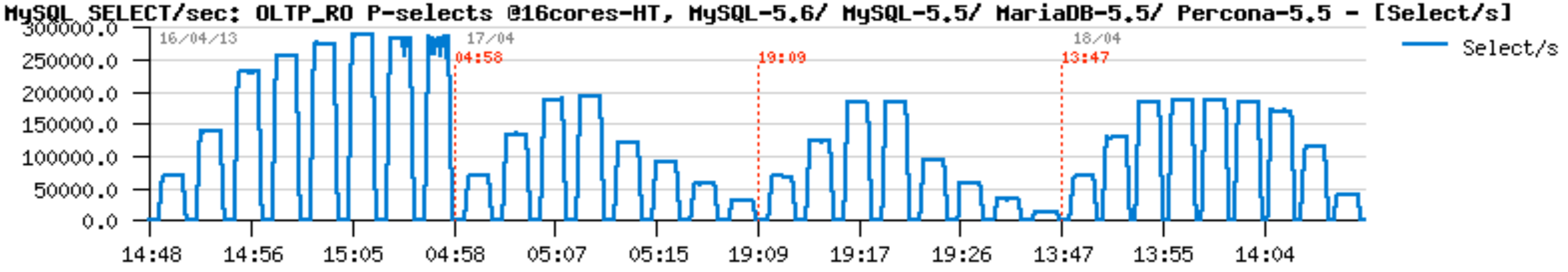
Workloads : Read-Only In-Memory

- Generally CPU / RAM bound + internal contentions ;-)
- 5.5 :
  - kernel_mutex
  - LOCK_open
  - + many other remane 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 spin lock delay
  - Adaptive hash index (AHI) : still unclear..
  - Memcached plugin

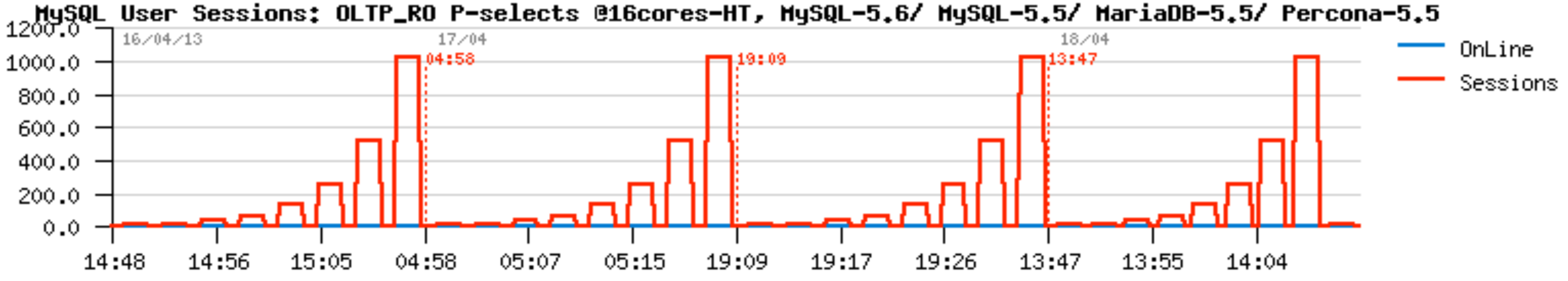
InnoDB: Read-Only Transactions in 5.6 (Apr.2013)



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]



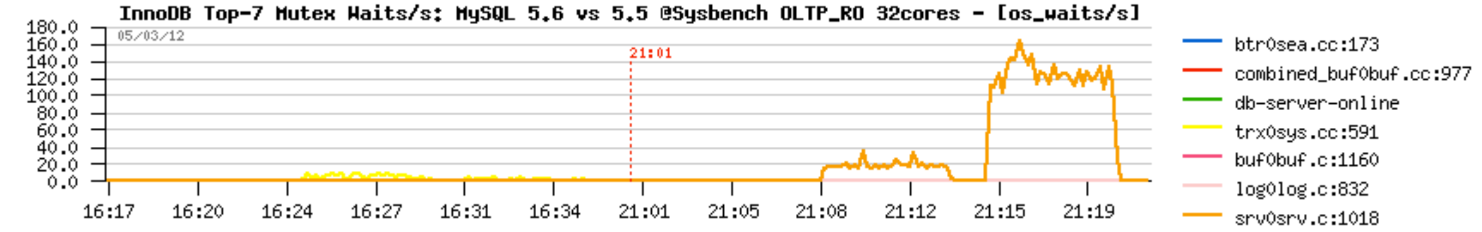
MySQL SELECT/sec: OLTP_RO P-selects @16cores-HT, MySQL-5.6/ MySQL-5.5/ MariaDB-5.5/ Percona-5.5 - [Select/s]

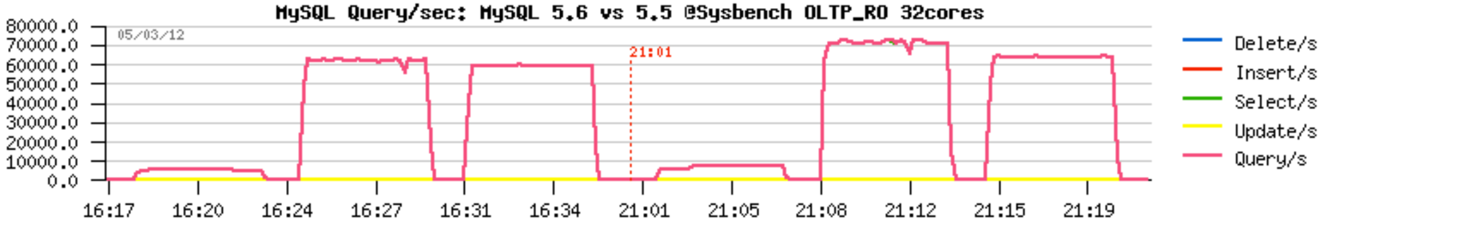


MySQL User Sessions: OLTP_RO P-selects @16cores-HT, MySQL-5.6/ MySQL-5.5/ MariaDB-5.5/ Percona-5.5

InnoDB : false sharing of cache-line = true killer

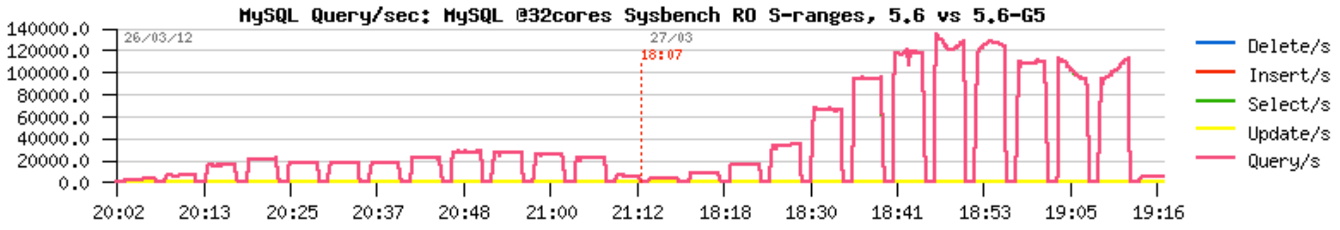
- RO or RW Workloads
  - Same symptoms in 5.5 & 5.6 : no QPS improvement between 16 and 32 user sessions:

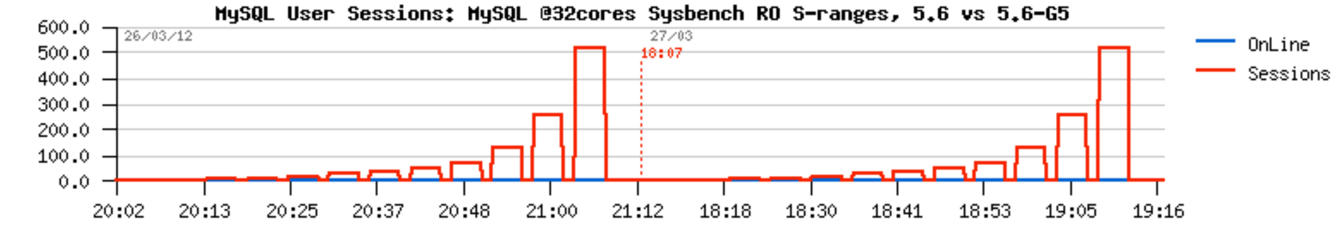




InnoDB : false sharing of cache-line fixed!

- 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..



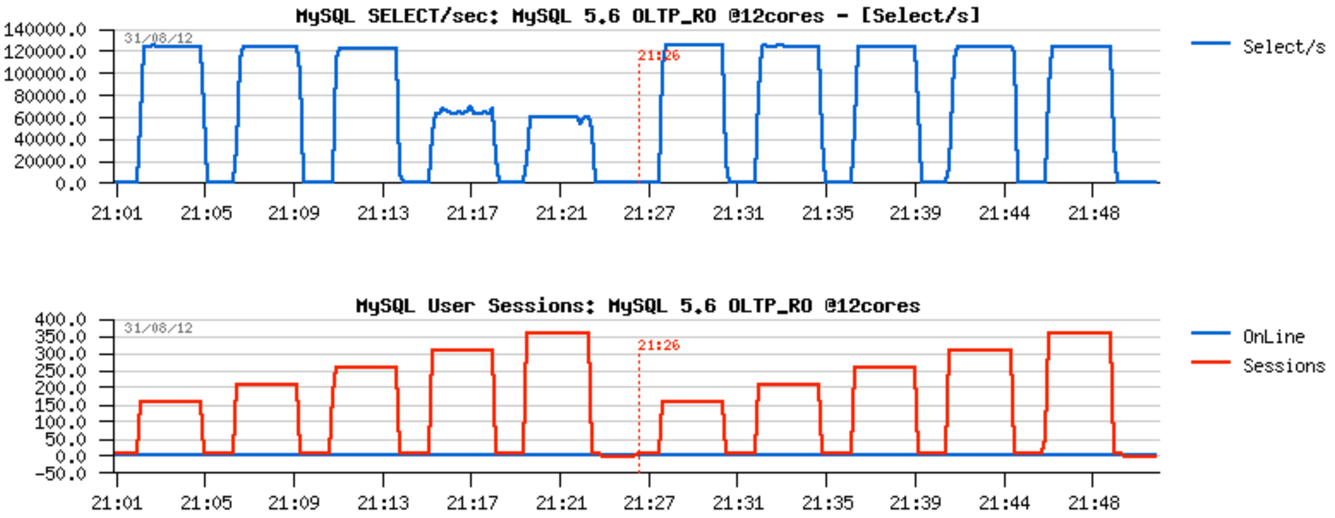


MySQL Internals: “killer” LOCK_open mutex

- 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

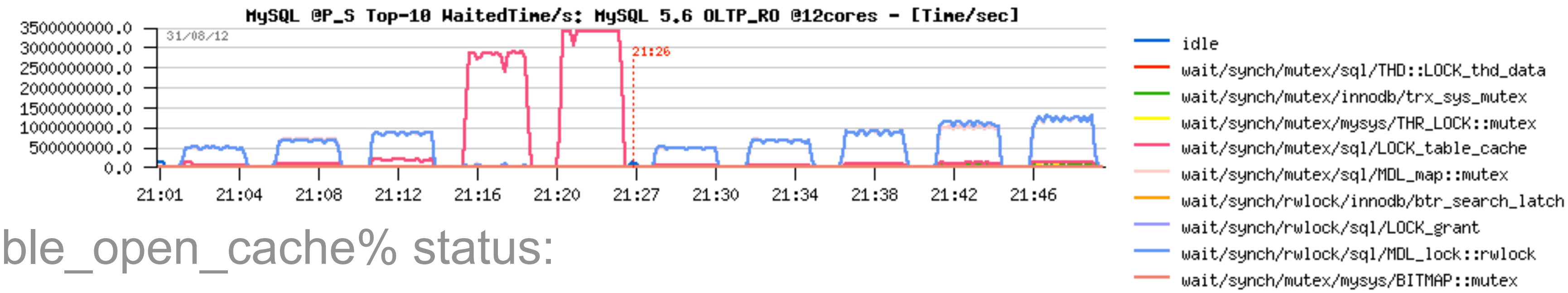
MySQL 5.6 Internals : low table_open_cache

- MySQL 5.6 :
  - Not big enough “table_open_cache” setting

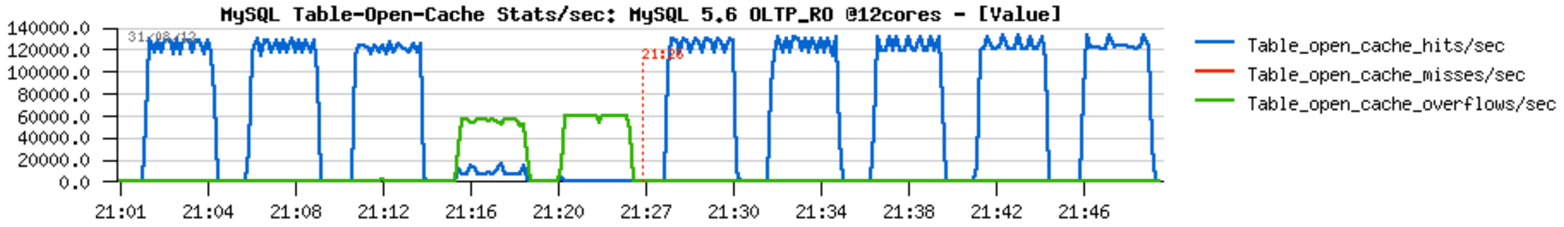


MySQL 5.6 Internals : low table_open_cache (2)

- MySQL 5.6 :
  - Not big enough “table_open_cache” setting
  - PFS Waits monitoring: LOCK_table_cache become the most hot:

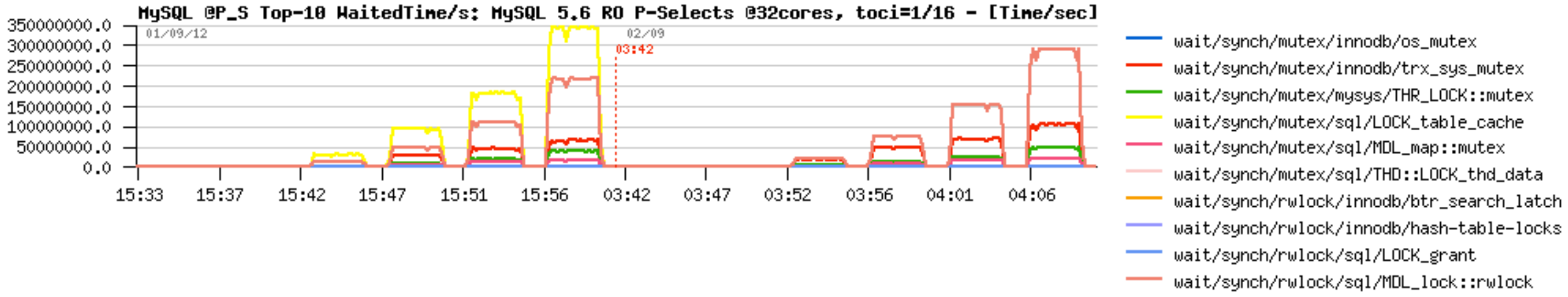


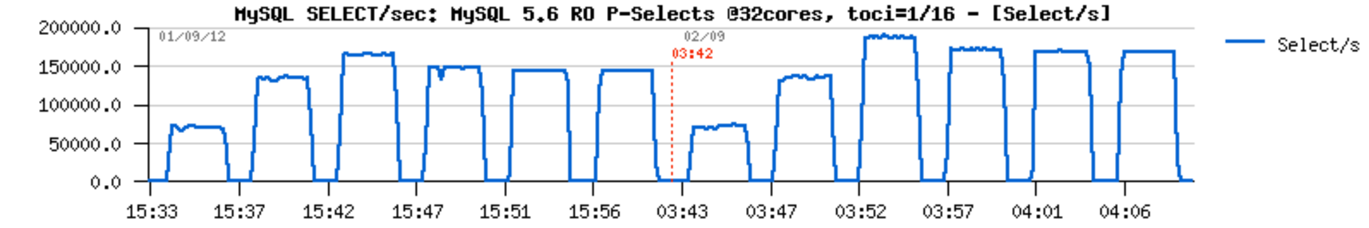
  - Table_open_cache% status:

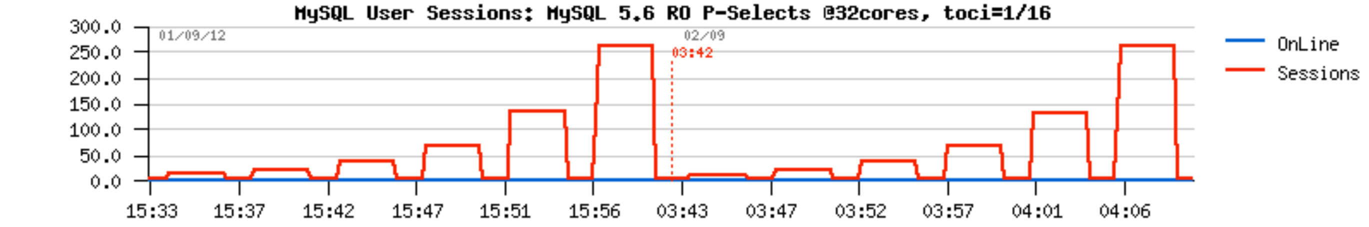


MySQL 5.6 Internals : table_open_cache_instances

- MySQL 5.6 :
  - When LOCK_table_cache wait is on top, the gain is usually well visible:





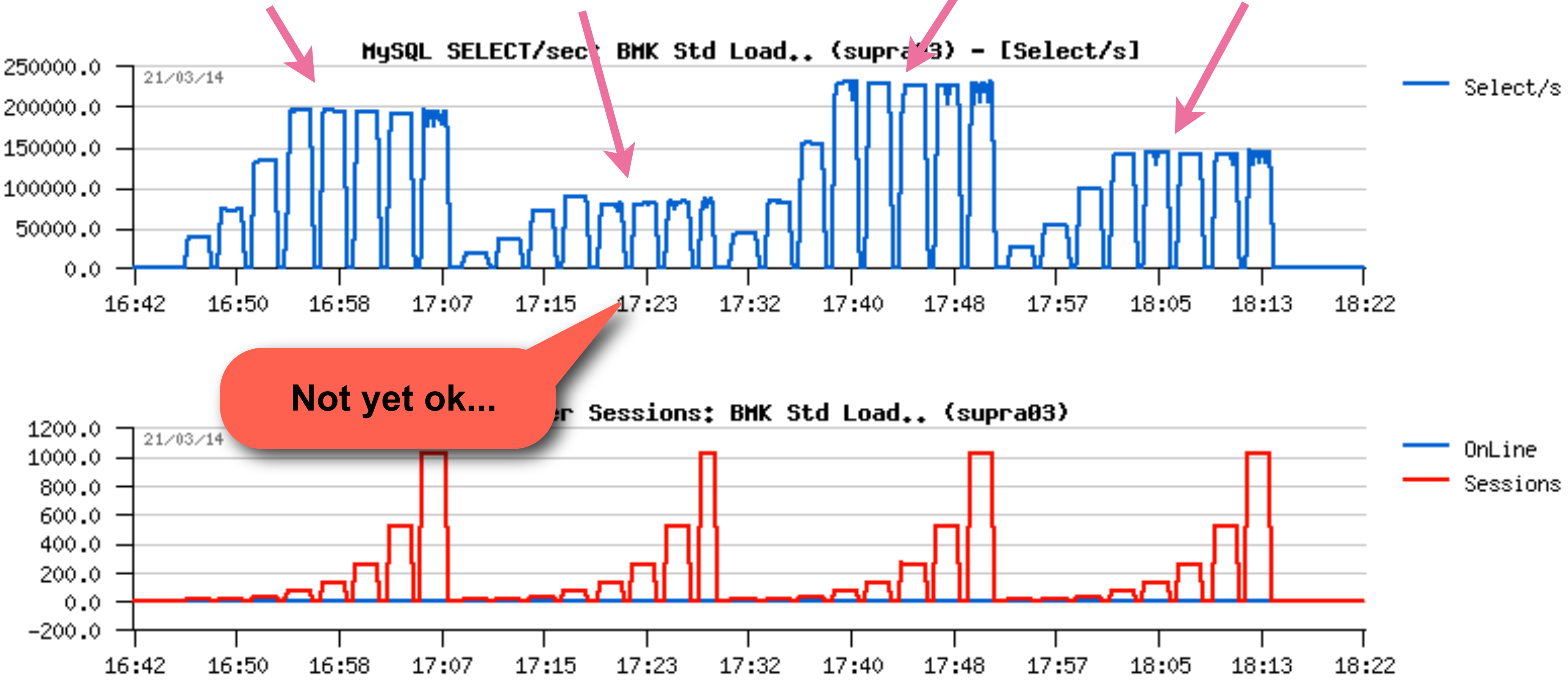


Workloads : Read-Only In-Memory @MySQL 5.7

- 5.7 :
  - trx_sys : redesigned TRX list! (yet better than RO transactions)
    - made MDL very hot !
  - MDL : lock free since DMR4 !!
    - made THR_lock very hot!! fix in pipe ;-)
  - 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

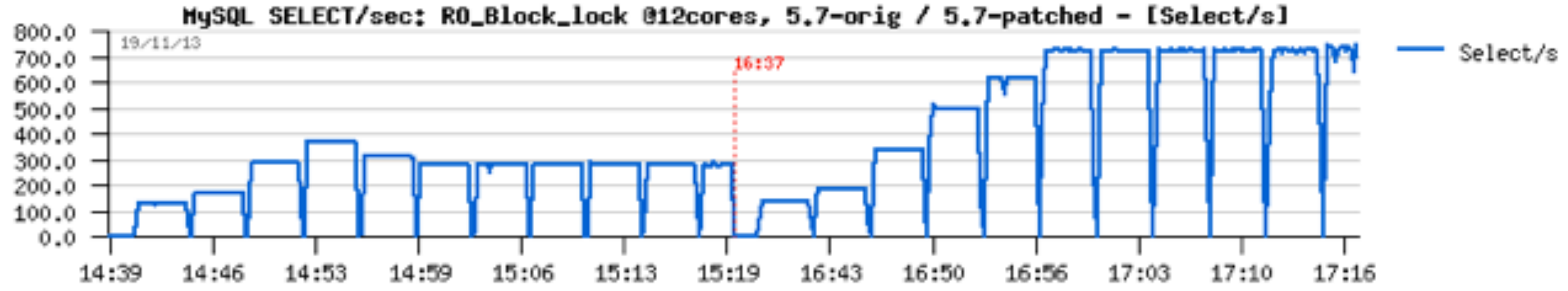
- Simple ranges, Distinct ranges, SUM ranges, Ordered ranges

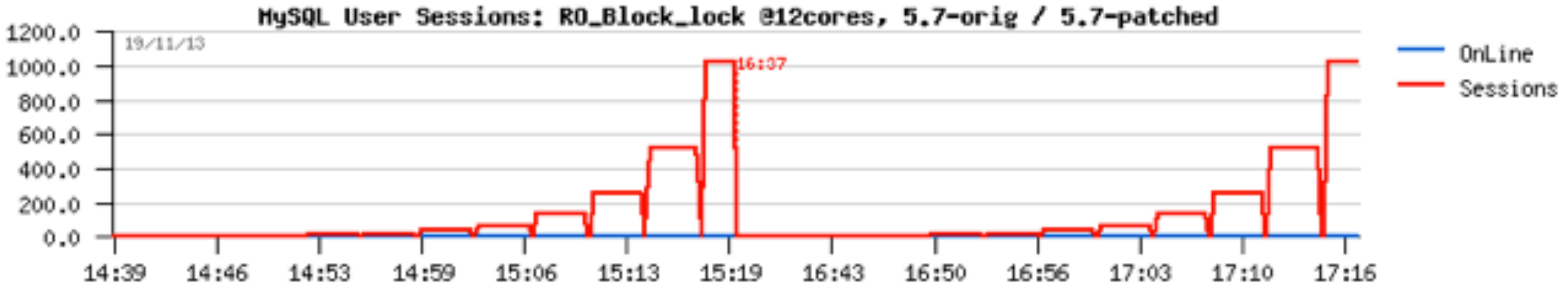


Not yet ok...

InnoDB block lock contentions...

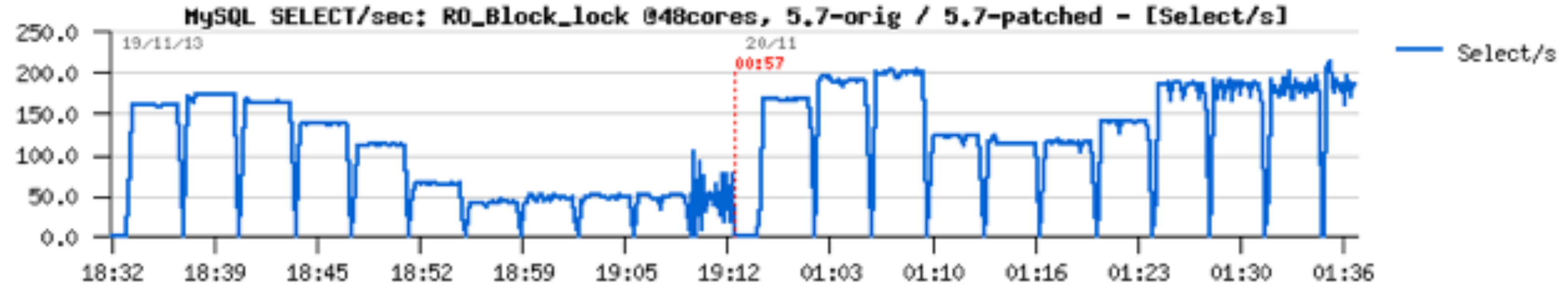
- Being here from a long long time (by design)..
- Improved in 2013, but not yet fully fixed..
- Can be seen as :

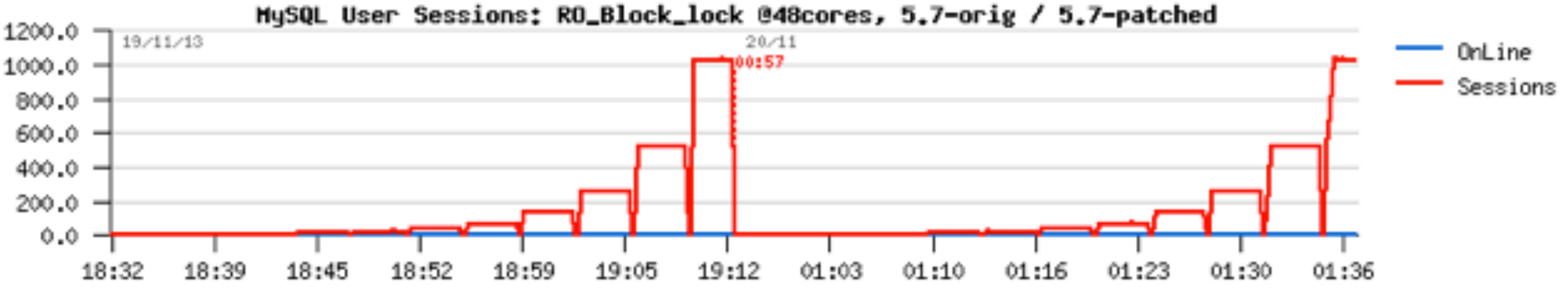




InnoDB block lock contentions... (cont.)

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



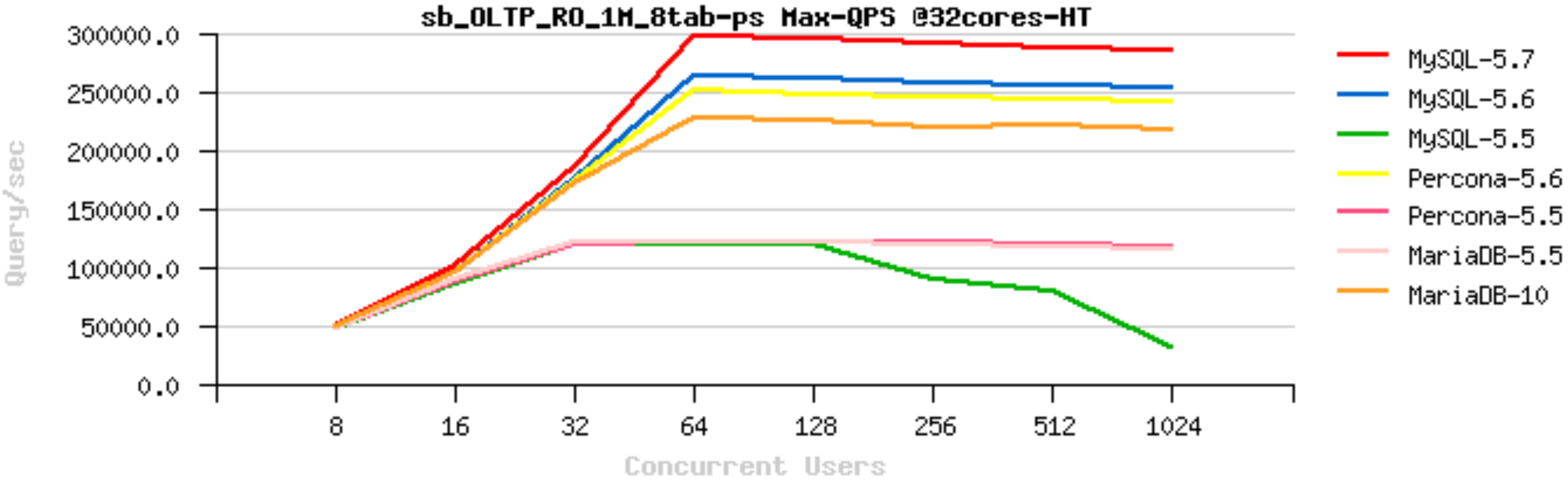


InnoDB block lock contentions... (cont.)

- 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)..

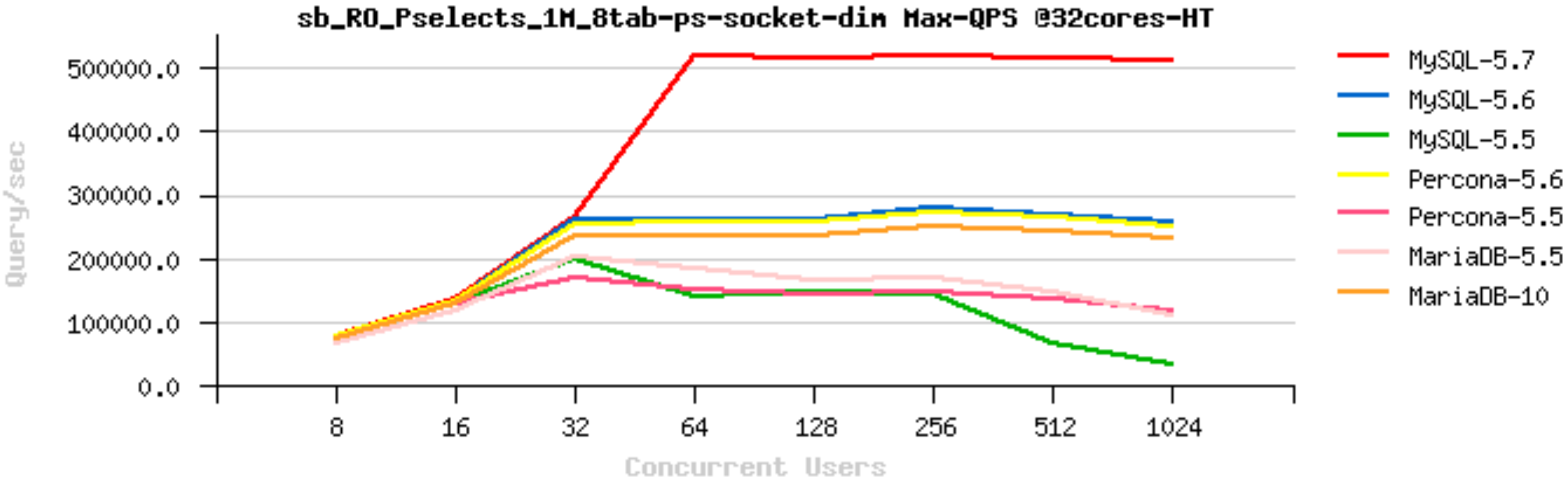
RO In-Memory @MySQL 5.7

- Sysbench OLTP_RO 8-tables, 32cores-HT :



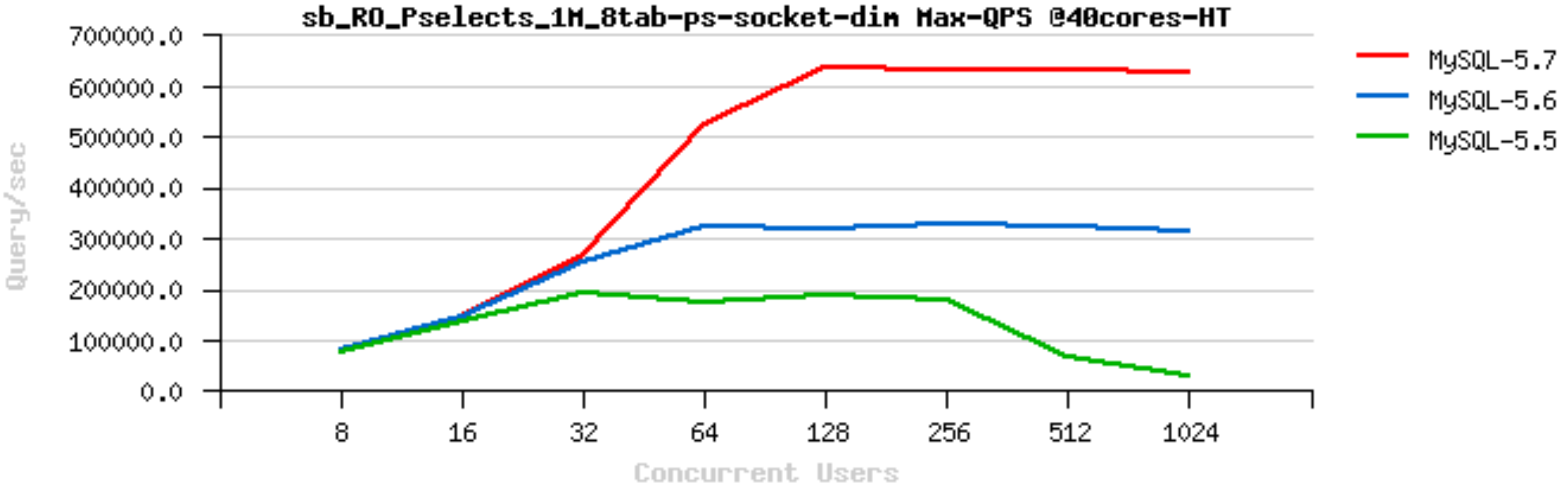
RO In-Memory @MySQL 5.7

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



RO In-Memory @MySQL 5.7

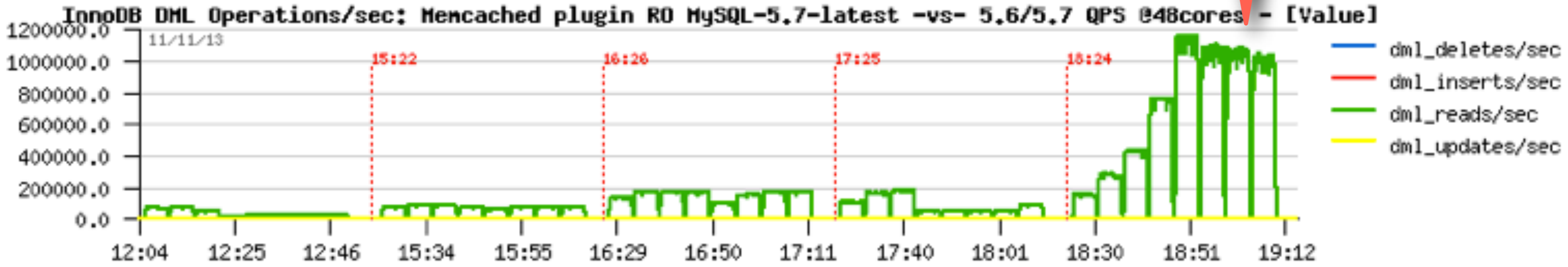
- **635K QPS** Sysbench Point-Selects 8-tab, 40cores-HT :

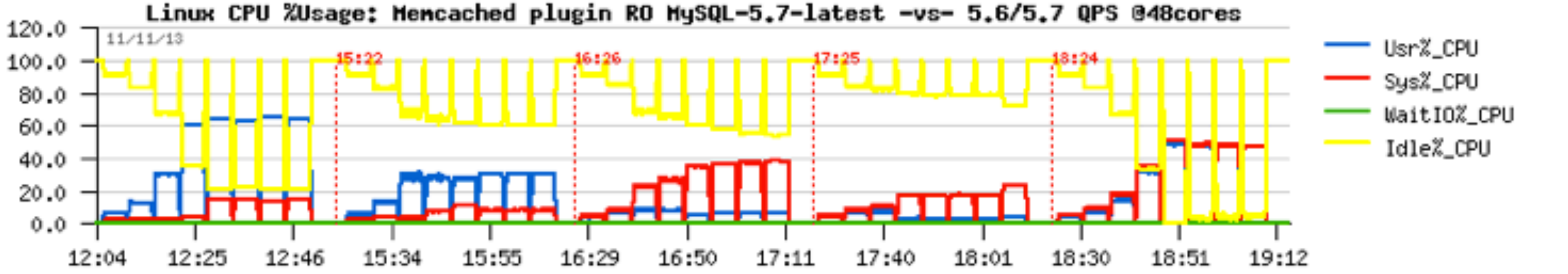


InnoDB Memcached @MySQL 5.7

- **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..
  - SDD : 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

Read+Write Workloads : In-Memory

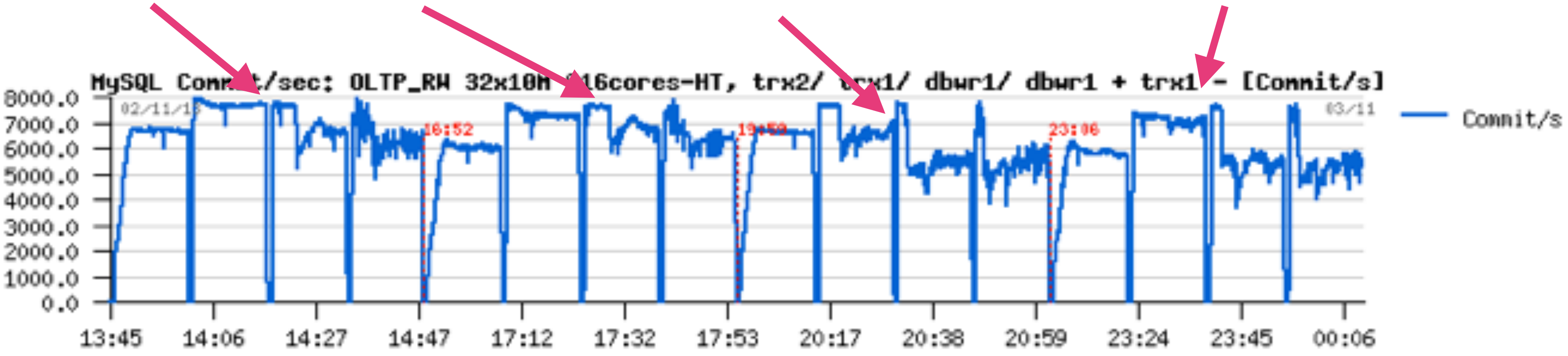
- Main points :
  - Processing itself / Data Safety
  - Internal contentions / Design limitations
  - Flushing / Checkpoint
  - Purge

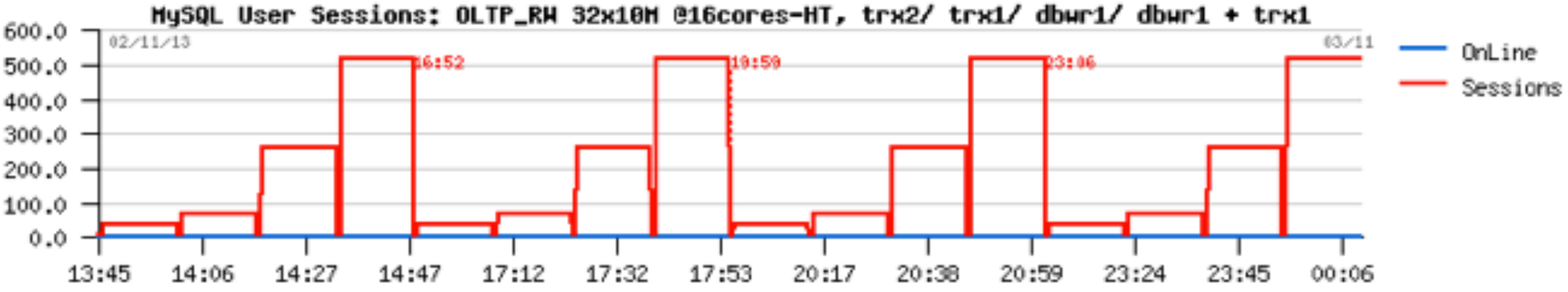
Read+Write Workloads : In-Memory

- Processing itself
  - 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, low on “good” storage
  - InnoDB double write buffer : **KILLER** ! overhead + bottleneck..
    - need a fix / re-design / etc. in urgency ;-)
    - Fusion-io atomic writes is one of (**true** support in MySQL 5.7)
    - Facebook solution is very attractive too
    - but a true re-design is still preferable ;-)

Impact of “safety” options..

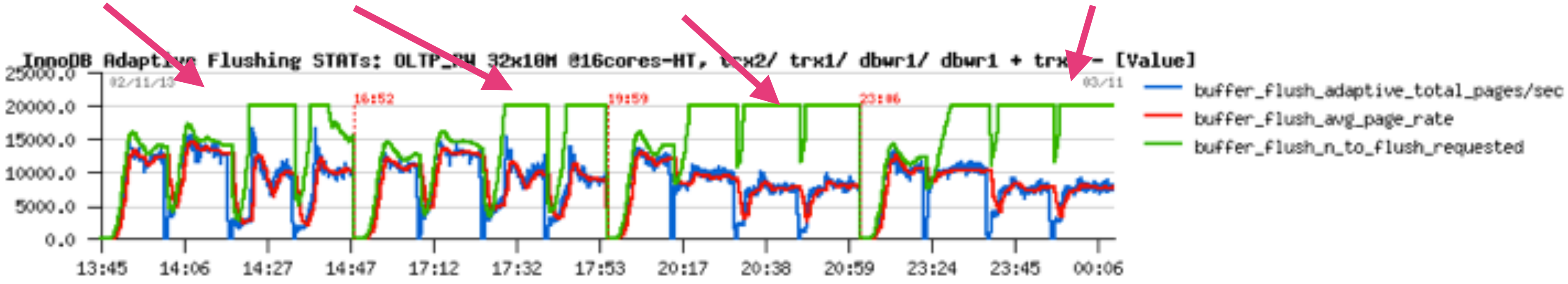
- OLTP_RW 32x10M-tables @Percona-5.6
  - (trx=2)(trx=1 + chksum=1)(dblwr=1)(trx=1 + chksum=1 + dblwr=1)

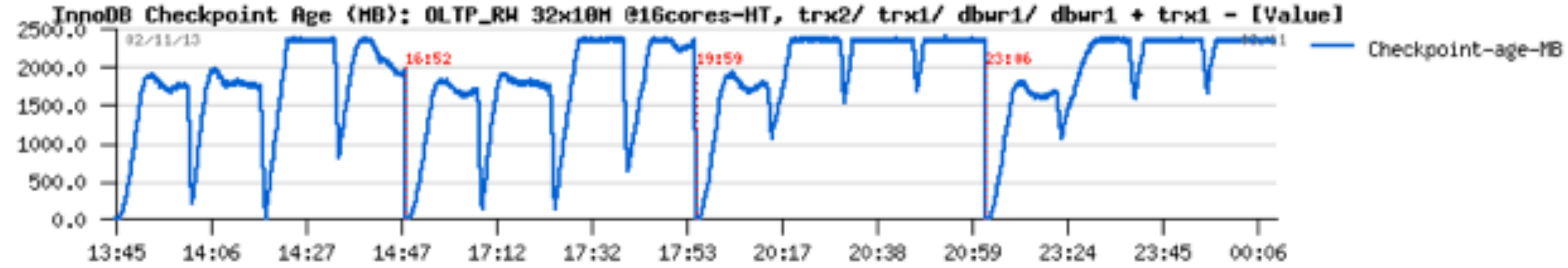




Impact of “safety” options..

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



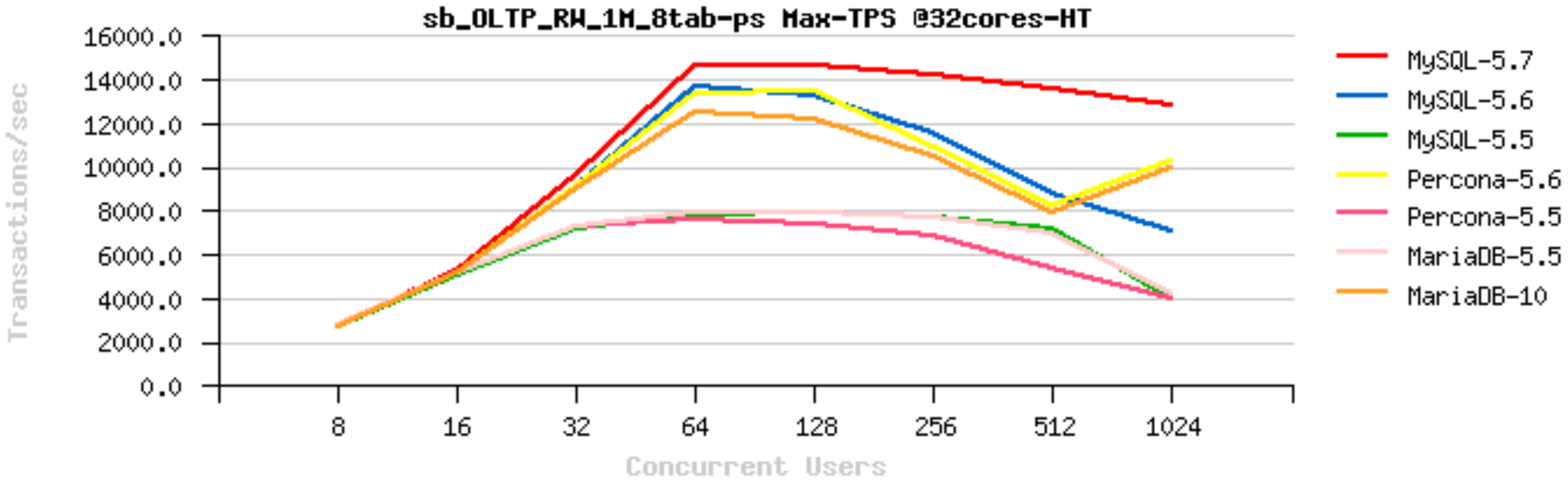


Read+Write Workloads : In-Memory

- 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 !
    - 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 ;-)

RW In-Memory @MySQL 5.7

- Sysbench OLTP_RW 8-tables 32cores-HT :

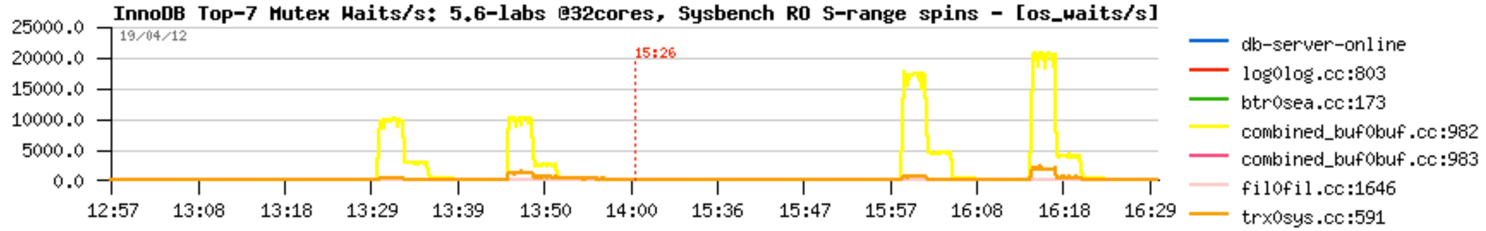


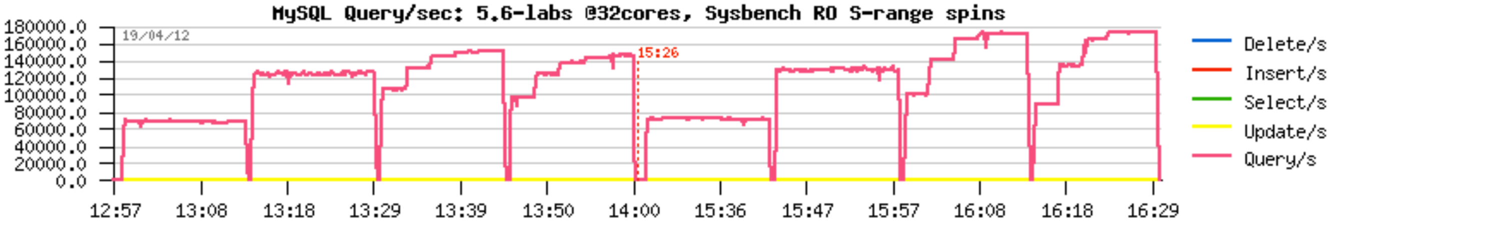
High Concurrency Tuning

- If bottleneck is due a concurrent access on the same data (due application design) – ask 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 ;-)
  - Try to increase InnoDB spin wait delay (dynamic)
  - Try innodb_thread_concurrency=N (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..

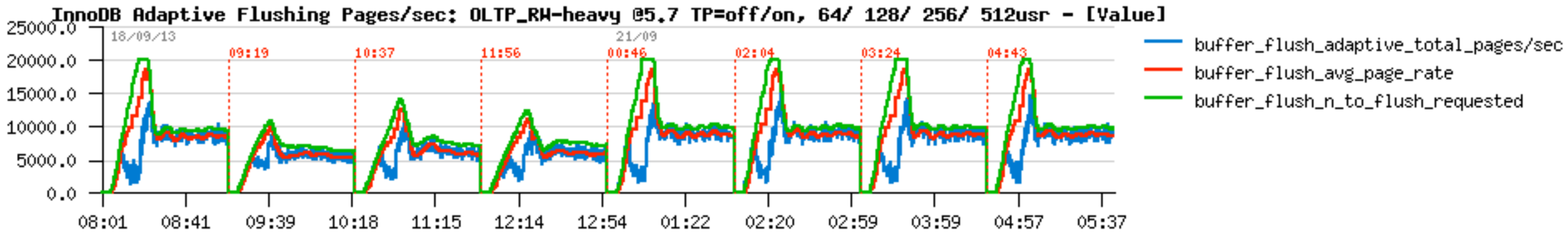
InnoDB Spin Wait Delay

- 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 here anymore..
  - So, innodb_spin_wait_delay is entering in the game:

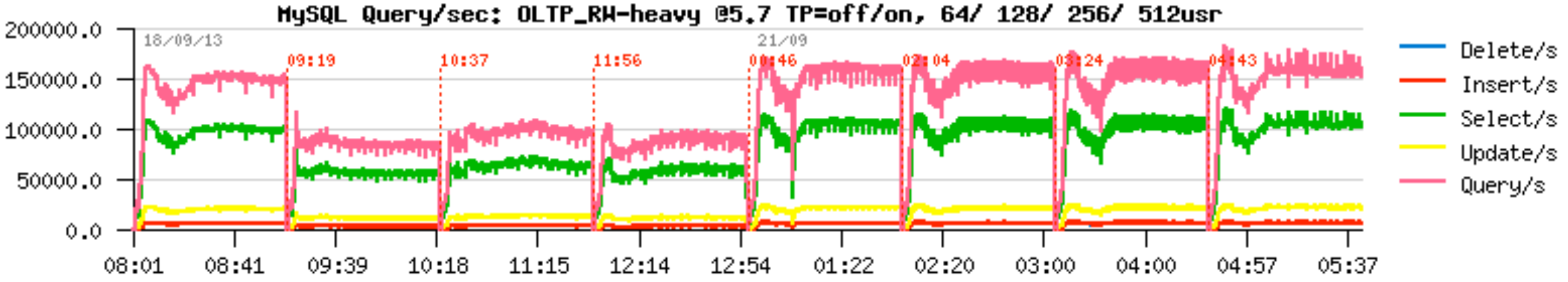




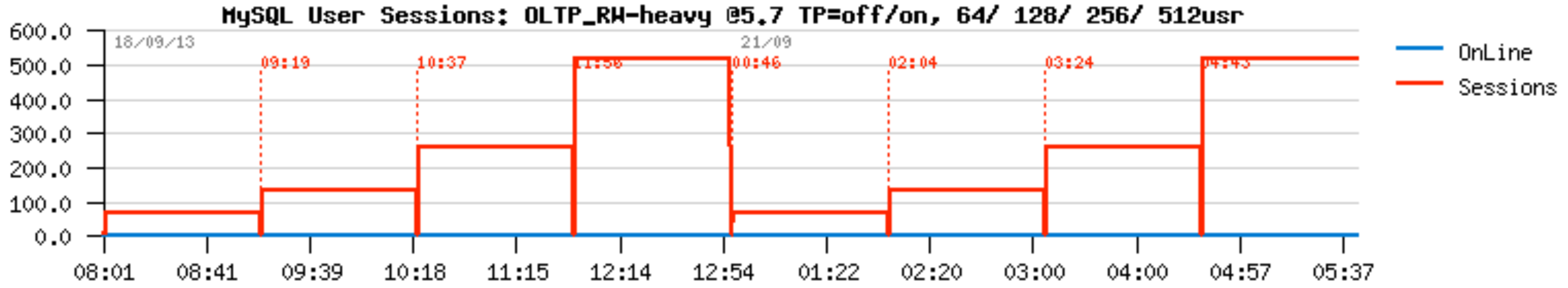
Thread Pool in old MySQL 5.7 @Heavy OLTP_RW



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



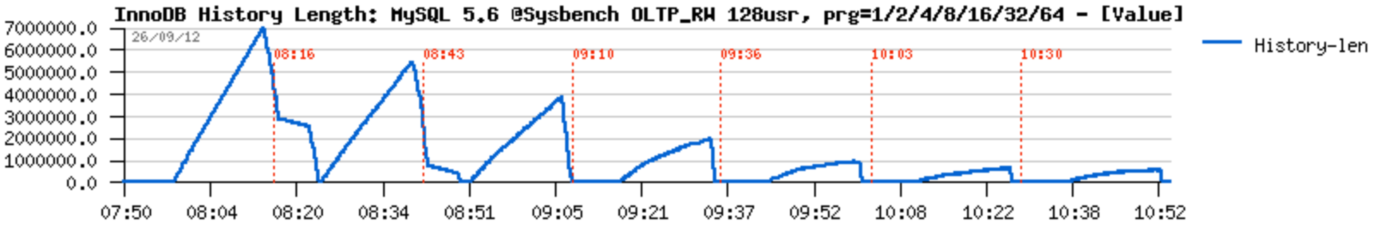
MySQL User Sessions: OLTP_RW-heavy @5.7 TP=off/on, 64/ 128/ 256/ 512usr

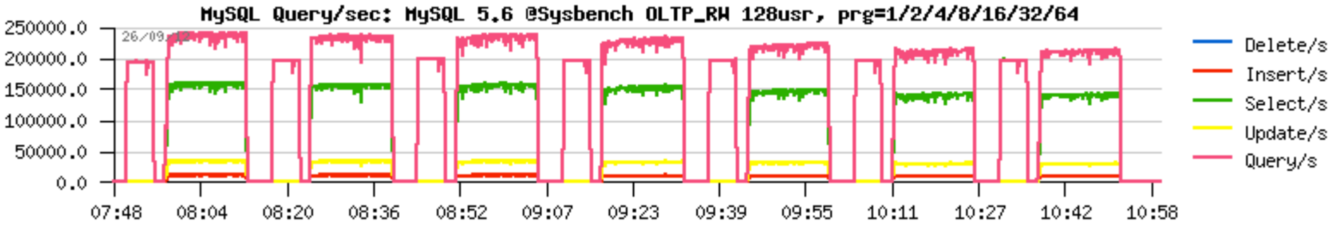
Read+Write Workloads : In-Memory

- 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

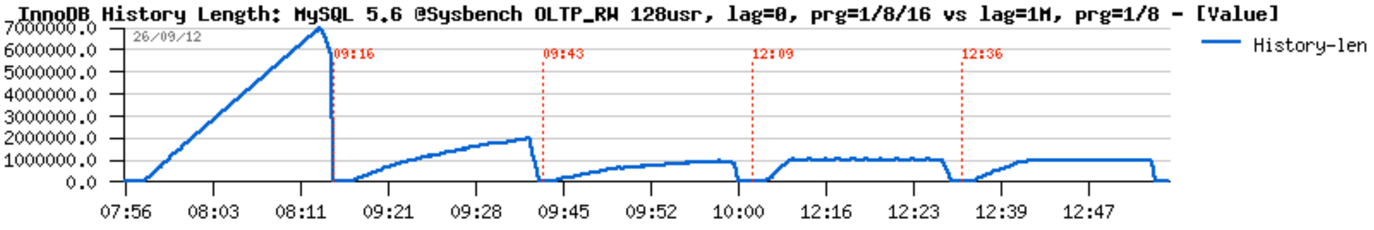
- Several Purge Threads :
  - NOTE #1 : activation is auto-magical (I'm serious ;-))
  - NOTE #2 : look well on the graphs - purge is not free !!!

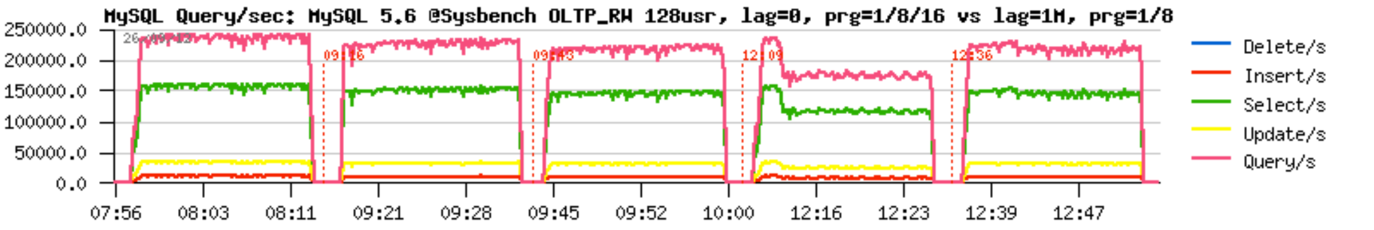




InnoDB : Purge improvement since 5.6

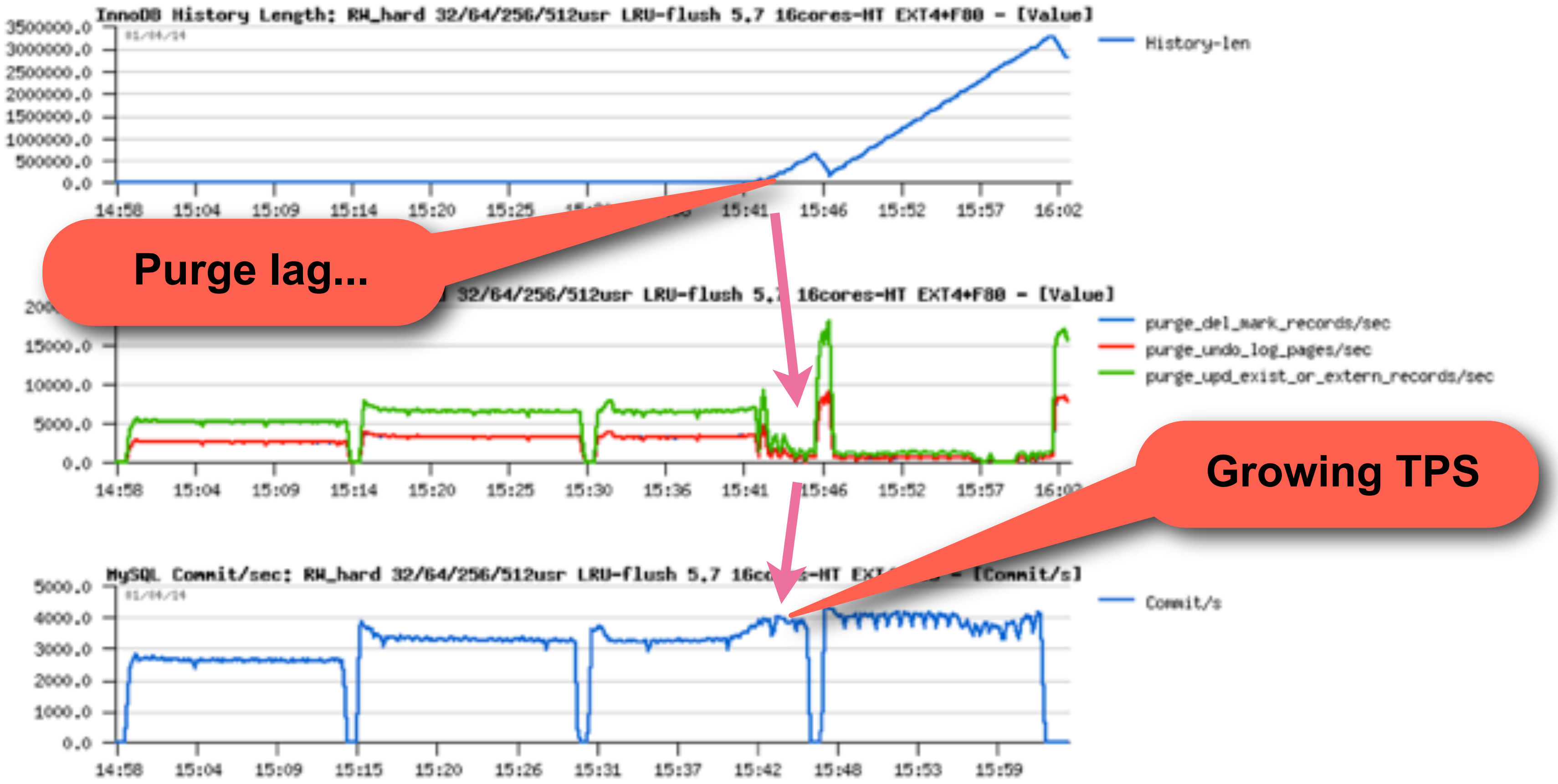
- Fixed max purge lag code!
  - innodb_max_purge_lag
  - innodb_max_purge_lag_delay <= configurable!
- Setting innodb_max_purge_lag=1M:





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? ;-) but not fair...



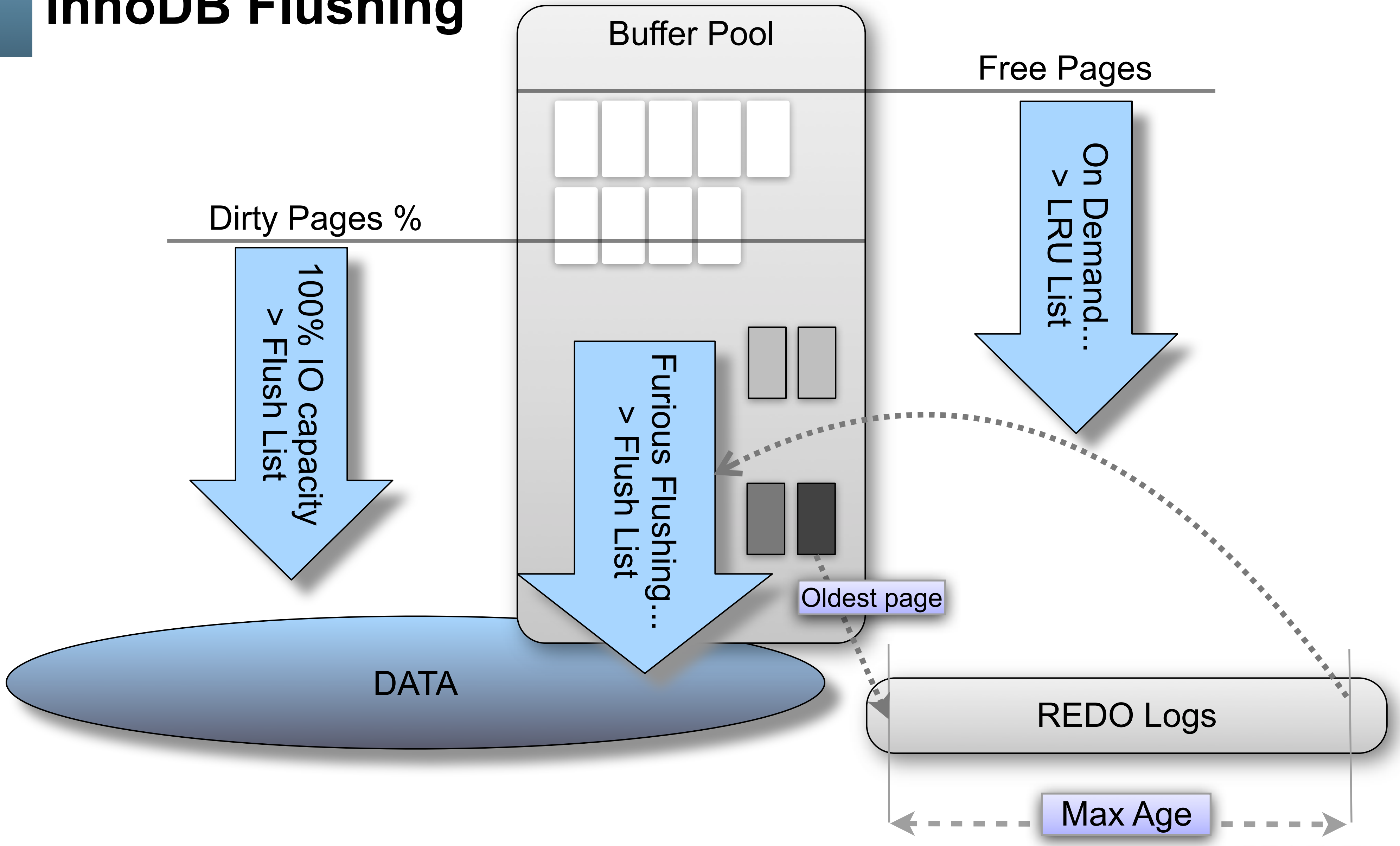
Purge lag...

Growing TPS

Read+Write Workloads : In-Memory

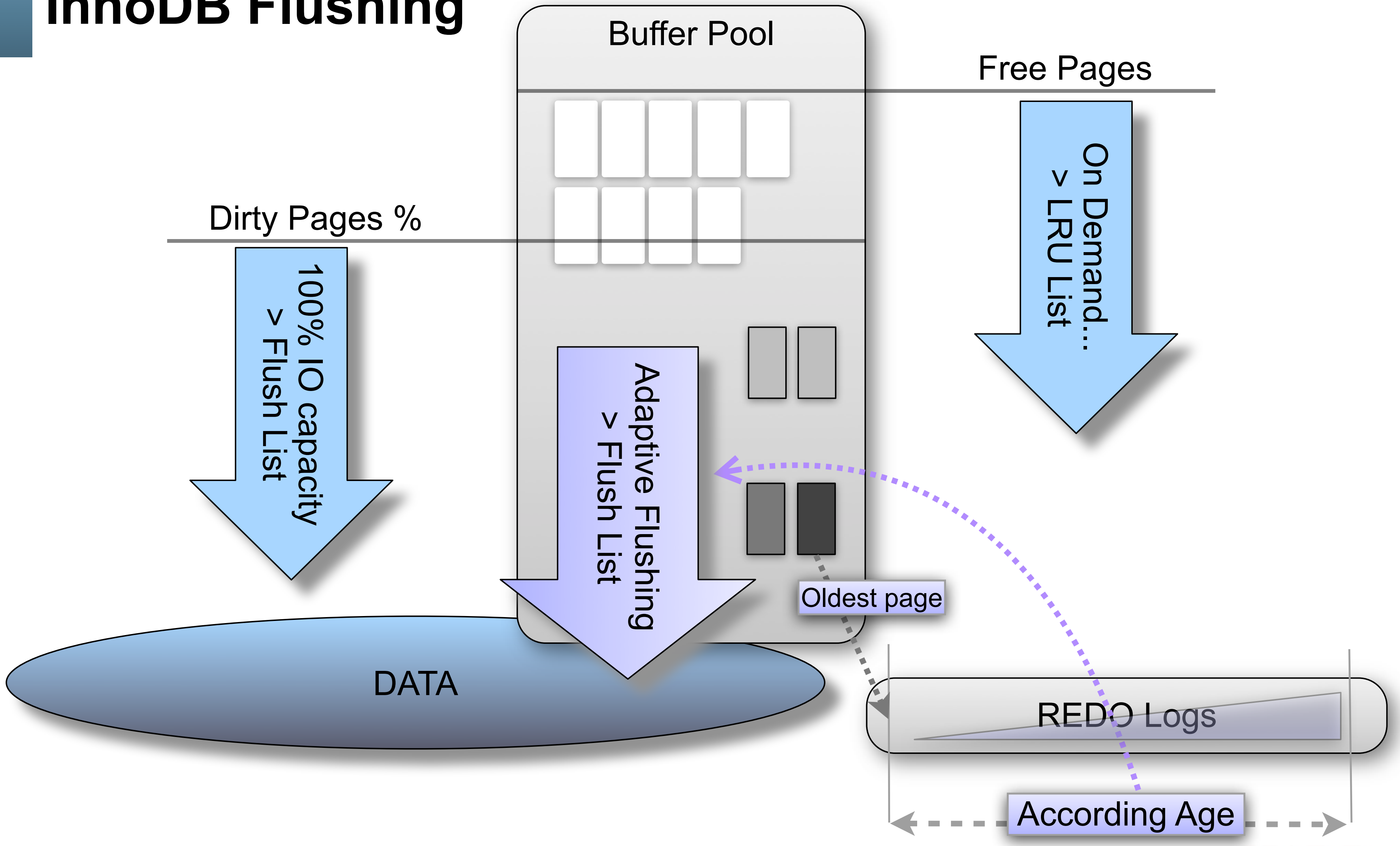
- InnoDB Flushing...
  - 5.5 : no comments.. ;-)
  - 5.6 :
    - Improved Adaptive Flushing (step 1)
    - Cleaner Thread
  - 5.7 :
    - multiple Cleaner Threads
    - improved LRU flushing
    - improved Adaptive Flushing Design (step 2)

InnoDB Flushing

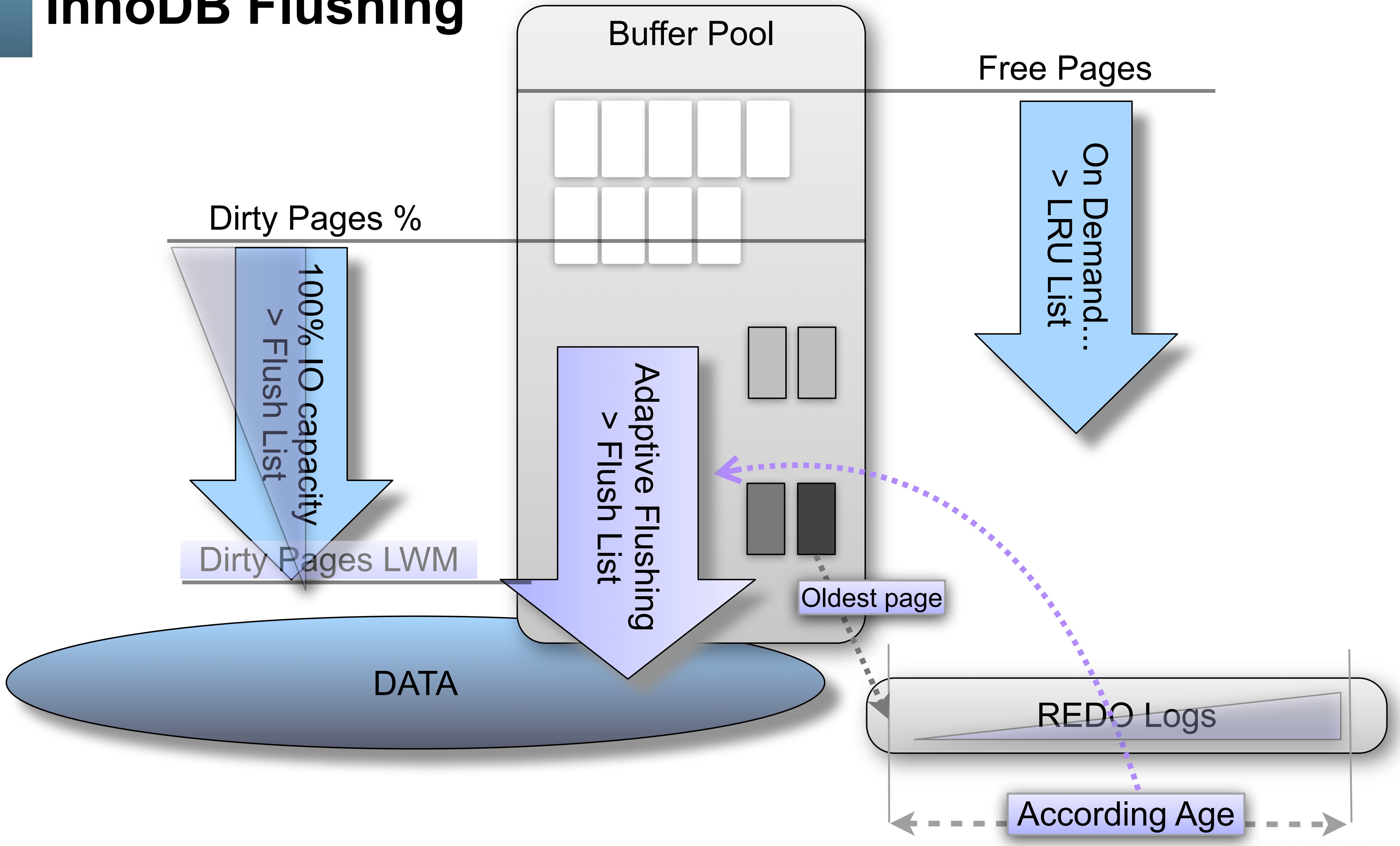


ORACLE

InnoDB Flushing

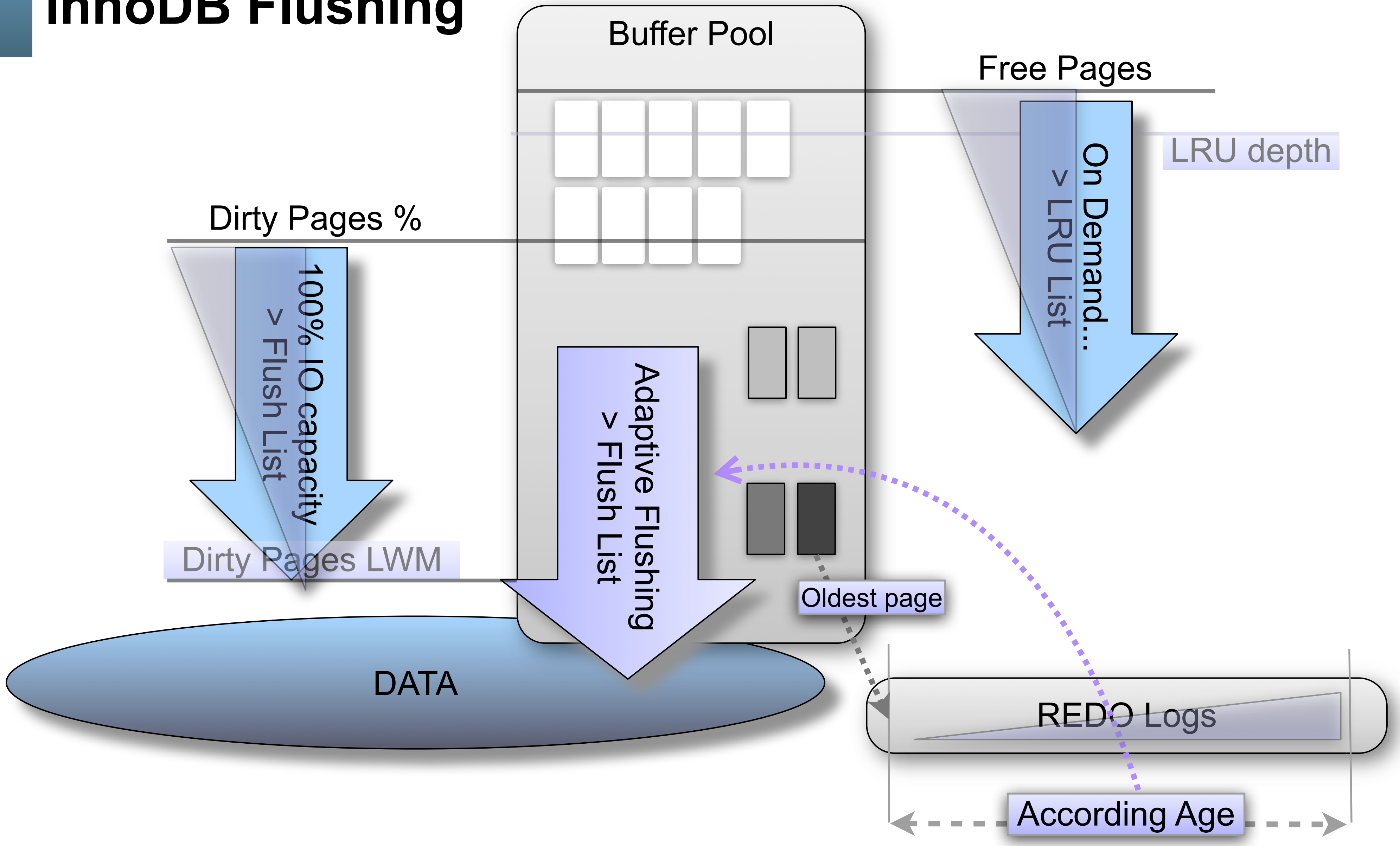


InnoDB Flushing



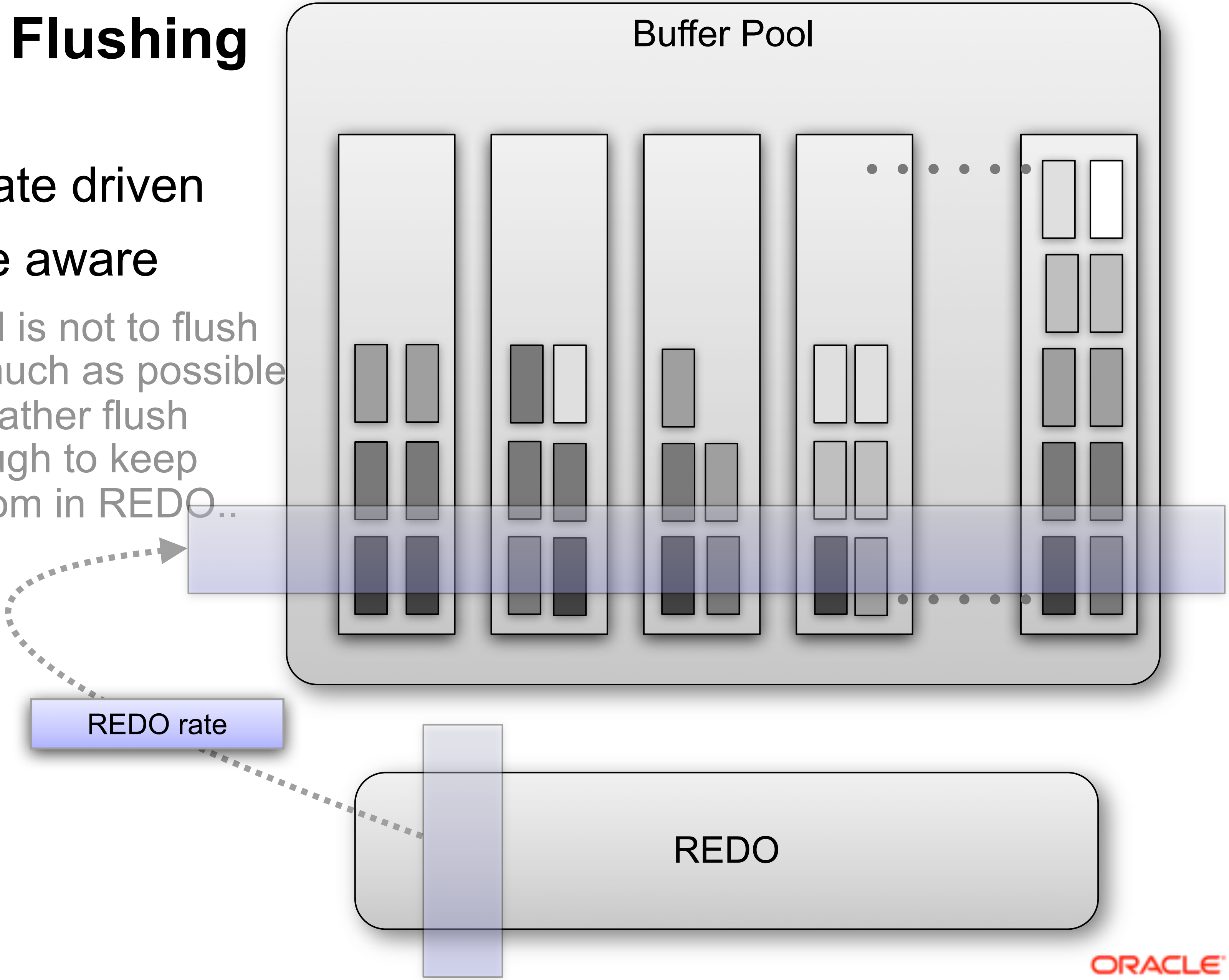
ORACLE

InnoDB Flushing



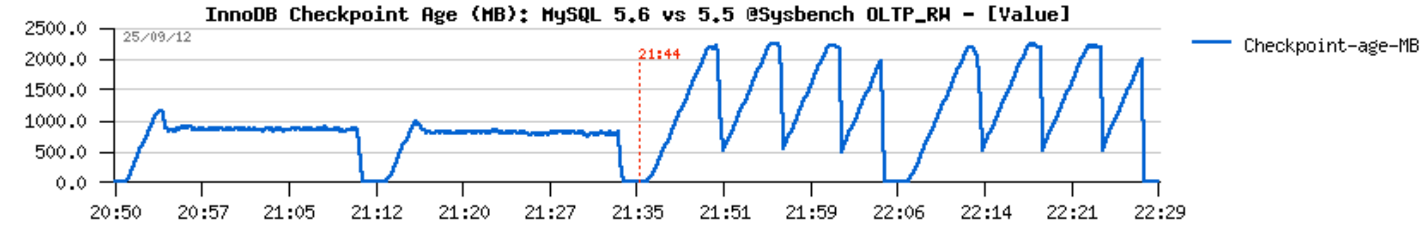
InnoDB Flushing

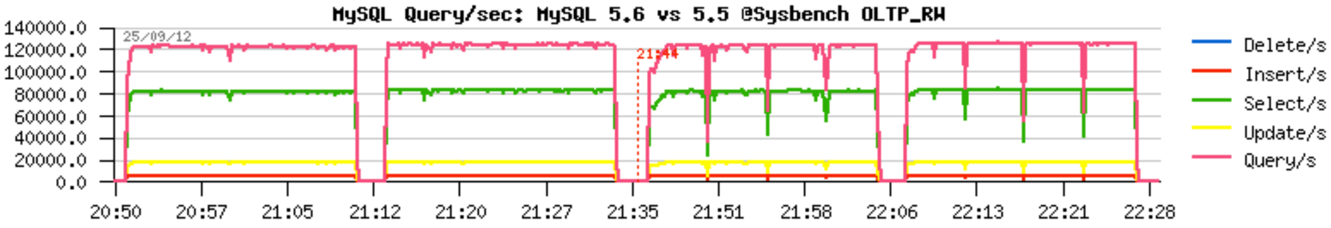
- REDO rate driven
- LSN Age aware
  - the goal is not to flush as much as possible but rather flush enough to keep a room in REDO..



Adaptive Flushing: MySQL 5.6 vs 5.5

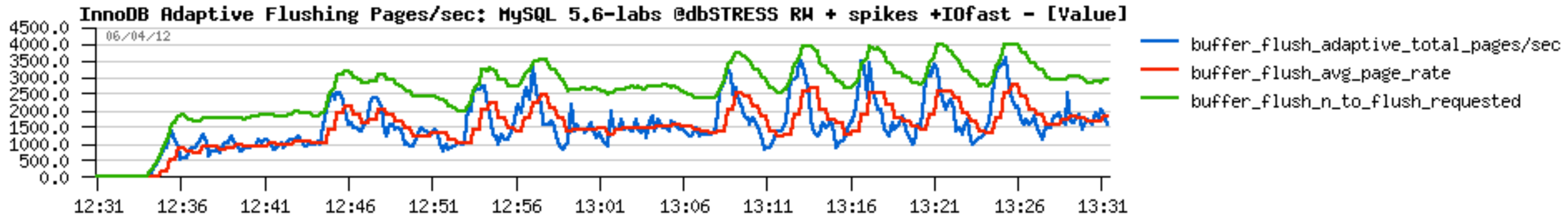
- OLTP_RW Workload:
  - Same IO capacity
  - Different logic..

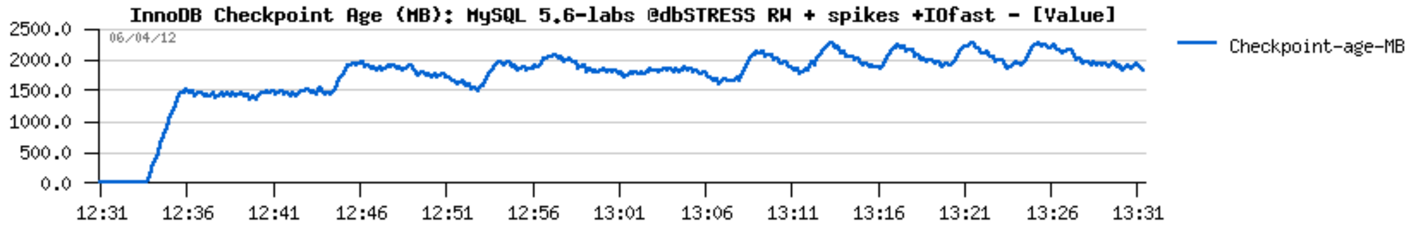


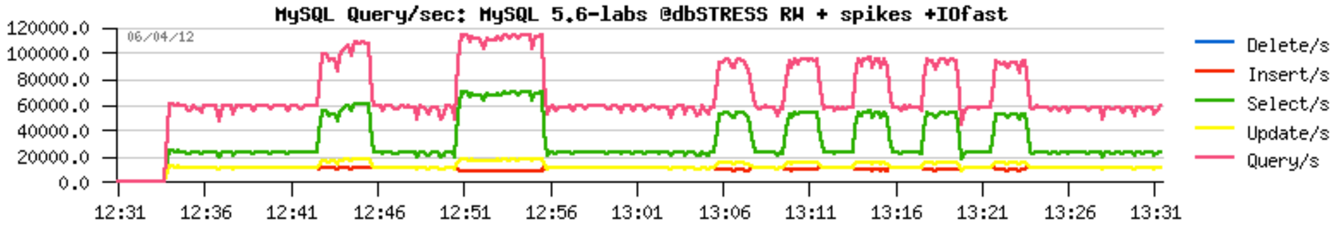


InnoDB : Resisting to activity spikes in 5.6

- dbSTRESS R+W with spikes

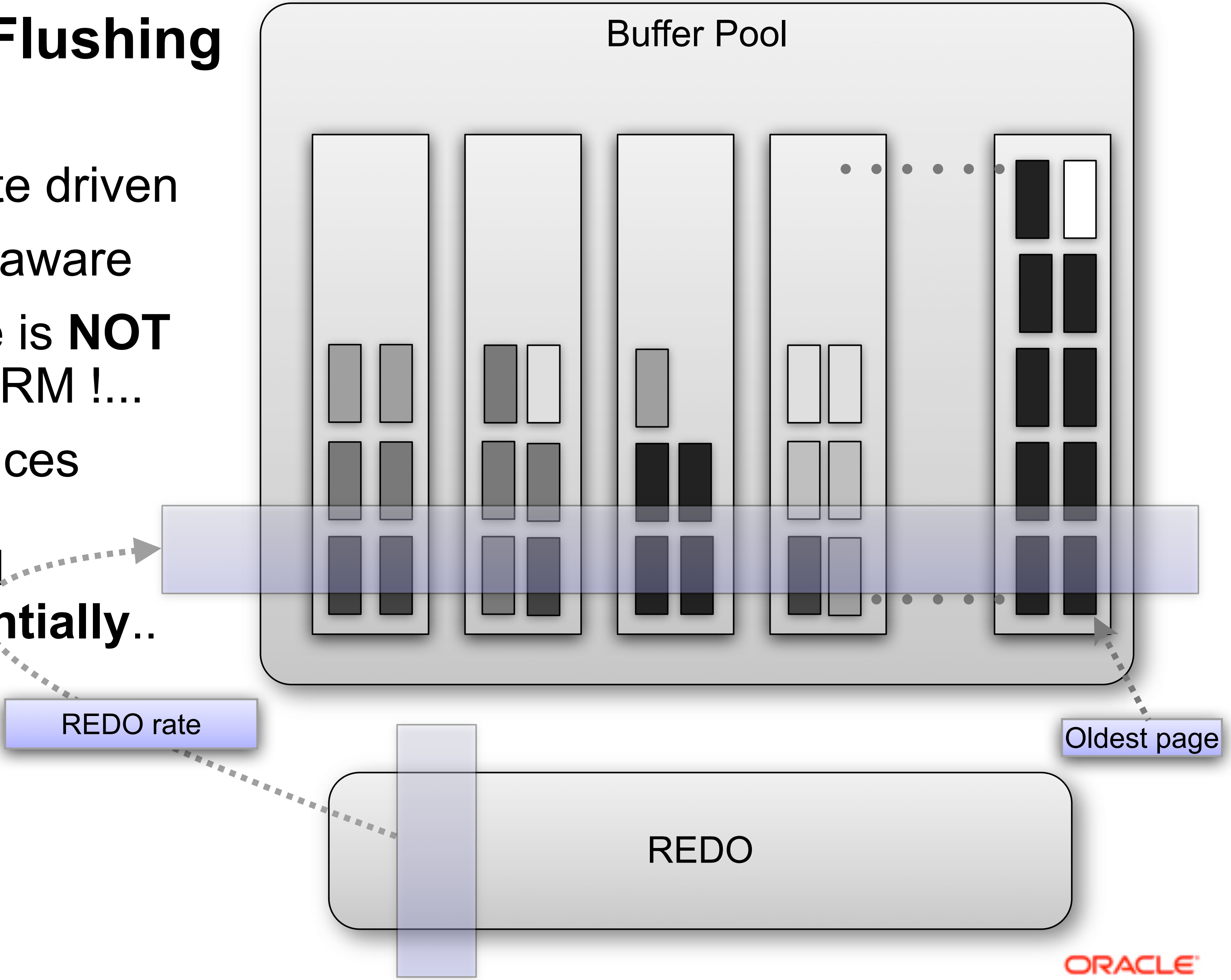






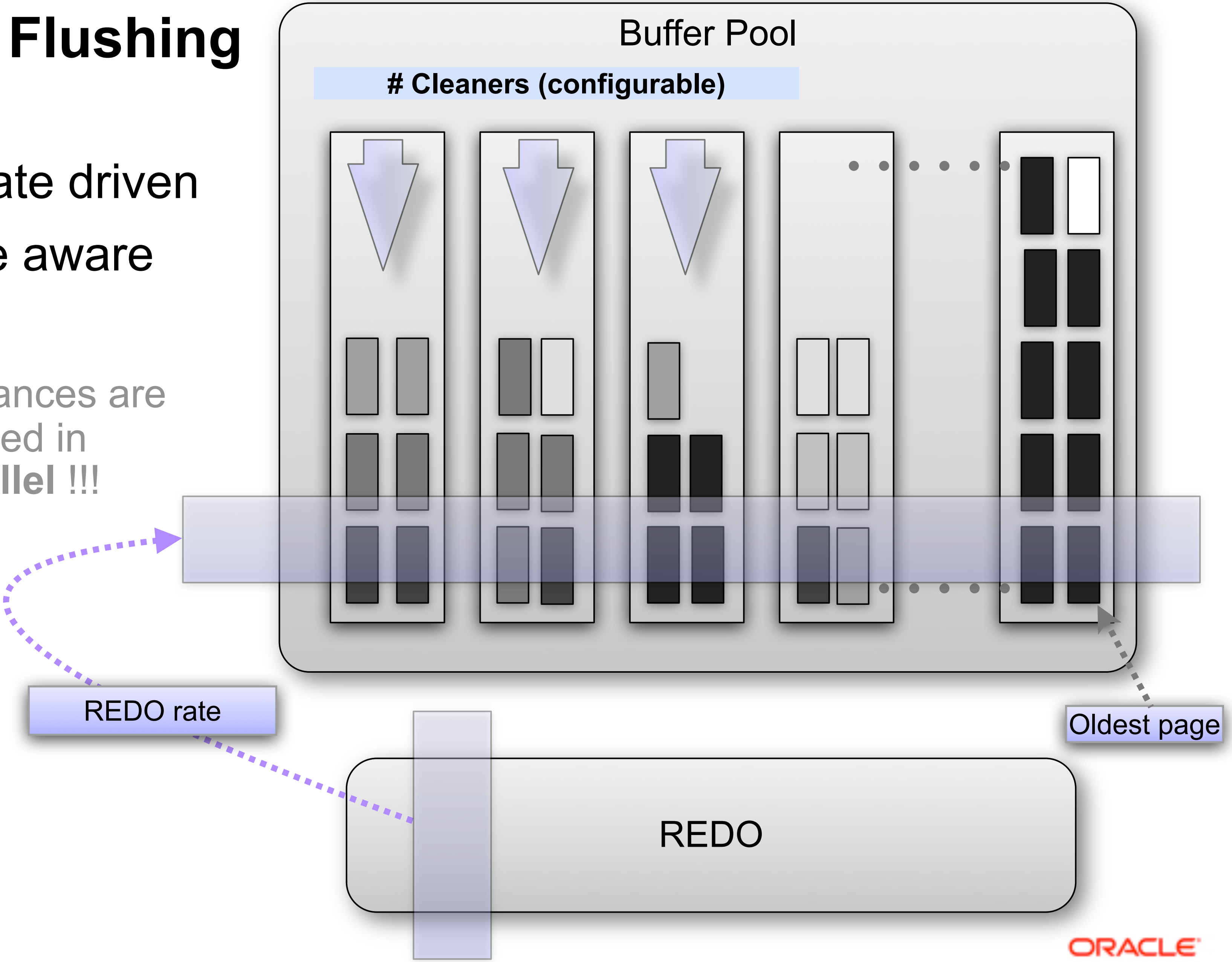
InnoDB Flushing

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



InnoDB Flushing

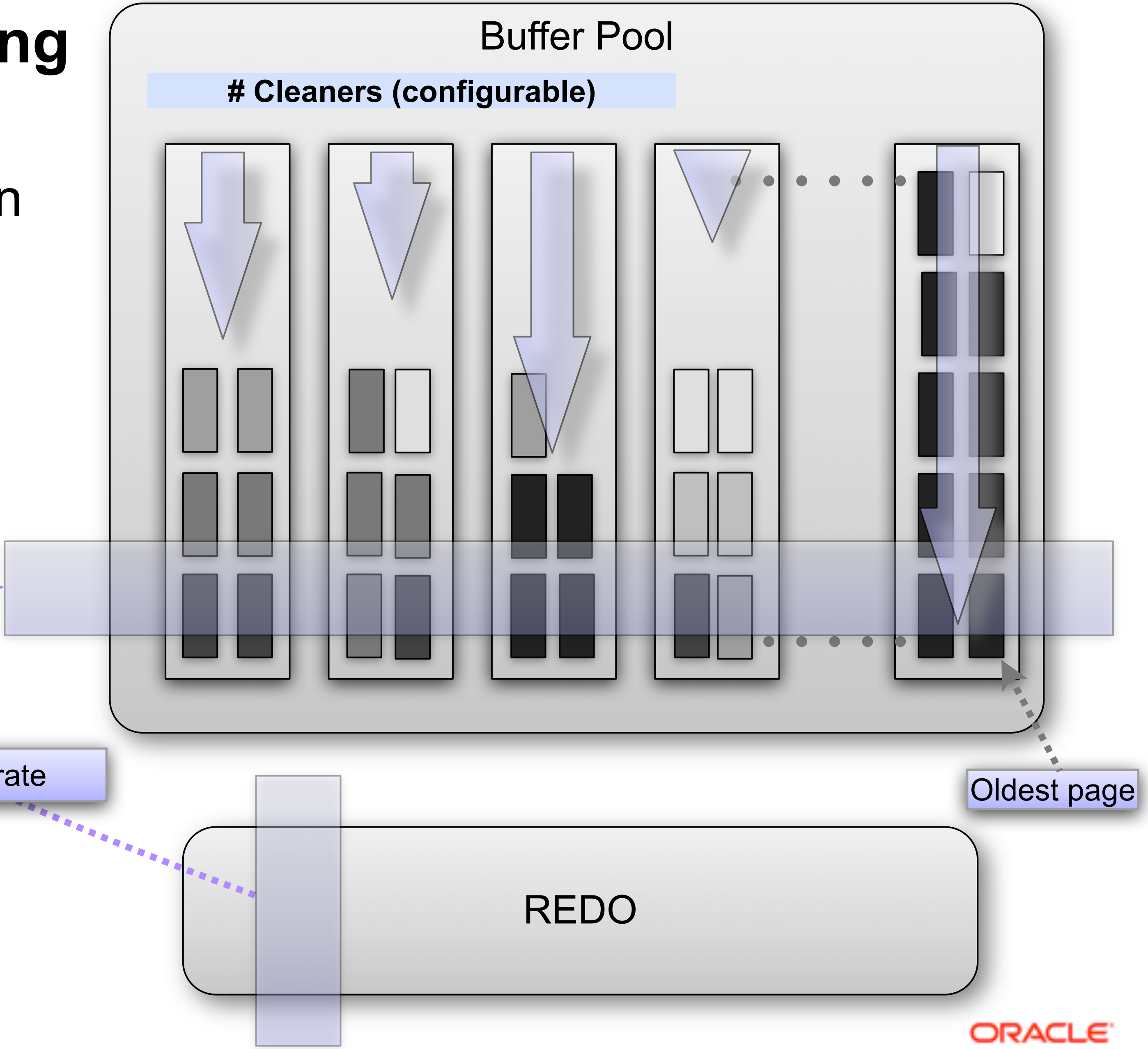
- REDO rate driven
- LSN Age aware
- 5.7 :
  - BP Instances are flushed in **parallel** !!!



ORACLE

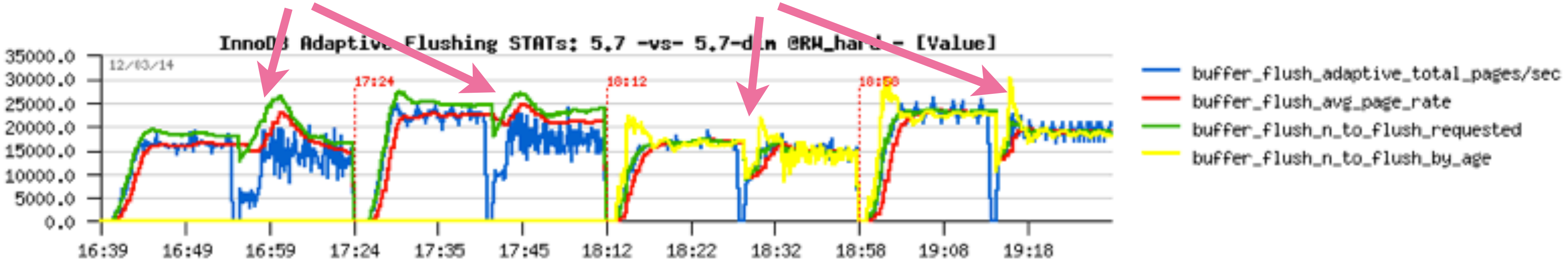
InnoDB Flushing

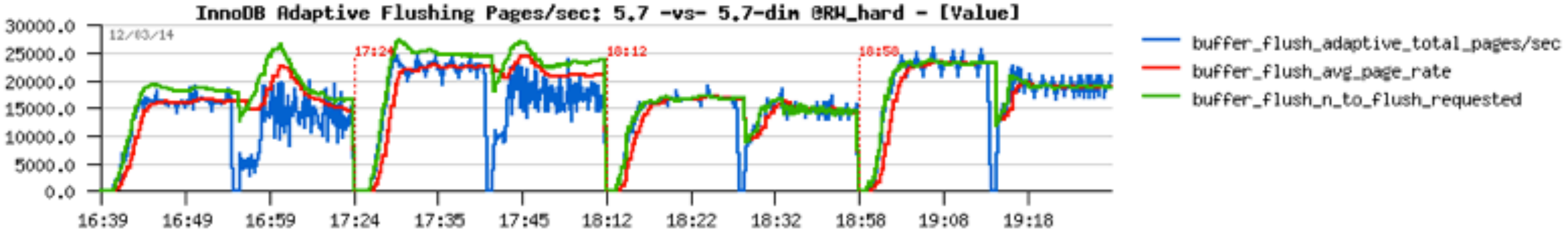
- REDO rate driven
- LSN Age aware
- 5.7 :
  - BP Instances are flushed in **parallel** !!!
  - Flushing rate is **adapted to Age distribution** within each BP instance !!!



InnoDB Flushing in 5.7

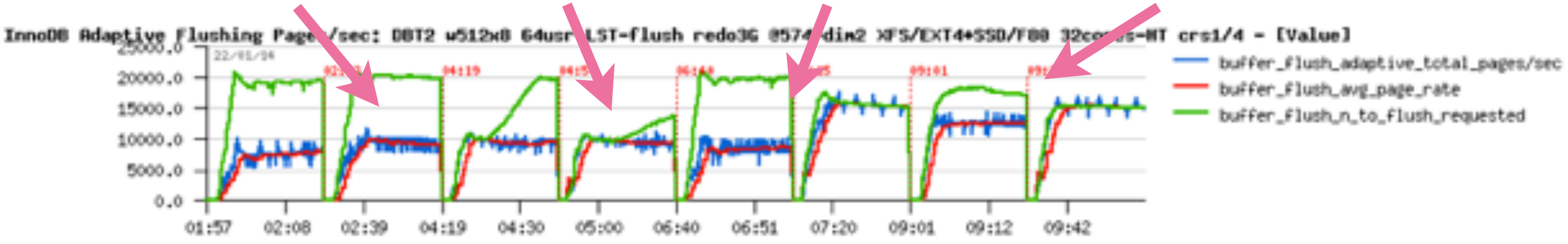
- Considering Age distribution :
  - Parallel Only -vs- Parallel + Age aware

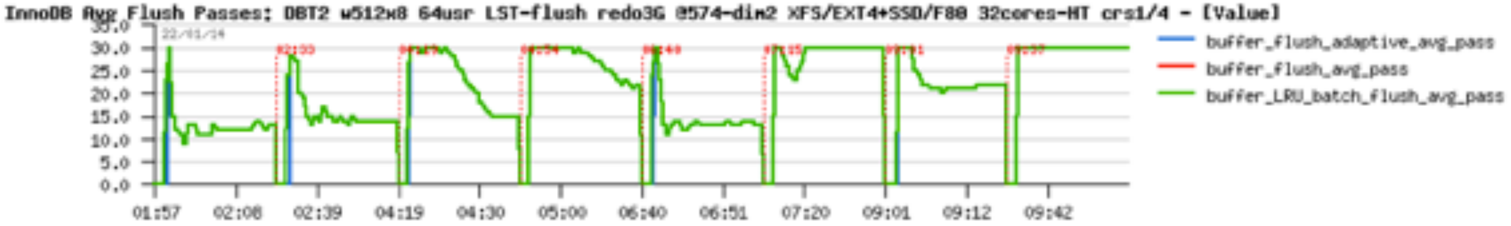


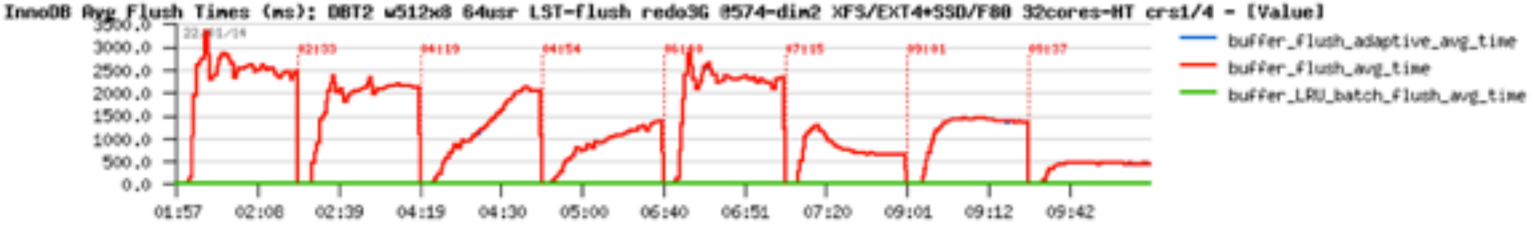


InnoDB Flushing in 5.7

- 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

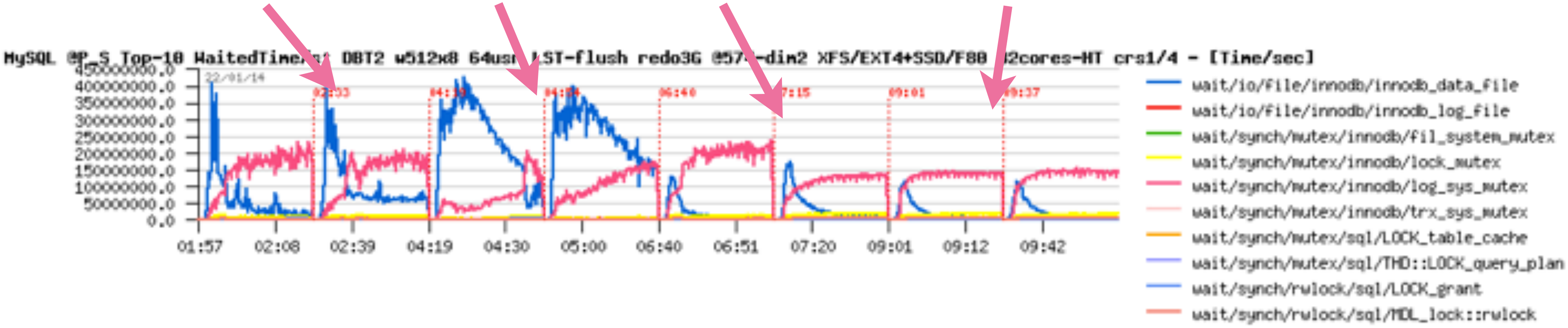


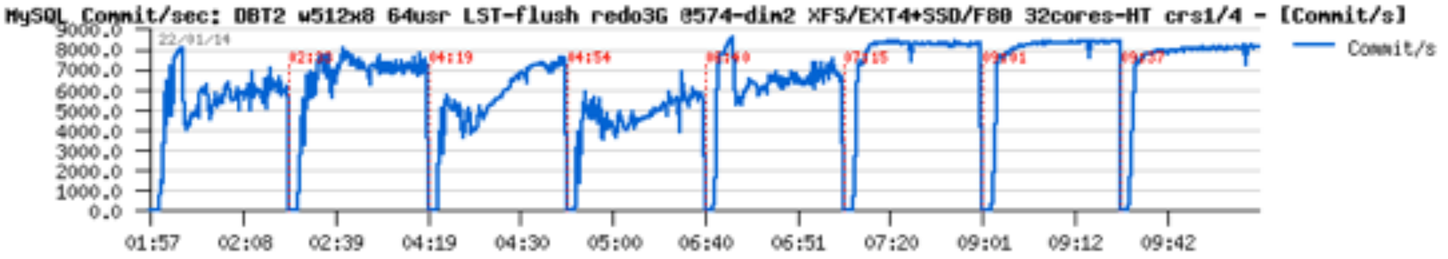




InnoDB Flushing in 5.7

- 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



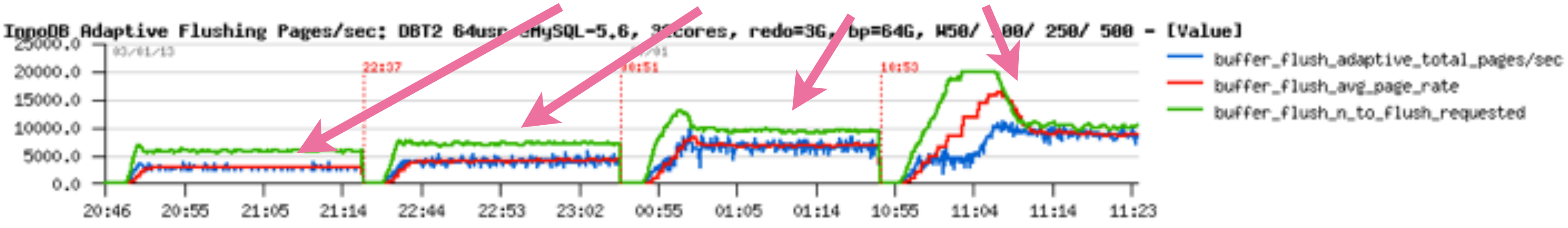


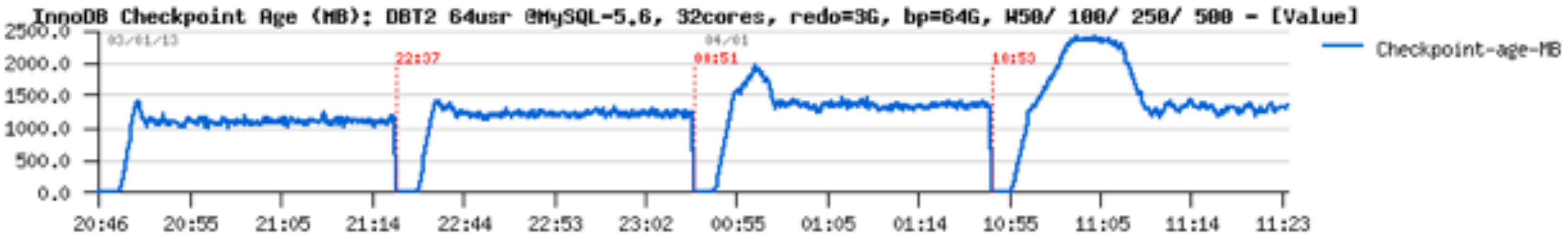
RW IO-bound

- 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 “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



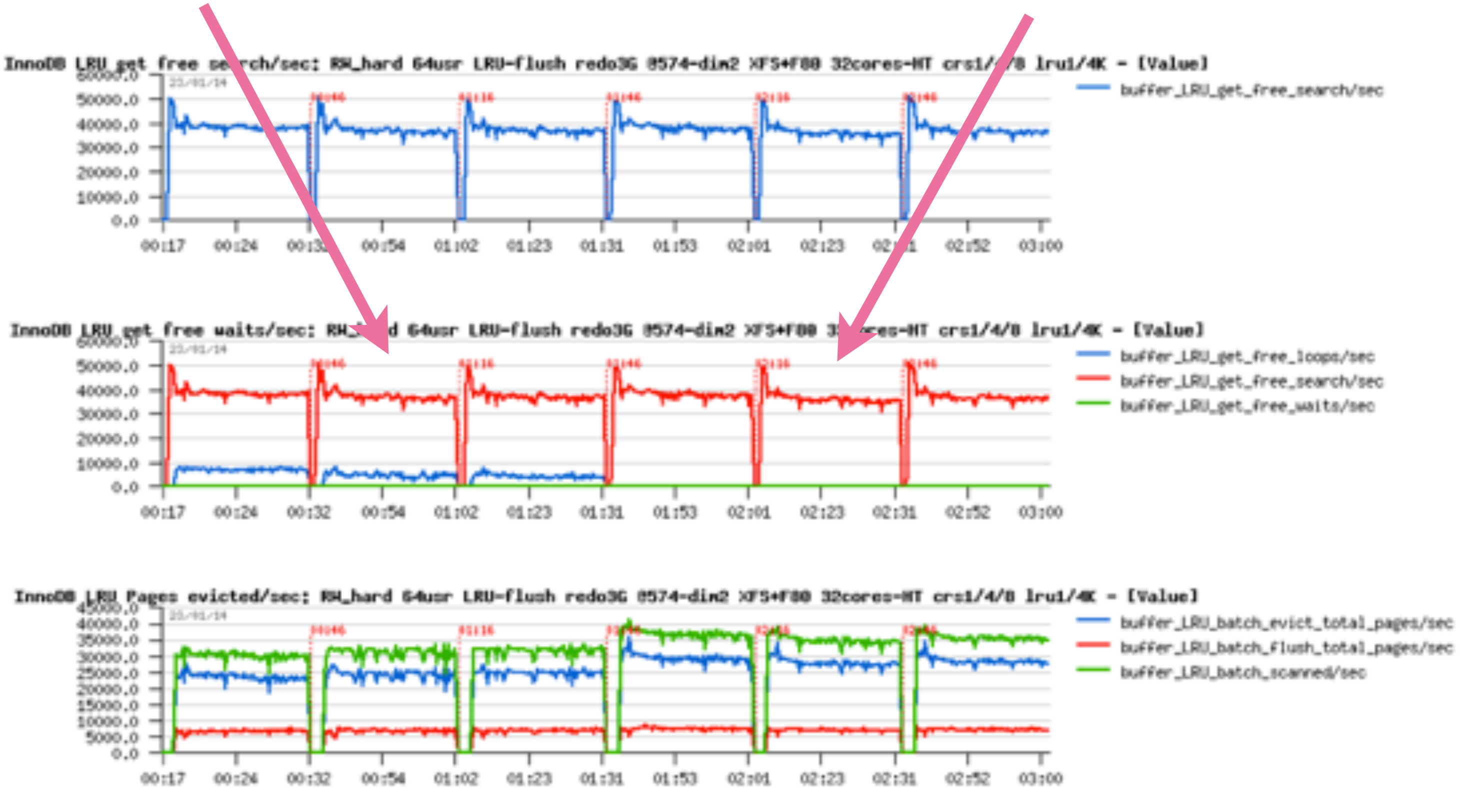


RW IO-bound “Out-of-Memory”

- 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..
- Better that Cleaner is always keeping free lists non-empty ;-))

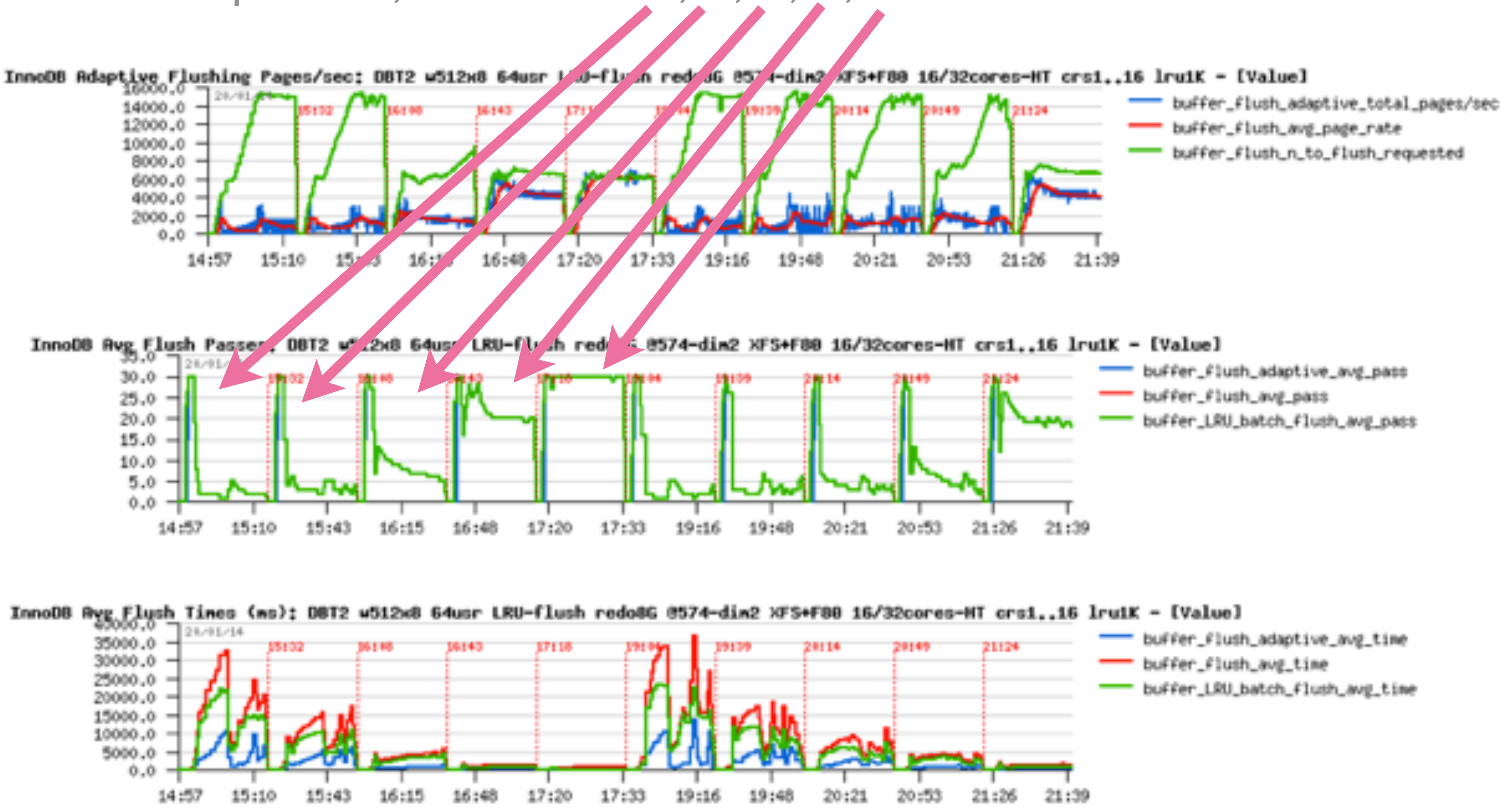
RW IO-bound “Out-of-Memory”

- 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..

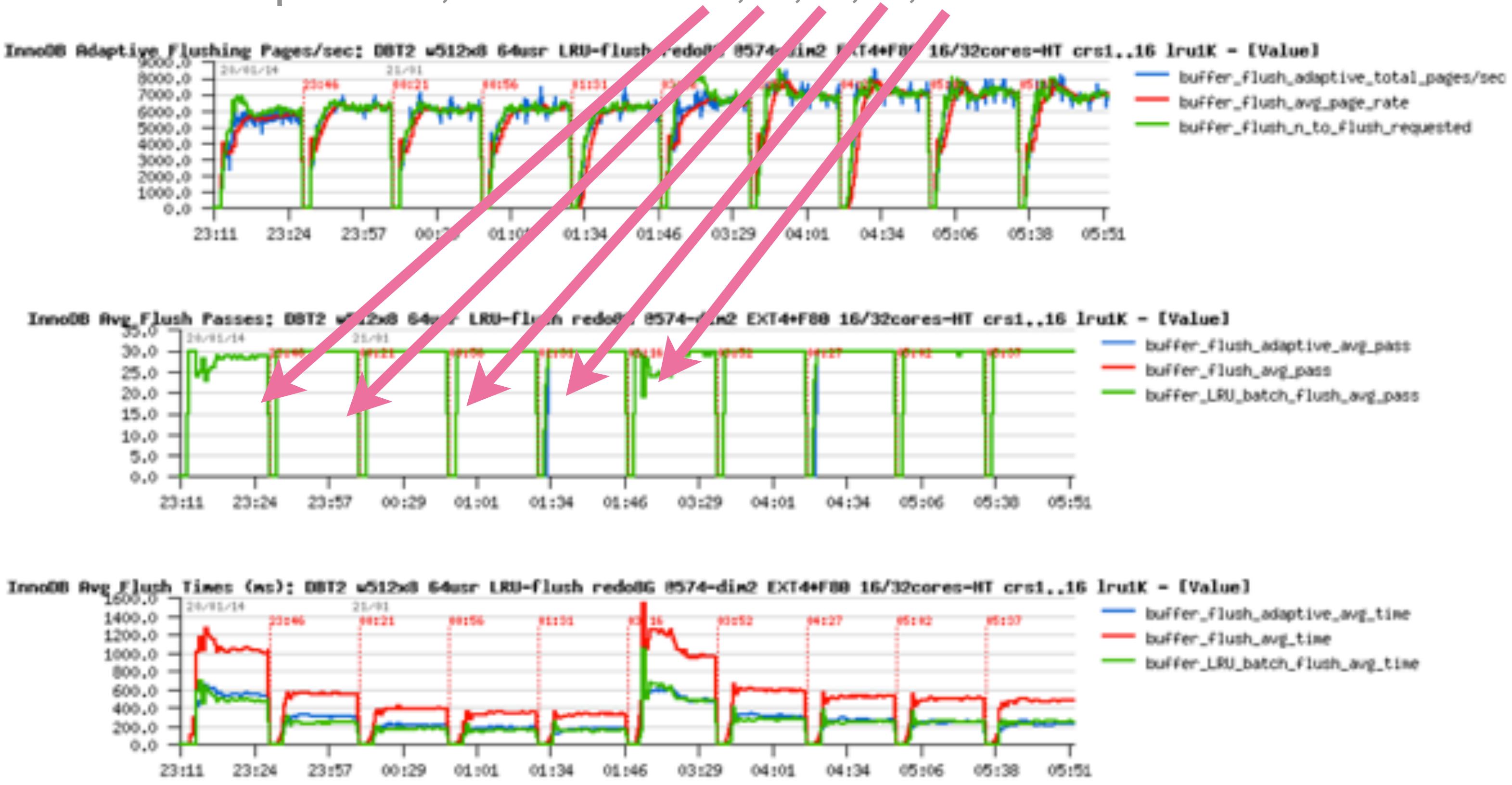
- DBT2 Workload, 64 users, **XFS**
  - LRU depth=1K, cleaners= 1, 2, 4, 8, 16 16cores-HT / 32cores-HT



ORACLE

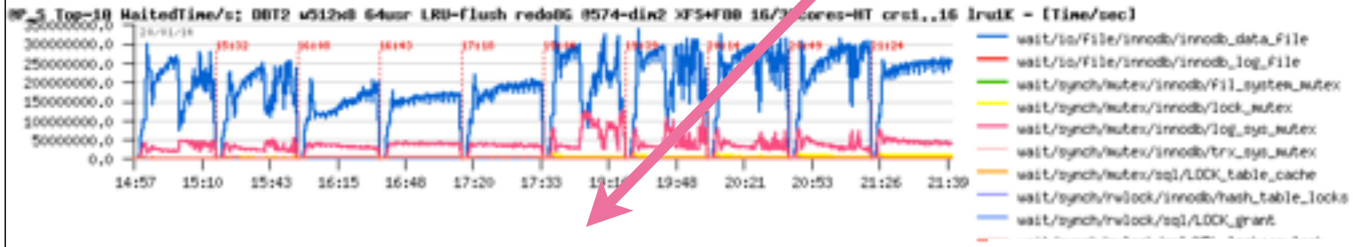
RW LRU-bound : FS impact..

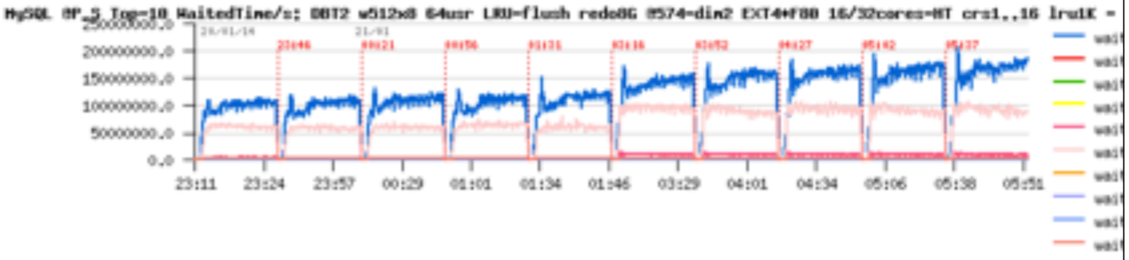
- DBT2 Workload, 64 users, **EXT4**
  - LRU depth=1K, cleaners= 1, 2, 4, 8, 16 16cores-HT / 32cores-HT

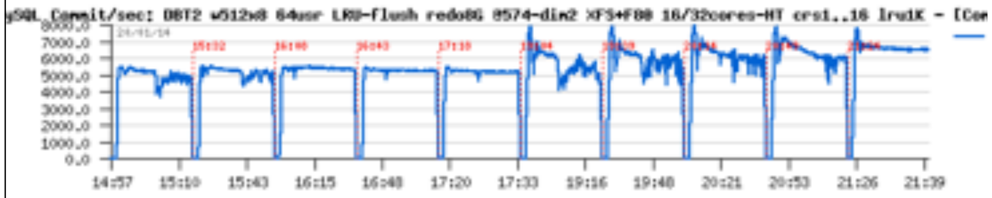


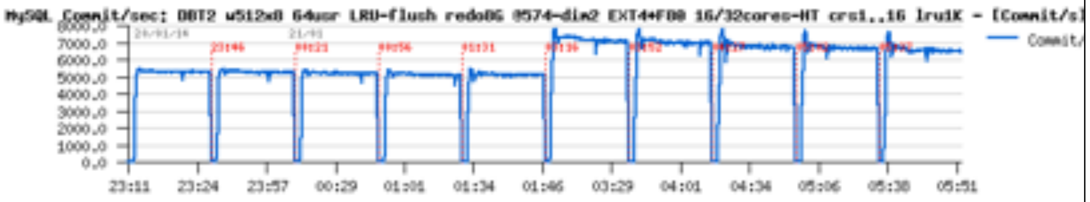
RW LRU-bound : FS impact..

- 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 : “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 :
  - 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

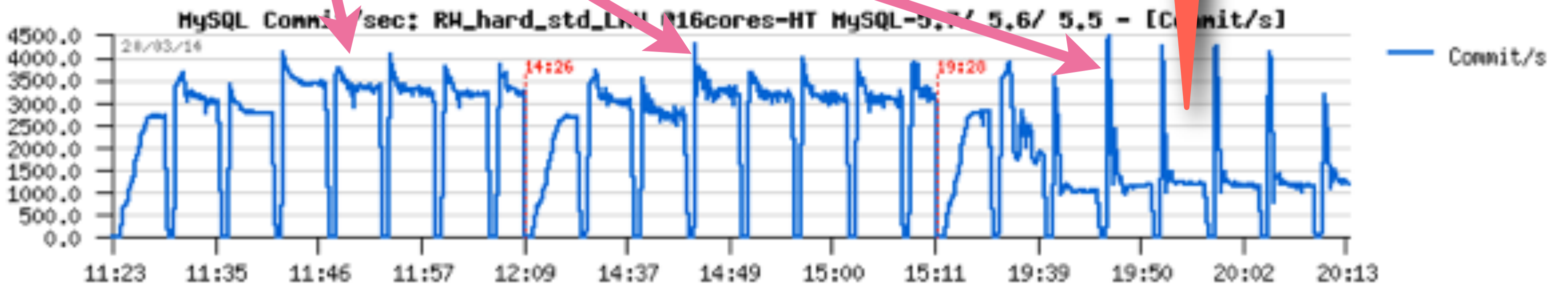
RW IO-bound Workloads

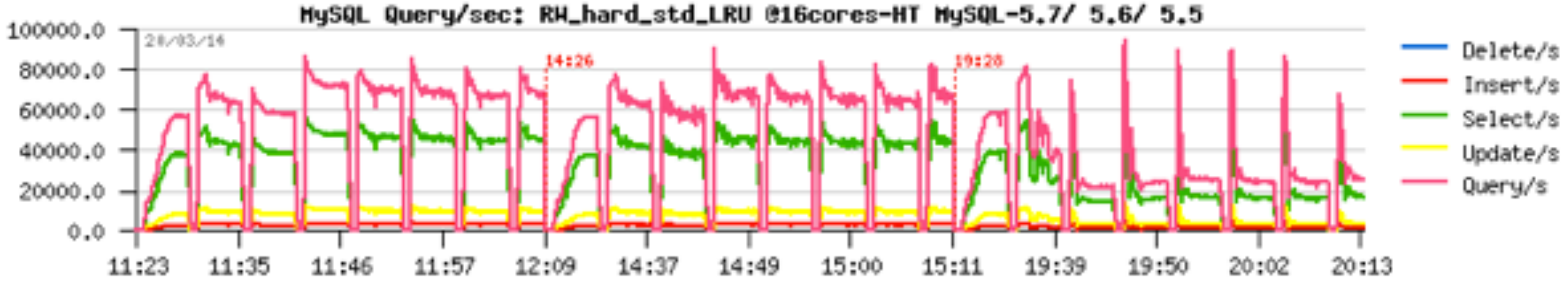
- Workloads :
  - Sysbench OLTP_RW 10Mx32-tables UNIFORM / PARETO
  - DBT2 512W x8 databases
  - LinkBench 150G data (150M ids)
- User Concurrency :
  - 32, 64, 256, 512 users
  - 15-20 min for each test level
- Test Conditions :
  - LRU-bound (BP size is less than 1/4 or 1/3 of db size)
  - LIST-bound (BP size is big enough to fit the whole db set)
- Engines:
  - MySQL 5.7 latest, 5.7.3, 5.6, 5.5 (Percona 5.6 just for PARETO)

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

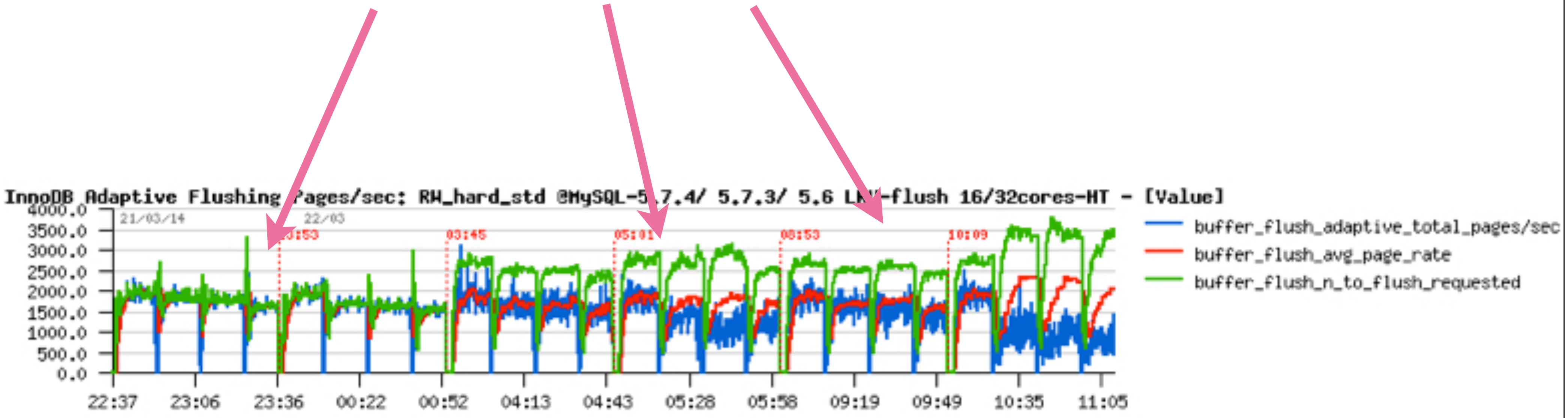
Please, upgrade me to 5.6 !!!





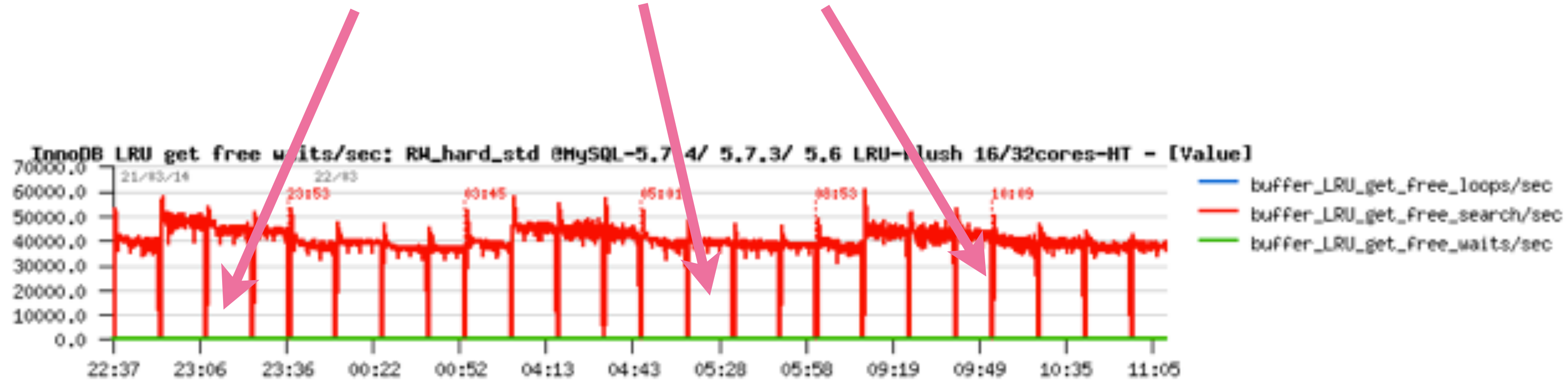
OLTP_RW 10Mx32-tab Uniform : LRU-bound

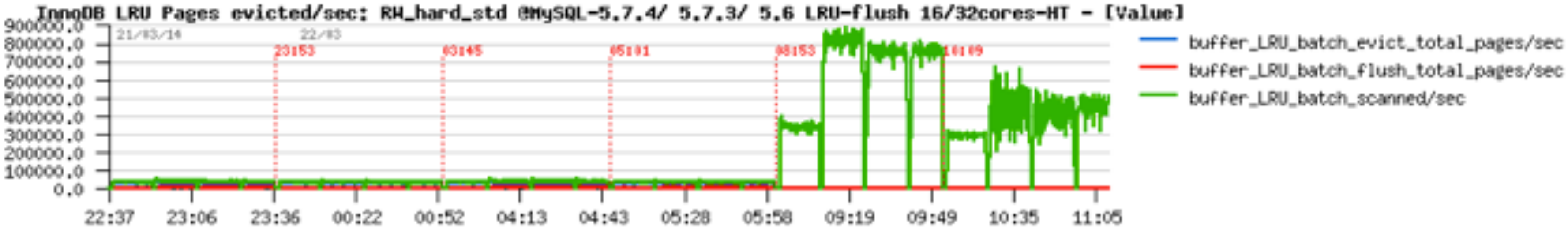
- Focus on : flush list
- Engines: 5.7 latest, 5.7.3, 5.6



OLTP_RW 10Mx32-tab Uniform : LRU-bound

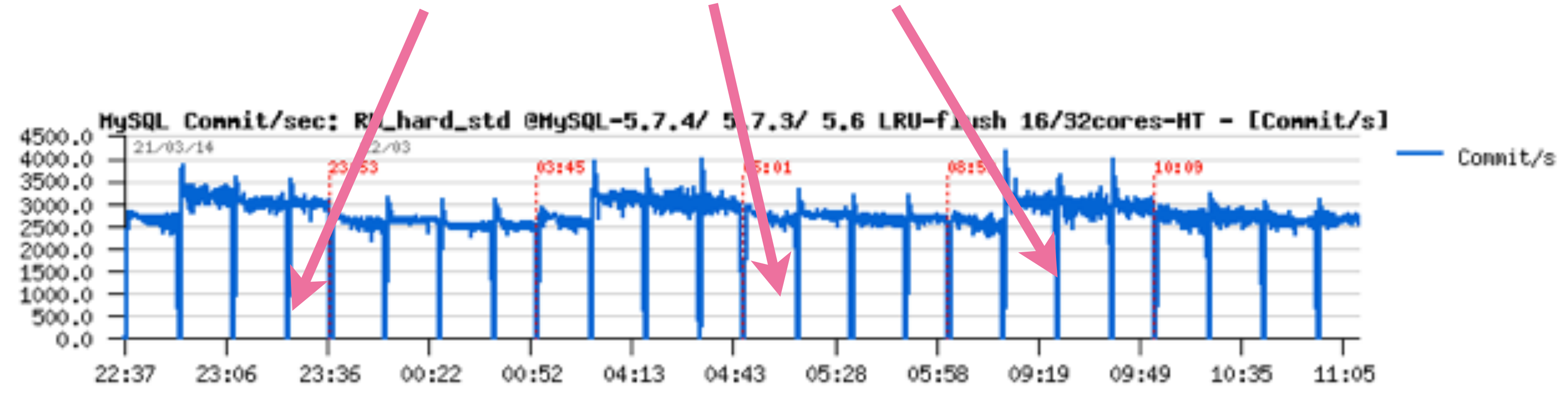
- Focus on : page scan & LRU flushing
- Engines: 5.7 latest, 5.7.3, 5.6

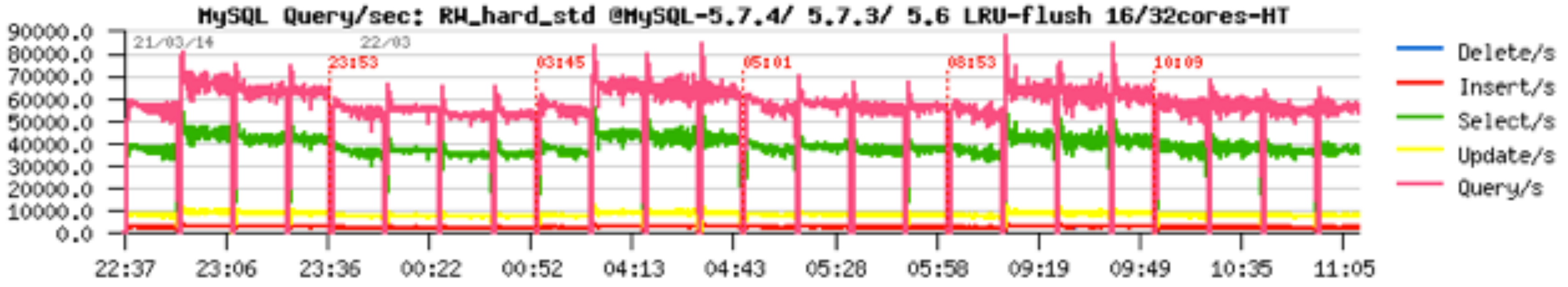




OLTP_RW 10Mx32-tab Uniform : LRU-bound

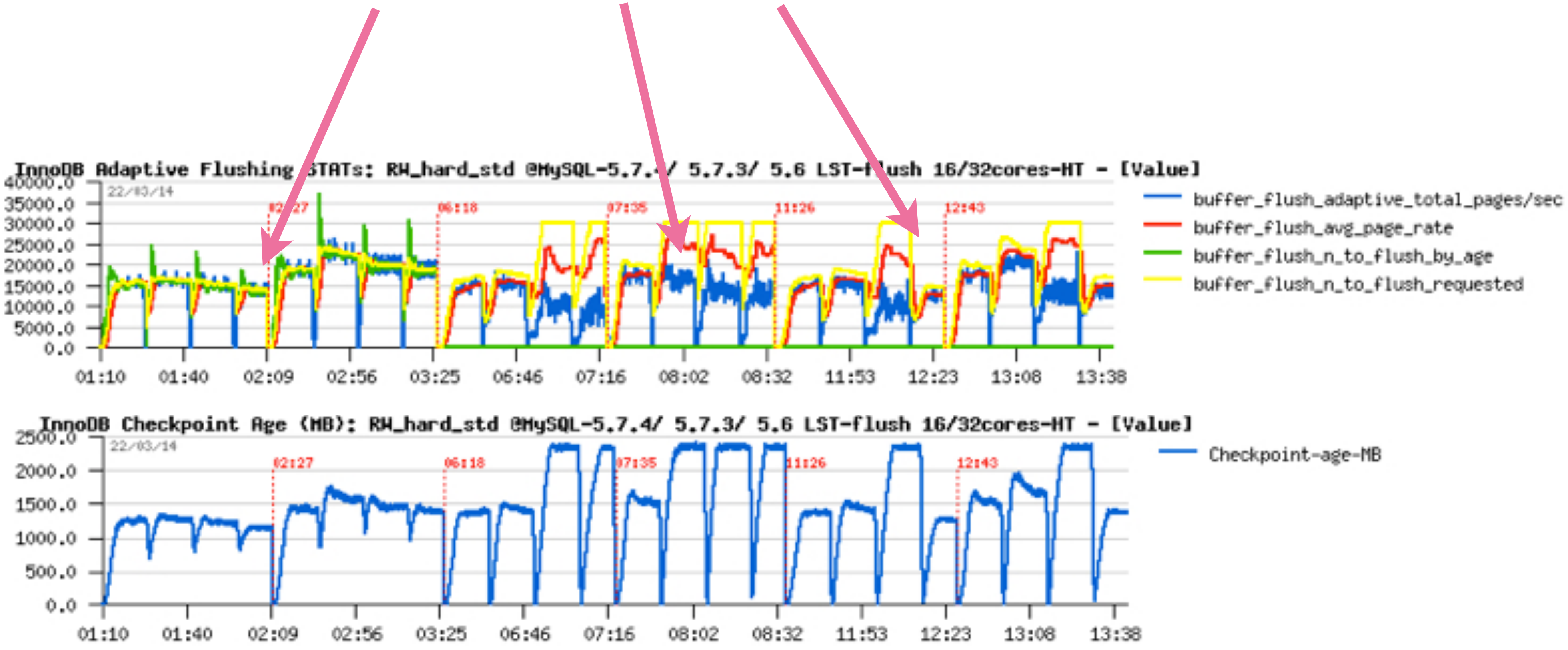
- Focus on : TPS & QPS... (hmm.. : near the same?? ;-))
- Engines: 5.7 latest, 5.7.3, 5.6





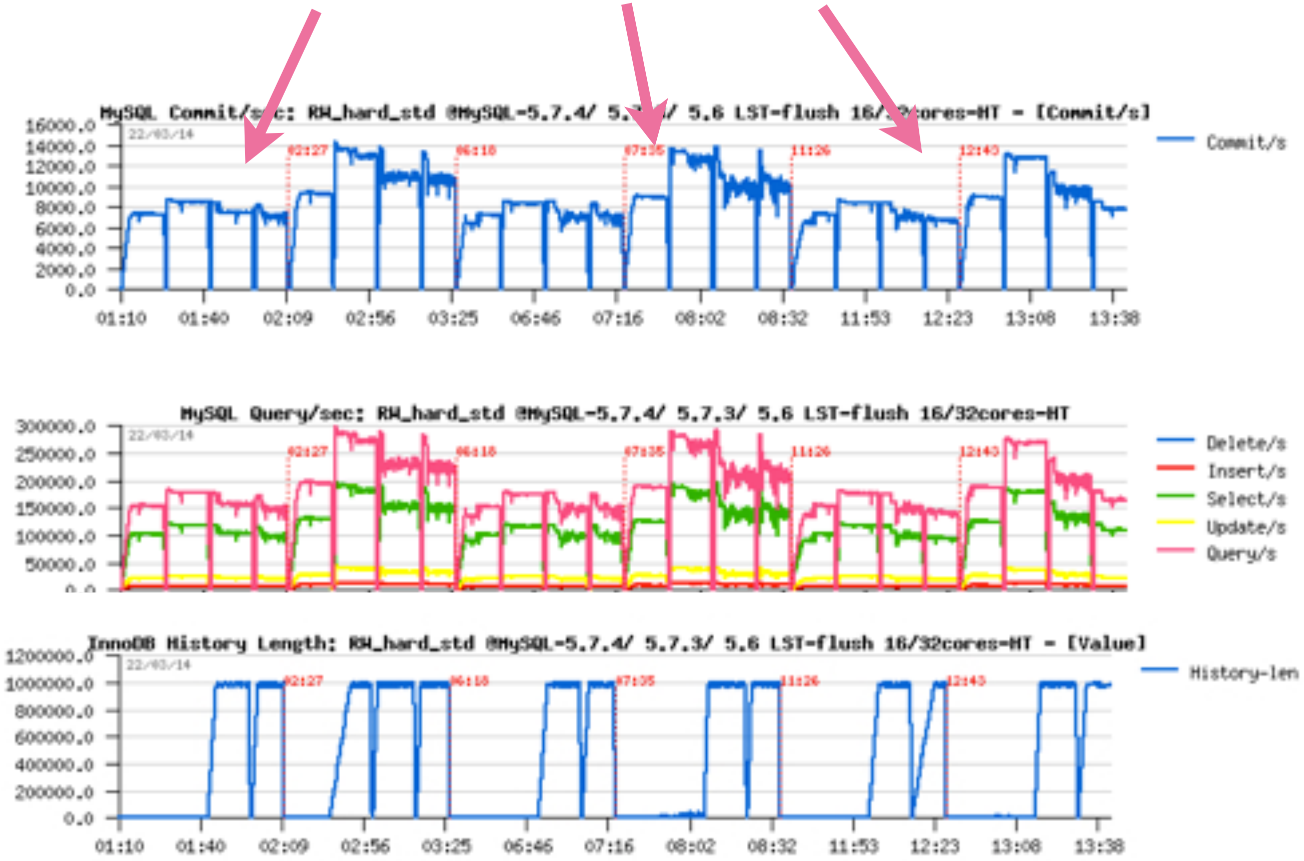
OLTP_RW 10Mx32-tab Uniform : LIST-bound

- Focus on : flush list (note: reaching 25K pages/sec on 5.7 now!)
- Engines: 5.7 latest, 5.7.3, 5.6



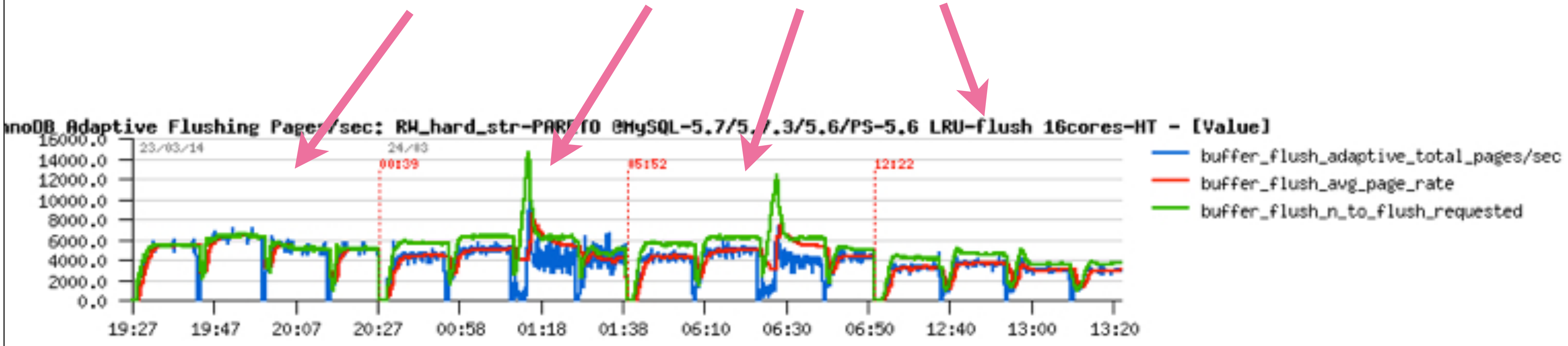
OLTP_RW 10Mx32-tab Uniform : LIST-bound

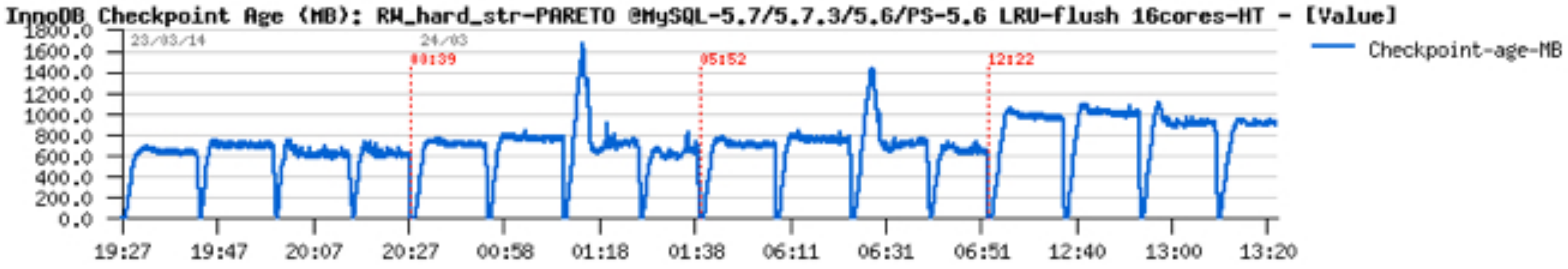
- Focus on : TPS / QPS / History Length impact
- Engines: 5.7 latest, 5.7.3, 5.6



OLTP_RW 10Mx32-tab Pareto : LRU-bound

- Focus on : flush list
- Engines: 5.7 latest, 5.7.3, 5.6, Percona 5.6

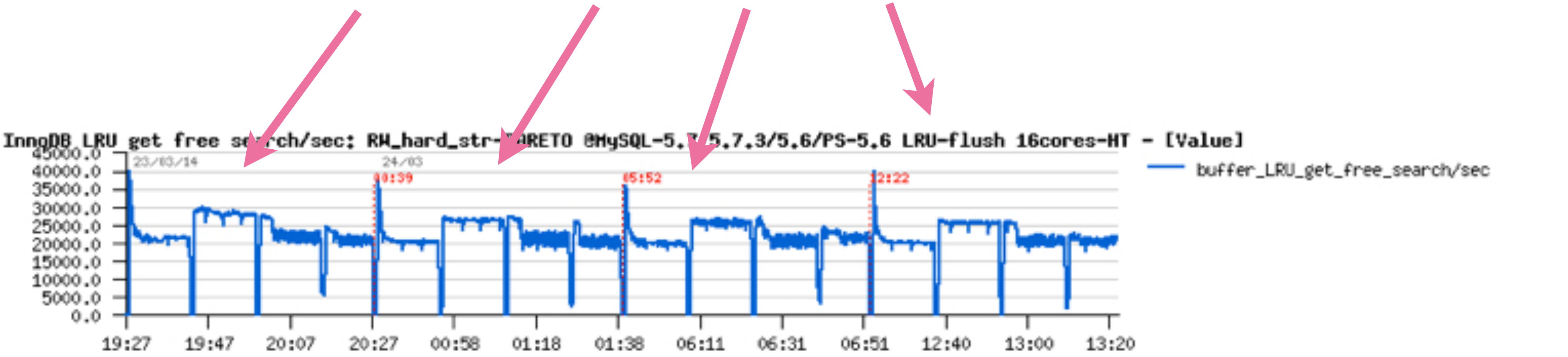


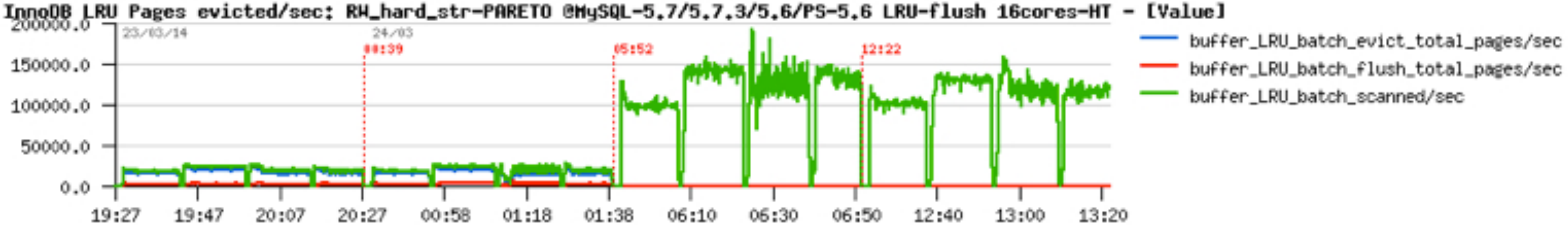


ORACLE

OLTP_RW 10Mx32-tab Pareto : LRU-bound

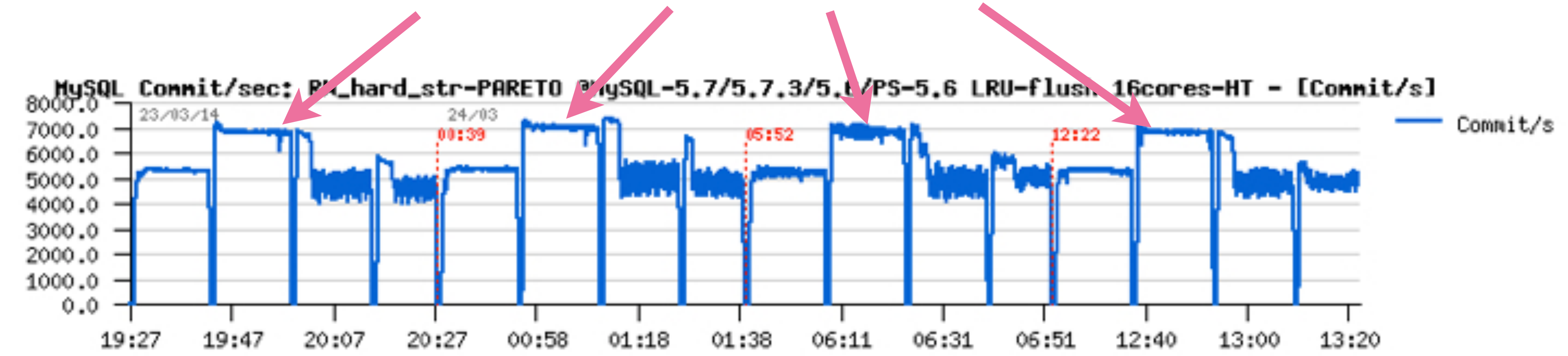
- Focus on : get free / LRU flushing
- Engines: 5.7 latest, 5.7.3, 5.6, Percona 5.6

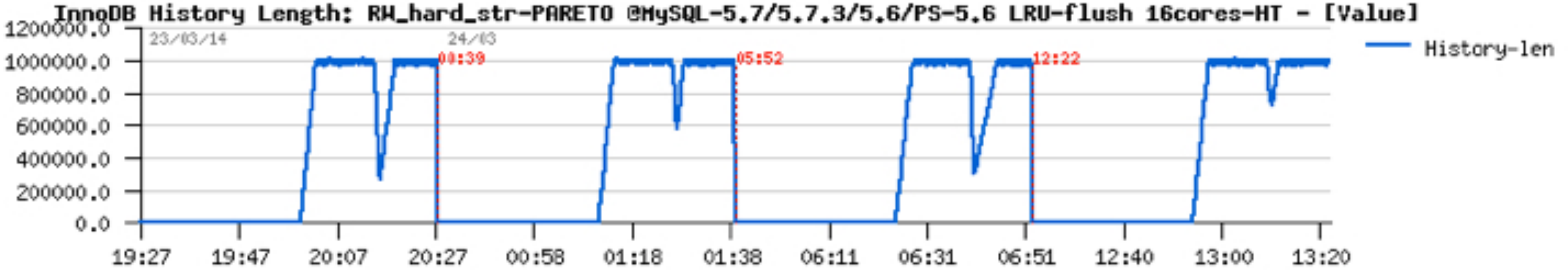




OLTP_RW 10Mx32-tab Pareto : LRU-bound

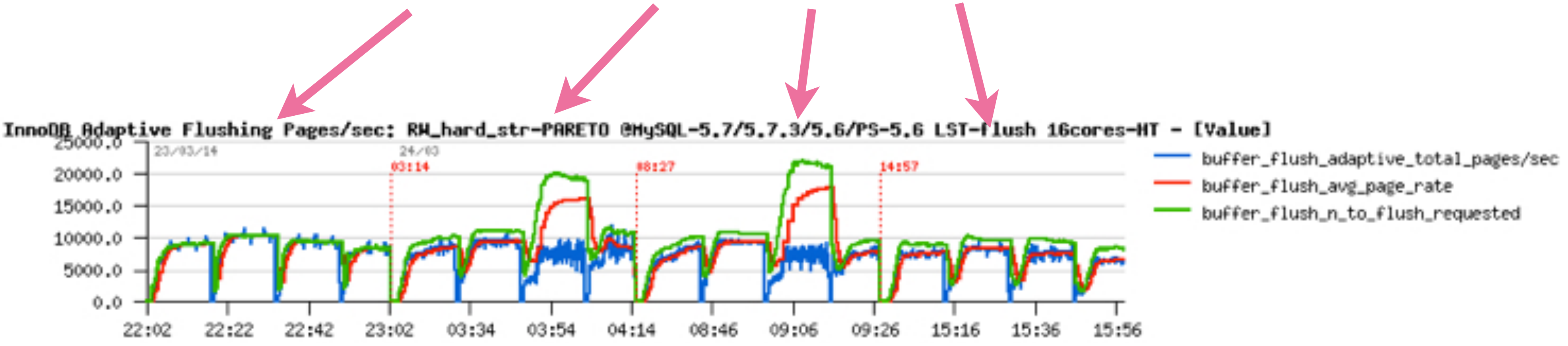
- Focus on : TPS / Purge lag
- Engines: 5.7 latest, 5.7.3, 5.6, Percona 5.6

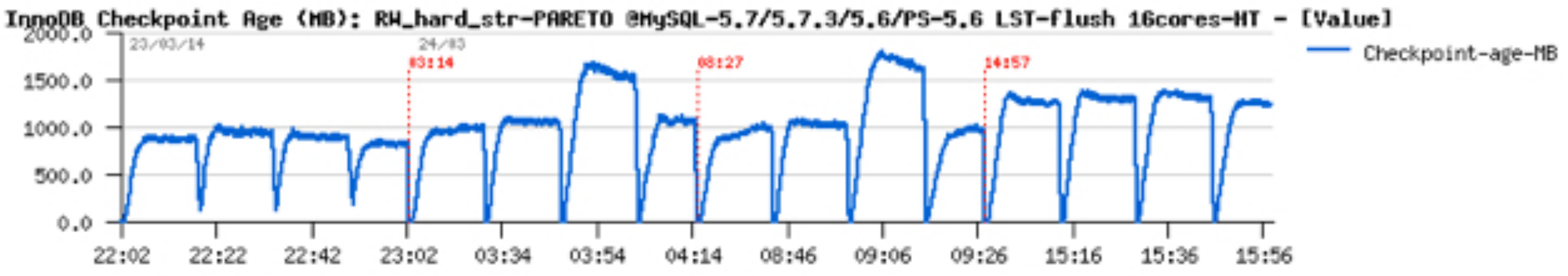




OLTP_RW 10Mx32-tab Pareto : LIST-bound

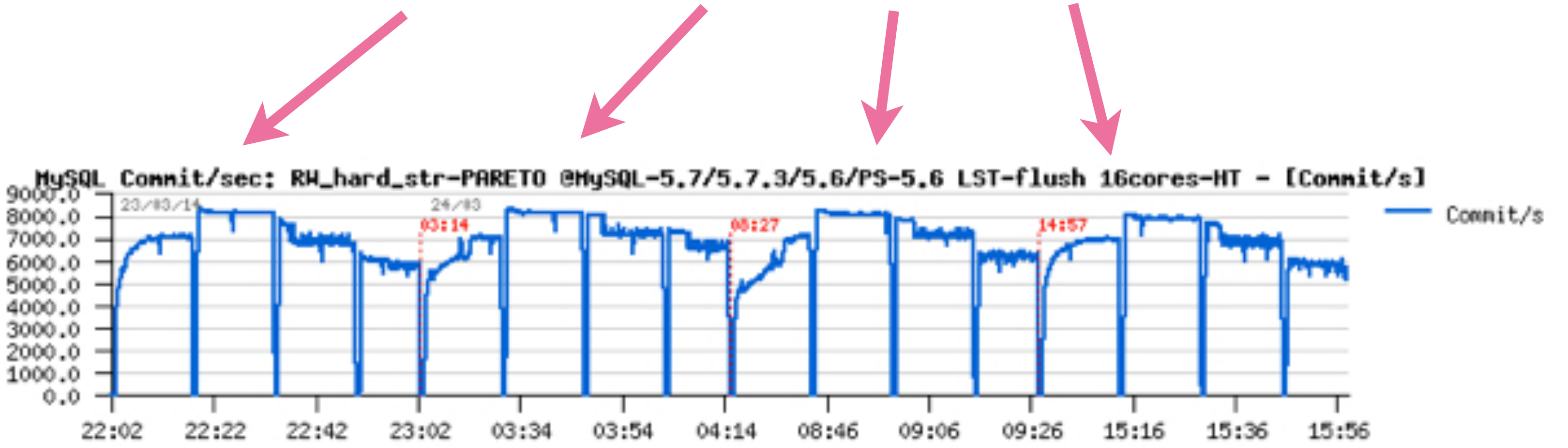
- Focus on : flush list
- Engines: 5.7 latest, 5.7.3, 5.6, Percona 5.6

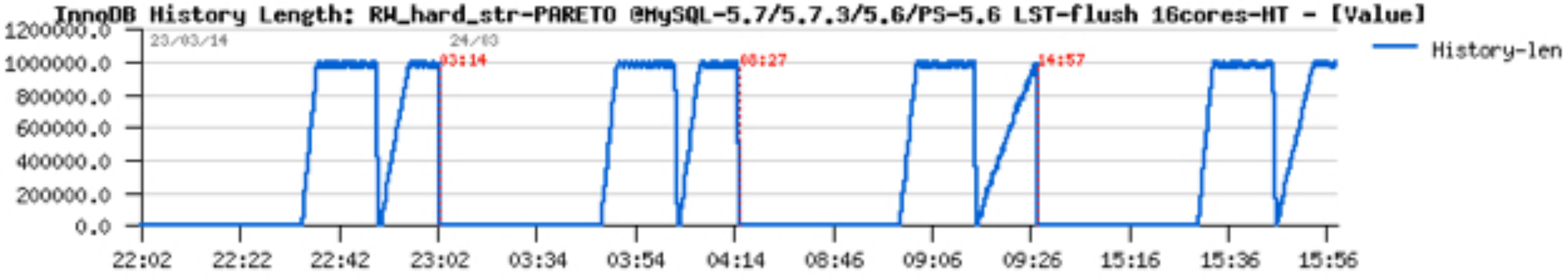




OLTP_RW 10Mx32-tab Pareto : LIST-bound

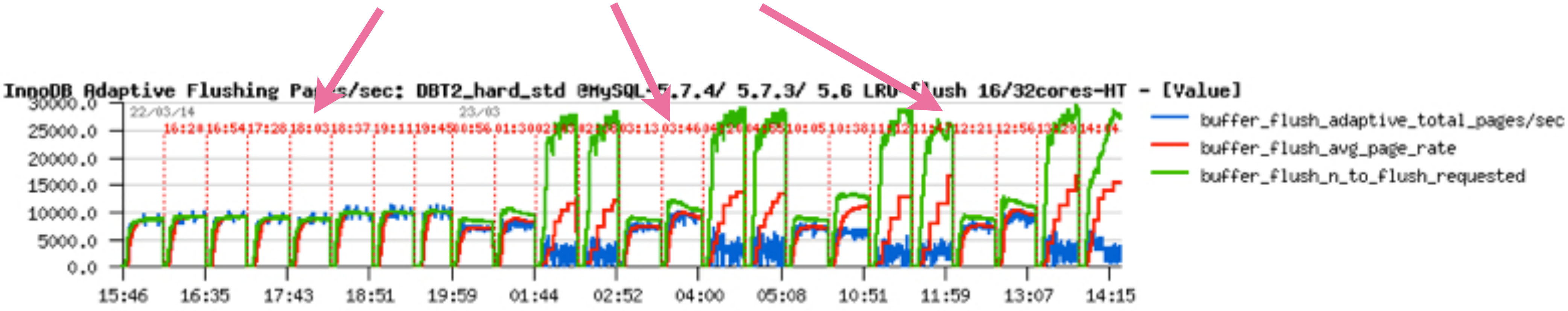
- Focus on : TPS / Purge lag
- Engines: 5.7 latest, 5.7.3, 5.6, Percona 5.6

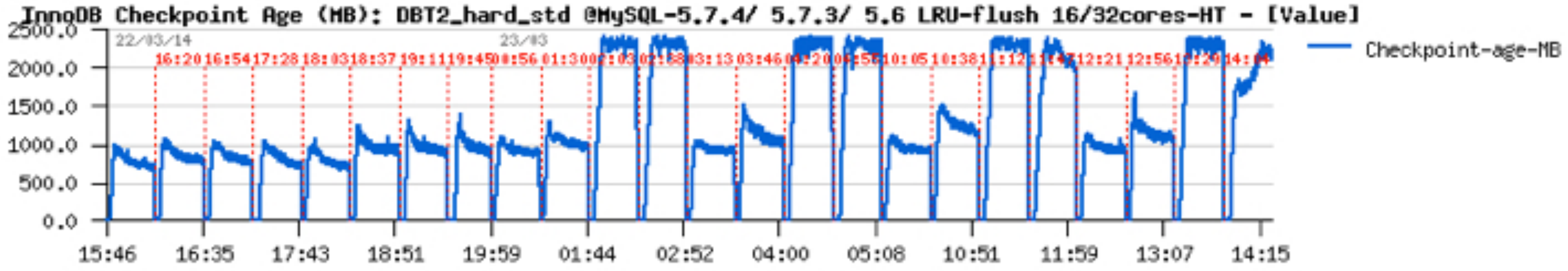




DBT2 512Wx8-db : LRU-bound

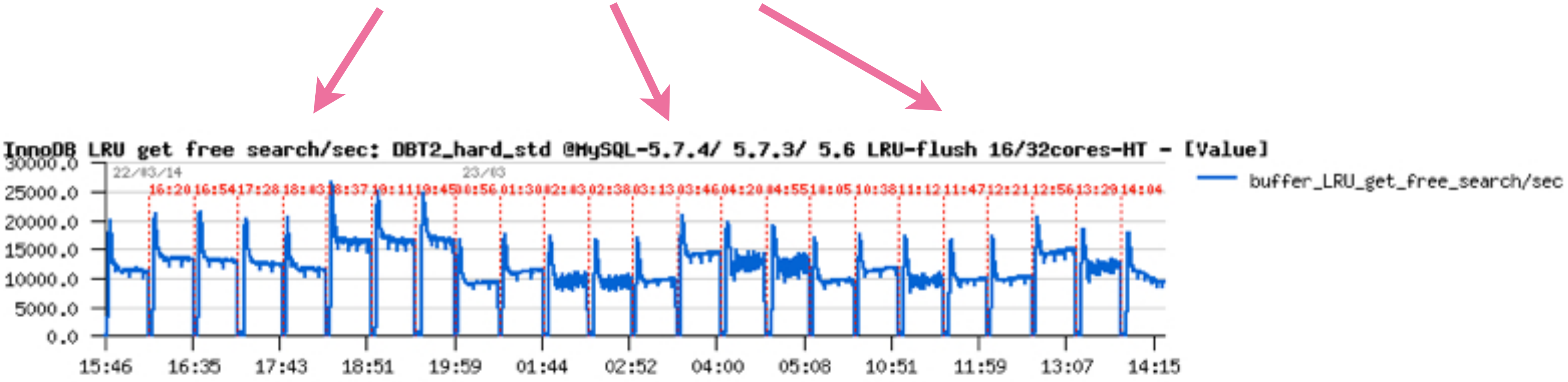
- Focus on : flush list
- Engines: 5.7 latest, 5.7.3, 5.6

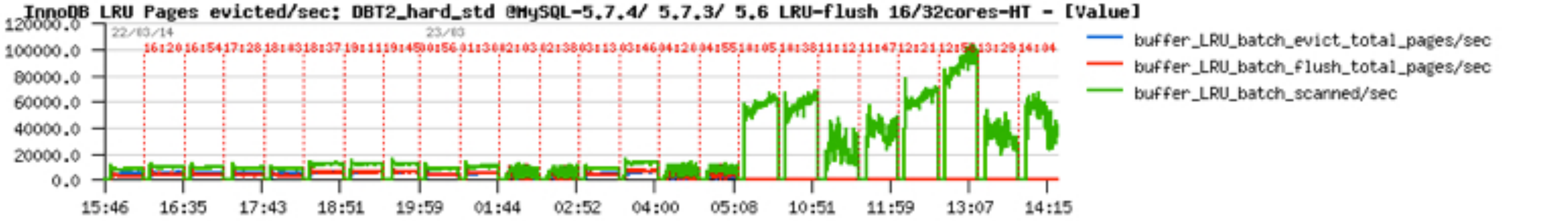




DBT2 512Wx8-db : LRU-bound

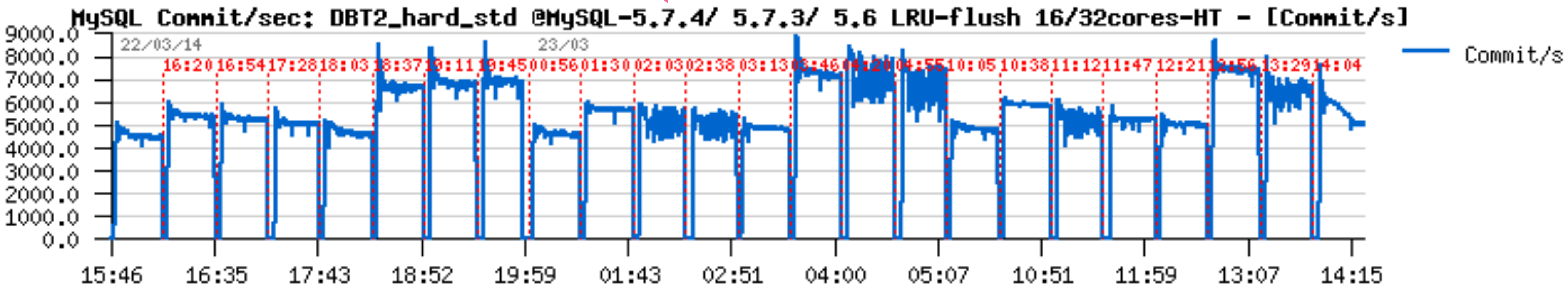
- Focus on : get free / page scan
- Engines: 5.7 latest, 5.7.3, 5.6





DBT2 512Wx8-db : LRU-bound

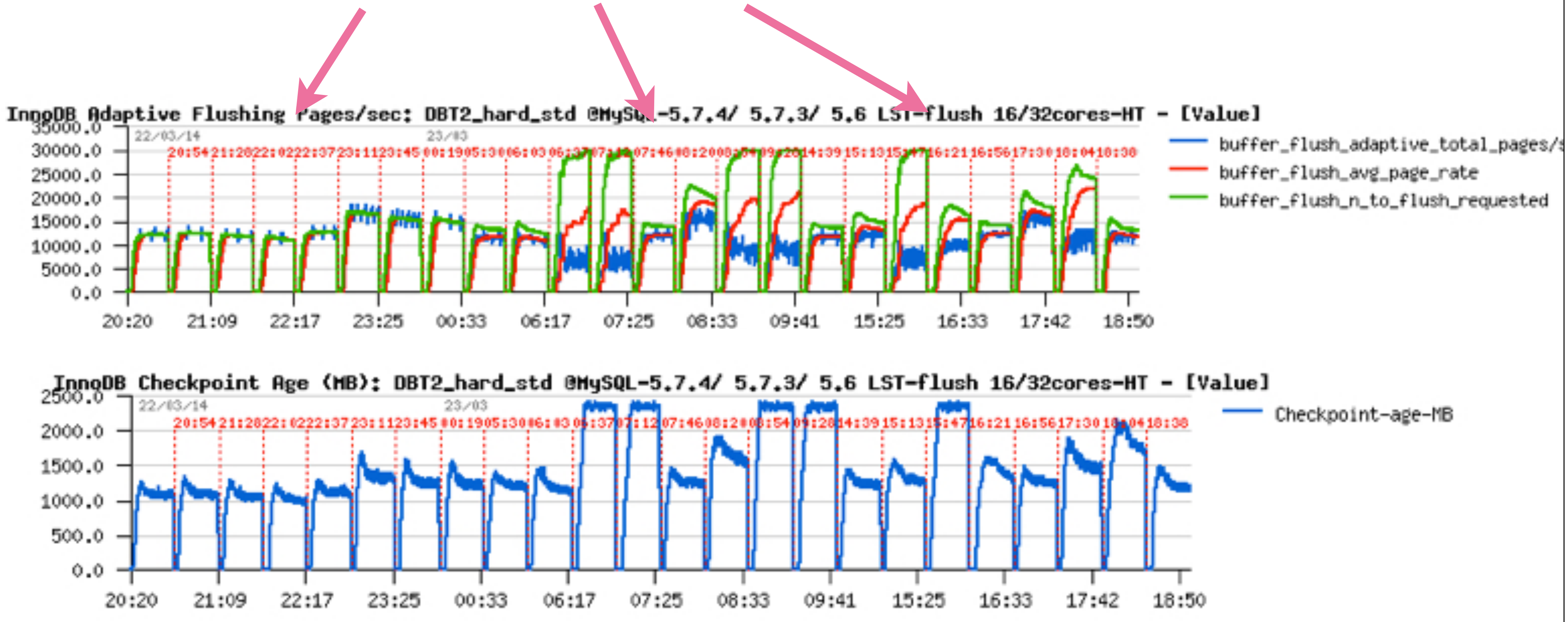
- Focus on : TPS
- Engines: 5.7 latest, 5.7.3, 5.6



- **Notes**:
  - no Purge lag = no TPS drop on 256 and 512 users..
  - innodb_thread_concurrency=64 is doing very well! ;-)

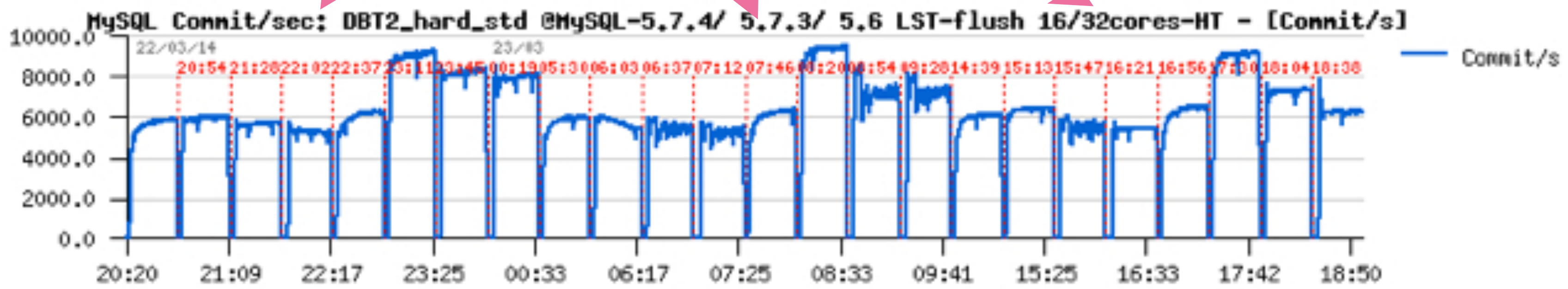
DBT2 512Wx8-db : LIST-bound

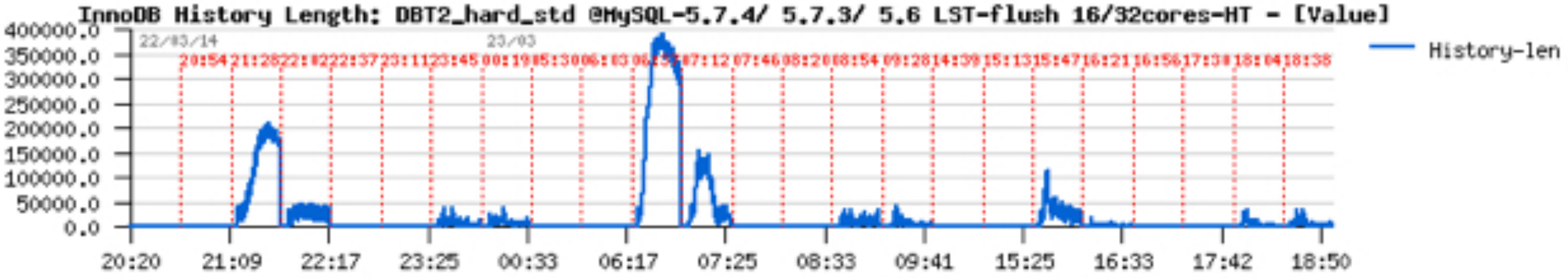
- Focus on : flush list
- Engines: 5.7 latest, 5.7.3, 5.6



DBT2 512Wx8-db : LIST-bound

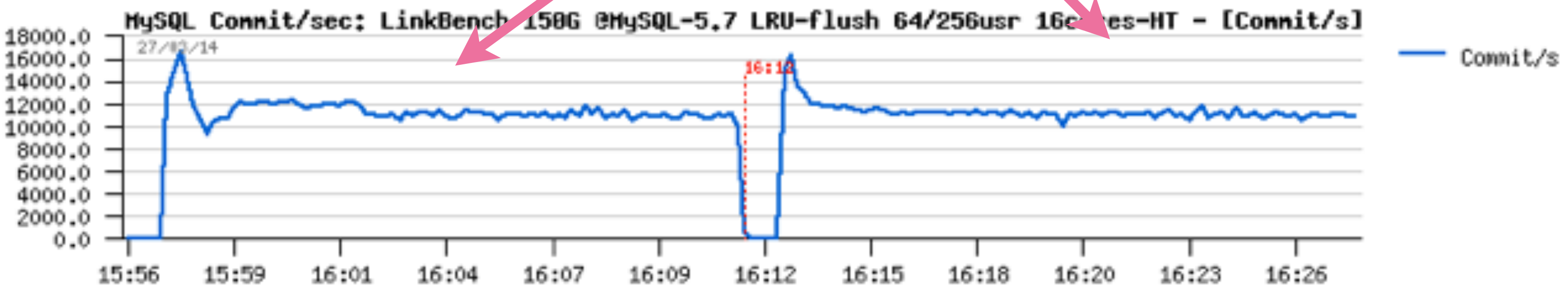
- Focus on : TPS (drops: see Checkpoint Age! not Purge Lag..)
- Engines: 5.7 latest, 5.7.3, 5.6

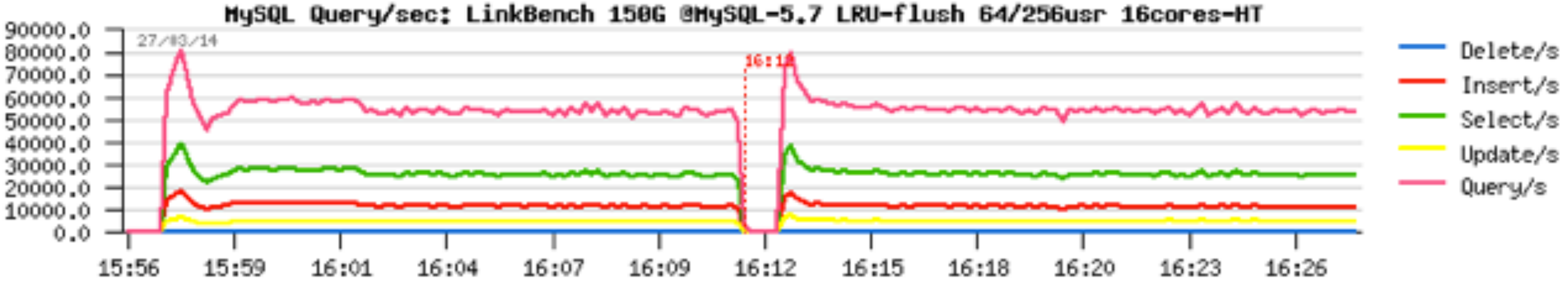




LinkBench 150G: LRU-bound

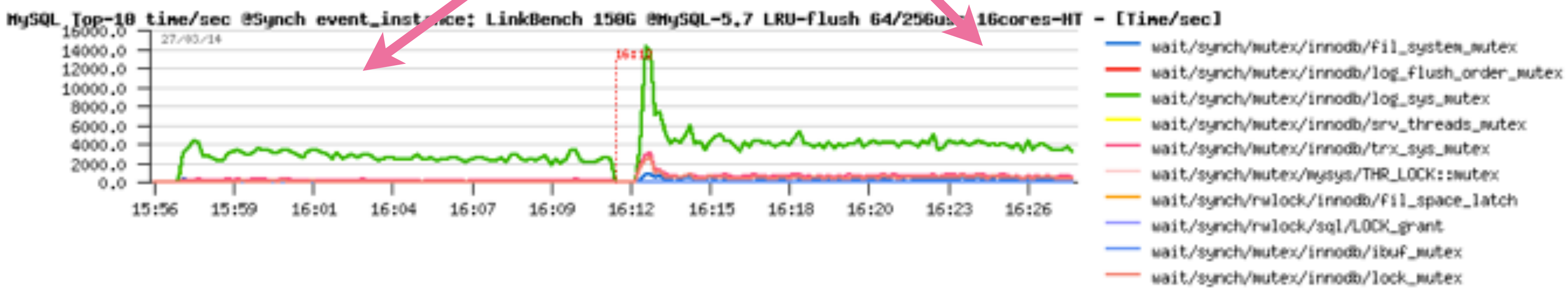
- Focus on : TPS
- Engines: 5.7 latest, 64 users / 256 users

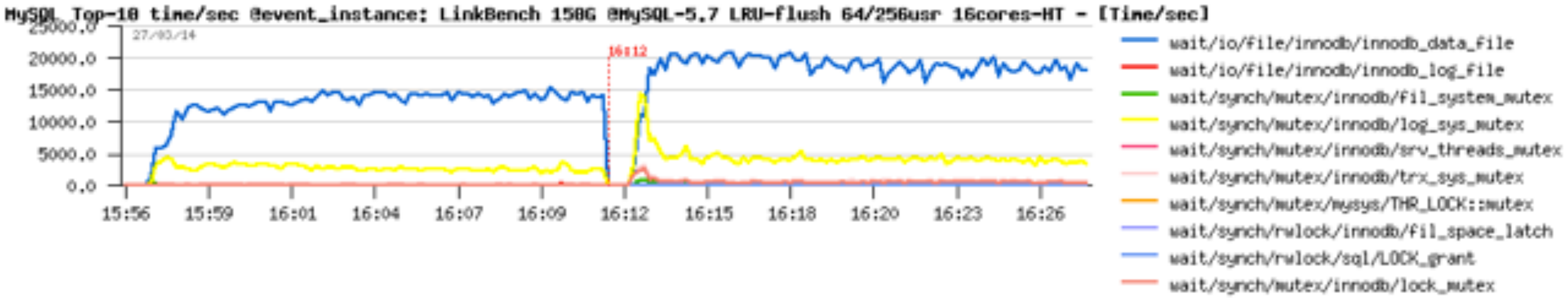




LinkBench 150G: LRU-bound

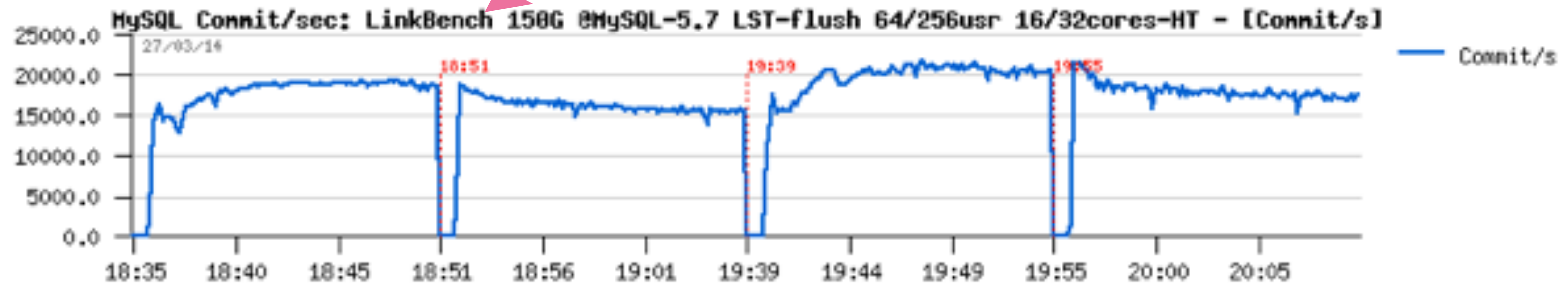
- Focus on : Lock contentions...
- Engines: 5.7 latest, 64 users / 256 users

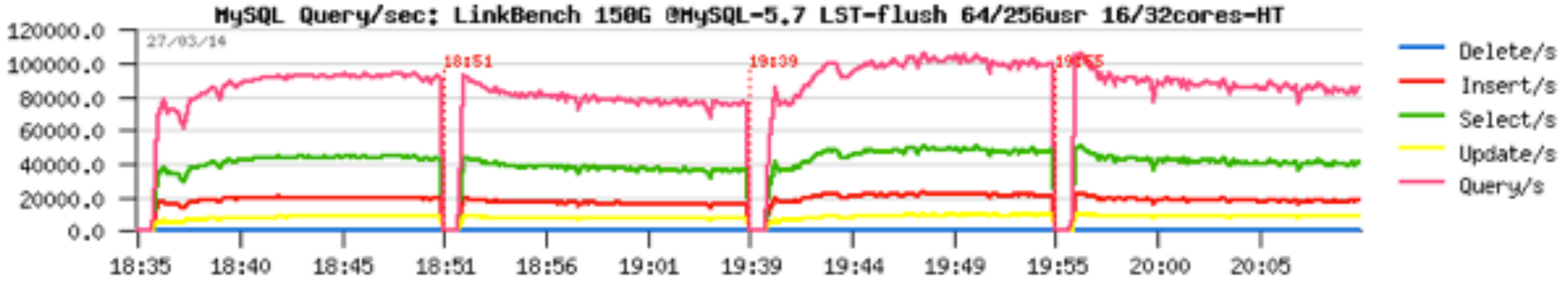




LinkBench 150G: Less LRU-bound (BP=96G)

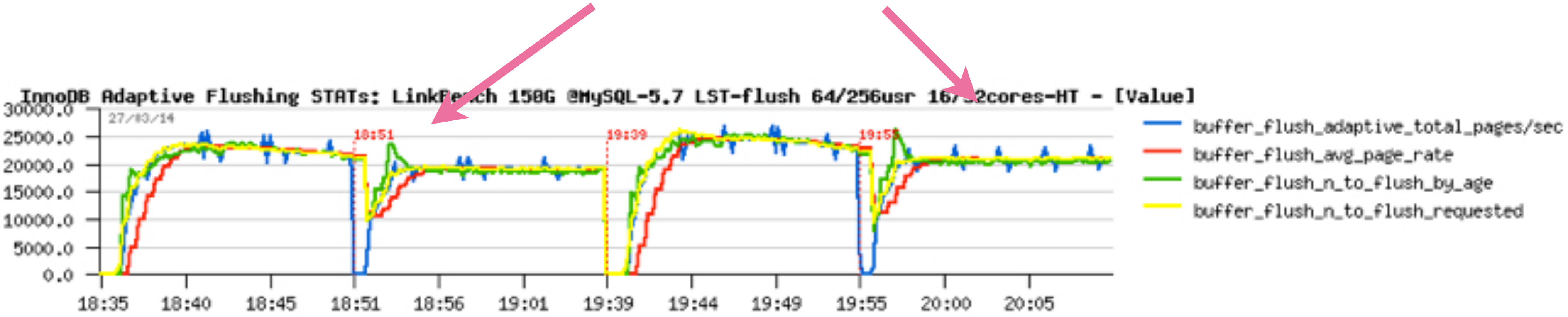
- Focus on : TPS / QPS
- Engines: 5.7 latest, 64 users / 256 users on 16/32cores-HT

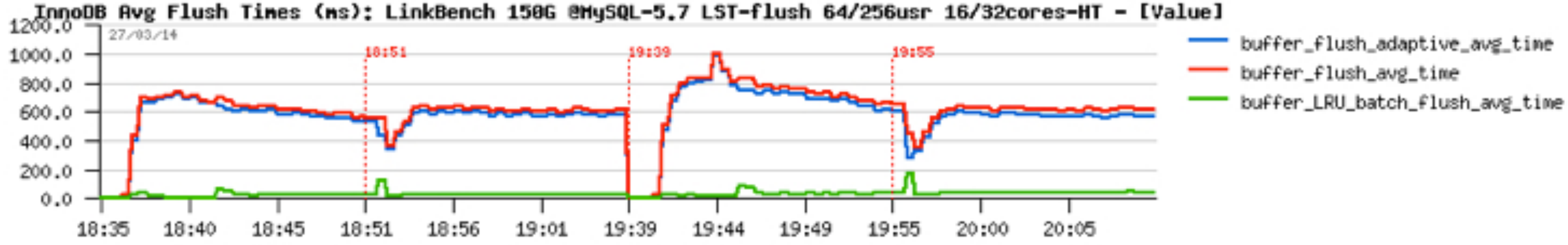




LinkBench 150G: Less LRU-bound (BP=96G)

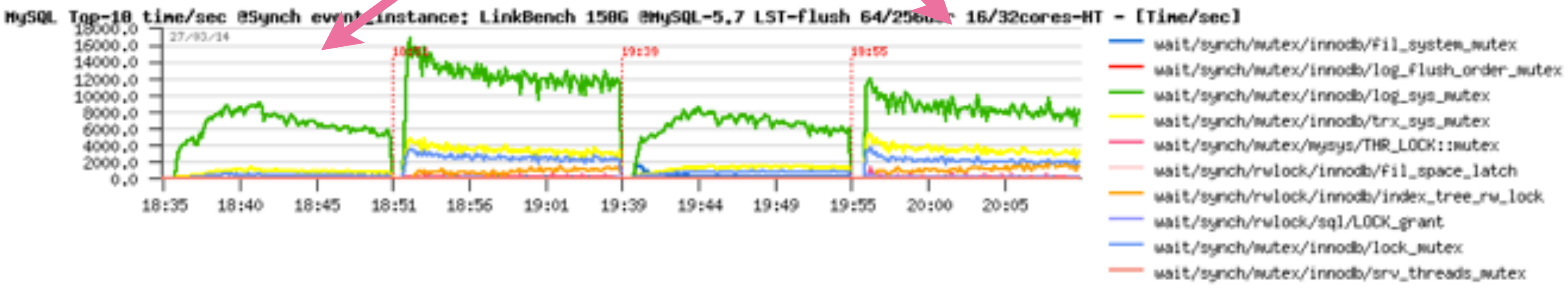
- Focus on : flush list rate / time
- Engines: 5.7 latest, 64 users / 256 users on 16/32cores-HT



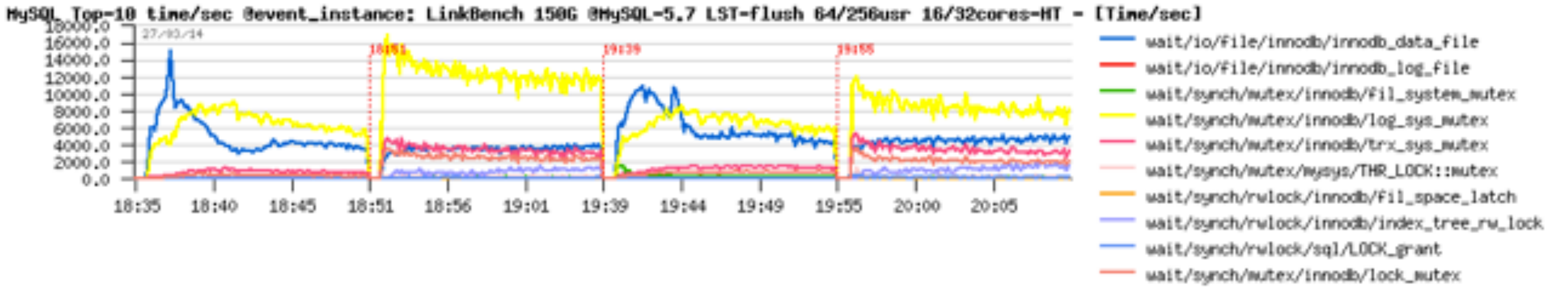


LinkBench 150G: Less LRU-bound (BP=96G)

- Focus on : Lock contentions
- Engines: 5.7 latest, 64 users / 256 users on 16/32cores-HT



MySQL Top-10 time/sec @Synch event_instance: LinkBench 150G @MySQL-5.7 LST-flush 64/256usr 16/32cores-HT - [Time/sec]



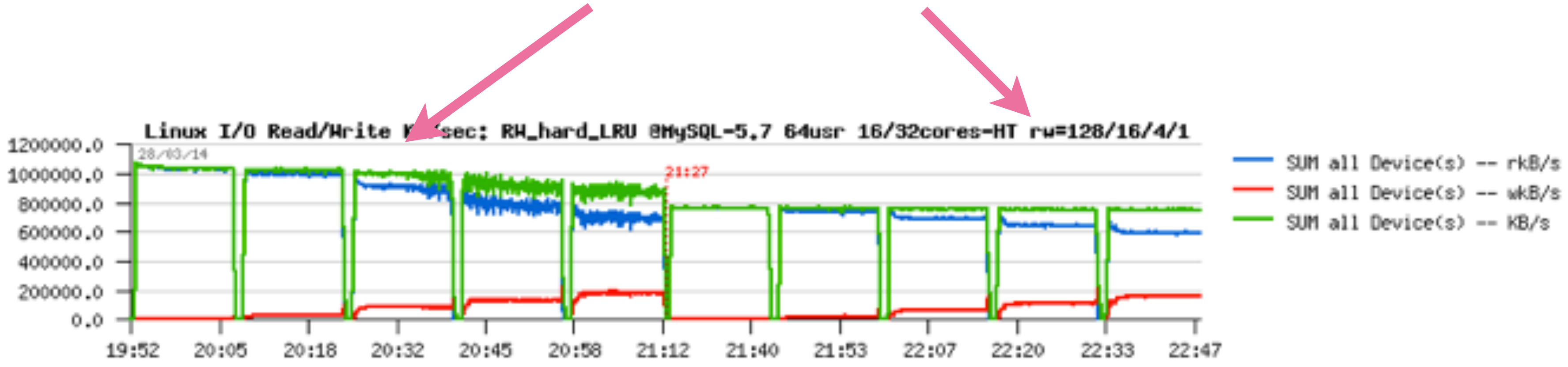
MySQL Top-10 time/sec @event_instance: LinkBench 150G @MySQL-5.7 LST-flush 64/256usr 16/32cores-HT - [Time/sec]

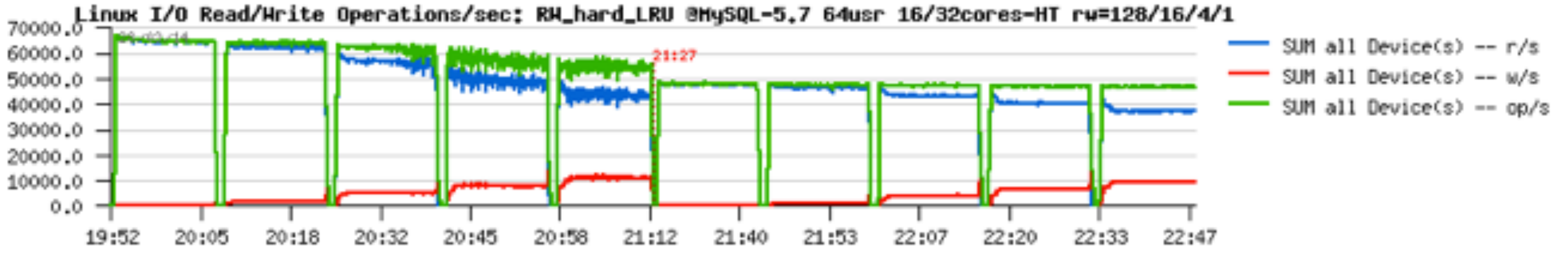
The RW IO-bound “mystery”..

- 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”

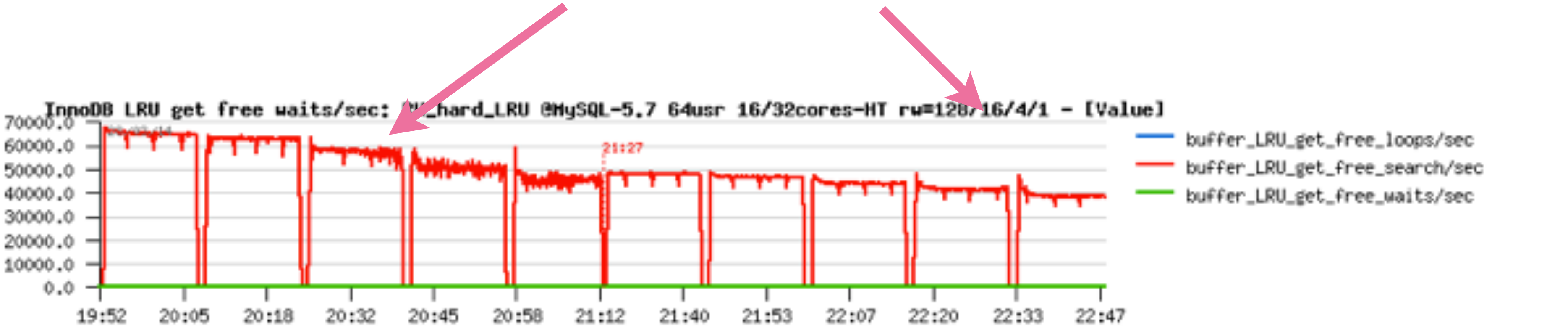
- Focus on : I/O stats
- Engines: 5.7 latest, 16cores-HT / 32cores-HT

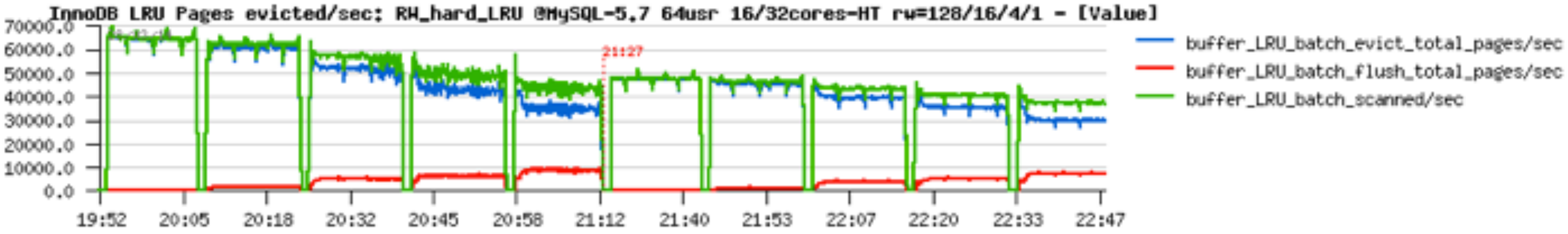




The RW IO-bound “mystery”

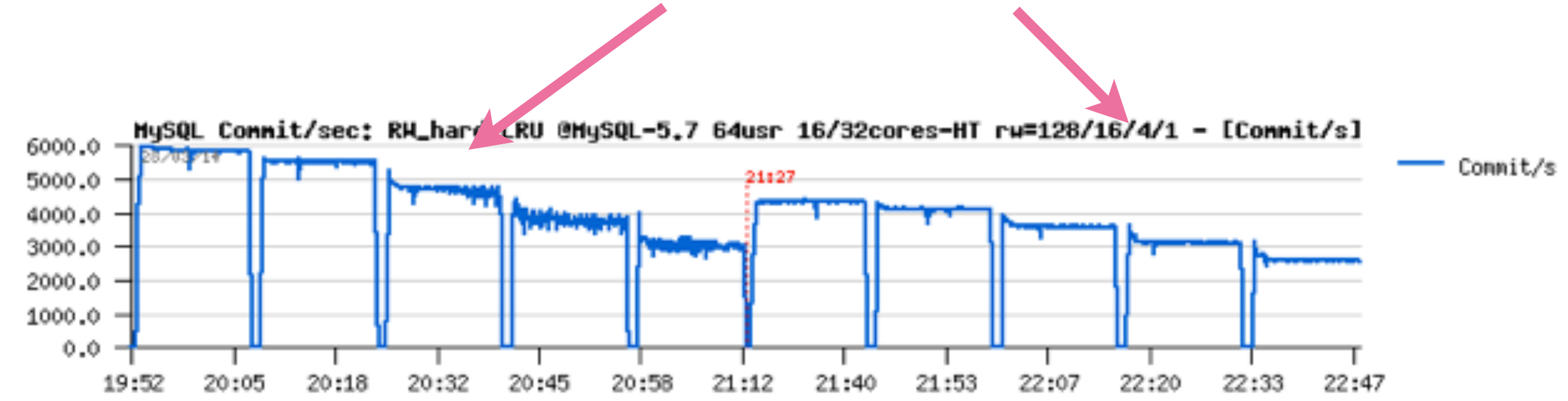
- Focus on : LRU stats
- Engines: 5.7 latest, 16cores-HT / 32cores-HT

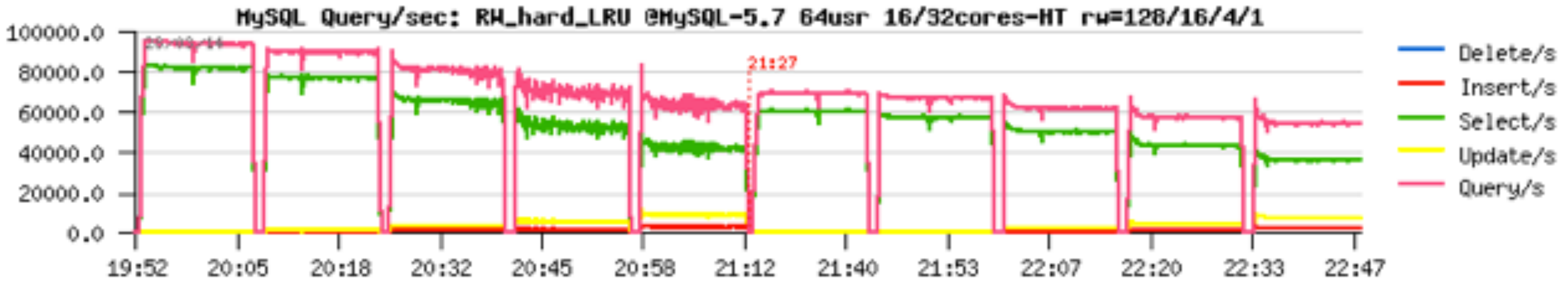




The RW IO-bound “mystery”

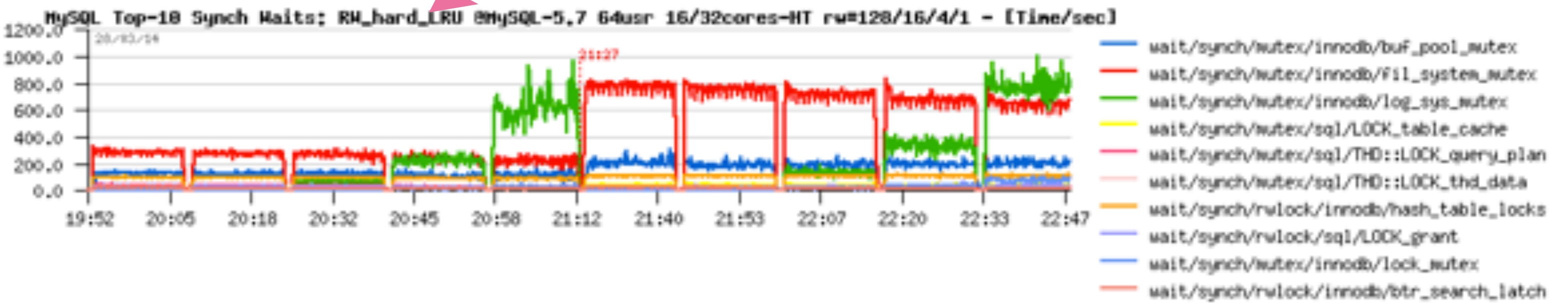
- Focus on : TPS / QPS (note: x2 times worse at the end!!)
- Engines: 5.7 latest, 16cores-HT / 32cores-HT

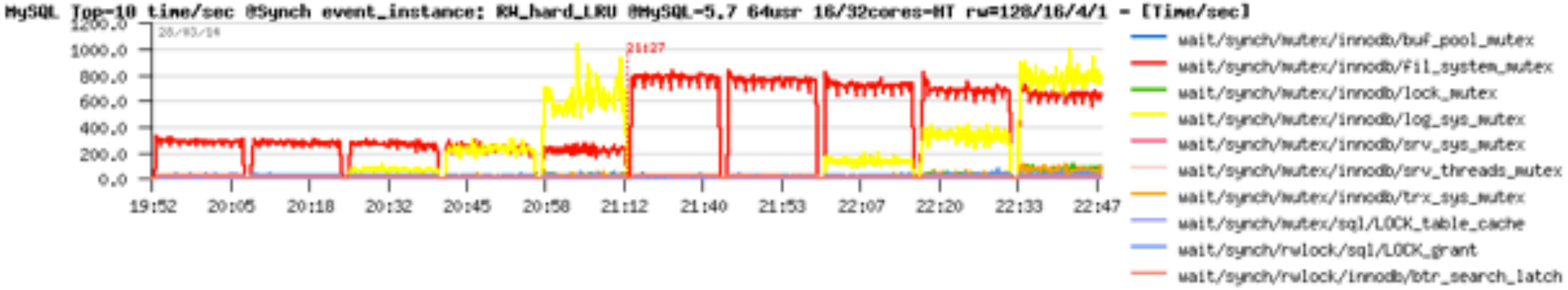




The RW IO-bound “mystery”

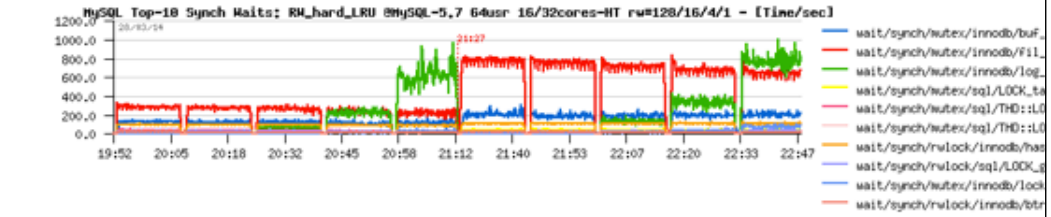
- Focus on : Lock contentions... (note: killing fil_sys + log_sys)
- Engines: 5.7 latest, 16cores-HT / 32cores-HT

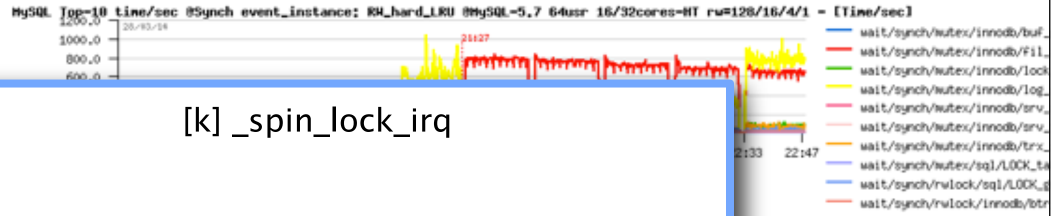




The RW IO-bound “mystery”

- 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
```

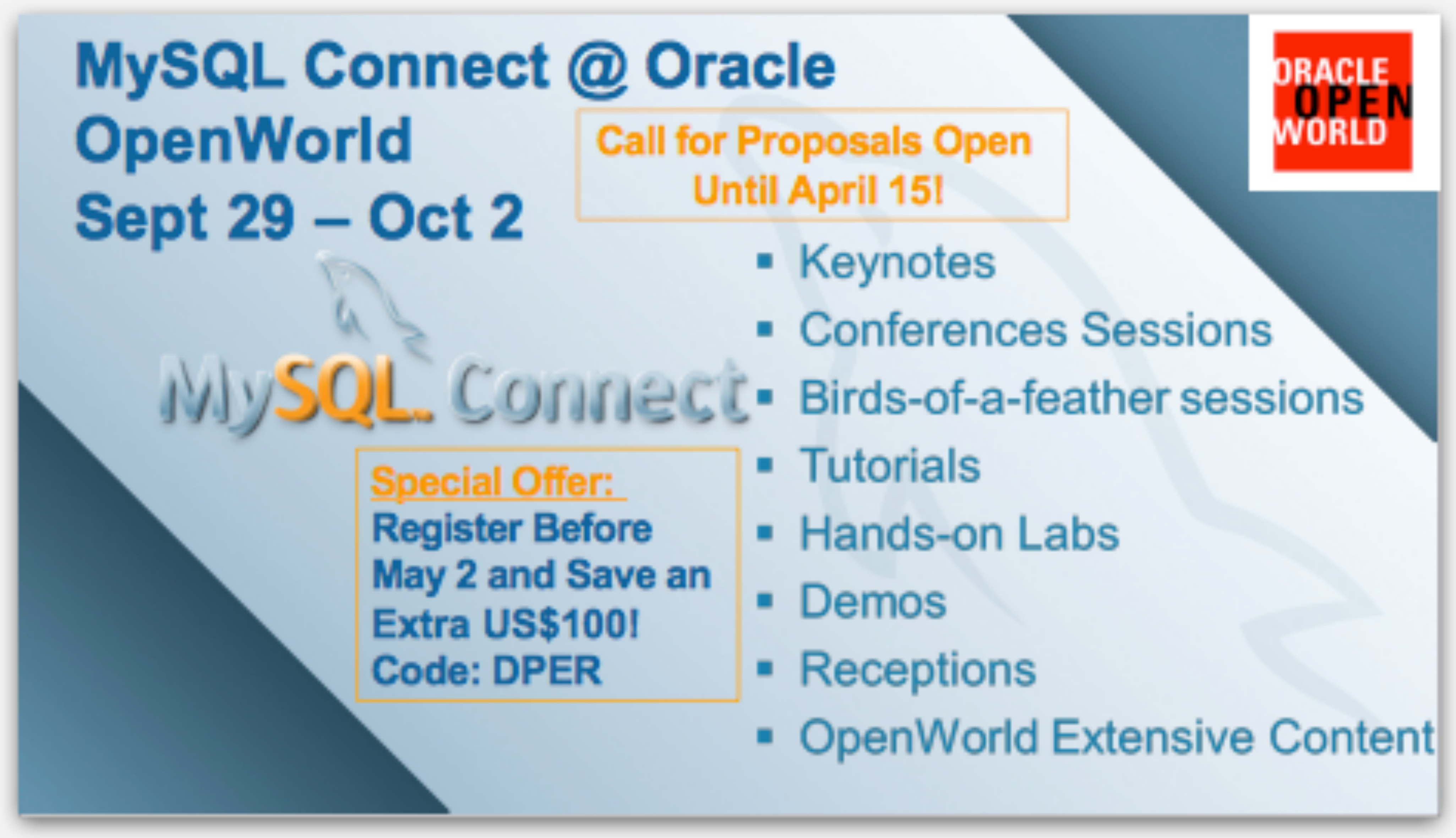
So, work continues..
stay tuned... ;-)

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.



MySQL Connect @ Oracle OpenWorld

Sept 29 – Oct 2

Call for Proposals Open Until April 15!

- Keynotes
- Conferences Sessions
- Birds-of-a-feather sessions
- Tutorials
- Hands-on Labs
- Demos
- Receptions
- OpenWorld Extensive Content

Special Offer:
Register Before May 2 and Save an Extra US$100!
Code: DPER

ORACLE
OPEN
WORLD
MySQL. Connect