

ORACLE®

MySQL Server and SQL Performance Tuning

Tom Disheroon

Senior Systems Consultant, MySQL





Agenda

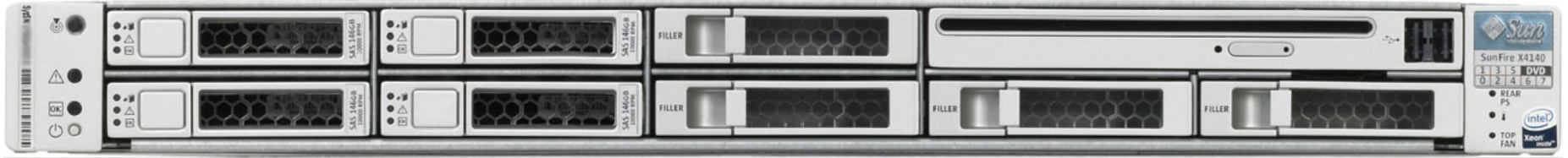
- Overview
- Hardware and Memory
- Basics
- Storage Engines
- MySQL Server Tuning
- Indexing
- Query Tuning Rules
- Schema
- The New Stuff
- What if I need more help?



Overview

- Cover the main steps
 - Show at least one example for each step
 - Examples are things I run into most commonly in the field
 - Include links to MySQL manual for additional information
- This will be technical
- Most everything you need comes with MySQL!
- You cannot become a performance tuning wizard in 45 minutes - PT Class is 4 day class
 - http://www.mysql.com/training/courses/performance_tuning.html
- MySQL Performance Forum
 - <http://forums.mysql.com/list.php?24>

Perfect Hardware?



- Pre 5.5 – No real scale up
- Now with 5.5 we scale up into 64 cores
- x86_64 - 64 bit for **more memory** is important
- Linux or Solaris best, Windows and Unix also fine
- Big gains on Windows with 5.5
- RAID 10 for most, RAID 5 OK if very read intensive
- Hardware RAID battery backed up cache critical!
 - More disks are always better!
 - 4+ recommended, 8-16 can increase IO performance if needed
- Memory should be at least 1/3 to 1/10 data size
 - The more, the better

ORACLE

Basics

- The MySQL server is controlled by “**System Variables**”

```
mysql> show variables like 'auto%';
```

Variable_name	Value
auto_increment_increment	1
auto_increment_offset	1
autocommit	ON
automatic_sp_privileges	ON

```
4 rows in set (0.00 sec)
```

```
shell> mysqladmin -uroot -S /tmp/mysql.sock variables | grep auto
| auto_increment_increment          | 1
| auto_increment_offset            | 1
| autocommit                      | ON
| automatic_sp_privileges          | ON
```

- Set Via:
 - my.cnf / my.ini
 - SET [GLOBAL] <variable>=<value>
 - client, i.e mysql
 - Can be local (session) or global

<http://dev.mysql.com/doc/refman/5.1/en/server-system-variables.html>

Basics

- You monitor a system's performance using “**Status Variables**”

```
mysql> show status like 'innodb_buf%';
```

Variable_name	Value
Innodb_buffer_pool_pages_data	142
Innodb_buffer_pool_pages_dirty	0

```
shell> mysqladmin -uroot -S /tmp/mysql.sock extended
```

Variable_name	Value
Aborted_clients	0
Aborted_connects	0

```
- shell> mysqladmin -u -p ... ex -i 15 -r | grep -v `0`  
http://dev.mysql.com/doc/refman/5.1/en/server-status-variables.html
```

- Enable the slow query log
<http://dev.mysql.com/doc/refman/5.1/en/slow-query-log.html>
 - Analyze using **mysqldumpslow**



Rules of Tuning

- Never make a change in production first
- Have a good benchmark or reliable load
- Start with a good baseline
- Only change 1 thing at a time
 - identify a set of possible changes
 - try each change separately
 - try in combinations of 2, then 3, etc.
- Monitor the results
 - Query performance - query analyzer, slow query log, etc.
 - throughput
 - single query time
 - average query time
 - CPU - top, vmstat
 - IO - iostat, top, vmstat, bonnie++
 - Network bandwidth
- Document and save the results



Where do I find a benchmark?

- Make your own
 - Can use general query log output
 - Could use MySQL Proxy and TCP Dump
- DBT2
 - <http://osdl dbt.sourceforge.net/>
 - <http://samurai-mysql.blogspot.com/2009/03/settingup-dbt-2.html>
- mysqlslap MySQL 5.1 +
 - <http://dev.mysql.com/doc/refman/5.1/en/mysqlslap.html>
- SysBench
 - <http://sysbench.sourceforge.net/>
- supersmack
 - <http://vegan.net/tony/supersmack/>
- mybench
 - <http://jeremy.zawodny.com/mysql/mybench/>

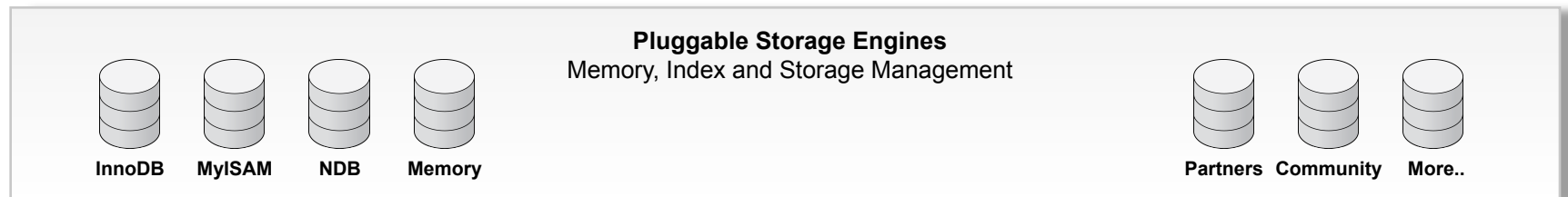
A photograph of an industrial facility with a complex network of large, brightly colored pipes. On the left, a series of yellow pipes run horizontally and curve downwards. On the right, blue pipes run vertically and curve. In the background, orange pipes are visible. The pipes are connected by various valves and flanges. The scene is lit by overhead industrial lights.

MySQL Storage Engines

**SOFTWARE.
HARDWARE.
COMPLETE.**

MySQL Supports Multiple Storage Engines

- Selecting the storage engine to use *is a tuning decision*



```
mysql> SHOW TABLE STATUS like 'Tommy%'\G
***** 1. row *****
      Name: TommyTest
      Engine: InnoDB
```

```
mysql> ALTER TABLE TommyTest ENGINE=MyISAM;
Query OK, 0 rows affected (0.40 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> SHOW TABLE STATUS like 'Tommy%'\G
***** 1. row *****
      Name: TommyTest
      Engine: MyISAM
```

ORACLE



InnoDB

- Transactional and fully ACID compliant
- Behavior most like traditional databases such as Oracle, DB2, SQL Server, etc.
- Data size is normally 2-3 X MyISAM, not with 5.5!!
- MVCC = Non-blocking reads in most cases
- Fast, reliable recovery from crashes with zero committed data loss
- **Always** clustered on the primary key
 - Lookups by primary key, very fast
 - Range scans on primary key also very fast
 - Non-Primary key lookups use the primary key to find the record, this means 2 key lookups
 - Important to keep primary key small

<http://dev.mysql.com/doc/refman/5.1/en/innodb.html>



MyISAM

- Formerly the faster read only engine
 - Most web applications
 - Perfect for web search databases
 - 80/20 read/modify or higher
 - pure inserts and deletes with partitions or merge engine
 - no transactions
 - reporting DB/ Data Warehouse
- Most compact data of all non-compressed engines
- Table locking
- Not **ACID** compliant, non-transactional
- Supports concurrent inserts
- Full-Text and Geospatial support

<http://dev.mysql.com/doc/refman/5.1/en/myisam-storage-engine.html>



MySQL Server Tuning

**SOFTWARE.
HARDWARE.
COMPLETE.**



InnoDB Tuning

- Unlike MyISAM InnoDB uses a single cache for both index and data
 - **Innodb_buffer_pool_size** - should be 70-80% of available memory.
 - It is not uncommon for this to be very large, i.e. 44GB on a system with 40GB of memory
 - Make sure its not set so large as to cause swapping!
 - `mysql>show status like 'Innodb_buffer%'` ;
- InnoDB can use direct IO on systems that support it, linux, FreeBSD, and Solaris.
 - **Innodb_flush_method** = O_DIRECT
- For more InnoDB tuning see
<http://dev.mysql.com/doc/refman/5.1/en/innodb-tuning-troubleshooting.html>

Cache hot application data in memory

DBT-2 (W200)	Transactions per Minute	%user	%iowait
Buffer pool 1G	1125.44	2%	30%
Buffer pool 2G	1863.19	3%	28%
Buffer pool 5G	4385.18	5.5%	33%
Buffer pool 30G (All data in cache)	36784.76	36%	8%

- DBT-2 benchmark (write intensive)
- 20-25GB hot data (200 warehouses, running 1 hour)
- Nehalem 2.93GHz x 8 cores, MySQL 5.5.2, 4 RAID1+0 HDDs
- RAM size affects everything. Not only for SELECT, but also for INSERT/UPDATE/DELETE
 - INSERT: Random reads/writes happen when inserting into indexes in random order
 - UPDATE/DELETE: Random reads/writes happen when modifying records



MyISAM Tuning

- The primary tuning factors in MyISAM are its two caches:
 - **key_buffer_cache** - should be 25% of available memory
 - system cache - leave 75% of available memory free
- Available memory is:
 - All on a dedicated server, if the server has 8GB, use 2GB for the **key_buffer_cache** and leave the rest free for the system cache to use.
 - Percent of the part of the server allocated for MySQL, i.e. if you have a server with 8GB, but are using 4GB for other applications then use 1GB for the **key_buffer_cache** and leave the remaining 3GB free for the system cache to use.
- You can define multiple key buffer's
- You can pre-load the key buffers
- For more details on configuring the MyISAM key cache see:

<http://dev.mysql.com/doc/refman/5.1/en/myisam-key-cache.html>

Monitoring the MyISAM Key Buffer Cache

```
mysql> SHOW STATUS like 'key%';
+-----+-----+
| Variable_name      | Value |
+-----+-----+
| Key_blocks_not_flushed | 0      | Dirty key blocks not flushed to disk
| Key_blocks_unused    | 28995  | Unused blocks in the cache
| Key_blocks_used      | 0      | Used blocks in the cache
| Key_read_requests    | 0      | Key read requests to the cache
| Key_reads            | 0      | times a key read request went to disk
| Key_write_requests   | 0      | Key write requests to the cache
| Key_writes          | 0      | times key write request went to disk
+-----+-----+
7 rows in set (0.00 sec)
```

- % of cache free : $\text{Key_blocks_unused} / (\text{Key_blocks_unused} + \text{Key_blocks_used})$
- Cache read hit % : $\text{Key_reads} / \text{Key_read_requests}$
- Cache write hit % : $\text{Key_writes} / \text{Key_write_request}$
- cat /proc/meminfo to see the system cache in linux
 - MemFree + Cached = memory available for system cache



Connections

- MySQL Caches the threads used by a connection
 - `thread_cache_size` - Number of threads to cache
 - Setting this to 100 or higher is not unusual
- Monitor `Threads_created` to see if this is an issue
 - Counts connections **not** using the thread cache
 - Should be less than 1-2 a minute
 - Usually only an issue if more than 1-2 a second
- Only an issue if you create and drop a lot of connections, i.e. PHP
- Overhead is usually about 250k per thread
- `Aborted_clients` -
<http://dev.mysql.com/doc/refman/5.1/en/communication-errors.html>
- `Aborted_connections` -
<http://dev.mysql.com/doc/refman/5.1/en/communication-errors.html>



Sessions

- Some session variables control space allocated by each session (connection)
 - Setting these to small can give bad performance
 - **Setting these too large can cause the server to swap!**
 - Can be set by connection
 - `SET SORT_BUFFER_SIZE=1024*1024*128`
 - Set small by default, increase in connections that need it
- `sort_buffer_size` - Used for ORDER BY, GROUP BY, SELECT DISTINCT, UNION DISTINCT
 - Monitor `Sort_merge_passes` < 1-2 an hour optimal
 - Usually a problem in a reporting or data warehouse database
- Other important session variables
 - `read_rnd_buffer_size` - Set to 1/2 `sort_buffer_size`
 - `join_buffer_size` - (BAD) Watch `Select_full_join`
 - `read_buffer_size` - Used for full table scans, watch `Select_scan`
 - `tmp_table_size` - Max temp table size in memory, watch `Created_tmp_disk_tables`



Query Cache

- MySQL's Jekyll and Hyde of performance tuning options, when it is useful it really helps, when it hurts, it really hurts
- MySQL Query Cache caches both the query and the full result set
 - `query_cache_type` - Controls behavior
 - 0 or OFF - Not used (buffer may still be allocated)
 - 1 or ON cache all unless `SELECT SQL_NO_CACHE` (**DEFAULT**)
 - 2 or DEMAND cache none unless `SELECT SQL_CACHE`
 - `query_cache_size` - Determines the size of the cache
- `mysql> show status like 'Qc%' ;`
- Gives great performance if:
 - Identical queries returning identical data are used often
 - No or rare inserts, updates or deletes
- Best Practice
 - Set to DEMAND
 - Add `SQL_CACHE` to appropriate queries
- See <http://dev.mysql.com/doc/refman/5.1/en/query-cache-configuration.html>

A low-angle photograph of a construction site. Several large blue and yellow tower cranes are visible against a clear blue sky. In the foreground, a building is under construction, featuring a section with large glass windows and a concrete structure with visible rebar. A red rectangular box is overlaid on the left side of the image, containing the word 'Indexing' in white text. To the right of this box, a white rectangular box contains the text 'SOFTWARE. HARDWARE. COMPLETE.' in black and red.

Indexing

**SOFTWARE.
HARDWARE.
COMPLETE.**



Indexes in MySQL

- Indexes allow for faster access to data
- Data accessed via an index is usually in sorted order
- Unique or Primary - Must refer to only one record
- Non-Unique - May refer to many records
- Can be on one or more columns
 - `CREATE INDEX IDX ON TAB1 (col1,col2,col3) ;`
- Can use prefix index for
 - `CHAR, VARCHAR, BINARY, and VARBINARY`
 - `CREATE INDEX PRE ON TAB1 (COL1(10)) ;`
 - Prefix is in bytes, **not** characters
 - Very useful for large strings
 - Works best when leading part of column is selective



How Indexes are Used - Filter

- You can use indexes to improve the access to filter data
 - `SELECT * FROM TAB WHERE CITY= 'MIAMI' ;`
- A filter Index returns, zero, one or more records
 - Usually zero or one for a unique or primary index
 - Zero, one, or more for a non-unique index
- If there is no `ORDER BY` clause then the data is returned in the order of the index
- The index will not be used if:
 - The table only has a few rows
 - The total number of rows is more than ~10% of the table
 - It is faster to do a full table scan without the index



How Indexes are Used - Join

- Indexes speed up joins
 - `SELECT X.A, Y.B FROM X,Y`
 - `WHERE X.C = 'FL' and Y.A = X.A ;`
- The Filter is on column C of table X
 - Table X needs an index on column C for the filter
- Table Y is joined to table X by column A
 - Table Y needs an index on column A
- MySQL reads each row of table X using the index on X.C with a value of 'FL'
- MySQL then uses the index on Y.A to join Y to A
- Optimizer may chose other path ...



How Indexes are Used - Sort (not for HASH index)

- MySQL can use indexes to speed up some ORDER BY operations
- SELECT Name from TAB1 ORDER By NAME
 - An Index on name can be used for the sort
- SELECT NAME FROM TAB WHERE NAME BETWEEN 'AAA' and 'CCC' ORDER By NAME ;
 - An index on name can still be used for the order by
- SELECT NAME FROM TAB WHERE NAME CITY = 'MIAMI' ORDER By NAME ;
 - You cannot use the NAME index for the order by.
 - Requires a sort.
- Sometimes a sort is faster than an index scan



Index Best Practices

- Too many indexes can slow down inserts/deletes
 - Use only the indexes you must have
 - Check often
 - `mysql>show create table tabname ;`
- Don't duplicate leading parts of compound keys
 - `index key123 (col1,col2,col3)`
 - `index key12 (col1,col2) <-` Not needed!
 - `index key1 (col1) <--` Not needed!
- Use prefix indexes on large keys
- Best indexes are 16 bytes/chars or less
- Indexes bigger than 32 bytes/chars should be looked at very closely
 - should have there own cache if in MyISAM
- For large strings that need to be indexed, i.e. URLs, consider using a separate column using the MySQL MD5 to create a hash key and index on it instead

Explain

- Order that the tables are accessed
- Indexes used
- Estimated number of rows accessed per table
- `select C.Name, Y.Name, Y.Population, Language from Country as C, City as Y, CountryLanguage as L where Y.Name = C.Name and L.CountryCode = Y.CountryCode and C.Name = 'Macao' ;`

```
explain select C.Name, Y.Name, Y.Population, Language from Country as C, City as Y, CountryLanguage as L where Y.Name = C.Name and L.CountryCode = Y.CountryCode and C.Name = 'Macao' ;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key      | key_len | ref              | rows | Extra              |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE      | C      | ALL  | NULL          | NULL     | NULL    | NULL             | 239 | Using where        |
| 1 | SIMPLE      | Y      | ALL  | NULL          | NULL     | NULL    | NULL             | 4079 | Using where; Using join buffer |
| 1 | SIMPLE      | L      | ref  | PRIMARY       | PRIMARY  | 3       | world.Y.CountryCode | 9 | Using index        |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

(Using the MySQL World database)

ORACLE



Explain - Details

- Tables are accessed from top to bottom
- Columns
 - Select Type - SELECT if no Union or Subquery
 - Table, uses aliases
 - Type - Most common ref or eq_ref
 - Possible Keys - Indexes the optimizer is considering
 - Key = The index the optimizer chose
 - Ref - What column in what table (using alias) is referenced by the index
 - Rows - Estimated number of rows **per reference**
 - Multiple these to get overall cost
- There are more values, see:

<http://dev.mysql.com/doc/refman/5.1/en/using-explain.html>

More Explain

- alter table Country add index c2 (Name) ;
- alter table City add index c2 (Name) ;

```
mysql> explain select C.Name, Y.Name, Y.Population, Language from Country as C, City as Y, CountryLanguage as L where Y.Name = C.Name and L.CountryCode = Y.CountryCode and C.Name = 'Macao' ;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	C	ref	c2	c2	52	const	1	Using where; Using index
1	SIMPLE	Y	ref	c2	c2	35	const	1	Using where
1	SIMPLE	L	ref	PRIMARY	PRIMARY	3	world.Y.CountryCode	9	Using index

3 rows in set (0.00 sec)

- The original cost was $239 * 4079 * 9 = 8,773,929$
- The new cost is $1 * 1 * 9 = 9$

A low-angle, upward-looking photograph of a modern skyscraper with a glass facade, seen through a series of dark, intersecting structural beams that form a grid-like pattern. The sky is a pale, overcast blue. The image has a clean, architectural feel.

Query Tuning Rules

**SOFTWARE.
HARDWARE.
COMPLETE.**



Queries I

- Often the # 1 issue in overall performance
- ***Always, Always have your slow query log on!***
<http://dev.mysql.com/doc/refman/5.1/en/slow-query-log.html>
 - Use: `log_queries_not_using_indexes`
 - Check it regularly
 - Use `mysqldumpslow` :
<http://dev.mysql.com/doc/refman/5.1/en/mysqldumpslow.html>
 - Best practice is to automate running `mysqldumpslow` every morning and email results to DBA, DBDev, etc.
- Understand and use `EXPLAIN`
<http://dev.mysql.com/doc/refman/5.1/en/using-explain.html>
- `Select_scan` - Number of full table scans
- `Select_full_join` - Joins without indexes
- MySQL Query Analyzer
<http://www.mysql.com/products/enterprise/query.html>



Queries II

- The IN clause in MySQL is very fast!
 - Select ... Where idx IN(1,23,345,456)
 - Much faster than a join
 - I have done tests with 80,000 items in the in list
 - 1,000-2,000 not unusual
- Don't wrap your indexes in expressions in Where
 - Select ... Where func(idx) = 20 [index ignored]
 - Select .. Where idx = otherfunc(20) [may use index]
 - Best practice : Keep index alone on left side of condition
- Avoid % at the start of LIKE on an index
 - Select ... Where idx LIKE('ABC%') can use index
 - Select ... Where idx LIKE('%XYZ') must do full table scan
- Use **union all** when appropriate, default is **union distinct!**
- Understand left/right joins and use only when needed

<http://dev.mysql.com/doc/refman/5.1/en/query-speed.html>

Schema

**SOFTWARE.
HARDWARE.
COMPLETE.**





Schemas

- Size = performance, smaller is better
 - Size right! Do not automatically use 255 for VARCHAR
 - Temp tables, most caches, expand to full size
- Use “procedure analyse” to determine the optimal types given the values in your table
 - <http://dev.mysql.com/doc/refman/5.1/en/procedure-analyse.html>
 - `mysql> select * from tab procedure analyse (64,2000) \G`
- Consider the types:
 - enum : <http://dev.mysql.com/doc/refman/5.1/en/enum.html>
 - set : <http://dev.mysql.com/doc/refman/5.1/en/set.html>
- Compress large strings
 - Use the MySQL COMPRESS and UNCOMPRESS functions
 - Very important in InnoDB pre 5.5, now with barracuda not so bad



**The NEW Stuff,
Making DBA Life EASIER!**

MySQL 5.5 Performance Schema

- **PERFORMANCE_SCHEMA** presents low level MySQL performance information
- Data can be cleared
- Filters with **WHERE** are allowed
- Must be enabled with **--performance_schema**

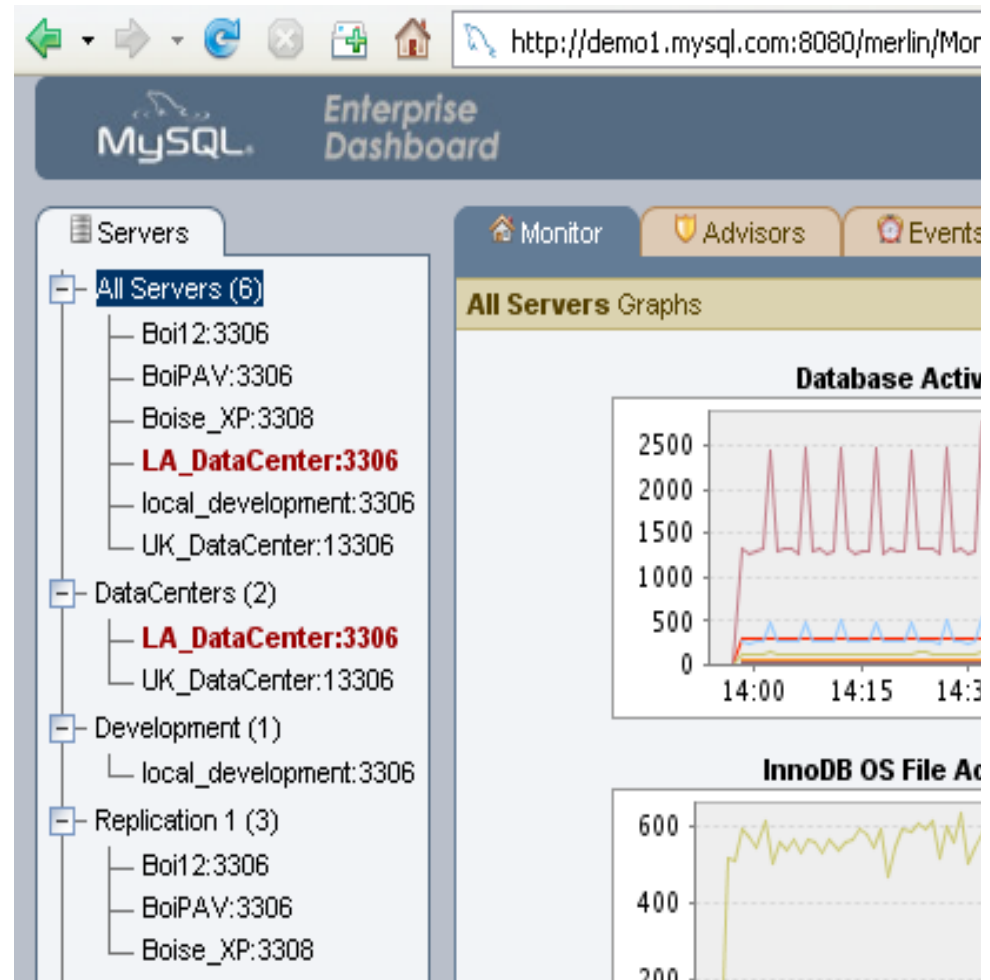
```
mysql> SELECT EVENT_ID, EVENT_NAME, TIMER_WAIT
-> FROM EVENTS_WAITS_HISTORY WHERE THREAD_ID = 13
-> ORDER BY EVENT_ID;
```

EVENT_ID	EVENT_NAME	TIMER_WAIT
86	wait/synch/mutex/mysys/THR_LOCK::mutex	686322
87	wait/synch/mutex/mysys/THR_LOCK_malloc	320535
88	wait/synch/mutex/mysys/THR_LOCK_malloc	339390
89	wait/synch/mutex/mysys/THR_LOCK_malloc	377100
90	wait/synch/mutex/sql/LOCK_plugin	614673
91	wait/synch/mutex/sql/LOCK_open	659925
92	wait/synch/mutex/sql/THD::LOCK_thd_data	494001
93	wait/synch/mutex/mysys/THR_LOCK_malloc	222489
94	wait/synch/mutex/mysys/THR_LOCK_malloc	214947
95	wait/synch/mutex/mysys/LOCK_alarm	312993

```
mysql> UPDATE SETUP_INSTRUMENTS
-> SET ENABLED = 'NO'
-> WHERE NAME = 'wait/synch/mutex/myisammrg/MYRG_INFO::mutex';
mysql> UPDATE SETUP_CONSUMERS
-> SET ENABLED = 'NO' WHERE NAME = 'file_summary_by_instance';
```

MySQL Enterprise Monitor

- Single, consolidated view into entire MySQL environment
- Auto-discovery of MySQL servers, replication topologies
- Customizable rules-based monitoring and alerts
- Query monitoring and analysis
- Identifies problems **before** they occur
- Reduces risk of downtime
- Makes it easier to scale out without requiring more DBAs



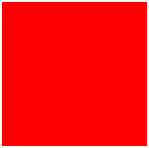
A Virtual MySQL DBA Assistant!

ORACLE



Demo Time


ORACLE





Learn More: Resources

- View MySQL Essentials Webinars (Part 1 – Part 7)
<http://mysql.com/news-and-events/web-seminars/mysql-essentials.html>
- MySQL Training Course – MySQL Performance Tuning
http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getCourseDesc?dc=D61820GC10&p_org_id=1001
- MySQL Performance Forum
<http://forums.mysql.com/list.php?24>
- Download MySQL 5.5
<http://www.mysql.com/downloads/mysql/>
- Download Free MySQL White Papers
<http://dev.mysql.com/why-mysql/white-papers/>
- Try MySQL Enterprise Edition (including MySQL Enterprise Monitor):
<http://www.mysql.com/trials/>



The presentation 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.

ORACLE®