ORACLE®

MySQL™

Optimizing MySQL Scalability and Performance

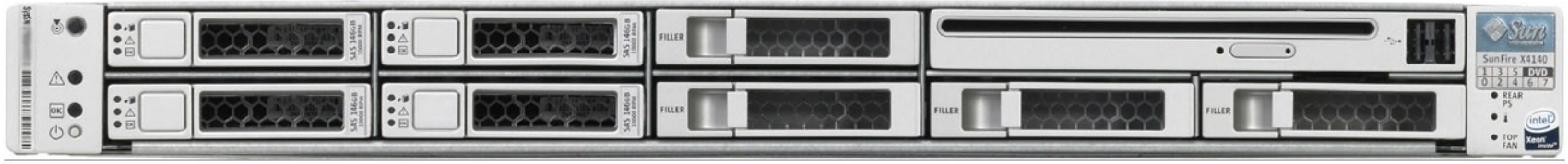
Agenda

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

Overview

- Cover the main steps
  - Show at least one example for each step
  - Examples are things 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

Hardware: The Perfect MySQL Server



- The more cores the better (especially for 5.5 and later)
- x86_64 - 64 bit for more memory is important
  - The more the better
- Fast HD (10-15k RPM SATA) or NAS/SAN……
  - 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
- …Or SSD (for higher throughput)
  - Intel, Fusion-IO good choices; good option for Slaves
- At least 2 x NICs for redundancy
- Slaves should be as powerful as the Master

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

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 Benchmarking

- 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, dstat
  - IO - iostat, top, vmstat, bonnie++, dstat
  - Network bandwidth
- Document and save the results

Were do I find a benchmark?

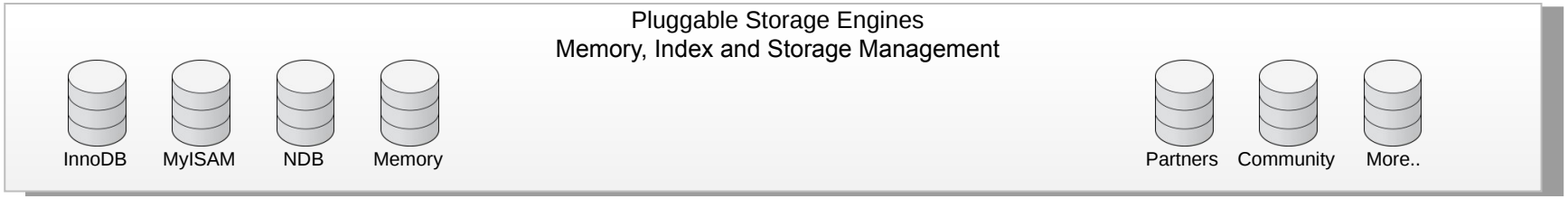
- Make your own
  - Can use general query log output
  - Could use MySQL Proxy and TCP Dump
- DBT2
  - http://osdldbt.sourceforge.net/
  - http://samurai-mysql.blogspot.com/2009/03/settingup-dbt-2.html
- mysqlslap MySQL 5.1+
  - http://dev.mysql.com/doc/refman/5.1/en/mysqlslap.html
- SysBench
  - http://sysbench.sourceforge.net/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

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
```

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 or foreign key support
  - reporting DB/ Data Warehouse
- Most compact data of all non-compressed engines
- Table level locking
- Not ACID compliant, non-transactional
- Full-Text and Geospatial support **
- Default SE before MySQL 5.5

InnoDB

- Transactional and fully ACID compliant
- Behaviour most like traditional databases such as Oracle, DB2, SQL Server, etc.
- Default SE from from MySQL 5.5
- MVCC = Non-blocking reads in most cases
- Row level locking
- 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
  - Important to keep primary key small

MySQL Server Tuning

SOFTWARE.
HARDWARE.
COMPLETE.

Server Tuning

- Thread_cache_size
  - Number of threads server cache for reuse.
  - It is costly to create new threads.
  - Look at status variable Threads_created to see if you need to raise thread_cache_size.
- Table_cache
  - Number of open tables for all threads.
  - It costly to open files.
  - Look at status variable Opened_tables to see if you need to raise table_cache.

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
  - Not much inserts, updates or deletes
  - Low amount on concurrency, single threaded "service"
  - Not to big, set to max 32Mb.
- Best Practice
  - Set to DEMAND
  - Add SQL_CACHE to appropriate queries

```
CREATE EVENT `flush_q_cache`
ON SCHEDULE EVERY 60 MINUTE
STARTS '2011-02-15 20:28:01'
ON COMPLETION NOT PRESERVE
ENABLE
DO FLUSH QUERY CACHE
```

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
  - Percent of the part of the server allocated for MySQL

- 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     |
| Key_blocks_unused      | 28995 |
| Key_blocks_used        | 0     |
| Key_read_requests      | 0     |
| Key_reads              | 0     |
| Key_write_requests     | 0     |
| Key_writes             | 0     |
+------------------------+-------+
7 rows in set (0.00 sec)
```

Dirty key blocks not flushed to disk
Unused blocks in the cache
Used blocks in the cache
Key read requests to the cache
times a key read request went to disk
Key write requests to the cache
times key write request went to disk

- % of cache free: `Key_blocks_unused / (Key_blocks_unused + Key_blocks_used)`
- Cache read hit %: (1-(`Key_reads/Key_read_requests))x100`
- Cache write hit %: (1-(`Key_writes/Key_write_request))x100`
- `cat /proc/meminfo` to see the system cache in Linux
  - MemFree + Cached = memory available for system cache

InnoDB Tuning

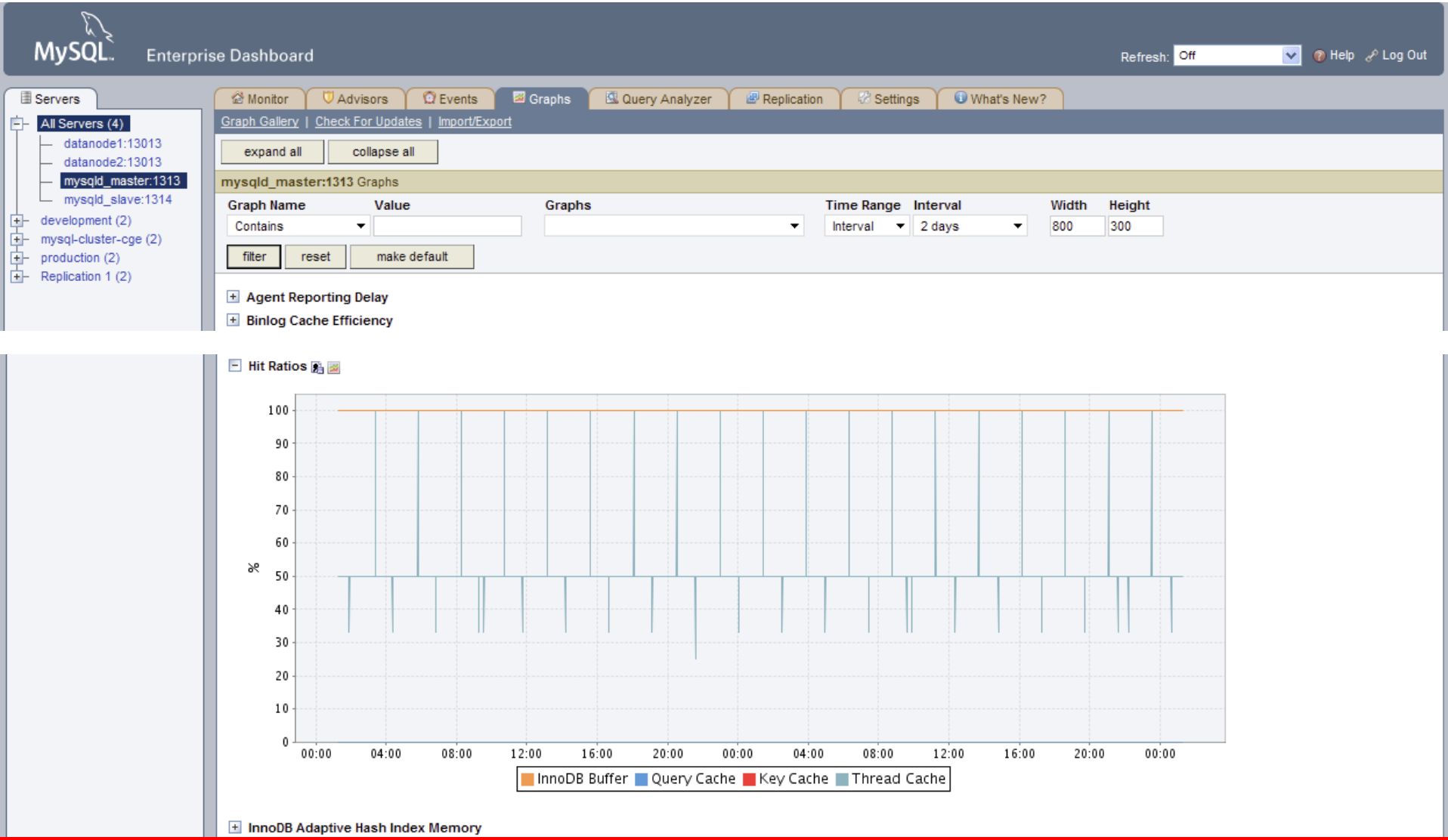
- Unlike MyISAM - InnoDB uses a single cache for both index and data.
  - `innodb_buffer_pool_size` - should be 70-80% of available memory.
  - It is not uncommon for this to be very large, i.e. 34GB on a system with 40GB of memory
  - `mysql>show status like 'Innodb_buffer%' ;`
- Innodb_log_file_size, default 5Mb, to small for production.
- innodb_flush_log_at_trx_commit, run InnoDB in “loose” mode.
- 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

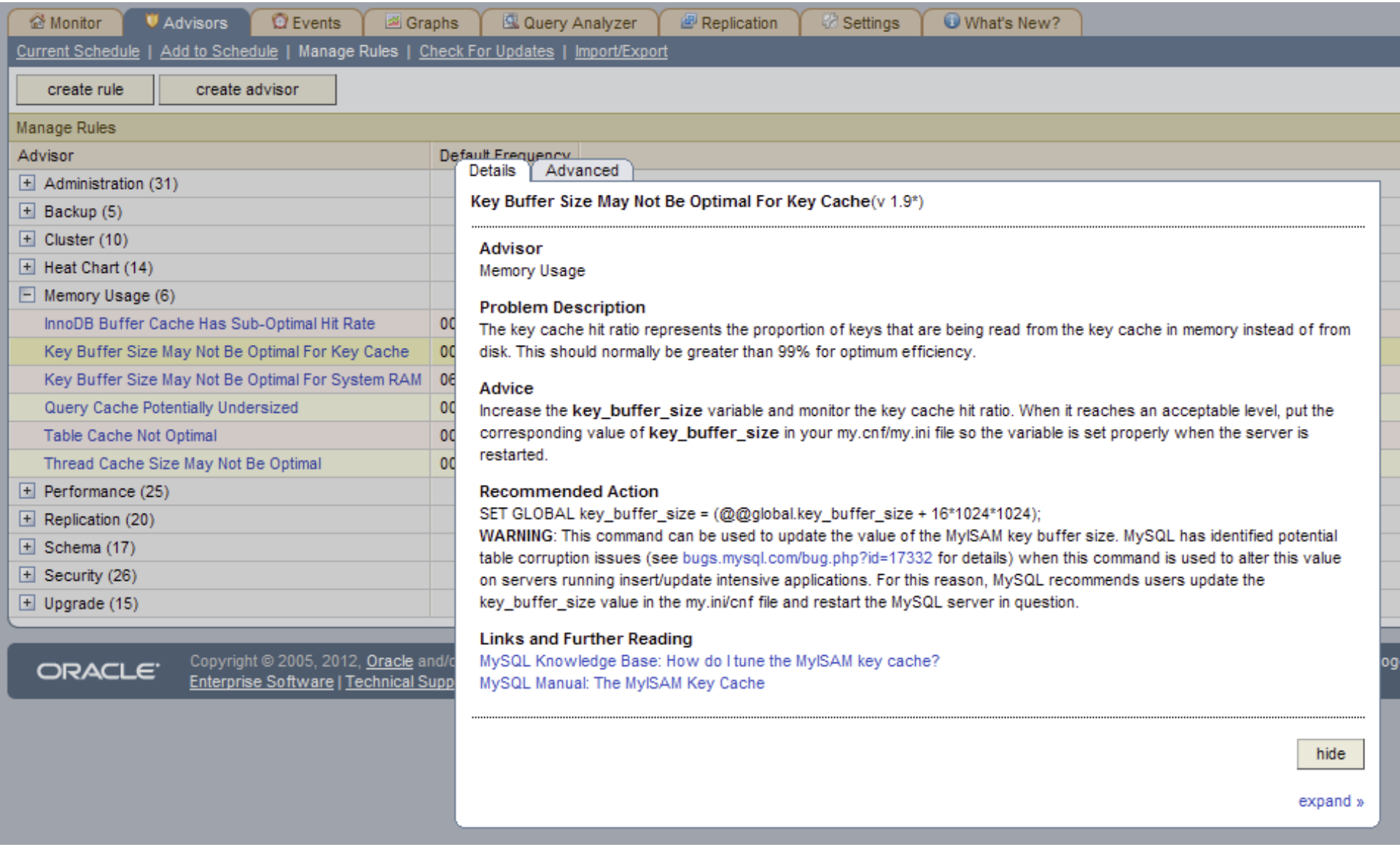
DBT-2 (W200)	Transactions per Minute	%user	%iowait
Buffer pool 1G	1125.44	2%	30%
Buffer pool 2G	1863.19	3%	28%
Buffer pool 5G	4385.18	5.5%	33%
Buffer pool 30G (All data in cache)	36784.76	36%	8%

- 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

To make it easy… MySQL Enterprise Monitor



To make it easy… MySQL Enterprise Monitor



MySQL Enterprise Advisors

140+ rules designed to enforce MySQL Best Practices

ADMINISTRATION

- Helps DBA better manage database processes
- Suggests improvements for smoother operations

SECURITY

- Protects MySQL Servers
- Uncovers Security loopholes

UPGRADE

- Monitors and Advises Bugs that affect current installation
- Provides update path to correcting MRU/QSP

CUSTOM

- Built by DBA to Enforce Organization specific best practices
- Create New or Tailor MySQL Advisors to fit needs

REPLICATION

- Makes suggestions for improving replication design
- Identifies potential replication bottlenecks

MEMORY

- Ensures optimum use of memory
- Minimizes disk access for read intensive systems.

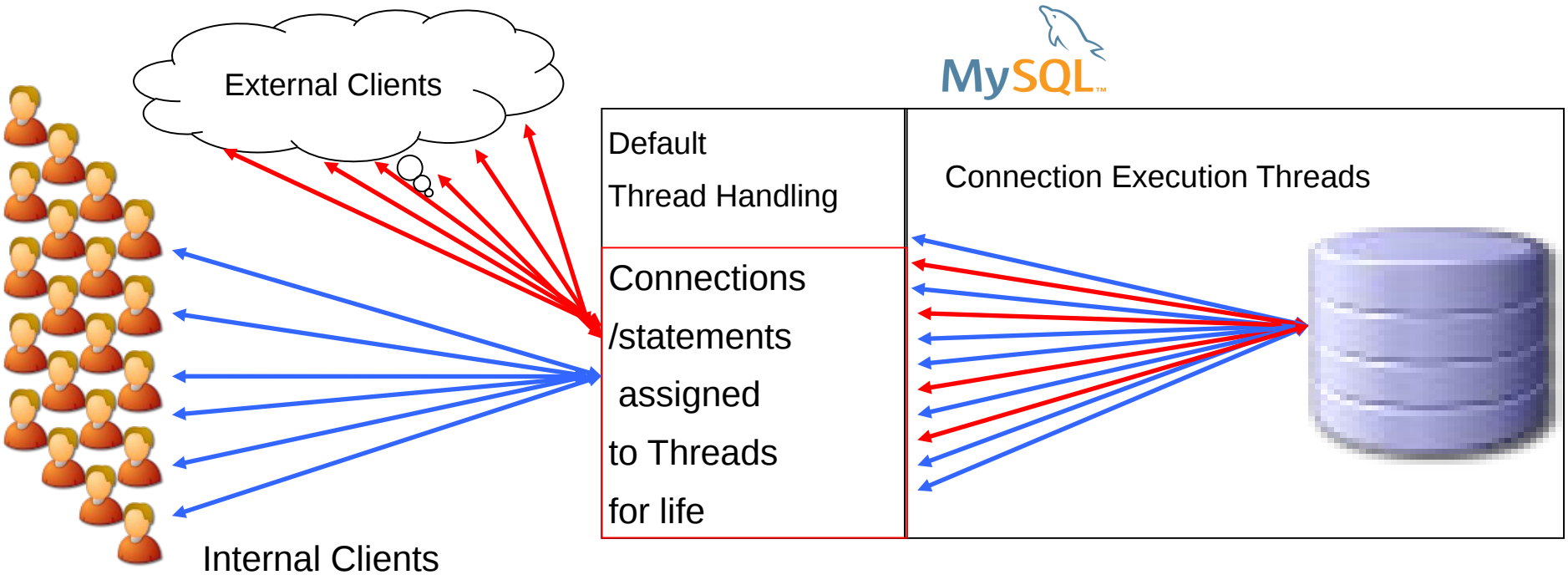
PERFORMANCE

- Makes suggestions for improving database speed
- Identifies potential performance bottlenecks

SCHEMA

- Helps DBA design better databases
- Uncovers Security loopholes

Default Thread Handling

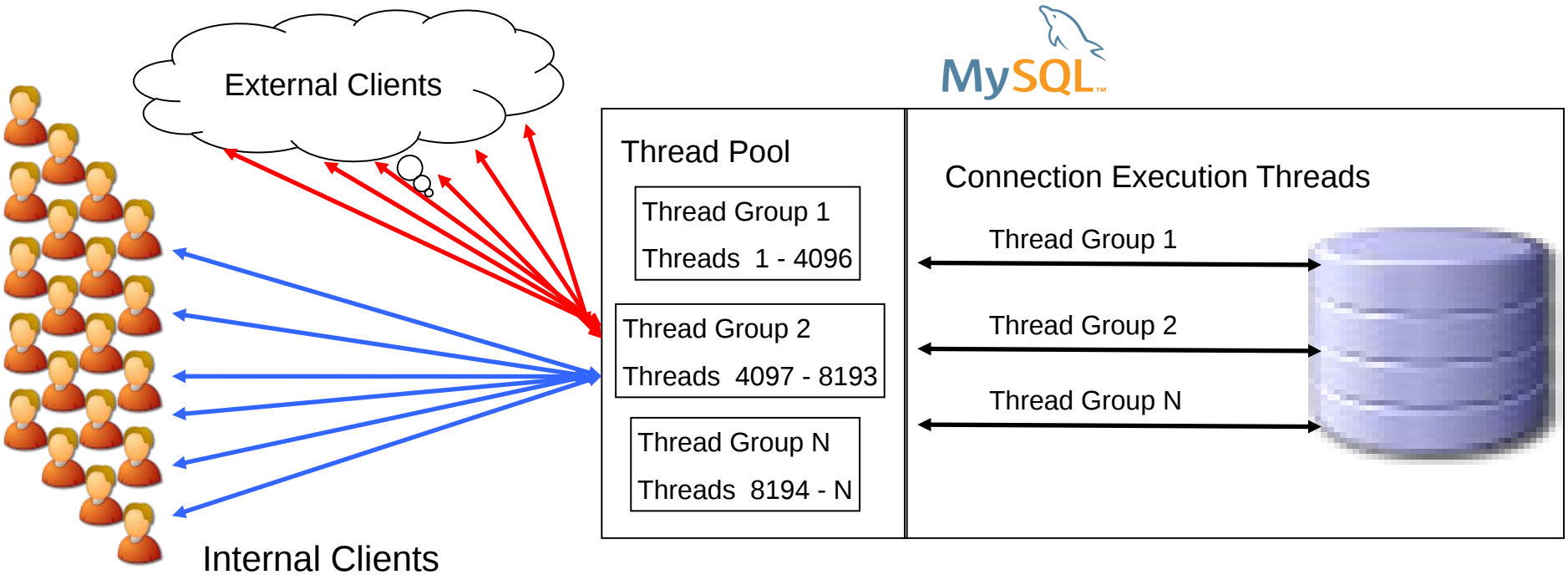


- Connections assigned to 1 thread for the life of the connection, same thread used for all statements
- No prioritisation of threads, statement executions
- Many concurrent connections = many concurrent execution threads to consume server memory, limit scalability

Connections

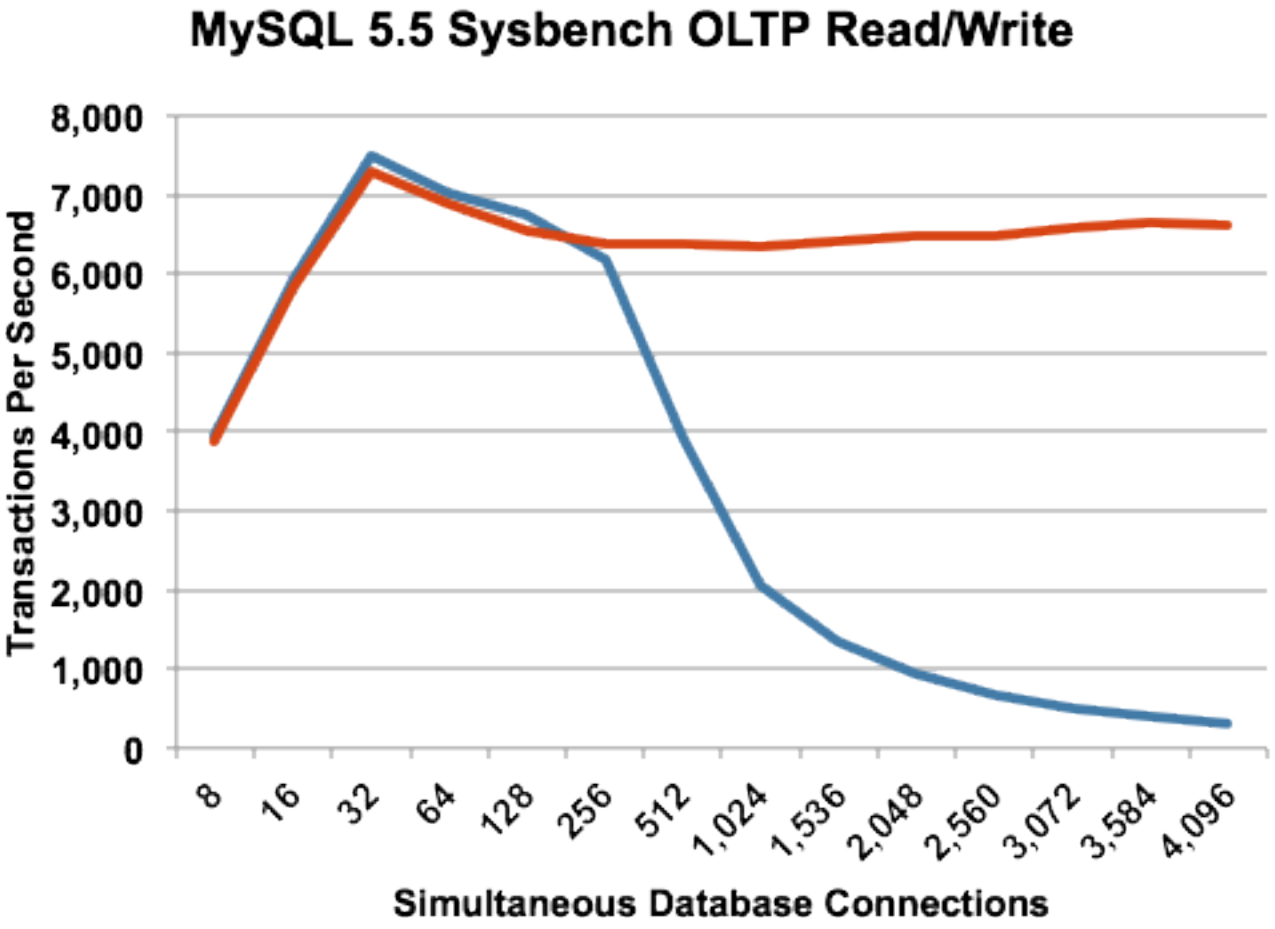
- MySQL caches the threads used by a connection
  - `thread_cache_size` - Number of threads to cache
  - Setting this to 100 or higher is not unusual
- Monitor `Threads_created` to see if this is an issue
  - Counts connections not using the thread cache
  - Should be less than 1-2 a minute
  - Usually 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

With Thread Pool Enabled



- Thread Pool contains configurable number of thread groups (default = 16), each manages up to 4096 re-usable threads
- Each connection assigned to thread group via round robin
- Threads are prioritised, statements queued to limit concurrent executions, load on server, improve scalability as connections grow

MySQL Enterprise Edition



MySQL Enterprise Edition

With Thread Pool

MySQL Community Server

Without Thread Pool

20x Better Scalability with Thread Pool

MySQL 5.5.16
Oracle Linux 6.1, Unbreakable Kernel 2.6.32
2 sockets, 24 cores, 2 X 12-core Intel(R) Xeon(R) X5670 2.93GHz CPUs
72GB DDR3 RAM
2 X LSI SCSI Disk (MR9261-8i) (597GB)

Indexing

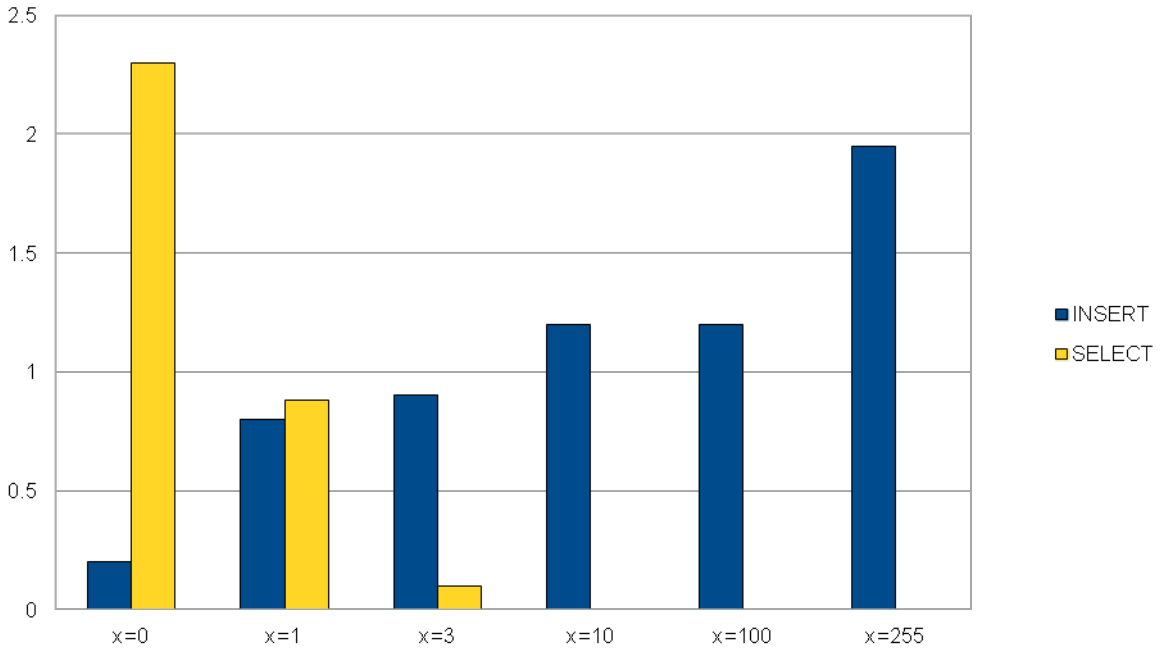
SOFTWARE.
HARDWARE.
COMPLETE.

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

Indexes in MySQL

- CREATE TABLE t (column char(255), KEY mykey (column***(x)***));
- INSERT 300k rows
- SELECT COUNT(*) FROM t WHERE column=”VALUE”;
- INSERT 5000 new rows



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 (> 16 text?)
- 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)
http://dev.mysql.com/doc/index-other.html
```

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, worst is ALL
  - 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

Do you really use your indexes?

```
create table test (
   i int primary key auto_increment,
   name varchar(12), key (name)
);
```

```
mysql> explain select * from test where lower(name)='ted'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: test
         type: index
possible_keys: NULL
          key: name
      key_len: 15
          ref: NULL
         rows: 5
        Extra: Using where; Using index
```

Do you really use your indexes?

```
create table test (
   i int primary key auto_increment,
   name varchar(12), key (name)
);
```

```
mysql> explain select * from test where name=lower('ted')\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: test
         type: ref
possible_keys: name
          key: name
      key_len: 15
          ref: const
         rows: 1
        Extra: Using where; Using index
```

Do you really use your indexes?

```
create table test (
   i int primary key auto_increment,
   name varchar(12), key (name)
);
```

```
mysql> explain select * from test where name like '%ed'\G
*************** 1. row ***************
   id: 1
  select_type: SIMPLE
        table: test
         type: index
possible_keys: NULL
          key: name
      key_len: 15
          ref: NULL
         rows: 5
        Extra: Using where; Using index
```

Do you really use your indexes?

```
create table test (
   i int primary key auto_increment,
   name varchar(12), key (name)
);
```

```
mysql> explain select * from test where name like 'te%'\G
*************** 1. row ***************
   id: 1
  select_type: SIMPLE
        table: test
         type: index
possible_keys: name
          key: name
      key_len: 15
          ref: NULL
         rows: 5
        Extra: Using where; Using index
```

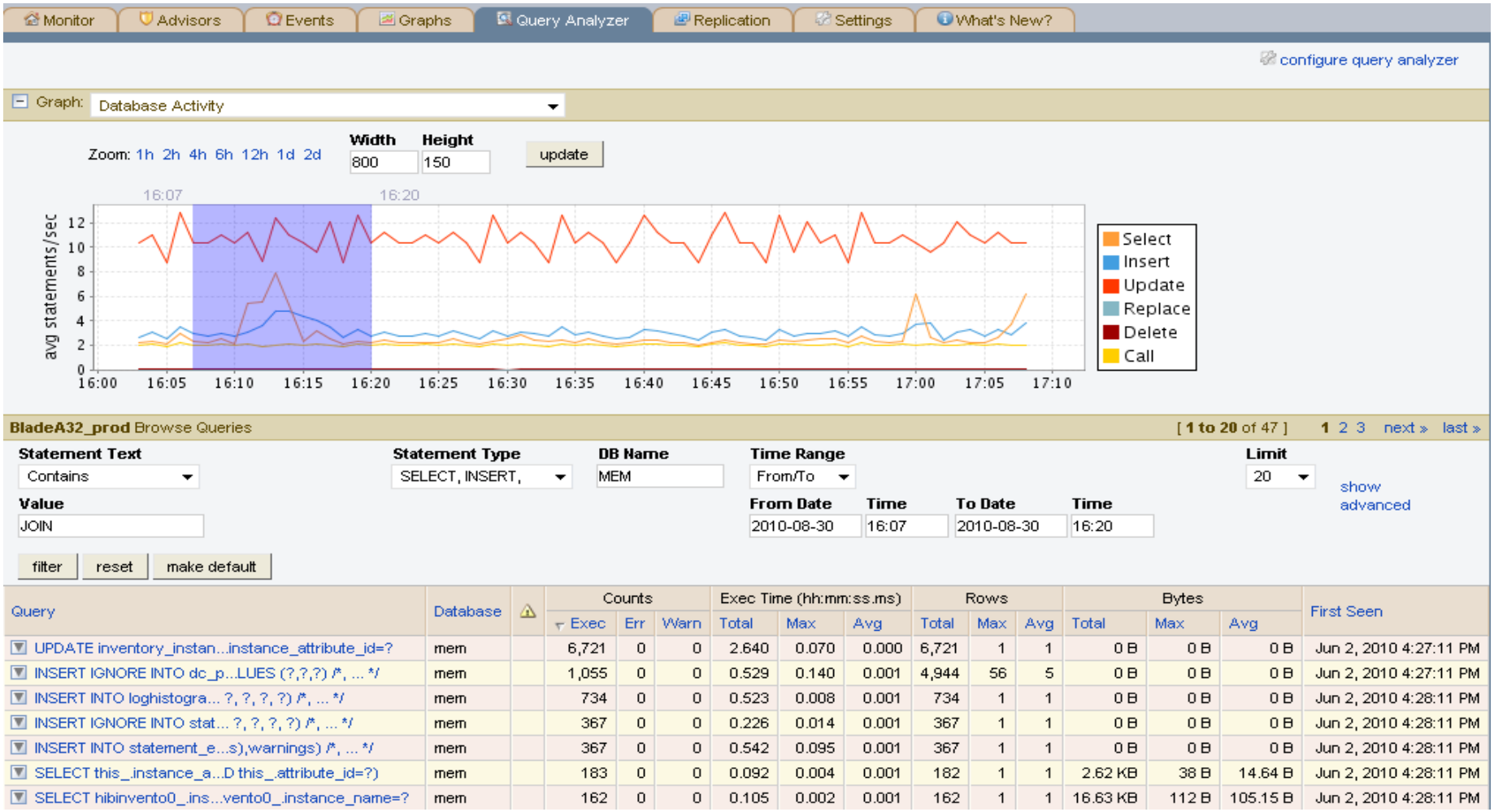
Query Tuning Rules

SOFTWARE.
HARDWARE.
COMPLETE.

Queries

- 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

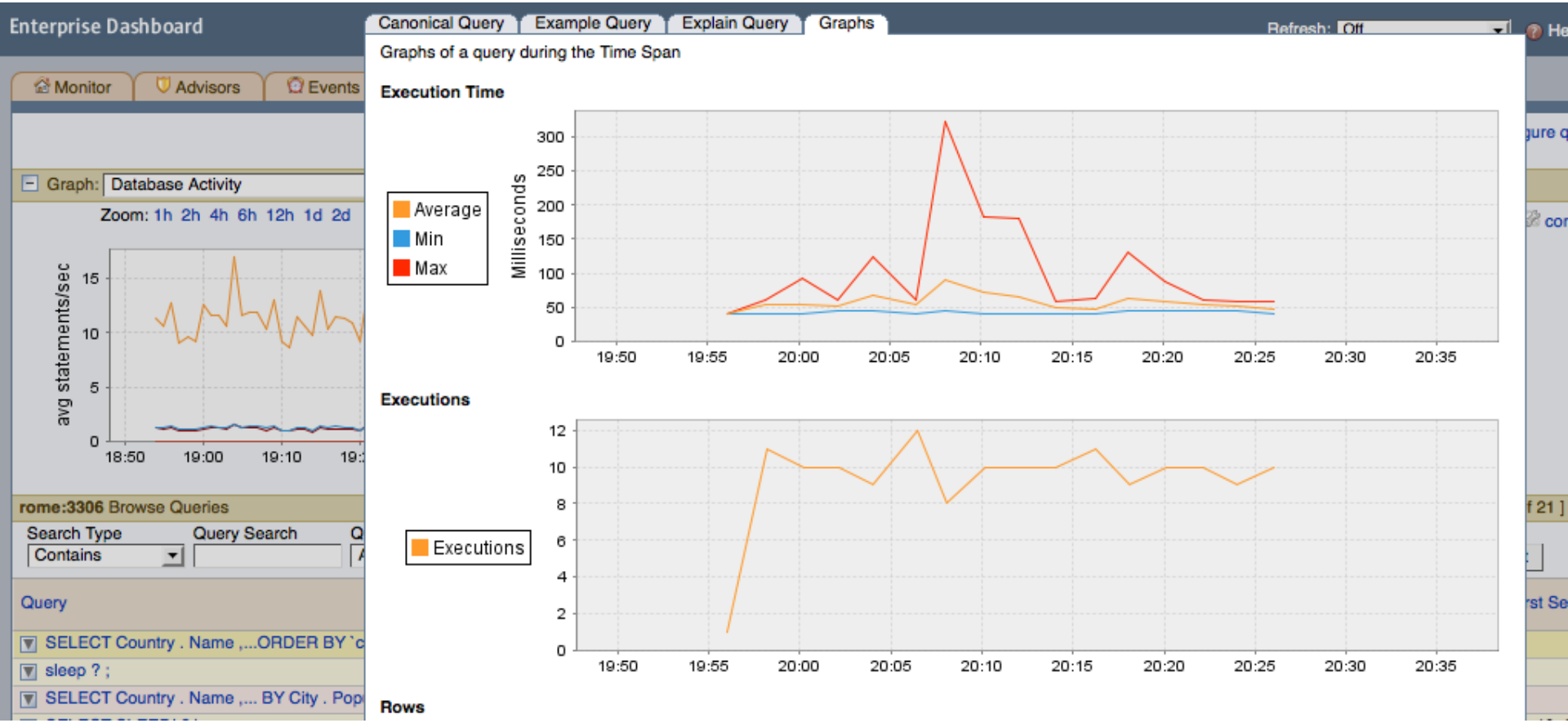
Query Analyzer



Query	Database		Counts			Exec Time (hh:mm:ss.ms)			Rows			Bytes			First Seen
			Exec	Err	Warn	Total	Max	Avg	Total	Max	Avg	Total	Max	Avg	
UPDATE inventory_instan...instance_attribute_id=?	mem		6,721	0	0	2.640	0.070	0.000	6,721	1	1	0 B	0 B	0 B	Jun 2, 2010 4:27:11 PM
INSERT IGNORE INTO dc_p...LUES (?,?,?) /*, ... */	mem		1,055	0	0	0.529	0.140	0.001	4,944	56	5	0 B	0 B	0 B	Jun 2, 2010 4:27:11 PM
INSERT INTO loghistogra... ?, ?, ?, ?) /*, ... */	mem		734	0	0	0.523	0.008	0.001	734	1	1	0 B	0 B	0 B	Jun 2, 2010 4:28:11 PM
INSERT IGNORE INTO stat... ?, ?, ?, ?) /*, ... */	mem		367	0	0	0.226	0.014	0.001	367	1	1	0 B	0 B	0 B	Jun 2, 2010 4:28:11 PM
INSERT INTO statement_e...s),warnings) /*, ... */	mem		367	0	0	0.542	0.095	0.001	367	1	1	0 B	0 B	0 B	Jun 2, 2010 4:28:11 PM
SELECT this_.instance_a...D this_.attribute_id=?)	mem		183	0	0	0.092	0.004	0.001	182	1	1	2.62 KB	38 B	14.64 B	Jun 2, 2010 4:28:11 PM
SELECT hibinvento0_.ins...vento0_.instance_name=?	mem		162	0	0	0.105	0.002	0.001	162	1	1	16.63 KB	112 B	105.15 B	Jun 2, 2010 4:28:11 PM

See all queries with execution statistics

Query Analyzer



View query performance over time

Query Analyzer

Sampled Query

truncated | full | formatted

```
SELECT
    hibinvento0_.instance_id AS instance1_5_,
    hibinvento0_.type_id AS type3_5_, hibinve
    hibinvento0_.insert_count AS insert5_5_,
    (
      SELECT
          concat(inv_ns.namespace, '.', t.type
        FROM
          inventory_namespaces
           AS inv_ns    JOIN
          inventory_types
           AS t    ON (inv_ns.namespace_id = t.
        WHERE
          t.type_id = hibinvento0_.type_id
    )
     AS clazz_
  FROM
```

Source Location

```
at sun.reflect.GeneratedMethodAccessor26.invoke(Unknown
at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown
at java.lang.reflect.Method.invoke(Unknown Source)
at com.mysql.jdbc.ReflectiveStatementInterceptorAdapter.postI
at com.mysql.jdbc.NoSubInterceptorWrapper.postPro
at com.mysql.jdbc.MysqlIO.invokeStatementIntercepto
at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.ja
```

Example query exec with variable substitution

Trace query exec back to source code

Full exec EXPLAIN

Canonical Query | Example Query | Explain Query | Graphs

Explain of a query that occurred during the Time Span (usually the slowest but not always).

Explain

id	select_type	table	type	possible_keys	key	key_len	ref	rows	extra
1	PRIMARY	hibinvento0_	const	instance_name,FKD4320F5BBDD9C29B	instance_name	771	const,const	1	
2	DEPENDENT SUBQUERY	t	const	PRIMARY,FKC2CE5FD6D77E2959	PRIMARY	4	const	1	
2	DEPENDENT SUBQUERY	inv_ns	const	PRIMARY	PRIMARY	4	const	1	

hide

expand »

Queries II

- Sub queries before MySQL 5.6 are slow!
- Don't wrap your "where" column in expressions
  - 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

http://dev.mysql.com/doc/refman/5.6/en/optimization.html

Schema

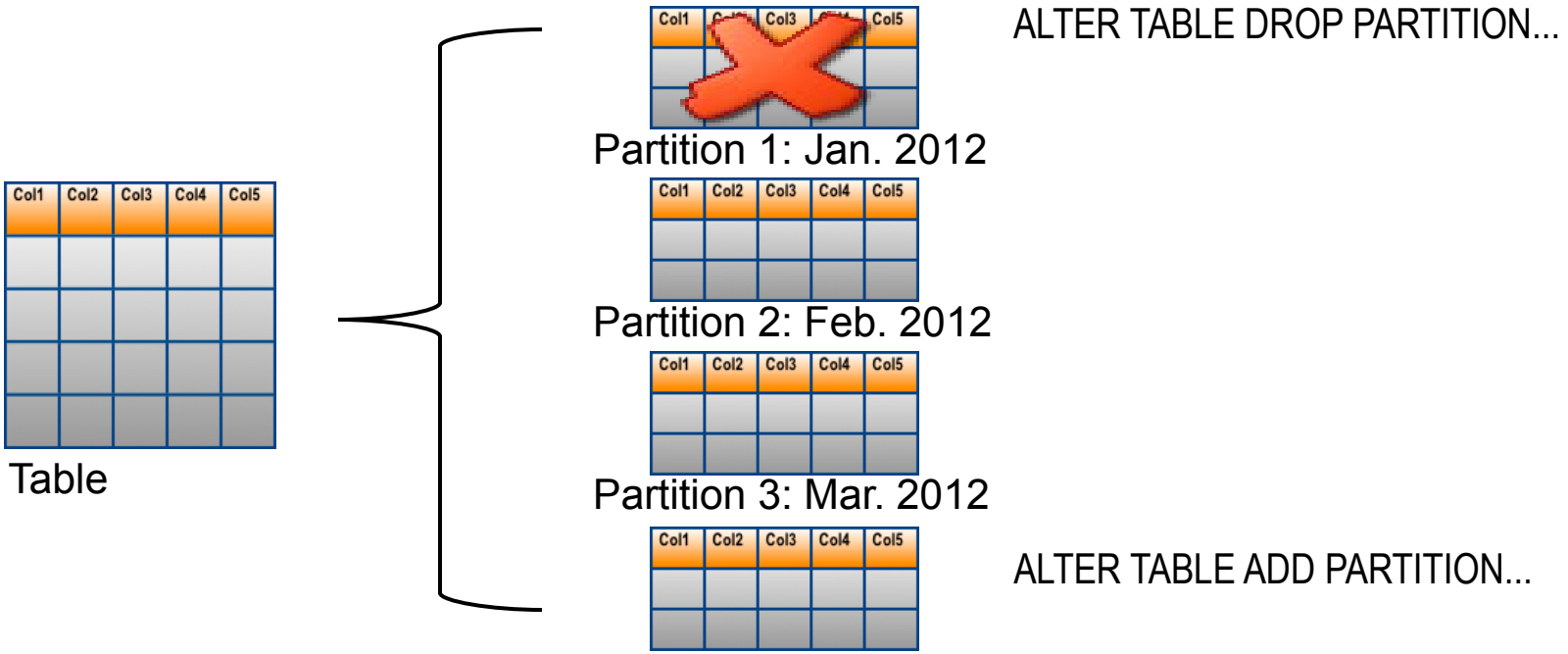
SOFTWARE.
HARDWARE.
COMPLETE.

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

Partitioning can help performance

- Easier to manage data
  - Drop/exchange partition
- Performance
  - Partition pruning
  - Smaller active indexes



Learn More: Resources

- MySQL Training Course – MySQL Performance Tuning
  - http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getCourseDesc?dc=D61820GC20
- MySQL Performance Forum
  - http://forums.mysql.com/list.php?24
- Download MySQL 5.6
  - 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/

Thank you!

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.


ORACLE®