
MySQL™ Connect

ORACLE

50 Tips for Boosting MySQL Performance

Arnaud ADANT
MySQL Principal Support Engineer

Safe Harbour Statement

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.

Program Agenda

- Introduction
- 50 MySQL Performance Tips
- Q & A

Introduction : Who I am

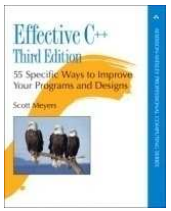
Arnaud ADANT

- http://blog.aadant.com
- 10 year+ Development
- MySQL Support for 3 years
- MySQL Performance
- I love my job !

Introduction : Why 50 ?

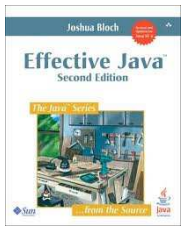
50 items is an effective format !

- *Effective C++*, Meyers, 1991



– 55 items

- *Effective Java*, Bloch, 2001



– 57 items

Tip 0. Never trust anyone, benchmark

HW and SW vendors are all lying !

- Test, test, test
- Benchmark, benchmark, benchmark
- Monitor, monitor, monitor
- One knob at a time
- Use sysbench, mysqlslap, monitoring tools

Tip 1. Make sure you have enough RAM

Depends on your active data and connections

- The active data should fit in the buffer pool
- MySQL connections and caches take memory
- ECC RAM recommended
- Extra RAM for
  - FS cache
  - Monitoring
  - RAM disk (tmpfs)

Tip 2. Use fast and multi-core processors

- Fast CPU is required for single threaded performance
- Recent servers have 32 to 80 cores.
- Enable hyper-threading
- MySQL can only scale to 16 cores in 5.5 and 32-48 cores in 5.6
- Same core count in fewer sockets is better
- Faster cores better than more but slower cores

Tip 3. Use fast and reliable storage

Will always help

- Good for IO bound loads
- HDD for sequential reads and writes
- Bus-attached SSD for random reads and writes
- Big sata or other disk for log files
- Several disks !
- Life time

Tip 4. Choose the right OS

MySQL is excellent on Linux

- L of LAMP
- Good on Solaris
- Oracle invests on Windows
- For pure performance, favor Linux

Tip 5. Adjust the OS limits

OS limits are extremely important !

- Max open files per process
  - **ulimit –n**
  - limits the number of file handles (connections, open tables, …)
- Max threads per user
  - **ulimit –u**
  - limits the number of threads (connections, event scheduler, shutdown)
- On Windows, ***MaxUserPort*** for TCP/IP, for 2003 and earlier

Tip 6. Consider using alternative malloc

Some malloc libraries are optimized for multi-core environment

- ***jemalloc*** is a good malloc replacement

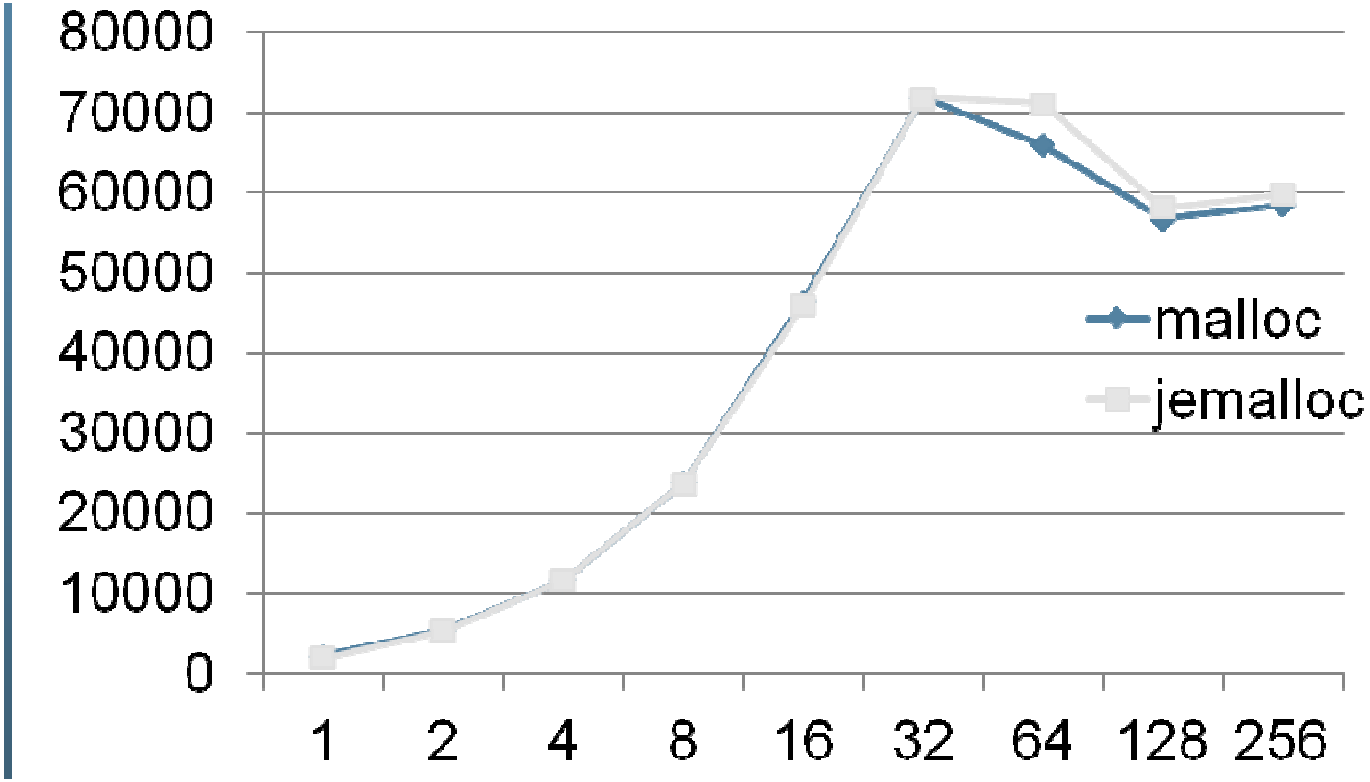
```
[mysqld_safe]
malloc-lib=/usr/lib64/libjemalloc.so.1
```

- ***tcmalloc*** shipped on Linux with MySQL

```
[mysqld_safe]
malloc-lib=tcmalloc
```

Tip 6. Consider using alternative malloc

- Sysbench OLTP RO
- High concurrency



Tip 7. Set CPU affinity

On Linux and Windows, CPU affinity helps concurrent WL

- taskset command on Linux

  taskset -c 1-4 `pidof mysqld`

  taskset -c 1,2,3,4 `pidof mysqld`

- On Windows :

  START /AFFINITY 0x1111 bin\mysqld –console

  START /AFFINITY 15 bin\mysqld –console

Tip 8. Choose the right file system

XFS for experts, ext4 or ext3

- **xfs** is excellent
  - With ***innodb_flush_method*** = O_DIRECT
  - supported by Oracle on OEL
  - less stable recently
- **ext4** best choice for speed and ease of use
  - fsyncs a bit slower than ext3
  - more reliable
- **ext3** is also a good choice
- **DYI nfs** is problematic with InnoDB

Tip 9. Mount options

For performance

- ext4 (rw,noatime,nodiratime,nobarrier,data=ordered)
- xfs (rw, noatime,nodiratime,nobarrier,logbufs=8,logbsize=32k)
- SSD specific
  - trim
  - ***innodb_page_size*** = 4K
  - ***Innodb_flush_neighbors*** = 0

Tip 10. Choose the best I/O scheduler

Use deadline or noop on Linux

- **deadline** is generally the best I/O scheduler
- echo deadline > /sys/block/{DEVICE-NAME}/queue/scheduler
- the best value is HW and WL specific
  - **noop** on high end controller (SSD, good RAID card …)
  - **deadline** otherwise

Tip 11. Use a battery backed disk cache

A good investment !

- Usually faster fsyncs
  - innoDB redo logs
  - binary logs
  - data files
- Crash safety
- Durability
- Applies to SSD

Tip 12. Balance the load on several disks

90 % of customers use a single disk !

- One disk is not a good idea
- Especially for HDD, read and write
- Separate :
  - ***datadir***
  - ***innodb_data_file_path***
  - ***innodb_undo_directory***
  - ***innodb_log_group_home_dir***
  - ***log-bin***
  - ***tmpdir***

Random, SSD

Sequential, spinning

Random, SSD, tmpfs

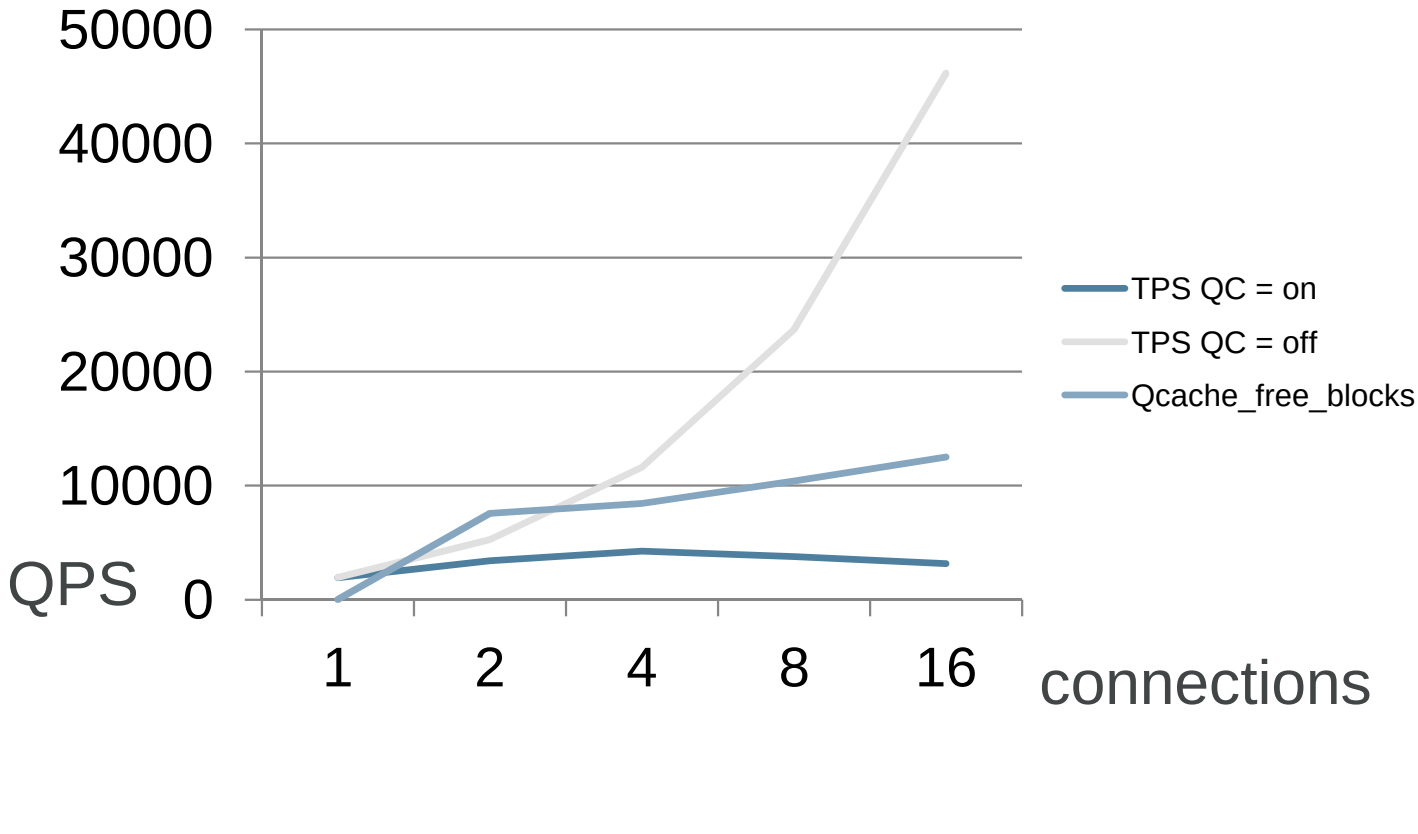
13. Turn Off the Query Cache

Single threaded bottleneck, only on low concurrency systems

- Only if ***threads_running*** <= 4
- Becomes fragmented
- Cache should be in the App !
- Off by default from 5.6
- **query_cache_type** = 0
- **query_cache_size** =0

Tip 13. Turn Off the Query Cache

- Sysbench OLTP RO
- QPS drops with connections and QC
- qcache_free_blocks > 5-10k



Tip 13. Turn Off the Query Cache

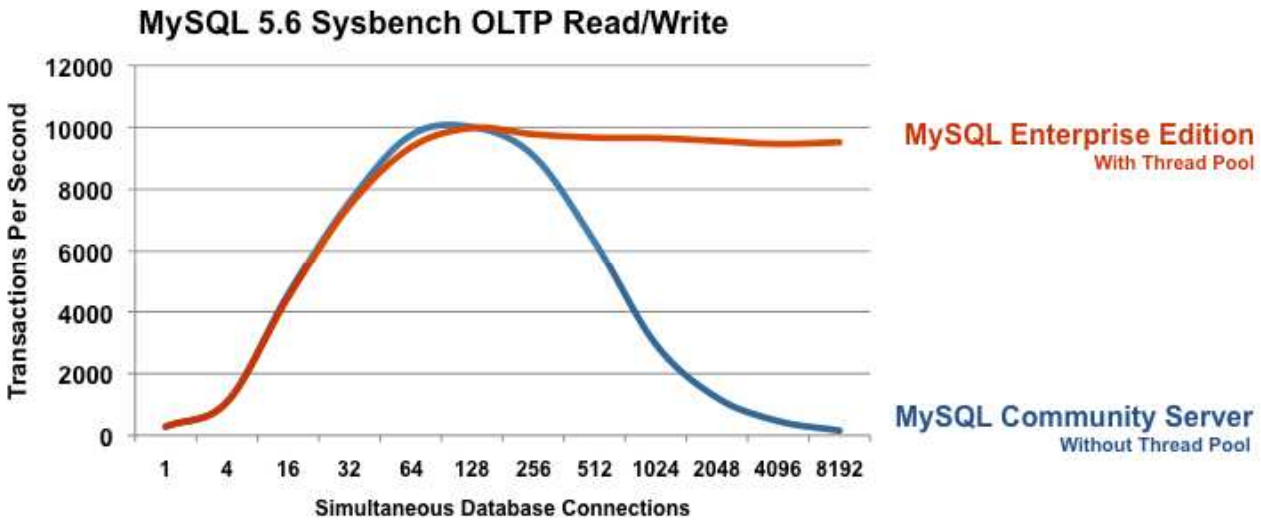
How to see there is a problem ?

- ***qcache_free_blocks*** > 5-10k
- stage/sql/Waiting for query cache lock

```
SELECT
        EVENT_NAME AS nm,
        COUNT_STAR AS cnt,
        sum_timer_wait,
       CONCAT(ROUND( sum_timer_wait / 1000000000000, 2), ' s') AS sec
FROM
        performance_schema.events_stages_summary_global_by_event_name
WHERE
        COUNT_STAR > 0
ORDER BY
        SUM_TIMER_WAIT DESC
LIMIT
        20;
```

Tip 14. Use the Thread Pool

- Stabilize TPS for high concurrency
- Useful if *threads_running* > hardware threads
- Decrease context switches
- Several connections for one execution thread

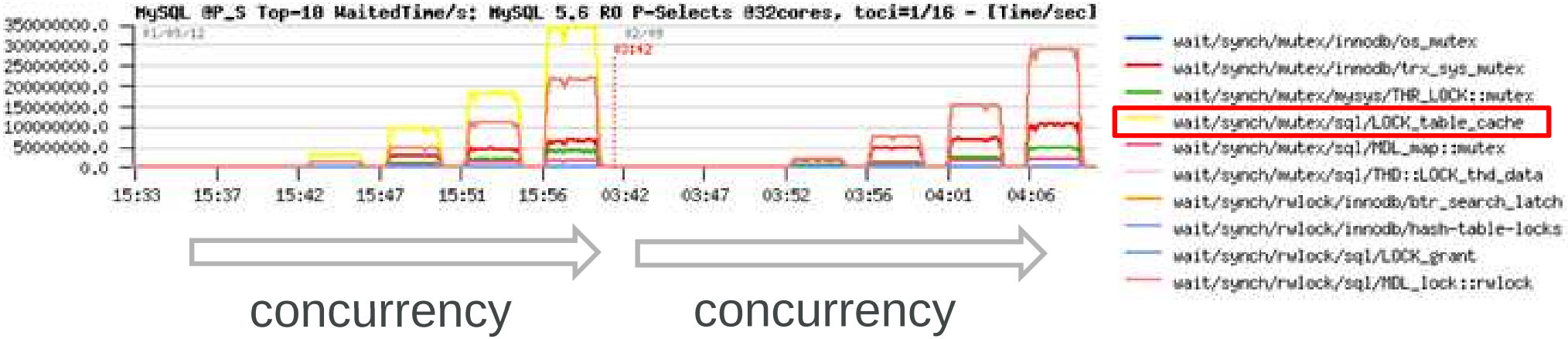


Tip 15. Configure table caching

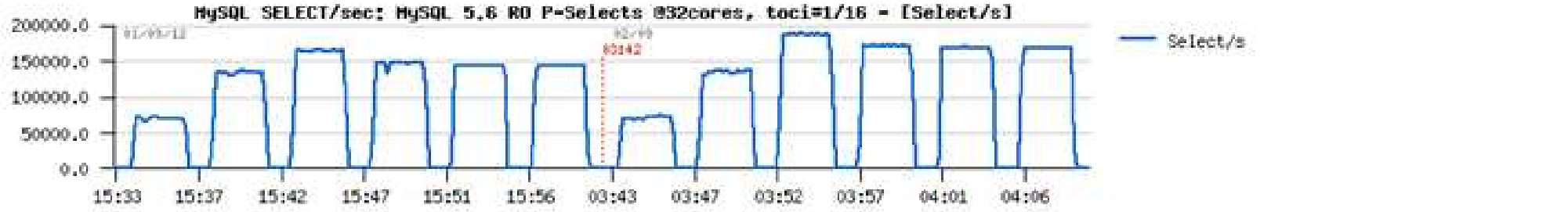
MySQL has at least one file per table : the FRM file

- ***table_open_cache***
  - not too small, not too big, used to size PS
  - *opened_tables / sec*
- ***table_definition_cache***
  - do not forget to increase
  - *opened_table_definitions* / sec
- ***table_cache_instances*** = 8 or 16
- ***innodb_open_files***
- ***mdl_hash_instances*** = 256

Tip 15. Configure table caching



TPS



MySQL Connect ORACLE

Tip 16. Cache the threads

Thread creation / initialization is expensive

- **thread_cache_size**
  - decreases *threads_created* rate
- capped by max user processes (see OS limits)
- 5.7.2 refactors this code

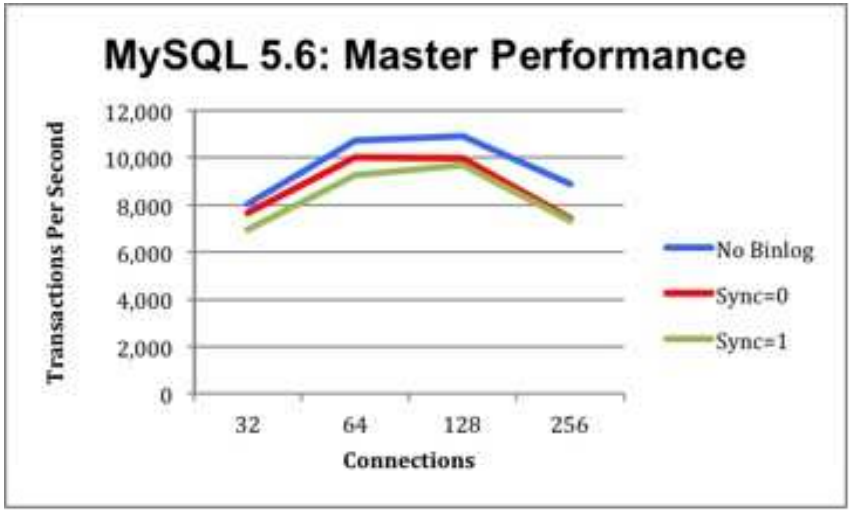
Tip 17. Reduce per thread memory usage

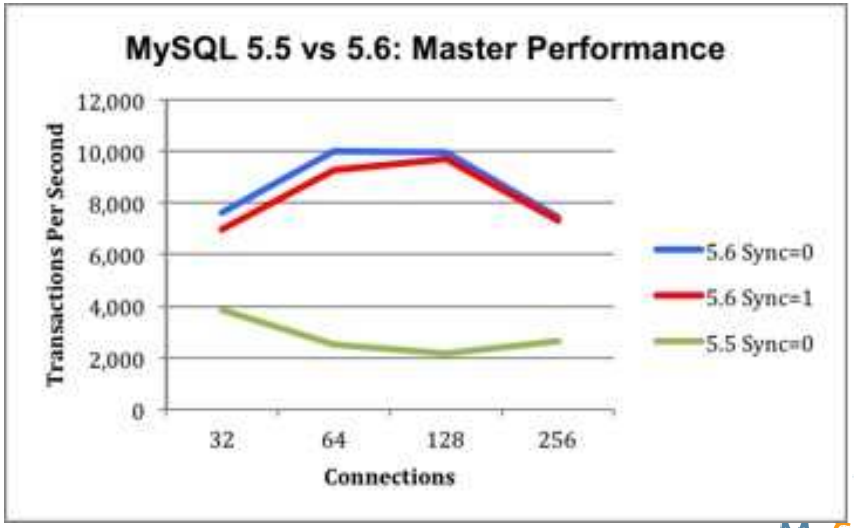
Memory allocation is expensive

- **max_used_connections * (**
  read_buffer_size +
  read_rnd_buffer_size +
  join_buffer_size +
  sort_buffer_size +
  binlog_cache_size +
  thread_stack +
  2 * net_buffer_length …
)

Tip 18. Beware of sync_binlog = 1

- sysbench RW
- ***sync_binlog*** = 1 was a performance killer in 5.5
- 5.6 binlog group commit fixed it
- Better with SSD or battery backed disk cache





Tip 19. Move your tables to InnoDB

InnoDB is the most advanced MySQL storage engine

- Scalable
- 99% of MyISAM use cases covered
- Online alter operations
- Full text engine
- Memcached API for high performance

Tip 20. Use a large buffer pool

50 – 80% of the total RAM

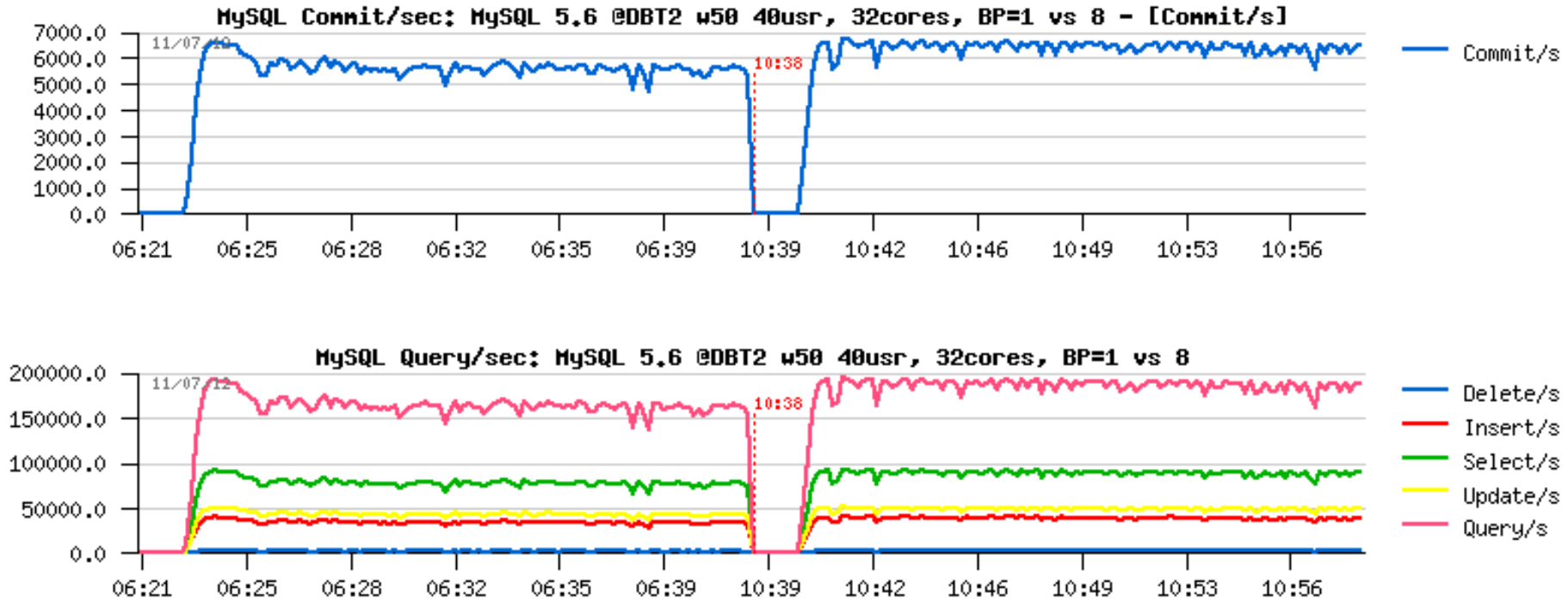
- **innodb_buffer_pool_size**
- Not too large for the data
- Do not swap !
- Beware of memory crash if swapping is disabled
- Active data <= ***innodb_buffer_pool_size*** <= 0.8 * RAM

Tip 21. Reduce the buffer pool contention

Key to achieve high QPS / TPS

- ***innodb_buffer_pool_instances*** >= 8
- Reduce *rows_examined* / sec (see Bug #68079)
- 8 is the default value in 5.6 !
- In 5.5, but even better in 5.6 and 5.7
- ***innodb_spin_wait_delay*** = 96 on high concurrency
- Use read only transactions
  - when possible

Tip 21. Reduce the buffer pool contention



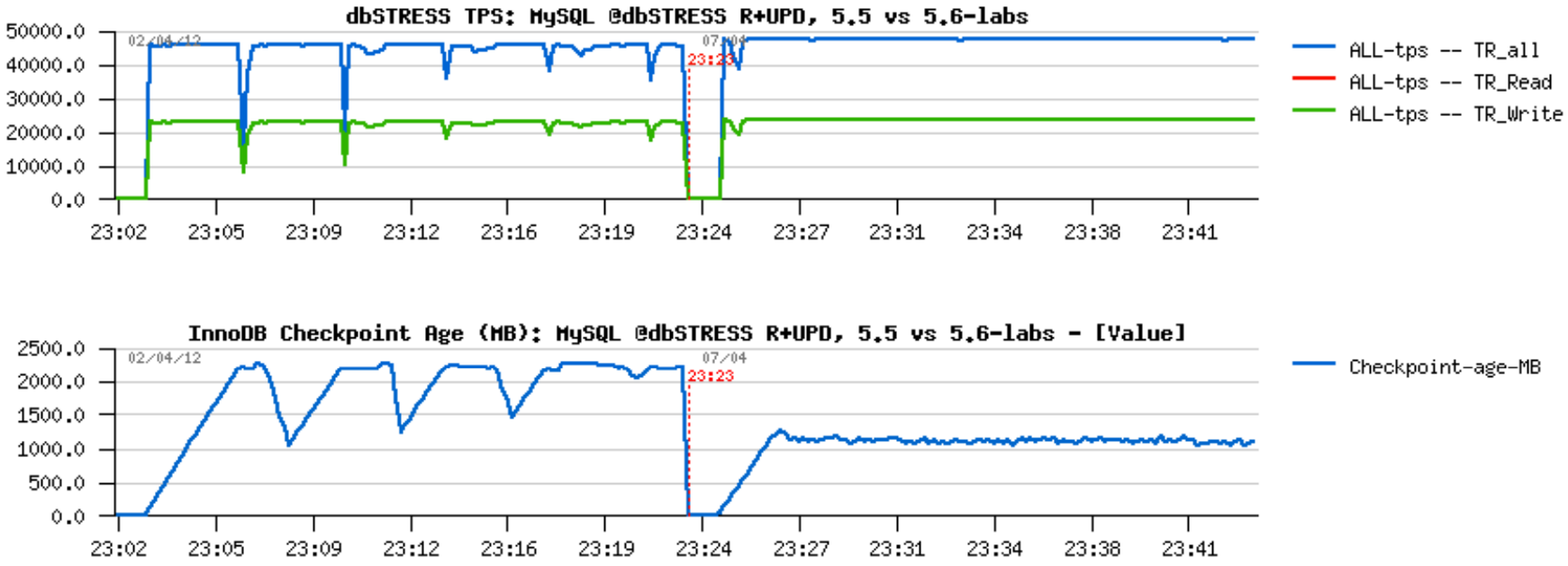
MySQL Connect ORACLE

Tip 22. Use large redo logs

A key parameter for write performance

- Redo logs defer the expensive changes to the data files
- Recovery time is no more an issue
- ***innodb_log_file_size*** = 2047M before 5.6
- ***innodb_log_file_size*** >= 2047M from 5.6
- Bigger is better for write QPS stability

Tip 22. Use large redo logs



Tip 23. Adjust the IO capacity

innodb_io_capacity should reflect device capacity

- IO OPS the disk(s) can do
- Higher for SSD
- Increase if several disks for InnoDB IO
- In 5.6, ***innodb_lru_scan_depth*** is per buffer pool instance
  so ***innodb_lru_scan_depth =***
  innodb_io_capacity / innodb_buffer_pool_instances
- Default ***innodb_io_capacity_max =***
  min(2000, 2 * innodb_io_capacity)

Tip 24. Configure the InnoDB flushing

Durability settings

- Redo logs *:*
  - ***innodb_flush_log_at_trx_commit*** = 1 // best durability
  - ***innodb_flush_log_at_trx_commit*** = 2 // better performance
  - ***innodb_flush_log_at_trx_commit*** = 0 // best performance
- Data files only :
  - ***innodb_flush_method*** = O_DIRECT // Linux, skips the FS cache
- Increase ***innodb_adaptive_flushing_lwm*** (fast disk)

Tip 25. Enable innodb_file_per_table

innodb_file_per_table = ON is the default in 5.6

- Increased manageability
- Truncate reclaims disk space
- Better with *innodb_flush_method* = O_DIRECT
- Easier to optimize
- But …
  - not so good with many small tables
  - more file handles (see OS limits)
  - more fsyncs

Tip 26. Configure the thread concurrency

innodb_thread_concurrency / innodb_max_concurrency_tickets

- No thread pool :
  - ***innodb_thread_concurrency*** = 16 - 32 in 5.5
  - ***innodb_thread_concurrency*** = 36 in 5.6
  - align to HW threads if less than 32 cores
- Thread pool :
  - ***innodb_thread_concurrency*** = 0 is fine

- ***innodb_max_concurrency_tickets*** : higher for OLAP, lower for OLTP

Tip 27. Reduce the transaction isolation

Default = repeatable reads

- Application dependent
- Read committed
  - it implies ***binlog_format*** = ROW
- Variable : ***transaction-isolation***
- Lower isolation = higher performance

Tip 28. Design the tables

Choose the charset, PK and data types carefully

- integer primary keys
  - avoid varchar, composite for PK
- latin1 vs. utf8
- the smallest varchar for a column
- keep the number of partitions low (< 10)
- use compression for blob / text data types

Tip 29. Add indexes

Indexes help decrease the number of rows examined

- for fast access to records
- for sorting / grouping
  - without temporary table
- covering indexes
  - contain all the selected data
  - save access to full record
  - reduce random reads

Tip 30. Remove unused indexes

Redundant indexes must be removed

- Too many indexes hurt performance
  - Bad for the optimizer
  - More IO, more CPU to update all the indexes
- Remove same prefix indexes
- Use ps_helper views
  - schema_unused_indexes

Tip 31. Reduce rows_examined

Maybe the most important tip for InnoDB and queries !

- Rows read from the storage engines
- Rows_examined
  - slow query log
  - P_S statement digests
  - MEM 3.0 query analysis
  - Handler%
- ***rows_examined > 10 * rows_sent***
  - missing indexes
  - query tuning !

Tip 31. Reduce rows_examined

Per query handlers

```
SHOW SESSION STATUS WHERE variable_name LIKE 'Handler%' OR variable_name LIKE '%tmp%';
select …
SHOW SESSION STATUS WHERE variable_name LIKE 'Handler%' OR variable_name LIKE '%tmp%';
```

- Diff
- Sum Handler%

```
+----------------------------+---------+
| Variable_name              | Value   |
+----------------------------+---------+
| Created_tmp_disk_tables    | 3       |
| Created_tmp_files          | 0       |
| Created_tmp_tables         | 31      |
| Handler_commit             | 0       |
| Handler_delete             | 0       |
| Handler_discover           | 0       |
| Handler_prepare            | 0       |
| Handler_read_first         | 3       |
| Handler_read_key           | 3738453 |
| Handler_read_next          | 3383374 |
| Handler_read_prev          | 0       |
| Handler_read_rnd           | 290540  |
| Handler_read_rnd_next      | 374464  |
| Handler_rollback           | 0       |
| Handler_savepoint          | 0       |
| Handler_savepoint_rollback | 0       |
| Handler_update             | 0       |
| Handler_write              | 1560294 |
```

Tip 31. Reduce rows_examined

- slow query log

```
# Time: 130921 12:02:46
# User@Host: prod[prod] @  [11.11.123.456]
# Query_time: 17.801964  Lock_time: 0.000135 Rows_sent: 363  Rows_examined: 111606
SET timestamp=1378380766;
SELECT…..;
```

- show engine innodb status

```
--------------
ROW OPERATIONS
--------------
1 queries inside InnoDB, 0 queries in queue
26 read views open inside InnoDB
Main thread id 10, state: sleeping
Number of rows inserted 355526809, updated 113575734, deleted 110305731, read 184654786269
1.07 inserts/s, 0.43 updates/s, 0.00 deletes/s, 194048.73 reads/s
```

- ps_helper

```
mysql> select * from statement_analysis;
+-------------------------------------------+-----------+------------+---------------+-------------+-------------+----------+--------------+--------------+-----
| query                                     | full_scan | exec_count | total_latency | max_latency | avg_latency | rows_sent | rows_sent_avg | rows_scanned | dig
+-------------------------------------------+-----------+------------+---------------+-------------+-------------+----------+--------------+--------------+-----
| BEGIN                                     |           |       8149 | 424.73 ms     | 4.67 ms     | 52.12 us    |        0 |            0 |            0 | 751
| SELECT * FROM `schema_table_st ... _latency| *         |          1 | 151.17 ms     | 151.17 ms   | 151.17 ms   |       72 |           72 |          288 | e6a
_
```

Tip 32. Reduce rows_sent

Another performance killer

- **rows_sent <= rows_examined**
- Network transfers
- CPU involved
- Apps seldom need more than 100 rows
- LIMIT !

Tip 33. Reduce locking

Make sure the right execution plan is used

- UPDATE, SELECT FOR UPDATE, DELETE, INSERT SELECT
- Use a PK ref, UK ref to lock
- Avoid large index range and table scans
- Reduce *rows_examined* for locking SQL
- Locking is expensive (memory, CPU)
- Commit when possible

Tip 34. Mine the slow query log

Find the bad queries

- Dynamic collection
- The right interval
- Top queries
  - Sort by query time desc
    - perl mysqldumpslow.pl –s t slow.log
  - Sort by rows_examined desc
- Top queries at the 60s range

Tip 34. Mine the slow query log

Log everything for 60 seconds to the slow query log :

```
SET @saved_slow_query_log = @@slow_query_log;
SET @saved_long_query_time = @@long_query_time;
SET global slow_query_log = 1;
SET global long_query_time = 0;
select sleep(60);
SET global slow_query_log = @saved_slow_query_log;
SET global long_query_time = @saved_long_query_time;
```

Tip 35. Use the performance_schema

The performance_schema is a great tool

- ps_helper :
  - good entry point
  - ready to use views
  - IO / latency / waits / statement digests
  - ideal for dev and staging
  - https://github.com/MarkLeith/dbahelper/archive/master.zip
- For high performance systems, ***performance_schema*** = 0

Tip 36. Tune the replication thread

Usually replication is a black box

- Slow query log with
  - **log-slow-slave-statements** is now dynamic (Bug #59860) from 5.6.11
- Performance_schema (Bug #16750433)
  - Not instrumented before 5.6.14
- binlog_format = ROW
  - `show global status like 'Handler%'`

Tip 37. Avoid temporary tables on disk

Writing to disk is not scalable !

- Large temporary tables on disk
  - ***handler_write***
  - ***created_tmp_disk_tables***
  - monitor **tmpdir** usage
- Frequent temporary tables on disk
  - High ***created_tmp_disk_tables / uptime***
  - `show global status like '%tmp%';`
- In 5.6, included in statement digests and ps_helper
- Available in MEM 3.0

Tip 38. Cache data in the App

Save MySQL resources !

- Good for CPU / IO
- Cache the immutable !
  - referential data
  - memcached
- Query cache can be disabled
- Identify frequent statements
  - perl mysqldumpslow.pl –s c slow60s.log

Tip 39. Avoid long running transactions

A frequent cause of performance drops

- Usually the oldest transactions

```
---TRANSACTION F08, ACTIVE 25445 sec
2 lock struct(s), heap size 376, 4 row lock(s), undo log entries 3
MySQL thread id 1, OS thread handle 0xd9c, query id 19 localhost 127.0.0.1 root
```

- High *history_list_length*
- Prevent the purge
- Decrease performance
- Kill if abandoned

Tip 40. Close idle connections

Idle connections consume resources !

```
mysql> SHOW FULL PROCESSLIST;
+----------+-----+-----------+------+-------------+-------+--------
| Id       | User| Host      | db   | Command     | Time  | State
+----------+-----+-----------+------+-------------+-------+--------
|  2058862 | app |           | prod | Sleep       |  3516 |
|  6554823 | app |           | prod | Sleep       |  3513 |
|  8568860 | app |           | prod | Sleep       |  2778 |
```

- Either kill or refresh them !
  - Connection pools : validation query

Tip 41. Close prepare statements

Prepare and close must be balanced

- com_stmt_prepare – com_stmt_close ~= 0

```
mysql> SHOW GLOBAL STATUS;
+-----------------------------------+------------
| Variable_name                     | Value
+-----------------------------------+------------
....
| Com_stmt_close                    | 489869
| Com_stmt_execute                  | 209441621
| Com_stmt_fetch                    | 0
| Com_stmt_prepare                  | 1312599
```

Tip 42. Configure Connector / J

Connectors must be tuned too !

- JDBC property for maximum performance :
  - **userConfigs=maxPerformance**
  - Use if the server configuration is stable
  - Removes frequent
    - SHOW COLLATION
    - SHOW GLOBAL VARIABLES
- Fast validation query : /* ping */

Tip 43 - 45

43. Do not use the information_schema in your App
44. Views are not good for performance
    - temporary tables (on disk)
45. Scale out, shard

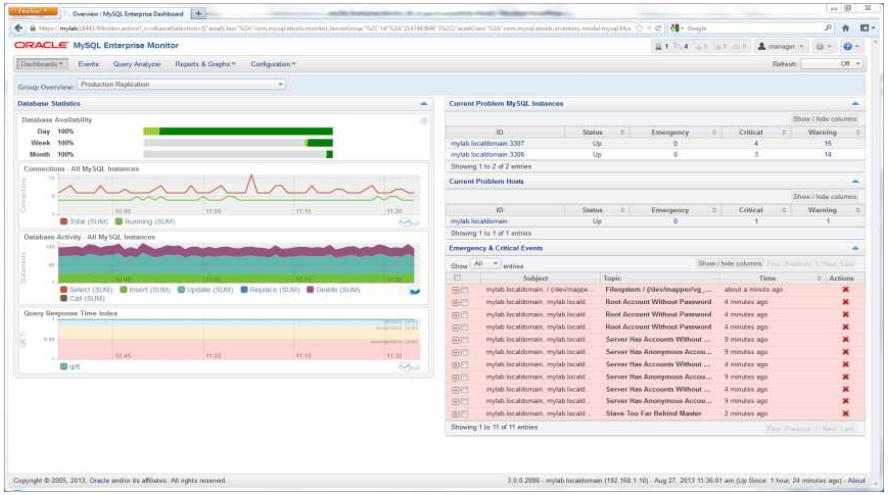
Tip 46. Monitor the database and OS

A monitoring tool is DBA's friend :

- alerts
- graphs
- availability and SLAs
- the effect of tuning
- query analysis

MySQL Enterprise Monitor 3.0 is GA !

- **SLA monitoring**
- **Real-time performance monitoring**
- **Alerts & notifications**
- **MySQL best practice advisors**



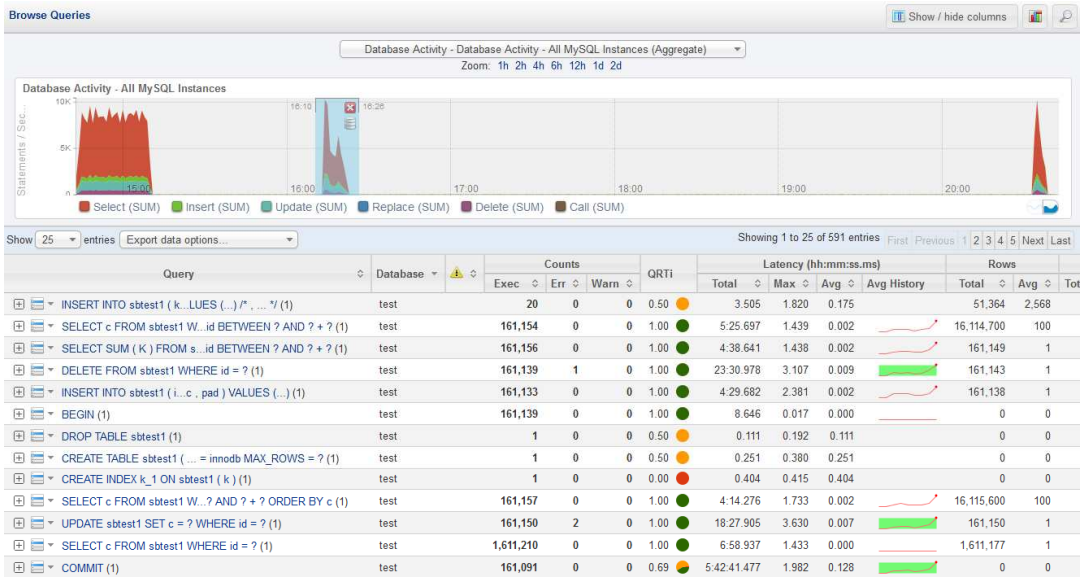
"The MySQL Enterprise Monitor is an absolute must for any DBA who takes his work seriously."

- Adrian Baumann, System Specialist
Federal Office of Information Technology & Telecommunications

MySQL Connect ORACLE

Query Analyzer

- **Real-time query performance**
- **Visual correlation graphs**
- **Find & fix expensive queries**
- **Detailed query statistics**
- **Query Response Time index (QRTi)**



"With the MySQL Query Analyzer, we were able to identify and analyze problematic SQL code, and triple our database performance. More importantly, we were able to accomplish this in three days, rather than taking weeks."

Keith Souhrada
Software Development Engineer
Big Fish Games

MySQL Connect

ORACLE

Tip 47. Backup the database

Why is it a performance tip ?

- Backup is always needed !
- Use MEB instead of mysqldump
  - especially on large instances
- mysqldump eats MySQL resources
- mysqlbackup copies the data files (in parallel)

Tip 48. Optimize table and data files

Fragmentation decreases performance

- Fragmentation …
  - On disk only due to FS
  - Inside InnoDB table spaces
  - Occurs when modifying existing data
- alter table … engine=InnoDB
  - Fixes everything
- Still blocking in 5.6

Tip 48. Optimize table and data files

How to estimate fragmentation ?

- There is no general formula
  - except for fixed length records
- ```
  create table t_defrag as
                          select * from t
                          limit 10000;
  ```
  - Fragmentation if Avg_row_length(t) > Avg_row_length(t_defrag)

Avg_row_length from `show table status`
  ```

Tip 49. Upgrade regularly

- Security vulnerability fixes
- Bug fixes
- Performance improvements
- Ready for the next GA
- Do not upgrade without testing.

Tip 50. Perform regular health checks

MySQL Support can help

- **MySQL Enterprise Monitor 3.0**
  - does it for you
- Use Community tools
- Open a Service Request
  - send the MEM **support diagnostics** zip to MySQL Support

“If everybody follows your advise we'll be looking for new jobs”

Anonymous Support Colleague

Questions & Answers


MySQL™ Connect