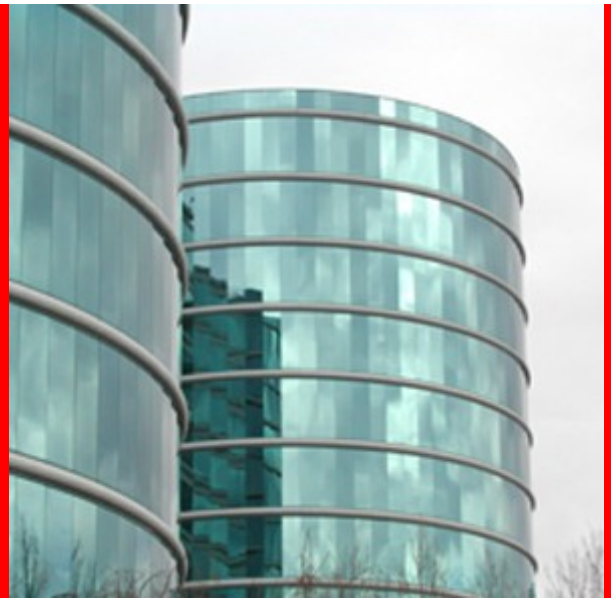


What's new in MySQL 5.5? Performance and Scalability Benchmarks

Mikael Ronström
Senior MySQL Architect

The preceding 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.

Outline of talk

- Analysis of MySQL Server and InnoDB changes
- Analysis of important InnoDB configuration parameters
- Analysis of Partitioning as Performance Booster
- Impact of Powersave mode on Benchmarks
- Scalability Analysis of MySQL 5.5.4-m3
- Analysis of behaviour of MySQL 5.5.4-m3

Log_sys mutex Improvement

- Zero impact on read-only benchmarks
- Improves performance by 5% when added to MySQL 5.5.3-m3 baseline on 16-core MySQL Server
- SHOW ENGINE INNODB MUTEX shows that log_sys mutex have decreased contention
- The new log_flush_order mutex have very little contention
- Combined with other patches the impact is smaller most likely

Split Buffer Pool into multiple instances

- Sysbench RW on 16-cores improves 10%
- Works very well together with Split Rollback Segment Mutex
- Improves Read Only performance as well
- Very large improvement on 32-core/threaded

Split out Buffer Pool Page Hash into array of mutexes

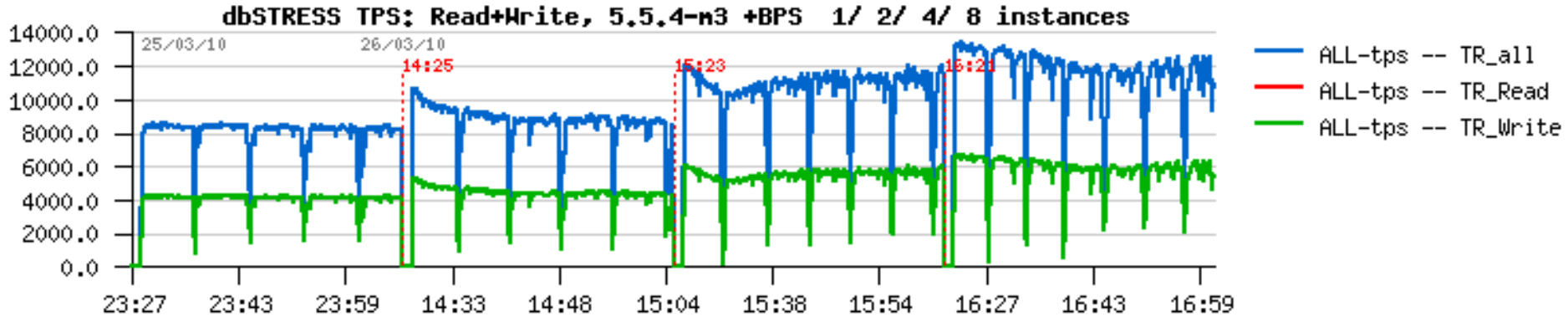
- About 10% improvement of Sysbench RW on 16-core
- No improvement or even decrease of Read Only performance
- Very good scalability on 32-core/thread

Split-out Page Hash vs. Multiple Buffer Pool instances

- Multiple Buffer Pool instances better Read Only performance
- Multiple Buffer Pool instances can combined with split-out page hash later if it makes sense
- Multiple Buffer Pool instances worked better on 32-core servers

Analysis of new InnoDB Buffer Pool instances

- –innodb-buffer-pool-instances=x



Split-out Flush List from Buffer Pool mutex

- No impact of Read Only performance
- A few percent improvement of ReadWrite workloads

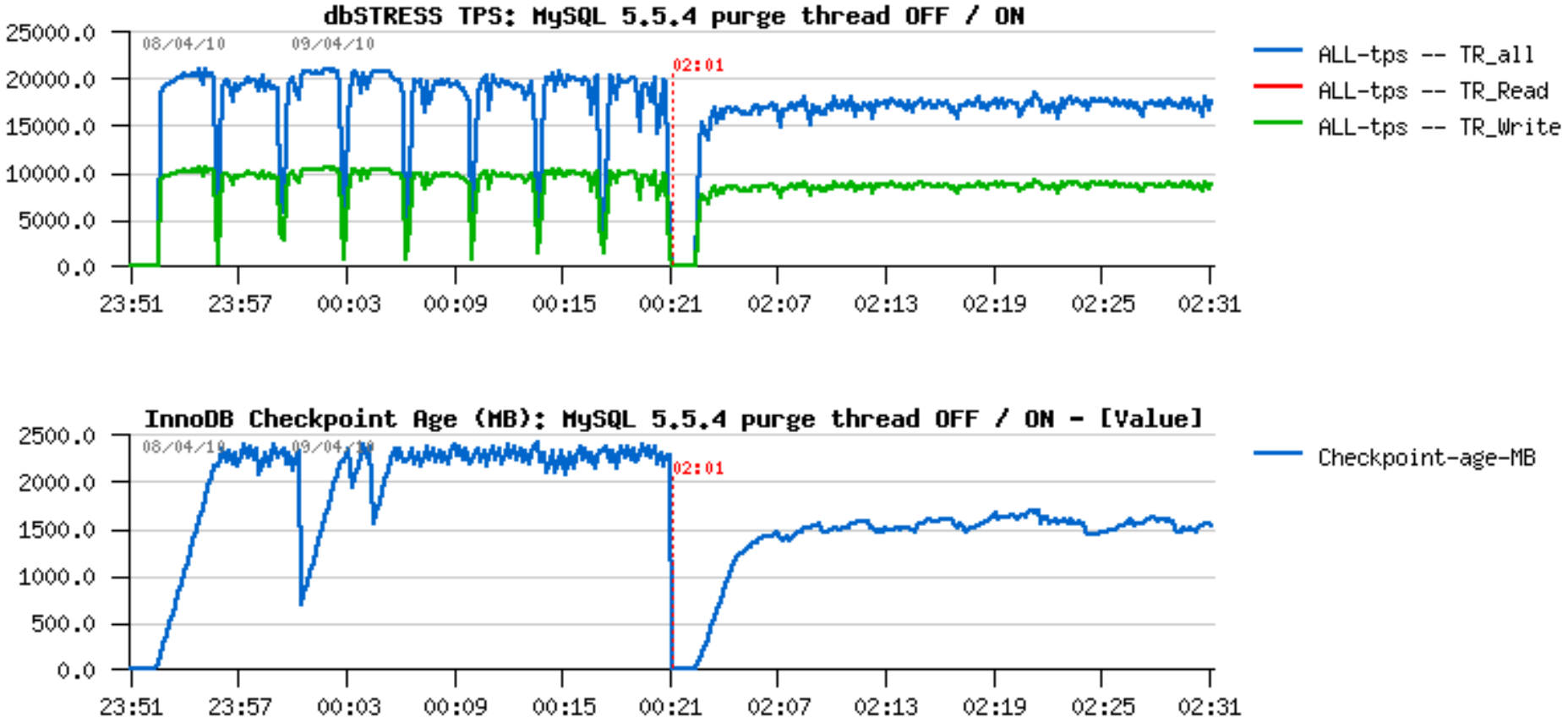
Split Rollback Segment into 128 Rollback Segments

- Combined with multiple buffer pool instances works very well and gives dramatic improvements on 32-core servers

Split Purge Thread into separate thread from Master Thread

- Required to get stable performance
- Mean performance slightly impacted positively or negatively
- Max performance decreases
- Min performance significantly increases
- Higher mean performance can often happen due to History Length continously increasing, will eventually lead to out of disk space

dbSTRESS: Read+Write & Purge Thread



Remove LOCK_alarm mutex

- Standalone improved 2%, improved both Read Only and Read Write

Improvement of LOCK_open, step 1, remove hash calculation from LOCK_open

- Improved performance of ReadOnly/ReadWrite a few percent

Remove many uses of LOCK_thread_count

- Standalone no improvement
- Combined with LOCK_open improvement and LOCK_alarm it improved ReadOnly/ReadWrite a few percent

Introduction of MDL locking framework

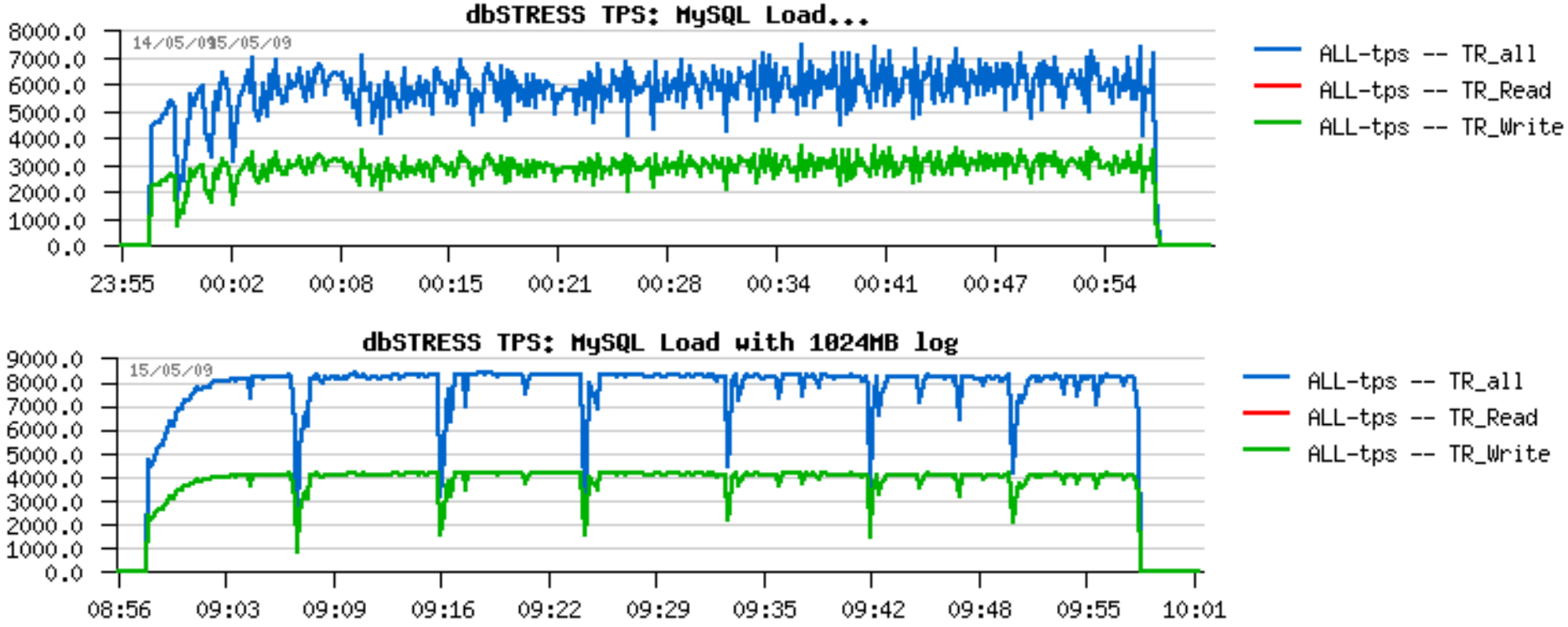
- Removed drop at high number threads due to change of how LOCK_open gets TABLE objects
- Improved performance a few percent

Improvement of LOCK_open based on MDL framework

- Improved performance a few percent

Analysis of InnoDB Log File Size

- 128MB => 1024MB: 6000 TPS => 8000 TPS



Analysis of InnoDB Log Buffer Memory

- No specific benchmarks executed
- Large buffer means less contention on log_sys mutex

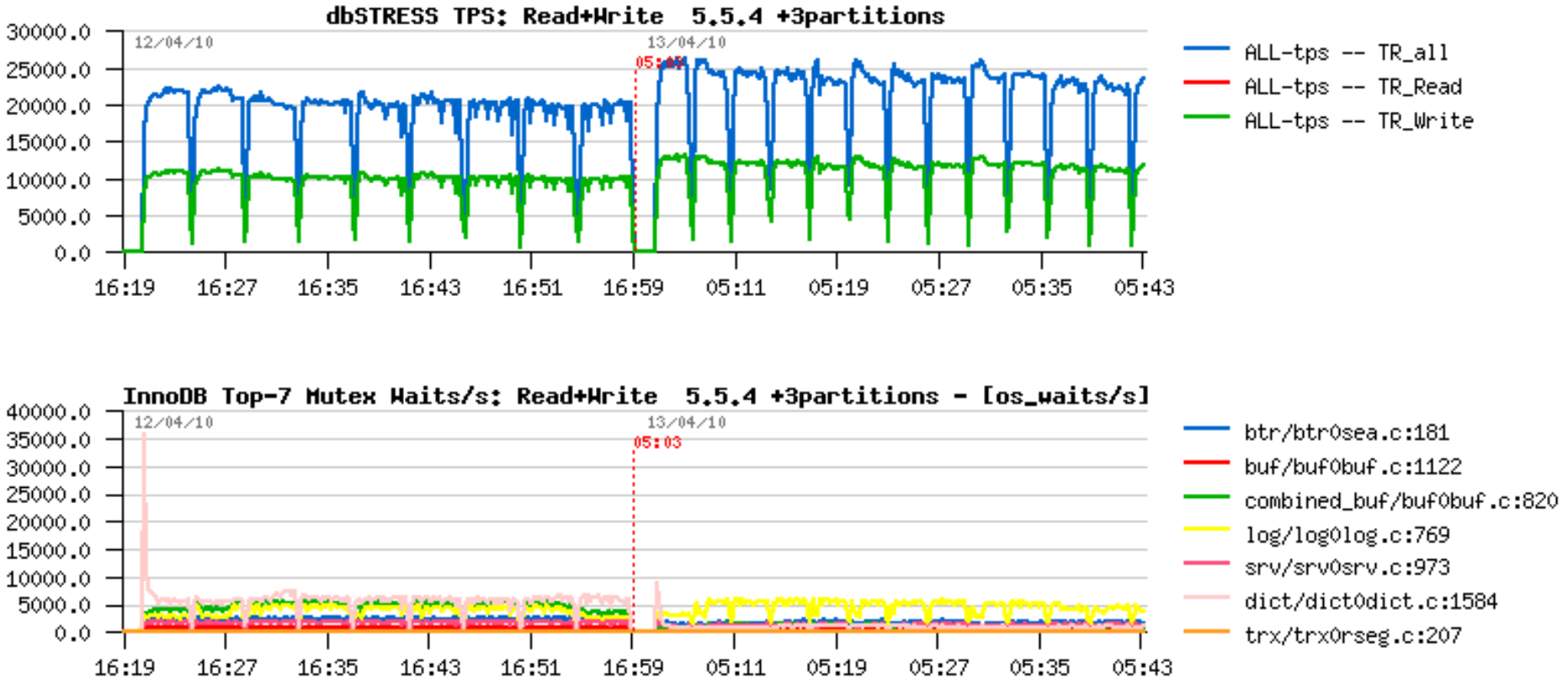
Analysis of use of InnoDB Adaptive Hash

- For Sysbench RO/RW on 16-cores improved performance by about 3-4% to not activate it

Analysis of Partitioning as Performance Booster

- Improves performance by splitting the InnoDB Index mutex

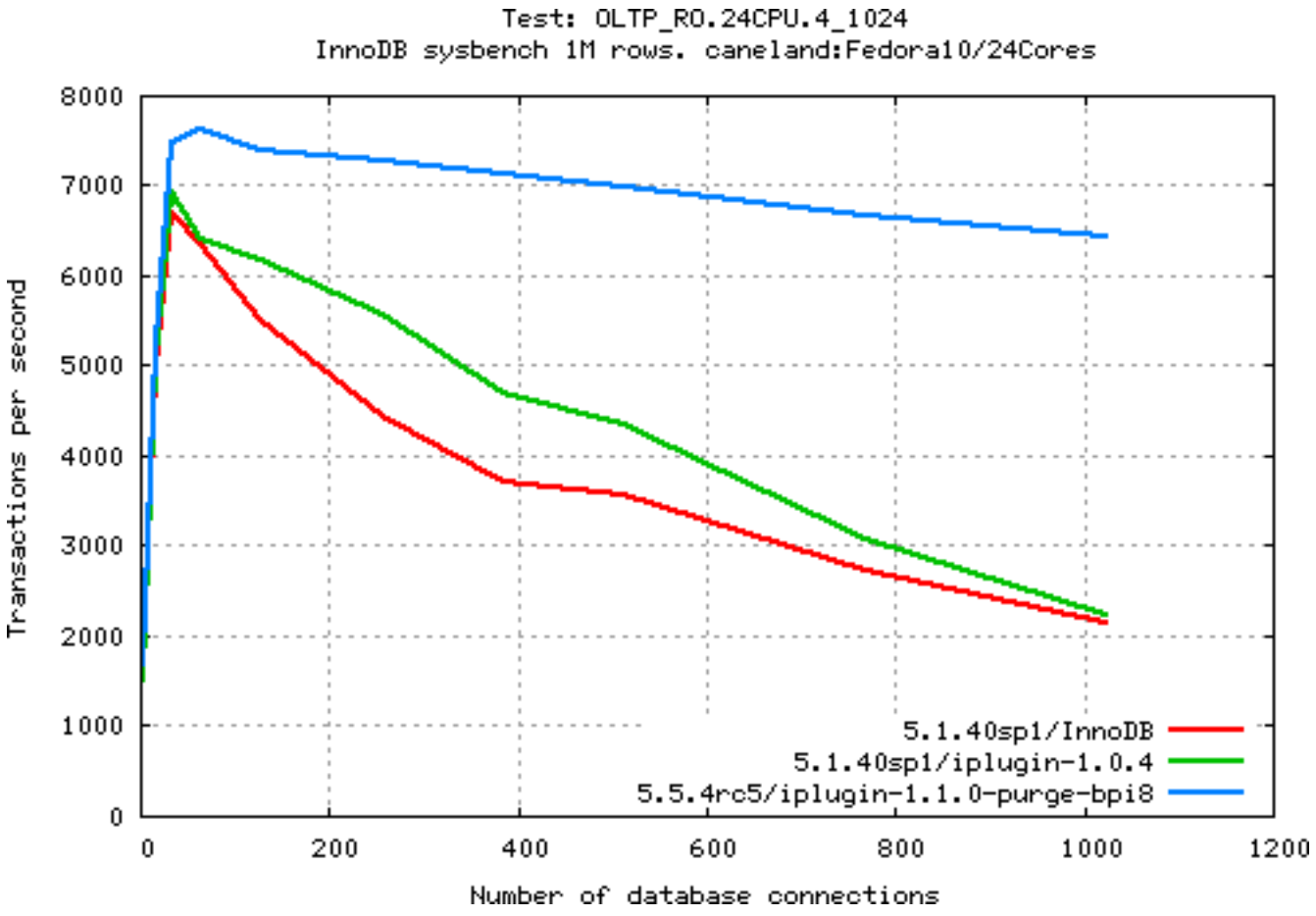
dbSTRESS: Using Partitions



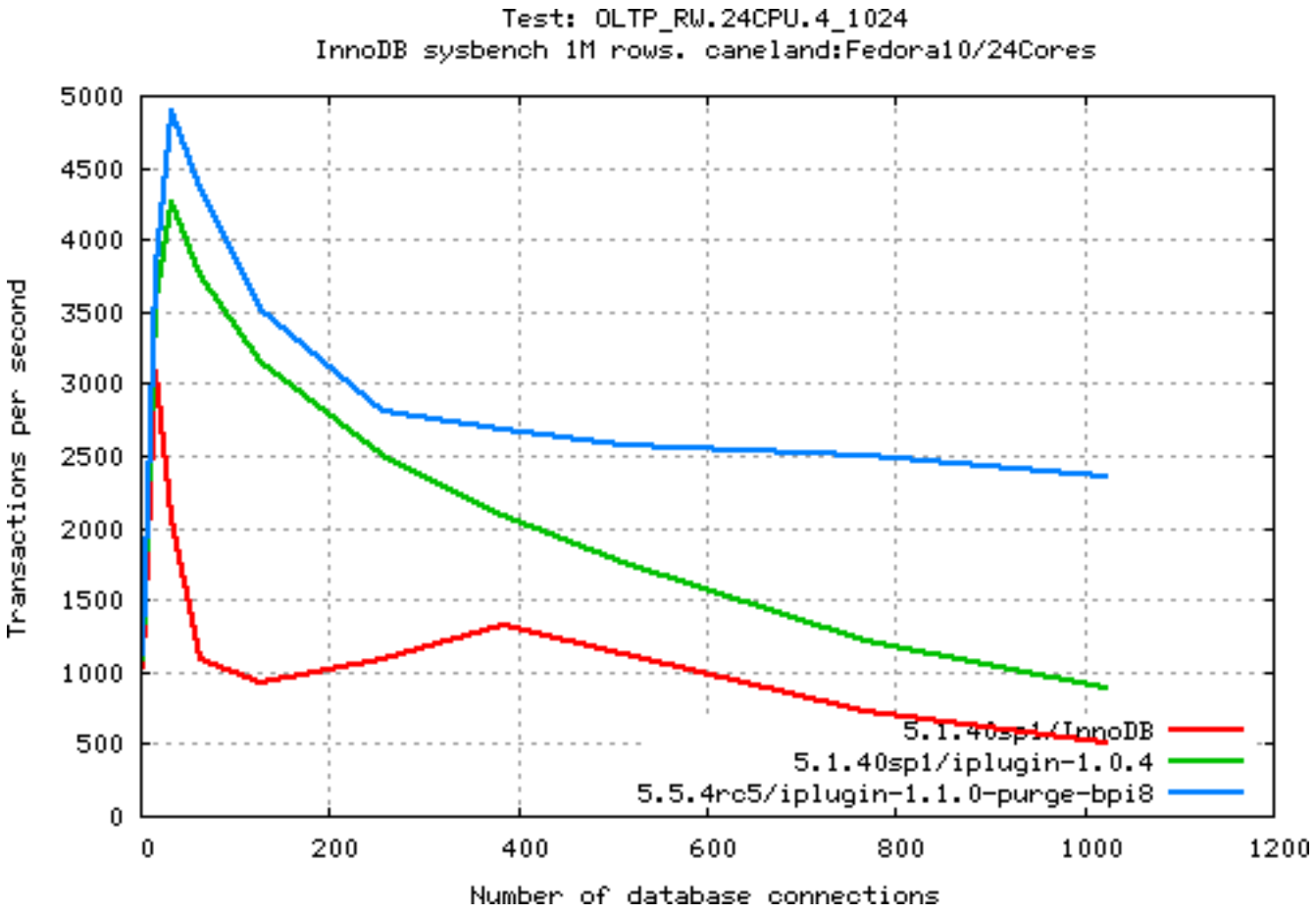
Impact of Powersave mode on Benchmarks/Applications

- /etc/init.d/cpuspeed on Linux
- Performance can drop significantly at low number of active connections (@16 threads performance drops to about half)
- Performance at 32 threads drops about 10%
- Performance at 64+ threads same

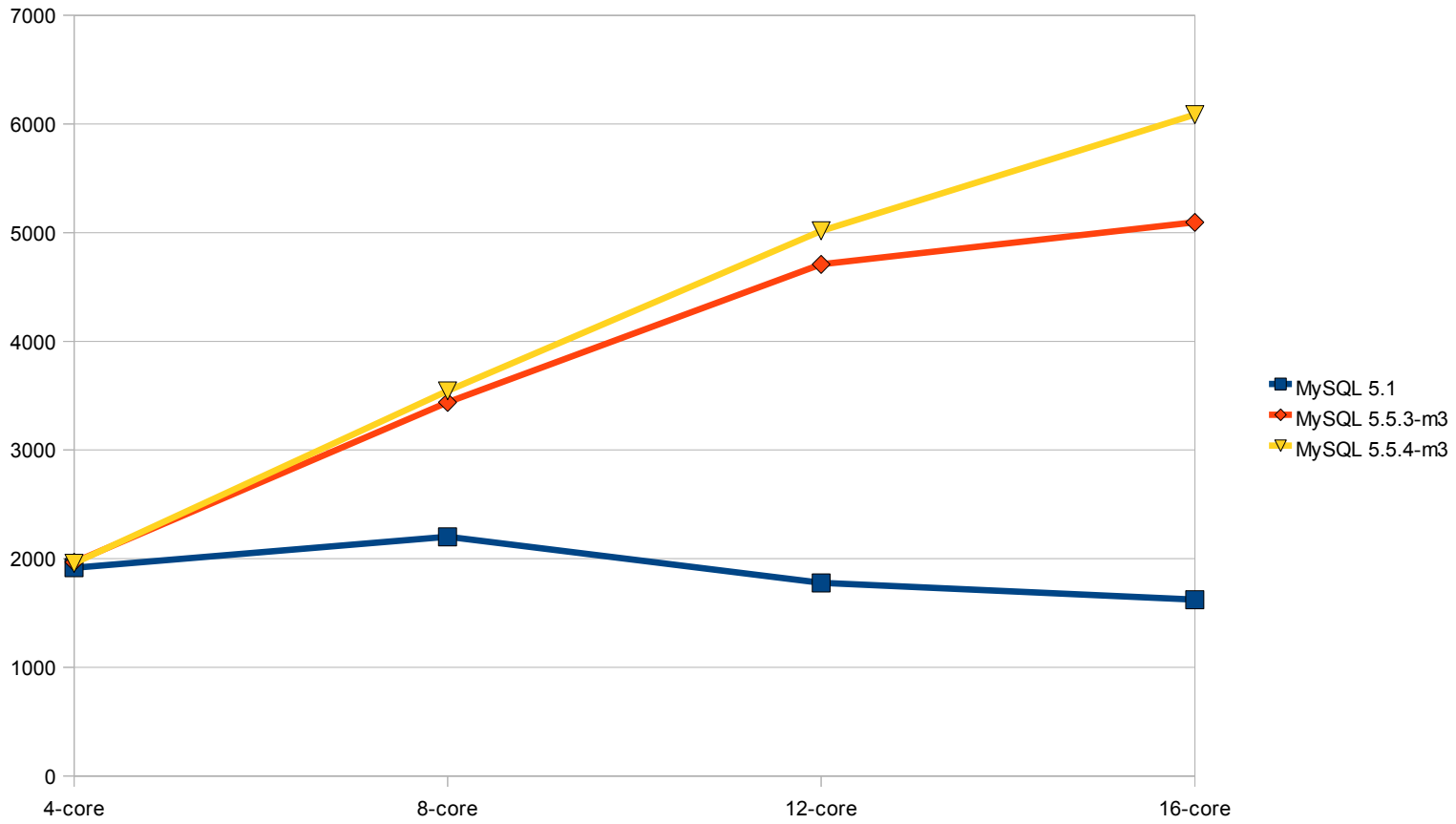
OLTP RO


Test: OLTP_RO.24CPU.4_1024
InnoDB sysbench 1M rows. caneland:Fedora10/24Cores
Transactions per second
Number of database connections
0
1000
2000
3000
4000
5000
6000
7000
8000
0
200
400
600
800
1000
1200
5.1.40sp1/InnoDB
5.1.40sp1/iplugin-1.0.4
5.5.4rc5/iplugin-1.1.0-purge-bpi8

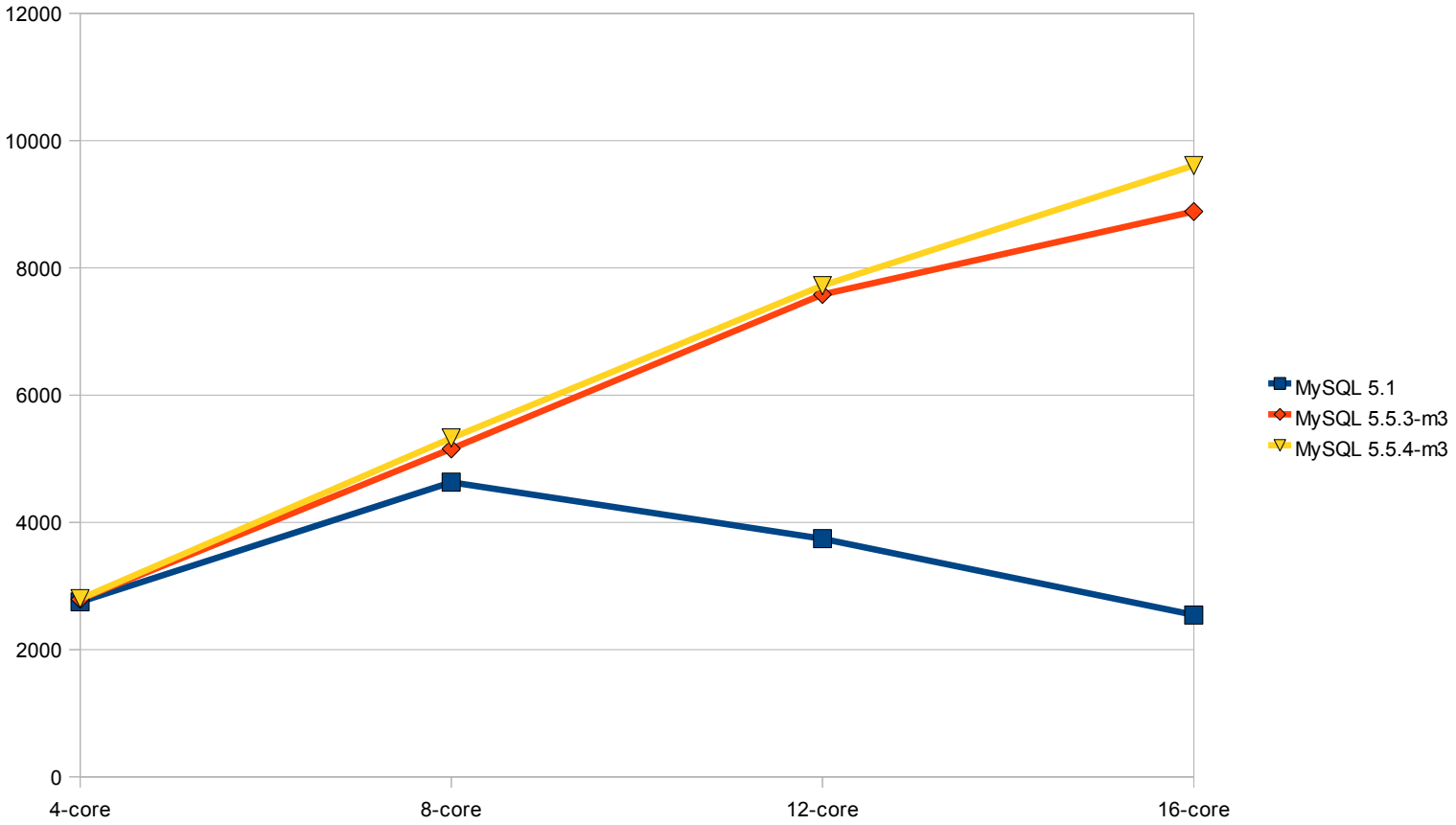
OLTP RW


Test: OLTP_RW.24CPU.4_1024
InnoDB sysbench 1M rows. caneland:Fedora10/24Cores
Transactions per second
Number of database connections
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
0
200
400
600
800
1000
1200
5.1.40sp1/InnoDB
5.1.40sp1/iplugin-1.0.4
5.5.4rc5/iplugin-1.1.0-purge-bpi8

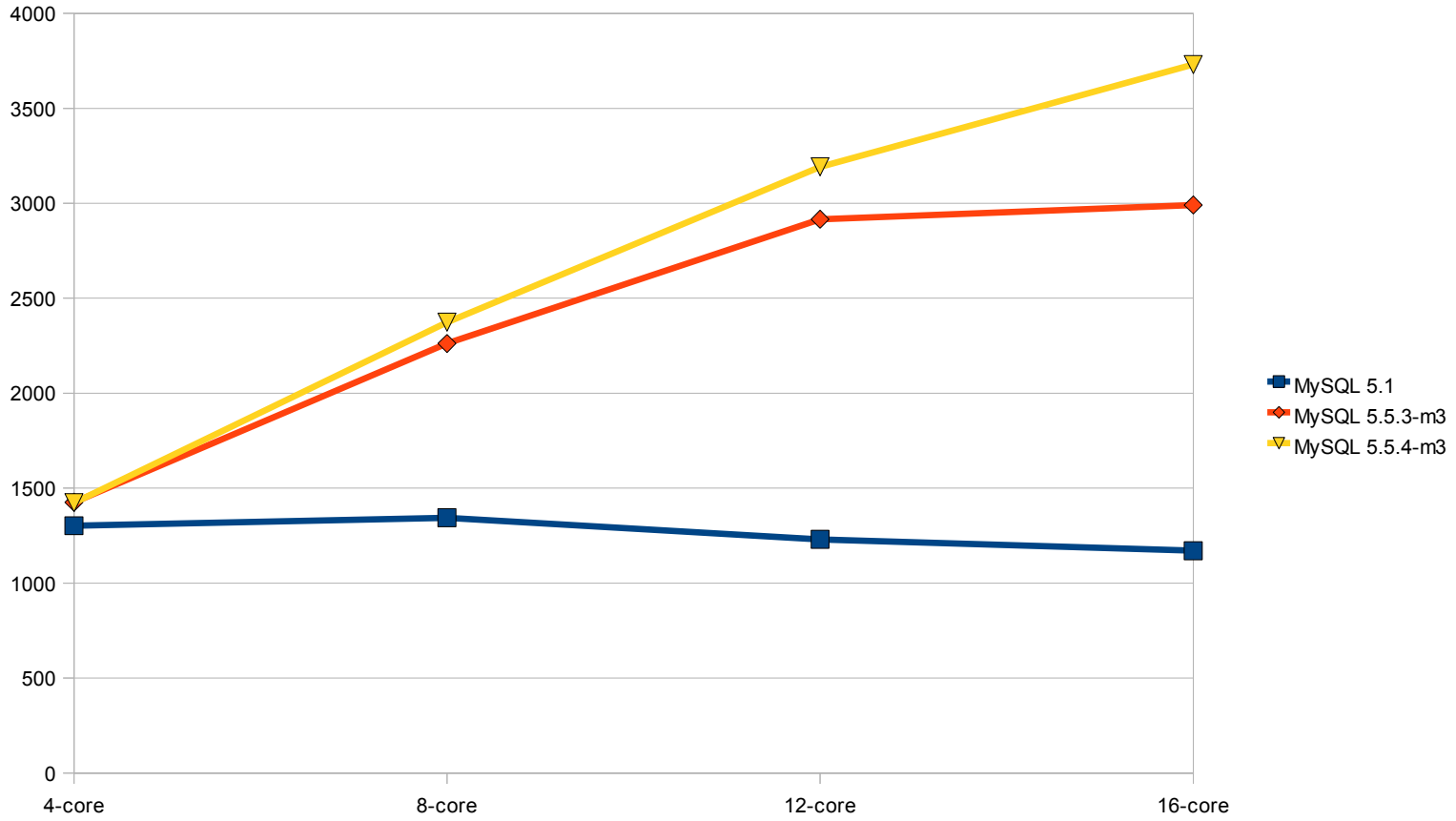
OLTP RW Scalability (4->16 cores)



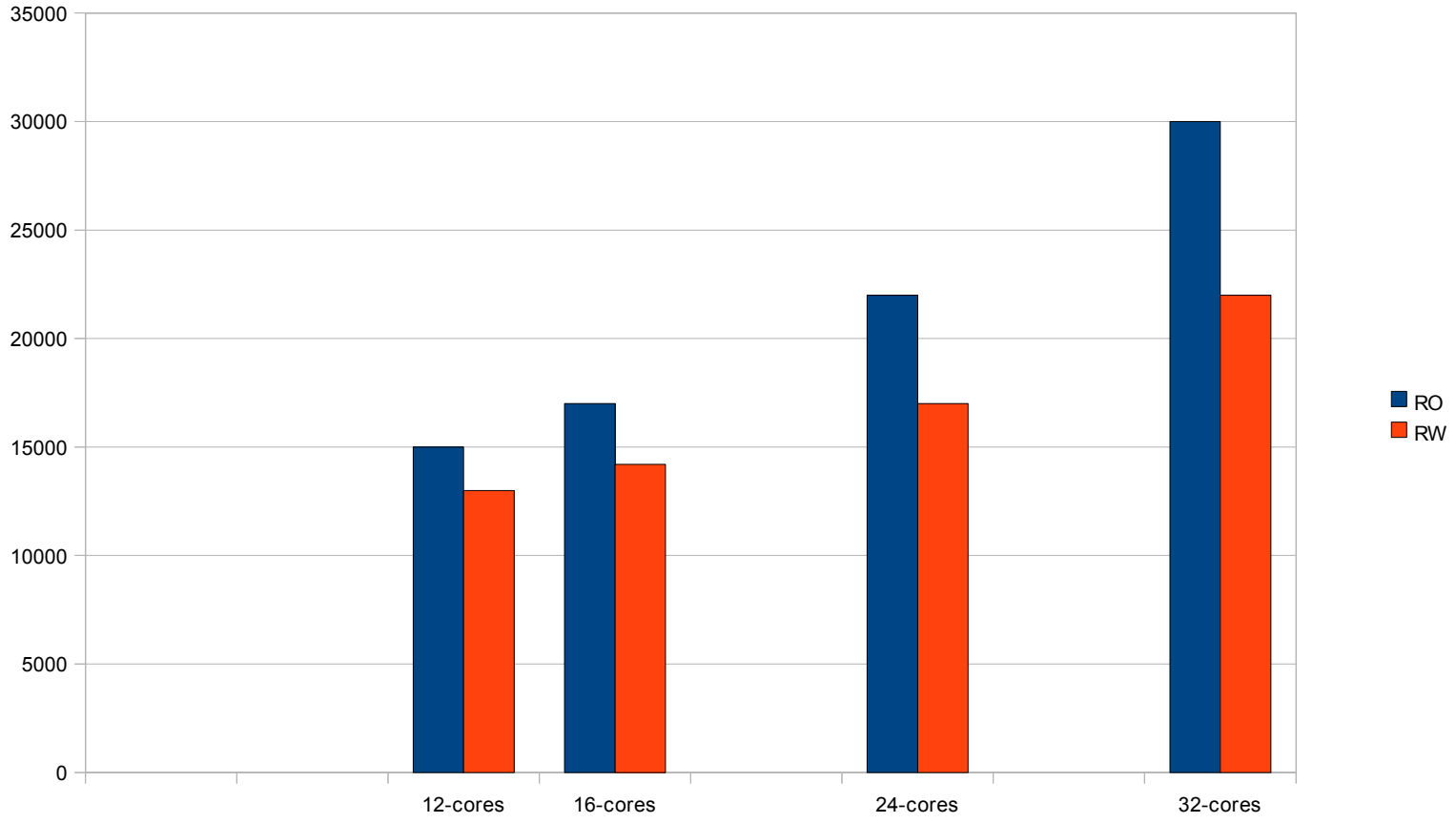
OLTP RO Scalability (4->16 cores)



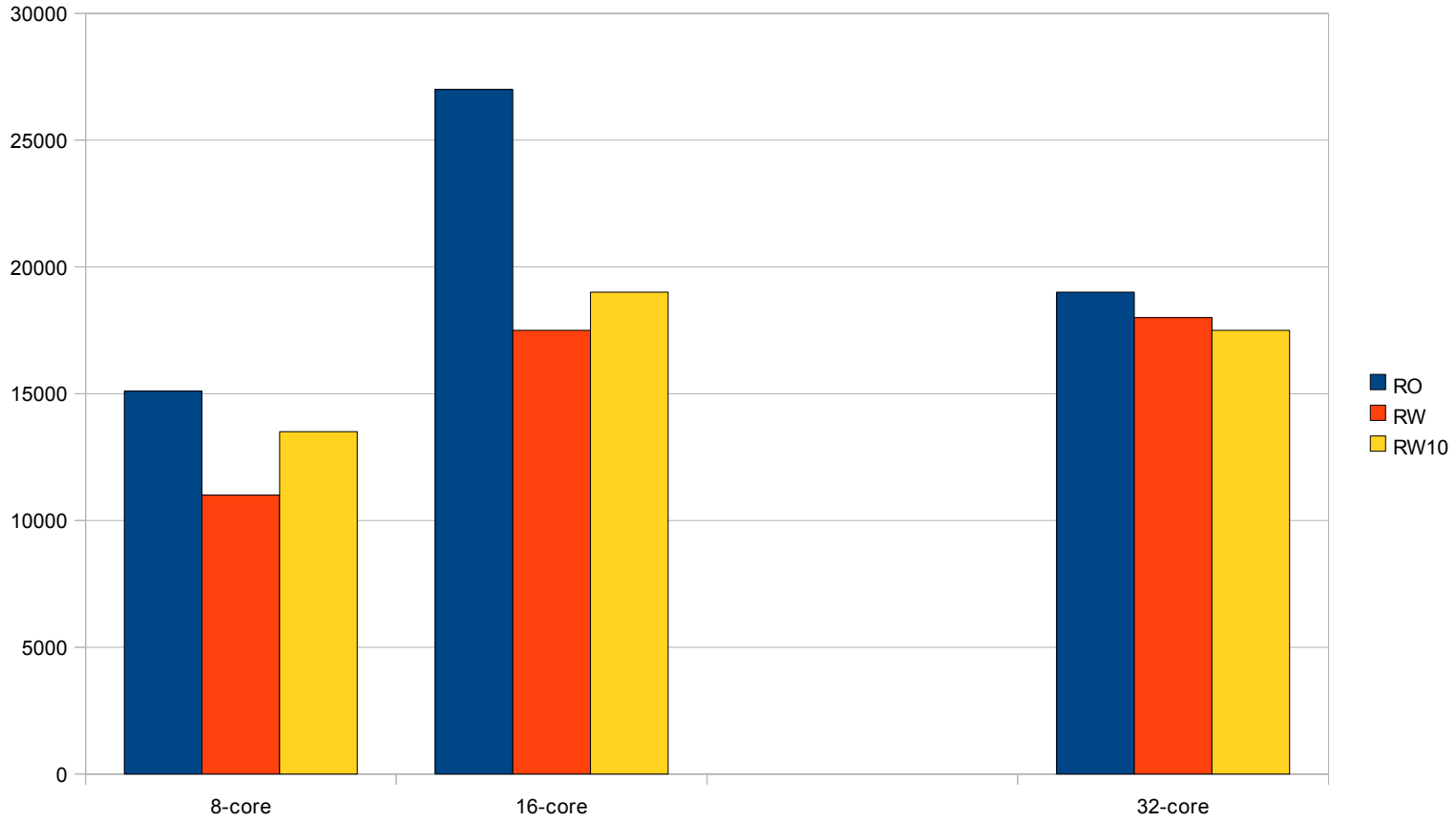
OLTP RW Write Intensive



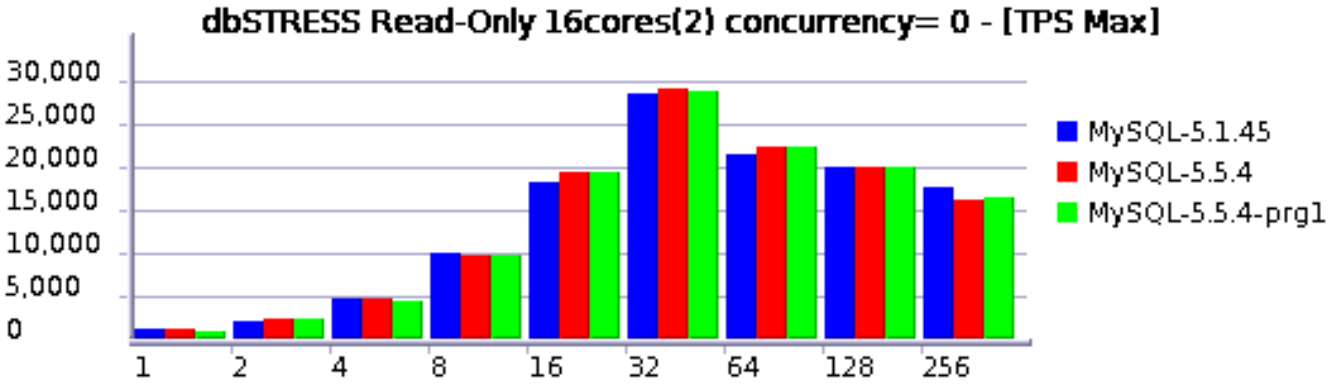
dbStress scalability 1 thread per core 12->32 cores

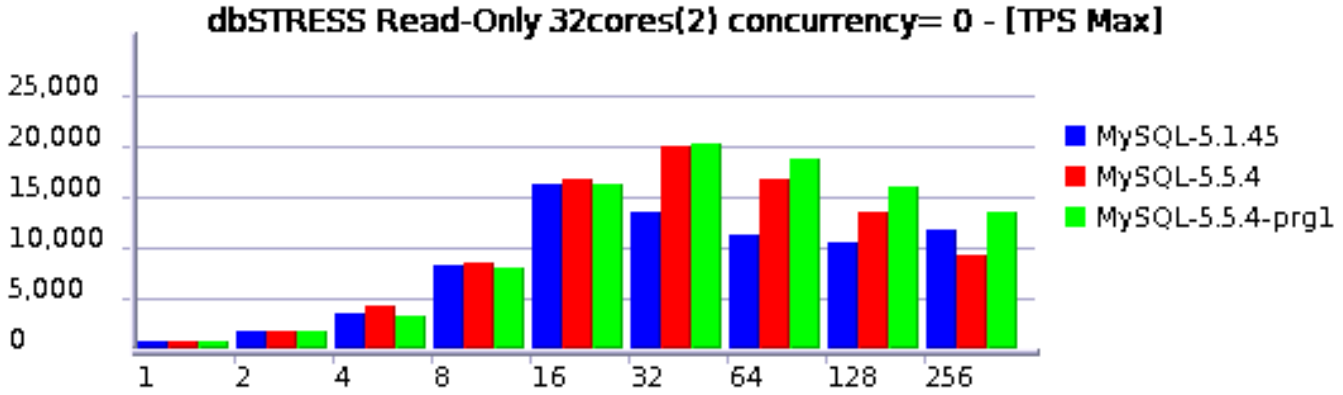


dbStress scalability 2 threads per core



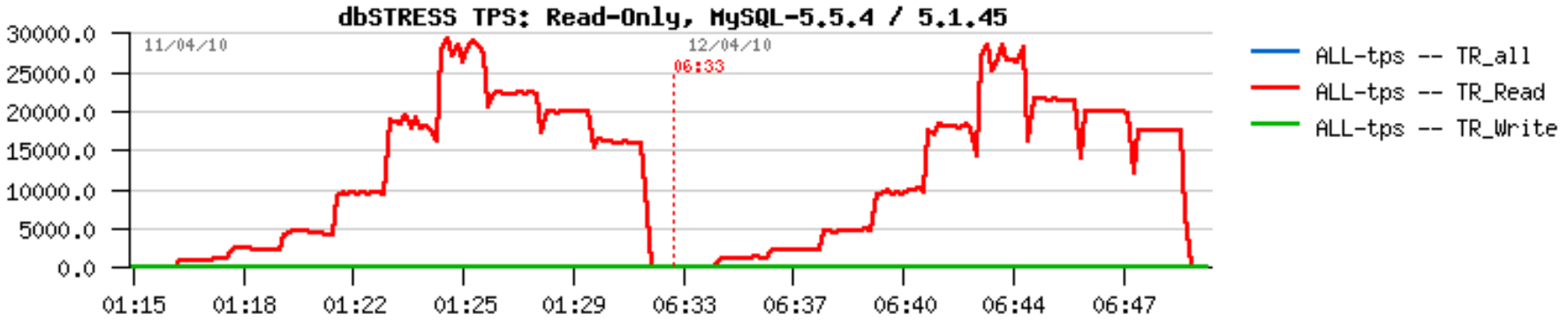
dbSTRESS: Read-Only

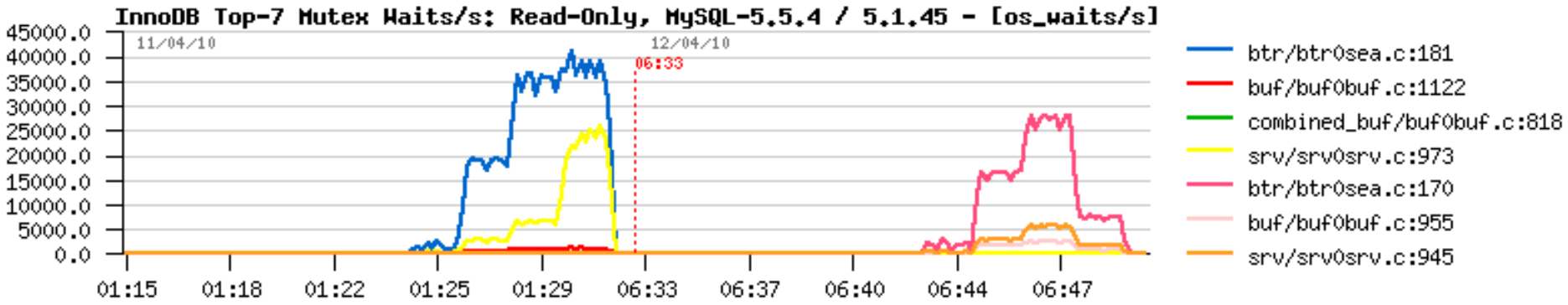




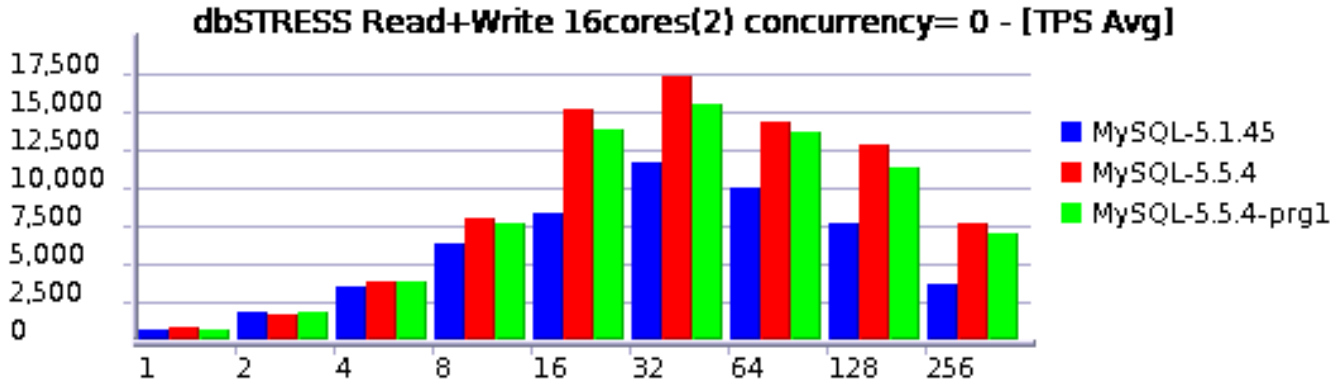
dbSTRESS: Read-Only Hot Mutexes

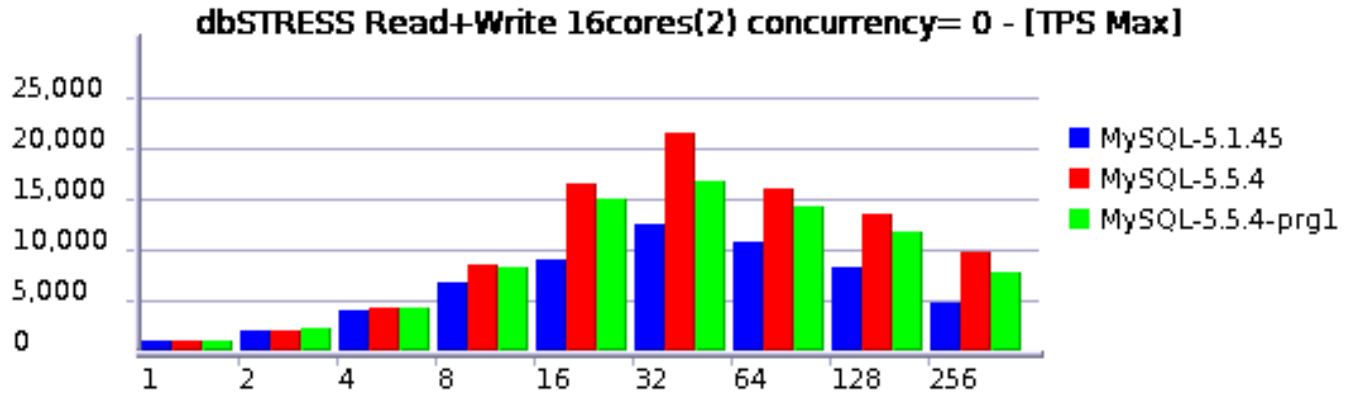
- kernel_mutex + B-tree + LOCK_open





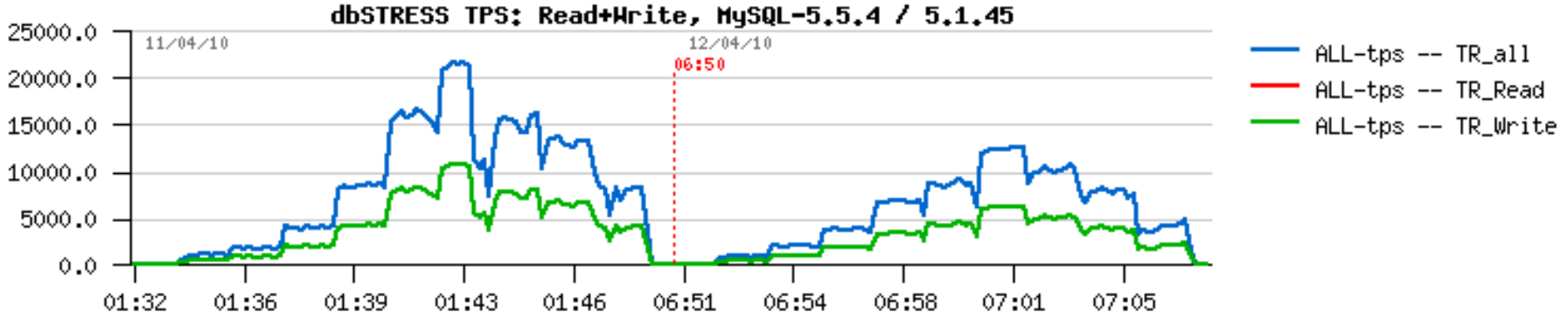
dbSTRESS: Read+Write @16cores

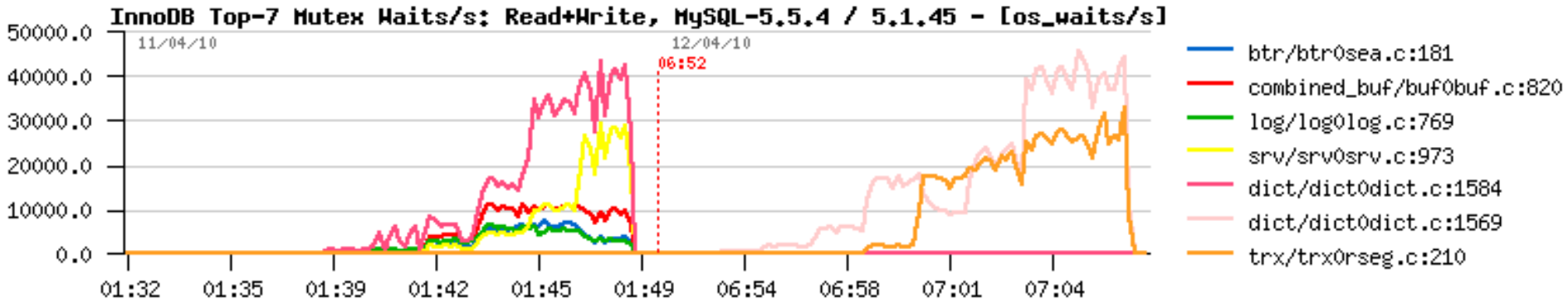




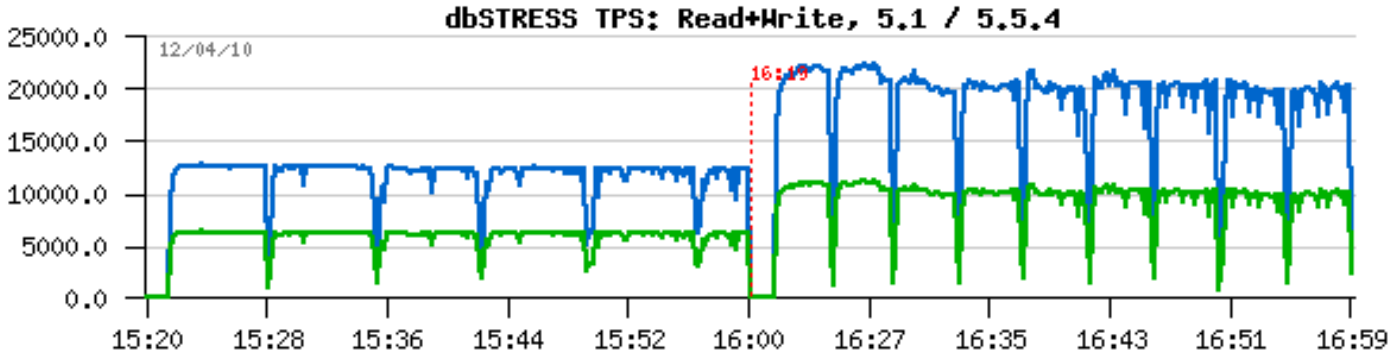
dbSTRESS: Read+Write Hot Contentions

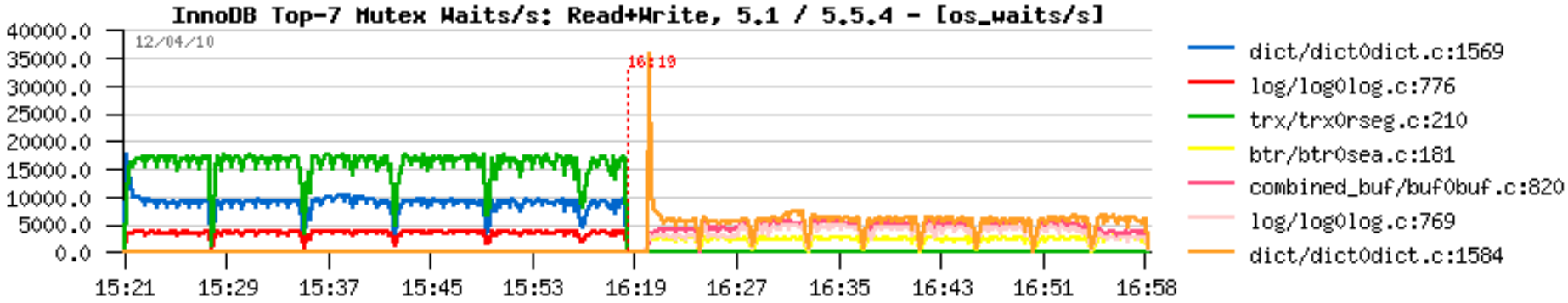
- Index mutex + kernel_mutex



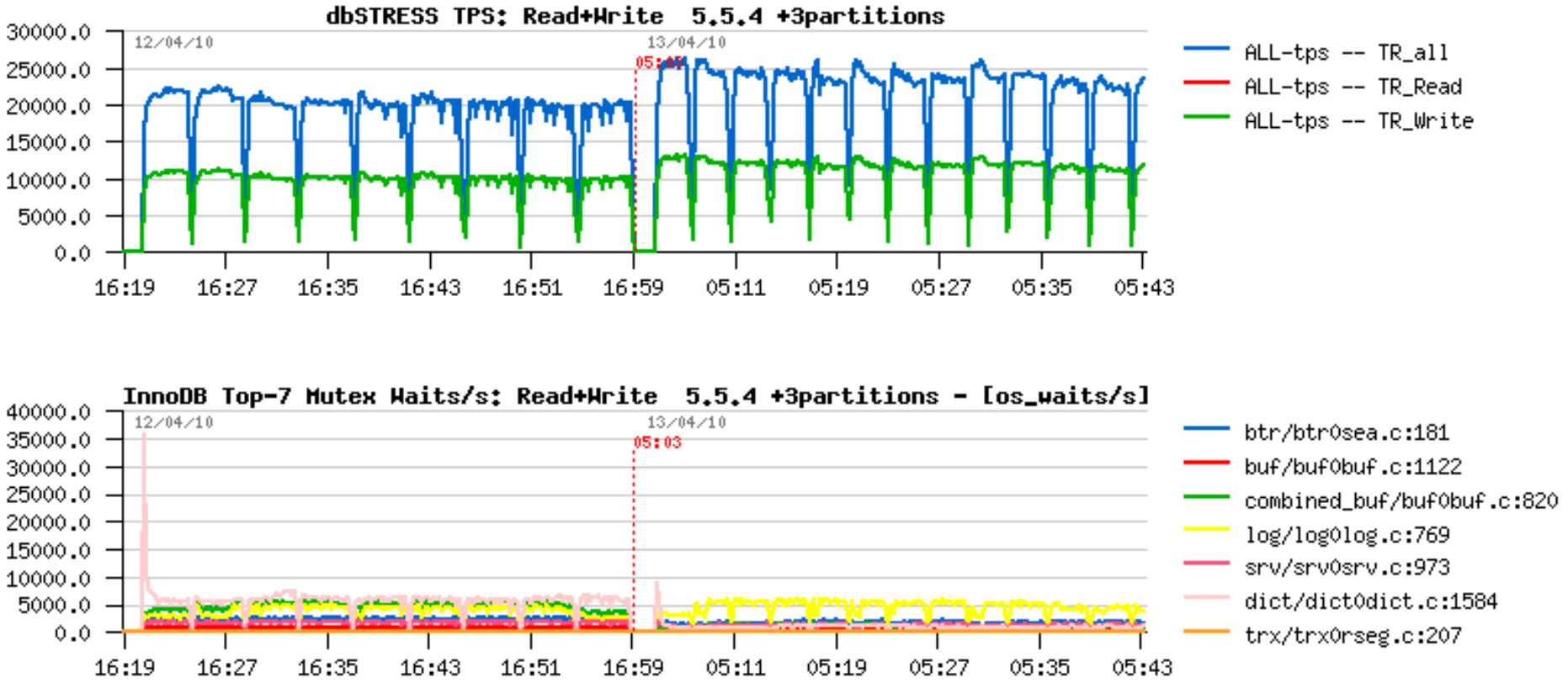


dbSTRESS: Read+Write Long 32sessons Test

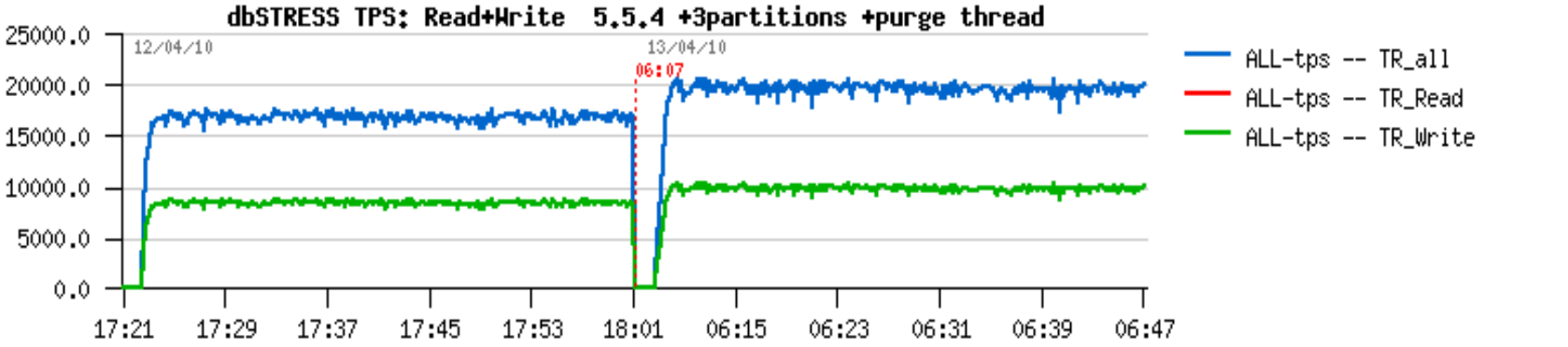


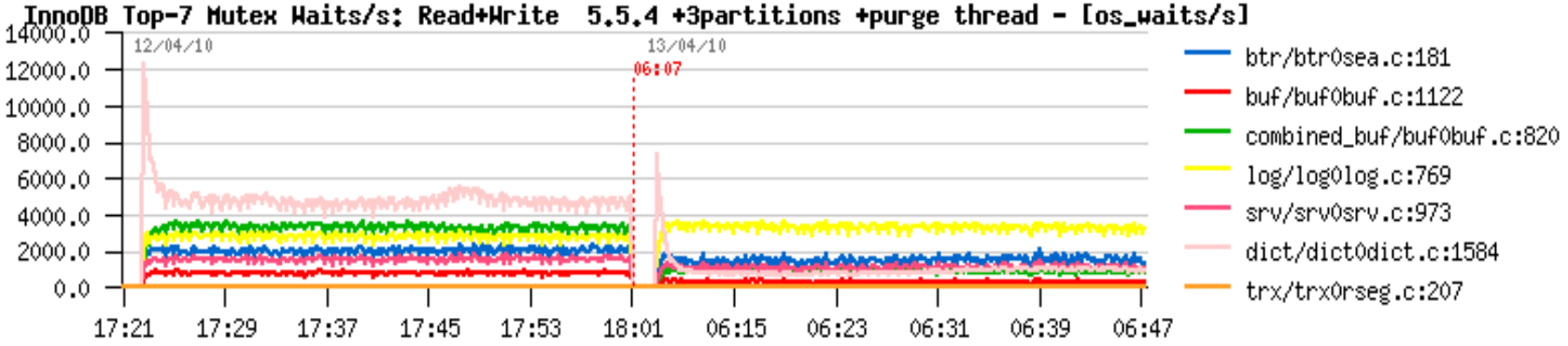


dbSTRESS: Using Partitions



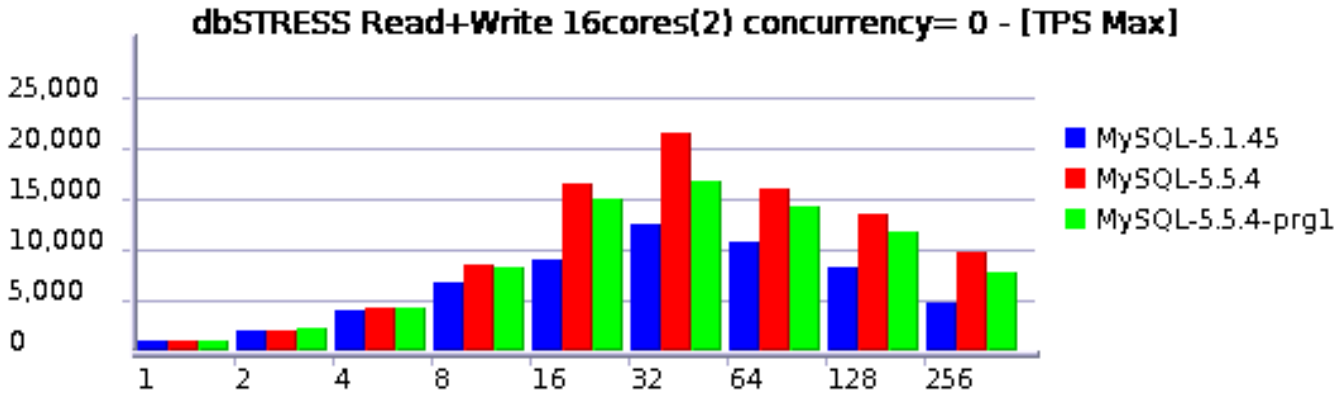
dbSTRESS: Partitions + Purge Thread

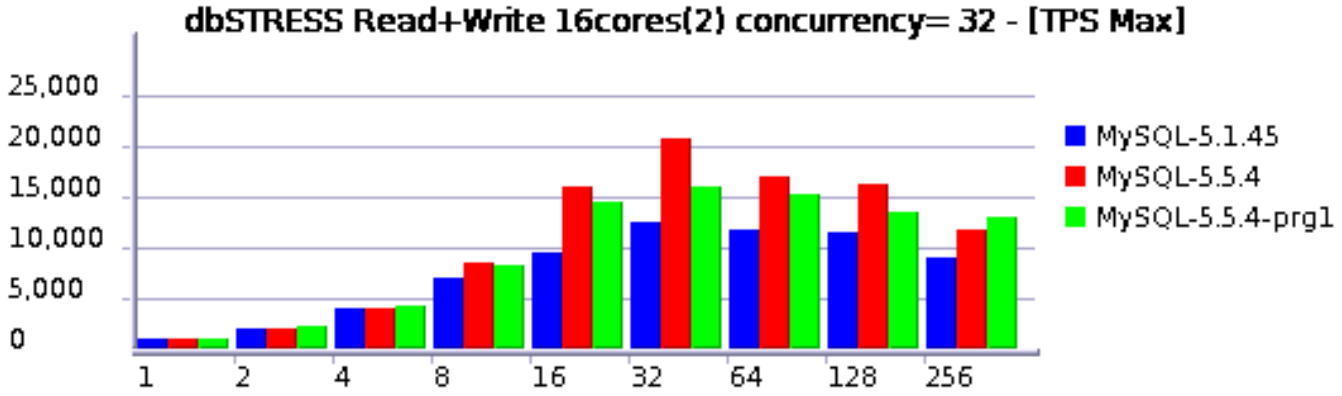




dbSTRESS: Read+Write & InnoDB Concurrency

- innodb_thread_concurrency= 0 / 32





Analysis of remaining scalability hogs

- Previously have been mainly hogged by global mutexes
- Now (especially for Read-only) also other effects becomes part of the picture such as False Cacheline sharing

Thank you for your attention

Questions?