DAMG 6210
Database Design

Nik Bear Brown
@NikBearBrown

SQL Optimization

Topics

- MySQL Architecture
- SQL Optimization
- Backup & Recovery

Benchmark

HW and SW vendors donk

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

MySQL - #1 Open Source Database...

- Thousands of downloads a day
- Millions of active installations
- Provides the date store for the AMP Vertical and other open source stacks
- Available under the GPL license and a commercial license
- Complementary products, such as client administration tools and MySQL Cluster
- MySQL AB founded in 1995
  - Now with operations around the world
  - Providing consulting, support, training and licensing

…and How it Got That Way

- Free to download and use
- Extremely easy to setup, use and maintain
  - Created the “personal” RDBMS trend that is now followed by the large proprietary databases
- Open code base, API and documentation
  - Easy for communities to develop language bindings, such as PHP and Java
- New thinking
  - Fresh code base, written from scratch to fulfill a specific need
  - Defined new parameters of what an RDBMS is, and should do
- Brought the RDBMS to the masses
  - MySQL has been belittled as a “SQL enabled file system”
  - That’s not necessarily always a bad thing

MySQL Features

- Multiple storage-engine architecture
- ACID compliant transactions
- Standards based SQL, aiming for SQL-2003
- Syntax based query caching
- Master/Slave replication
- Compile and runtime feature flexibility to conserve resources
  - Embedded in hardware
  - Extremely high load applications
- Written in C, C++ and ASM
  - 80% in C
  - Parse tree and optimizer in C++
  - String functionality in ASM on some platforms

MySQL - Under the Hood

- Multithreaded - 1 connection means 1 thread
  - Decrease in overhead – increase in performance
  - Configurable per-thread parameters
    - read_buffer_size - I/O per thread
    - read_rnd_buffer_size - for ORDER BY
    - sort_buffer_size - rows in memory for ORDER BY and GROUP BY
    - tmp_table_size – memory temporary tables for GROUP BY
- Multithreaded – 1 server means 1 process
  - Limited addressable space, especially under 32bits
  - 64bit platforms are a must for even moderate datasets
  - 64bit platforms are a must to take advantage of buffers
  - Pay attention to the operating system’s kernel/user memory split in a process

MySQL -Pluggable Storage Engine Architecture

- Storage engines are the heart of a database
  - Provide the data structures and functionality for reliable data persistence and fast data access
  - They are very complex and it’s exponentially difficult to design one that fits all use cases (dataset size, query modeling, traffic patterns)
- MySQL can use multiple storage engines simultaneously
  - Each table can be implemented using a different storage engine
  - An SQL query can operate transparently across storage engines
  - Can cause issues with SQL optimization
- Developers can create their own storage engine for a specific use case

MySQL Configuration

- MySQL assumes little about your hardware system configuration.
- Optimal size of the MyISAM key_buffer used to allocate indices in memory.
- Number of open tables at a given point (it plays an important role when dealing with millions of tables).
- Max number of temporary tables

MySQL Configuration

- Number of connections per second, number of threads created per second, max number of connections
- Thread concurrency
- Thread memory allocation
- Long query time

MySQL Query Handling

- Use EXPLAIN SELECT to improve queries
- Use Indexes
- Simplify some of the WHERE clauses
- Use UNION of SELECT instead of only one SELECT with several conditions in the WHERE clause
- Disabled query cache, provided that the queries would not take advantage of that
- Avoid Table Scans

Other Optimizations

- File System tuning
- File System “mount” options tuning
- OS Linux Scheduler tuning
- OS Linux Kernel Swap tuning
- Tuning for Write-Heavy Loads

Storage Engine: Memory

- RAM based storage engine
  - Data is stored *only* in system memory
  - Schema persists on disk
- Very fast
  - No disk I/O
  - Basic data structures
- Quite limited
  - Fixed column widths – no VARCHAR
  - Limited indexes available

Storage Engine: MyISAM

- File based storage
  - .MYD – table data
  - .MYI – index data
  - .FRM – table definition (schema)
- Easily maintained
  - Architecture-independent data
  - Files can be copied across platforms
- Low overhead
  - No transactions
  - Large grained table level locking
  - Excels at mostly-read applications
  - One third the memory/disk footprint of transactional engines
- Limited
  - Write concurrency
  - Potential for corruption with limited recovery (no transactions)
  - Limited data dictionary (reduced optimizations)
  - Enjoys smaller datasets and simpler queries
- Made MySQL…
  - A “SQL enabled file system”
  - Belittled as a being a toy
  - *Number one*

Storage Engine: InnoDB

- ACID Compliant
  - Atomicity/Consistency/Isolation/Durability
  - Full transactional support and multi-versioning
  - Read Uncommitted, Read Committed, Repeatable Read, Serializable
  - Foreign keys constraints
- Locking and logging
  - Row-level and next-key locking
  - Consistent non-locking reads
  - Commit and rollback segments
- Fault tolerance and table spaces
  - Large datasets, raw partitions
  - Online backups
- Next generation indexing and data storage
  - Clustered and B-tree indexes
- Higher overhead
  - Substantial memory/disk footprint
  - Administration and maintenance
- Made MySQL…
  - Competitive in the enterprise database market
  - Ready to break out of commodity RDBMS use
  - *A target…*

Glitter on its own could be evil, but with rainbows...?
Finn

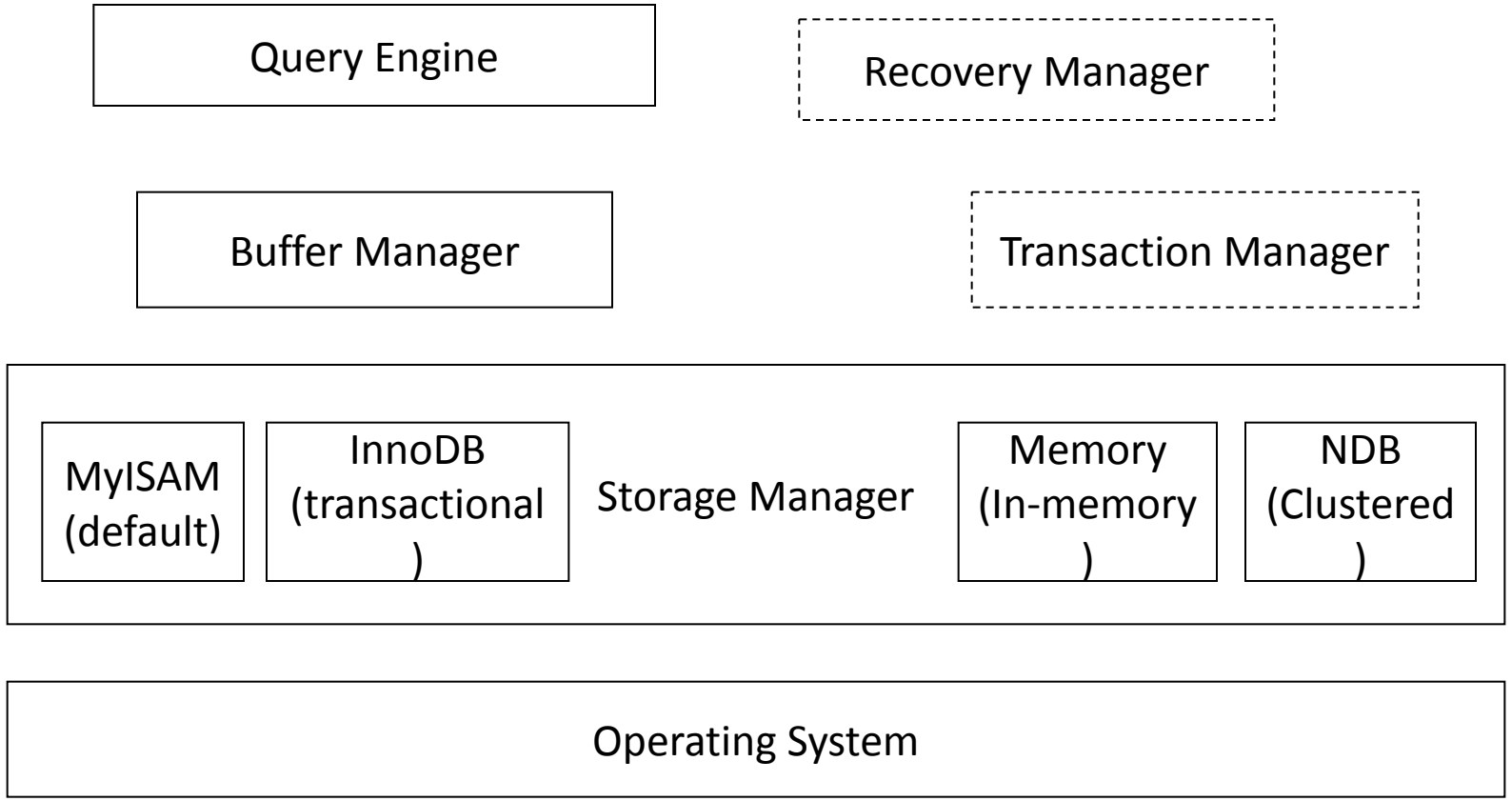
Storage Engine: NDB (Cluster)

- Designed to eliminate any single-point-of-failure
  - The Five-Nines of MySQL
- Shared-nothing data distribution
  - Data redundancy with synchronous replication
  - Transparent sub-second fail-over
  - Available even with multiple node failures
  - Network partitioning and load balancing algorithms
  - Hot backup and restore
- Acquisition of Ericsson's IP and staff
  - Implemented as a pluggable storage engine
  - Memory resident with disk persistence

MySQL Architecture

- Primary subsystems
  - The Query Engine
  - The Storage Manager
  - The Buffer Manager
  - The Transaction Manager
  - The Recovery Manager
- Support components
  - The Process Manager
    - Connectivity functionality
  - Function Libraries
    - General functions used by all the subsystems

MySQL Architecture



Query Engine

- Three components
  - Syntax parser
    - Translates SQL query to an internal form, RAT
    - Correctness of the syntax is established
    - User privileges are used to verify if the query makes legal references to tables (or fields)
  - Query optimizer
    - Selects the most efficient query execution plan
  - Execution component
    - Executes the query by making calls to other modules

The Storage Manager

- Responsible for file organization and indexing
- Interfaces with the OS to write to disk all of the data, such as
  - User tables, indexes and logs
- The component that makes MySQL special by
  - Offering different types of storage engines (or table types)
- Advantages of multiple storage engines
  - When new engines are added to the server, the server architecture allows supporting older file formats
  - Changes to storage engines do not cause changes elsewhere in MySQL
  - Different users/applications have a choice in the storage engine used
  - Easier to introduce new storage media (compact flash) which may require different approach

Storage Engines

- MyISAM (default)
  - extensions to the traditional ISAM
  - No support for transactions
  - No referential integrity
- InnoDB (Transactional)
  - Supports transactions with ‘ACID’ properties
  - Supports referential integrity
- Memory (In-memory)
  - Tables of this type are stored in RAM; therefore fast
  - But data will be lost in the event of a crash
- NDB (Clustered) – advanced storage method - not here
- You will be using mostly MyISAM or InnoDB

Buffer Manager

- Manages memory management issues
  - Between query engine and memory manager
- Maintains Query Cache which
  - Stores the result sets of queries
  - Repeated queries are answered directly from query cache
- New records are cached in before being written to disk
- Query cache is unique to MySQL
- Is responsible for enhanced response times
  - Studies show that MySQL works as fast as Oracle and SQL server (Microsoft’s RDBMS server)

Recovery Manager

- Performs data recovery in the event of loss of data
- Different for different storage engines (or table types)
- InnoDB table provides recovery management
- MyISAM table provides fixes to data inconsistencies due to server outage

Transaction Management

- Performs locking and transaction management
- Different for different storage engines (or table types)
- InnoDB table supports
  - transaction management where transactions obey ‘ACID’ requirements
  - row and table level locking
  - All isolation levels
- MyISAM table supports
  - Table level locking

Specifying the table type

- When you create a table you can specify the table type (storage engine) for that table
- For example

  CREATE TABLE test (id int(5) Primary Key, name varchar(25)) TYPE = InnoDB;
- You can alter the type of an existing table to a different type
- For example

  ALTER TABLE test TYPE = MyISAM;
- A single MySQL database can have tables of different types
- If you don't require transactions, it is best to use MyISAM
- You can use SHOW TABLE STATUS to display information about a table
- For example to display information about table called test

  SHOW TABLE STATUS like 'test' \G

Performance related issues

- MySQL allows users to ask for information about how their select queries are processed using EXPLAIN
- EXPLAIN is the command to be added before your select statement
- For example

```
mysql> explain select * from client where ClientNo='CR74';
+--------+------+--------------+------+---------+------+------+-------------+
| table  | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+--------+------+--------------+------+---------+------+------+-------------+
| client | ALL  | NULL         | NULL |    NULL | NULL |    4 | Using where |
+--------+------+--------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)
```

Please note that the ‘rows’ field is 4 – which means 4 rows (which is all the rows client table has) which requires examination

Please note that the ‘possible_keys’ field is NULL – which means no index defined

Example continued

- Now add the index to the client table

```
mysql> alter table client add index(ClientNo);
Query OK, 4 rows affected (0.05 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

- Now rerun the previous select query asking for explanation

```
mysql> explain select * from client where ClientNo='CR74';
+--------+------+--------------+----------+---------+-------+------+-------------+
| table  | type | possible_keys | key      | key_len | ref   | rows | Extra  |
+--------+------+--------------+----------+---------+-------+------+-------------+
| client | ref  | ClientNo      | ClientNo |       6 | const |    1 | Using where |
+--------+------+--------------+----------+---------+-------+------+-------------+
1 row in set (0.00 sec)
```

- Please note that now only one row needs to be examined and the query therefore runs faster
- Please also note that this table now has an index

Further performance enhancements

- You can improve the performance of your query further by defining smaller indexes
- In the previous example ClientNo is used as the index
  - it is five characters long
  - therefore defining only part of the clientNo as the index may not make much difference
- When longer fields are used as indexes
  - It helps to define the index on part of the field (ref: slide 16)
- Query Optimizer in MySQL inspects all the indexes before selecting one for use in query processing
- You can help MySQL by running the following statement to help in the process of selecting the correct index

  ANALYZE TABLE client;
- You can also instruct MySQL to reclaim space lost due to deletion of records (holes)

  OPTIMIZE TABLE client;

RAM (and more 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)

Real Tables, Real Problems

- Look for bad indexes.
  - Stories used a 20 byte VARCHAR as a primary key.
  - Comments used a 20 byte VARCHAR + an integer as a primary key.
  - Converted both to use integers.
  - Had to keep the old indexes around, of course.

Tip - 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

So spice! So spice!

- 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/Unix Kinda donks on Windows

- 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 ! DB can be Mad chubbs

- 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 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

Processor affinity, or **CPU** pinning enables the binding and unbinding of a process or a thread to a central processing unit (**CPU**) or a range of**CPUs**, so that the process or thread will execute only on the designated **CPU** or **CPUs** rather than any **CPU**.

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

- 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

deadline or noop on Linux are algebraic!

- **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

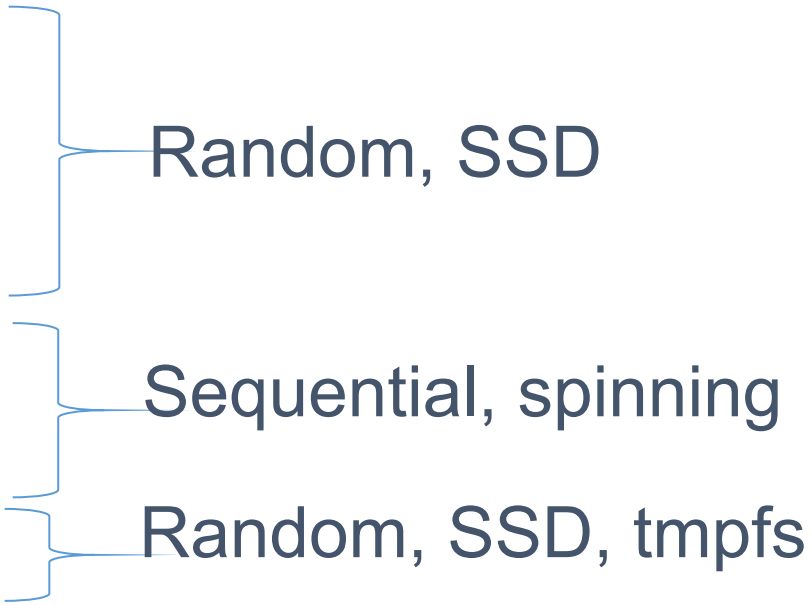
Mad cool

- 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 users 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***



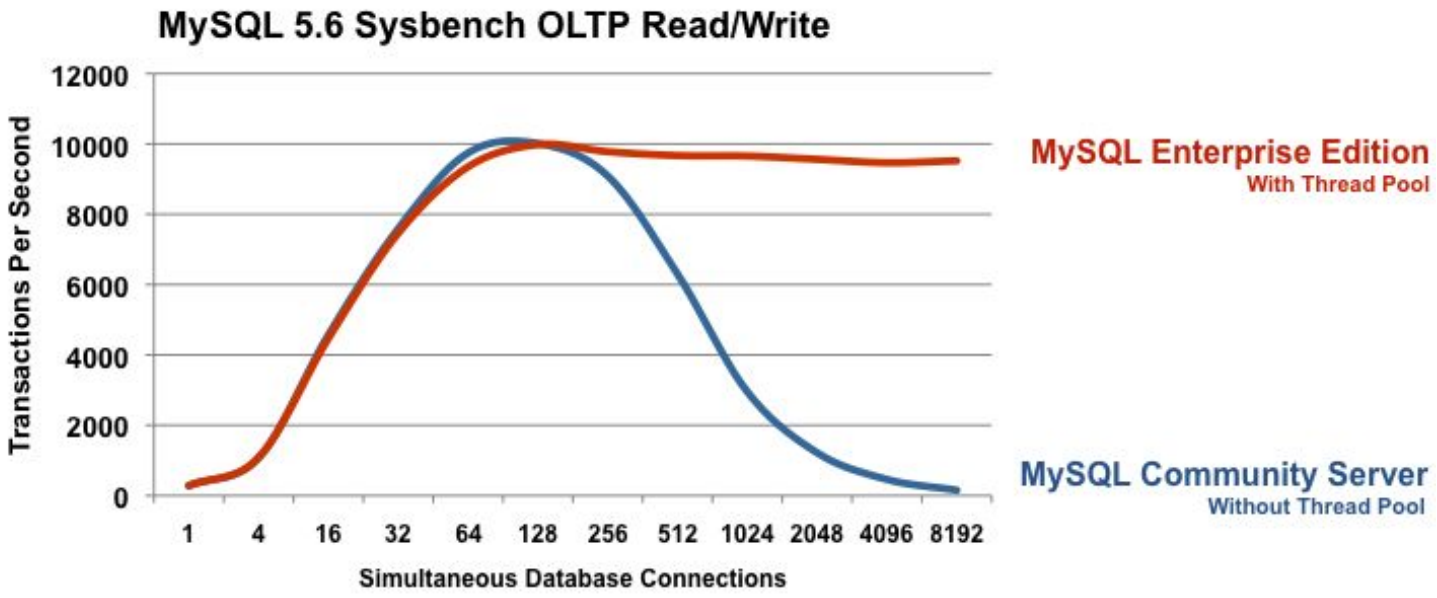
Tip 13. Turn Off the Query Cache

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 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 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

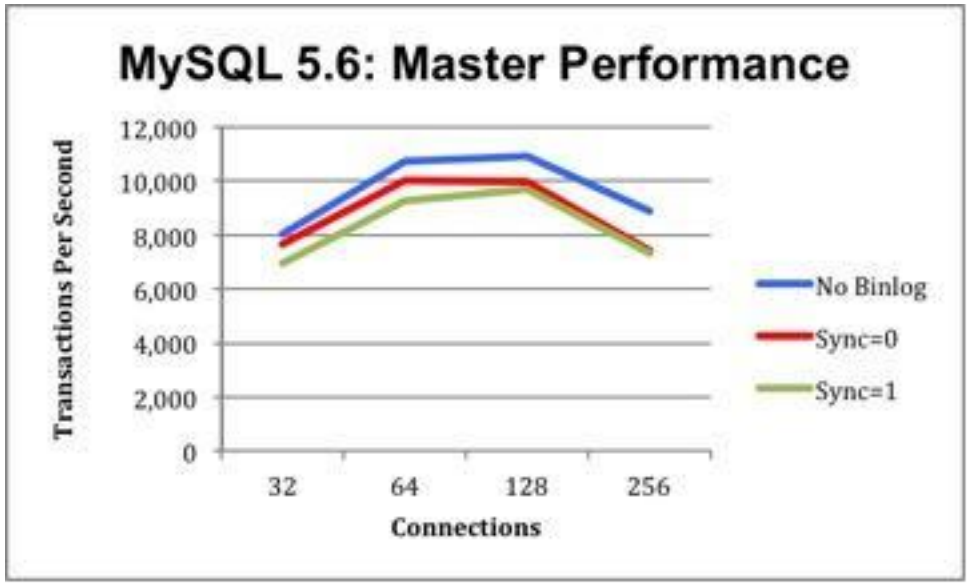
Tip 17. Reduce per thread memory usage

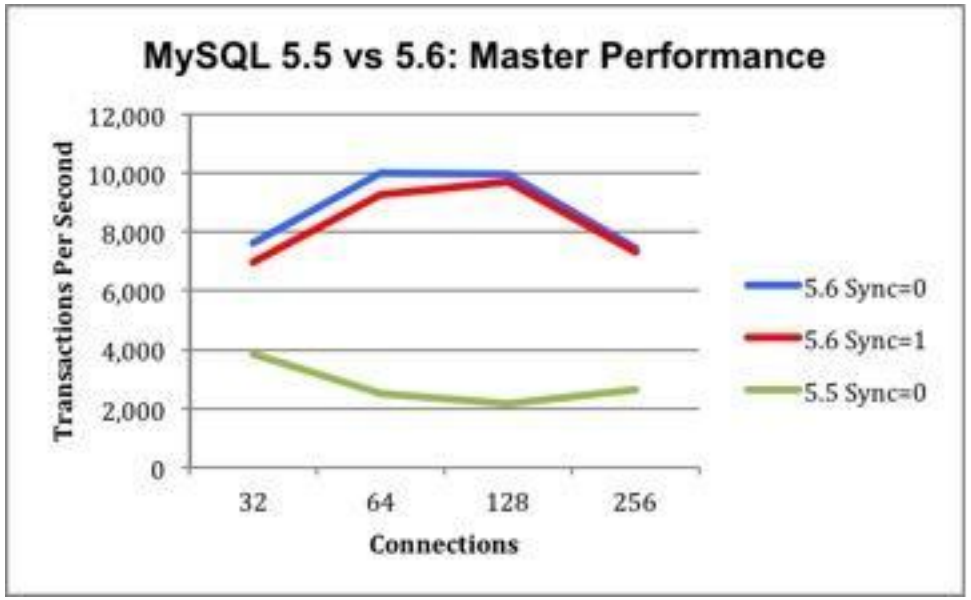
Memory allocation is expensive

- **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 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 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 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 */

Tips 43 - 45

43. Do not use the information_schema in your App
44. Normalize Db
45. Scale out, shard

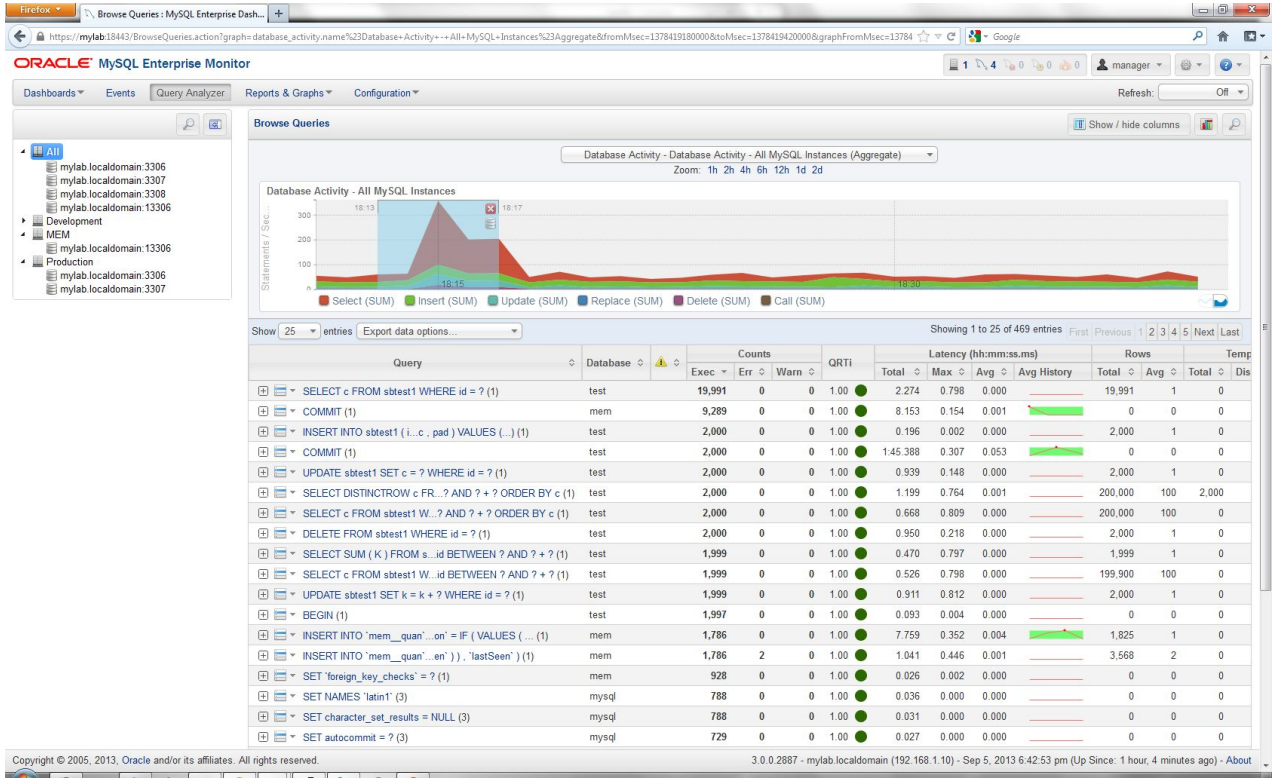
Tip 46. Monitor the database and OS

A monitoring tool is DBA’s friend :

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

MySQL Enterprise Monitor 3.0 is spice!

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

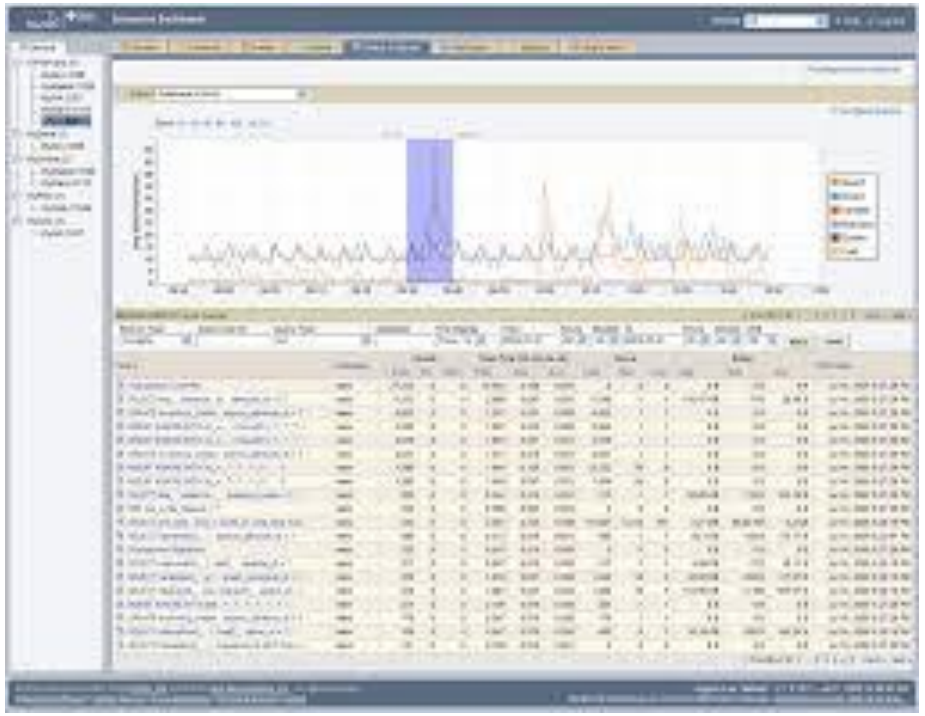


"The MySQL Enterprise Monitor is an absolute must for any DBA who takes his work seriously."

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

MySQL 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

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, workaround :
  - alter table t1 row_format=dynamic;

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 like t;
  insert into t_defrag select * from t
                                          limit 20000;
  ```
  - Fragmentation if Avg_row_length(t) > Avg_row_length(t_defrag)

Avg_row_length from `show table status`
  ```

Tip 49. Upgrade MySQL regularly

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

Tip 50 - MySQL Enterprise Monitor 3.0

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

Backup MySQL

- Learn the various ways to backup MySQL
- Understand the “right ways” to do it
- Know the dangers of the wrong way
- Select and appropriate tools and techniques
  - Data integrity
  - Speed
  - Flexibility

MySQL Backup

- Dump or Raw Backup
- On-line of Off-line
- Table Types and Consistency
- Storage Requirements
- Replication

Dump or Raw Backup?

- Dumps
  - Can be performed remotely
  - Take a lot of space
  - CPU intensive and slower
  - Are plain SQL, so easy to use later
  - Can do selective restore
  - Works for any table type
- Raw Backups
  - Faster, there is no translation
  - Restore “instantly”
  - Works for ISAM/MyISAM only

On-line or Off-line?

- On-line
  - Impacts running applications (can’t write)
  - Must deal with data consistency issues
  - Slower than off-line because of I/O contention
- Off-line
  - Very simple
  - Speed is not an issue
  - Can use normal backup software

Table Types & Consistency

- All tables need to flushed to disk
  - FLUSH TABLES
- Tables can’t be updated during backup
  - LOCK TABLES...
- Related tables treated as a group, lock all
  - FLUSH TABLES WITH READ LOCK
- Transactional tables have logs to backup

Storage Requirements

- How long can you store backups?
  - Consider staggering older backups
  - Never know when you'll want old data
- Raw backups take less space
  - You can keep just the .MYI header
  - Rebuild indexes later if need be
- All forms of backup can be compressed

Replication

- Backup your slave instead of your master
  - Master won’t be interrupted
  - Slave can probably be shut down
  - Or at least you can hold read locks for a long time and nobody will care
- It may be worth setting up a slave just for doing backups
- Remember to backup the master.info file

Tools and Techniques

- mysqldump
- mysqlhotcopy
- mysqlsnapshot
- InnoDB on-line backup
- Off-line backups
- Filesystem snapshots
- Roll your own

mysqldump

- Comes with MySQL
- Dumps local or remote tables/databases
- Variety of output formats
- Handles locking if you need it
- Works best for small and medium installations

mysqlhotcopy

- Comes with MySQL
  - Written by Tim Bunce of Perl DBI fame
- Raw backups of MyISAM tables
- Handles locking
- Regular expression support
  - Match certain databases or tables
- Can truncate indexes to save space

mysqlsnapshot

- Separate download
  - http://jeremy.zawodny.com/mysql/
- Used for setting up replication snapshots
- Or backing up a running server
- Always backs up everything
  - Either one big tar file or one per database
- Very quick to restore

InnoDB on-line Backup

- Separate download
  - http://www.innodb.com/
- Costs extra $$$
- The only good solution for InnoDB backups without replication
- Only does InnoDB tables

Off-line Backups

- Very easy
- Hard to screw up
- Use any backup tool you like
  - Just remember to backup all the MySQL stuff
  - You wouldn’t want compatibility problems after an upgrade
- Restores can be quick

Filesytem Snapshot

- Most “expensive” solution
- Need hardware/software support
  - Veritas, Network Appliance, LVM, EMC, etc.
- Fastest option available
- Uses least disk space
- Doesn’t guard against all failures

Roll your Own

- Not hard to do in Perl, Python, whatever
- Be sure to hold connection open or locks will be released
- Error checking is critical
- Use existing scripts as references
  - mysqlhotcopy
  - mysqlsnapshot