
ORACLE®

MySQL

ORACLE

MySQL Query Tracing And Profiling

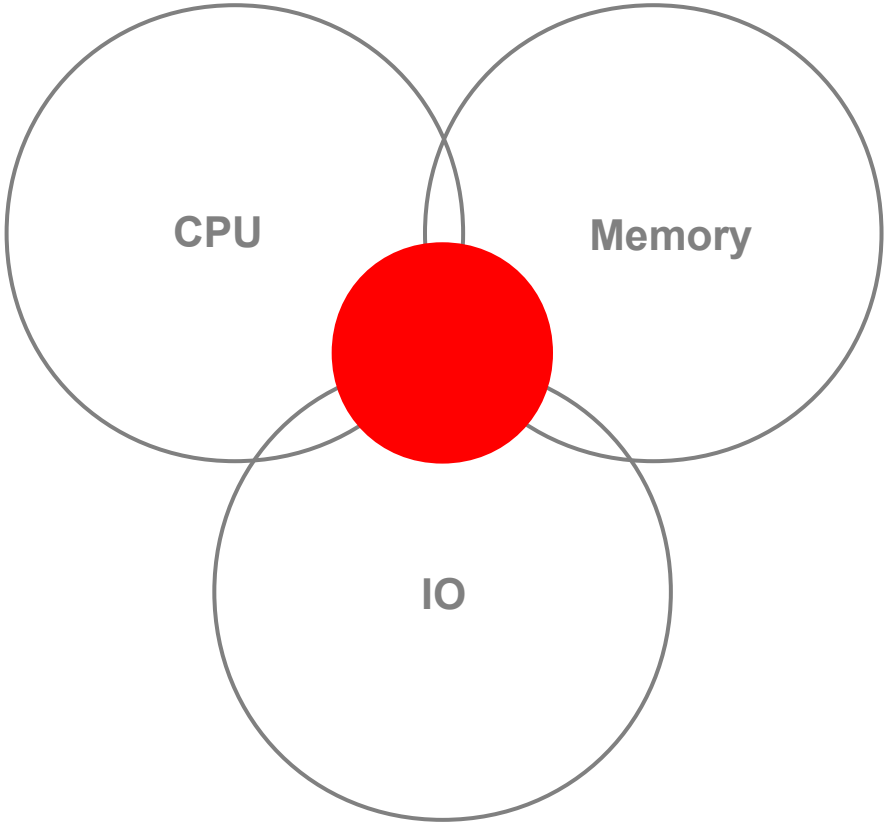
Johannes Schlüter
MySQL Engineering Connectors And Client Connectivity

- Performance
- Scalability

Performance Limiters

At least a few of them

- Frequency
- Scale
- ...



- Amount
- Speed
- CPU-Architecture (NUMA?)
- ...

- Disc
- Network
- ...

- If I …
  - Use a more CPU efficient algorithm
    - I might need more memory (and vice versa)
  - Add compression to save IO
    - I need more CPU
  - Offload work from the database
    - I need a bigger app/web server

→ Improving one piece might have bad impact on others

We're talking about the Web …

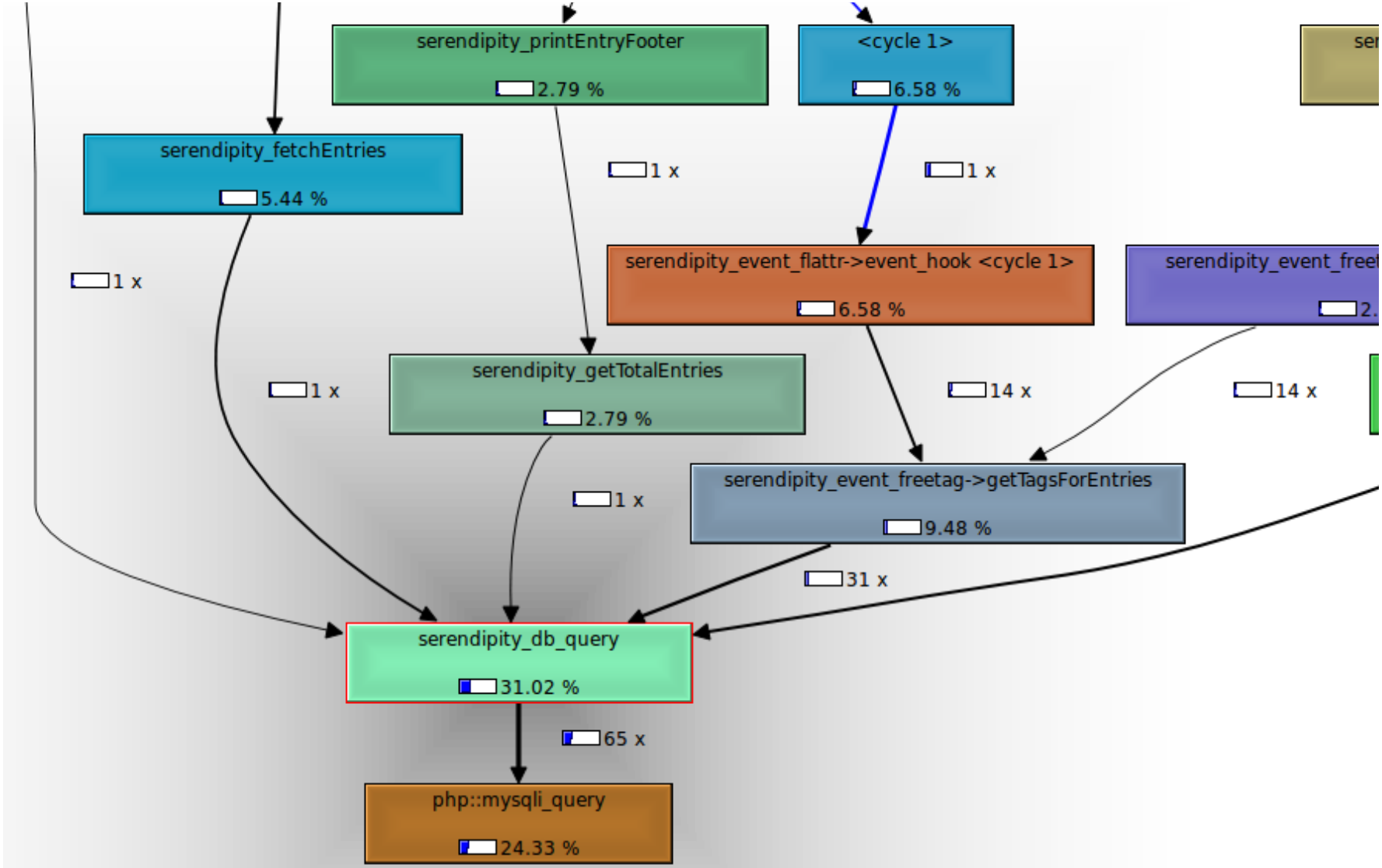
Measure What Matters!

- Goal has to be to server more users and/or serve faster
- Some tools:
  - Apache httpd's ab
  - Apache JMeter
  - Siege
- Measure the base line
- Decide what to focus on
  - Latency, throughput, ...

We're talking about the Web …

… don't forget the Frontend!

80 / 20


serendipity_printEntryFooter
2.79 %
<cycle 1>
6.58 %
serendipity_fetchEntries
5.44 %
1 x
1 x
1 x
serendipity_event_flattr->event_hook <cycle 1>
6.58 %
1 x
serendipity_getTotalEntries
2.79 %
14 x
14 x
1 x
serendipity_event_freetag->getTagsForEntries
9.48 %
31 x
serendipity_db_query
31.02 %
65 x
php::mysqli_query
24.33 %

PHP Profiling

- Xdebug
  - Free / OpenSource
  - $ pecl install xdebug
  - $ php -d xdebug.profiler_enale=1 script.php
  - KCacheGrind, WinCacheGrind, WebGrind
  - http://xdebug.org
- Zend Platform/Studio
  - commercial


MySQL

Finding the Slow/Expensive Queries

PHP-Based Database Wrapper

```
class MyPDO extends PDO {
    private $data = array();
    public function query($stmt) {
        $stats_before = mysqli_get_client_stats();
        $start = microtime(true);

        $retval = parent::query($stmt);

        $end = microtime(true);
        $stats_after = mysqli_get_client_stats();

        $this->data[] = array(
            'stmt' => $stmt,
            'time' => $end - $start,
            'bytes_sent' => $stats_after['bytes_sent'] - $stats_before['bytes_sent']
        );
        return $retval;
    }
}
```

mysqlnd Statistics

Client statistics	
bytes_sent	5381679
bytes_received	39375881
packets_sent	16117
packets_received	469633
protocol_overhead_in	1878532
protocol_overhead_out	64468
bytes_received_ok_packet	18436
bytes_received_eof_packet	67682
bytes_received_rset_header_packet	68915
bytes_received_rset_field_meta_packet	8416202
bytes_received_rset_row_packet	30690198
bytes_received_prepare_response_packet	0
bytes_received_change_user_packet	0
packets_sent_command	15279
packets_received_ok	1676
packets_received_eof	13522

- Around 150 statistic values collected
- mysqli_get_client_stats (), mysqli_get_connection_stats()

PHP-Based Database Wrapper

- ✔ Aware of the application
- ✔ Requires no changes to the Server

- ✗ Application changes needed
- ✗ Probably notable performance impact

PHP mysqlnd_uh

```
// php.ini
auto_prepend_file=autoprepend.php
```

```
// autoprepend.php
class ConnProxy extends MysqlndUHConnection {
    public function query($conn, $stmt) {
        // Same code as before ...
    }
}

mysqlnd_uh_set_connection_proxy(new ConnProxy());
```

PHP mysqlnd_uh

- ✔ Aware of the application
- ✔ Requires no changes to the server

- • Very little application change needed

- ✗ Probably notable performance impact
- ✗ No stable version

MySQL Proxy

- “MySQL Proxy is a simple program that sits between your client and MySQL server(s) that can monitor, analyze or transform their communication.“
- Can be programmed using LUA

```
function read_query( packet )
    if string.byte(packet) == proxy.COM_QUERY then
        local query = string.sub(packet, 2)
        print("we got a normal query: " .. query)
    end
end
```

MySQL Proxy

- ✔ Requires only changes to the configuration, not the code
- ✔ Can be placed on app server, database server or dedicated machine
- ✔ No actual code changes needed
- ✔ No server admin changes needed

- ✗ Requires knowledge of LUA and the MySQL protocol
- ✗ Not application-aware
- ✗ No GA version

Slow Query Log

- --slow_query_log=1

```
$ mysqldumpslow

Reading mysql slow query log from /usr/local/mysql/data/mysqld51-slow.log
Count: 1  Time=4.32s (4s)  Lock=0.00s (0s)  Rows=0.0 (0), root[root]@localhost
 insert into t2 select * from t1

Count: 3  Time=2.53s (7s)  Lock=0.00s (0s)  Rows=0.0 (0), root[root]@localhost
 insert into t2 select * from t1 limit N

Count: 3  Time=2.13s (6s)  Lock=0.00s (0s)  Rows=0.0 (0), root[root]@localhost
 insert into t1 select * from t1
```

DTrace

```
#!/usr/sbin/dtrace -qs

mysql*:mysqld::query-start {
  this->query = copyinstr(arg0);
  this->start = timestamp;
  this->in_filesort = 0;
}

mysql*:mysqld::filesort-start {
  this->in_filesort = 1;
}

mysql*:mysqld::query-done
/ this->in_filesort / {
  printf("%u: %s\n",
          timestamp – this->start,
          this->query);
}
```

DTrace

- ✔ Low impact on system
- ✔ No changes to the system
- ✔ Can be attached while running
- ✔ Access to all information on the system

- • Oracle Solaris Kernel feature (ported to BSD, MacOS)

- ✗ Deep system knowledge required
- ✗ System user with the needed rights required

Can the query be avoided?

Can it be cached?

Can it be simplified?

Can it be optimized?

Explain

```
mysql> EXPLAIN SELECT first_name, last_name, salary
            FROM employees e
            LEFT JOIN salaries s ON e.emp_no = s.emp_no
            WHERE salary > 67000
            AND e.last_name LIKE '%fg%'
            AND s.to_date = '9999-01-01';
+----+-------------+-------+------+----------------+
| id | select_type | table | type | possible_keys  |
+----+-------------+-------+------+----------------+
|  1 | SIMPLE      | e     | ALL  | PRIMARY        |
|  1 | SIMPLE      | s     | ref  | PRIMARY,emp_no |
+----+-------------+-------+------+----------------+
```

```
 +--------+---------+---------------------+--------+-------------+
 | key    | key_len | ref                 | rows   | Extra       |
 +--------+---------+---------------------+--------+-------------+
 | NULL   | NULL    | NULL                | 300363 | Using where |
 | emp_no | 4       | employees.e.emp_no |      4 | Using where |
 +--------+---------+---------------------+--------+-------------+
```

SHOW PROFILES;

```
mysql> SET profiling = 1;
mysql> SELECT first_name, last_name, salary
           FROM employees e
           LEFT JOIN salaries s ON e.emp_no = s.emp_no
           WHERE salary > 67000
           AND last_name LIKE '%fg%' AND to_date = '9999-01-01';
mysql> SHOW PROFILES;
+----------+------------------------------------------------+
| Query_ID | Duration   | Query                             |
+----------+------------+-----------------------------------+
|        1 | 0.33801275 | SELECT first_name, last_name, ...
                                ...  to_date = '9999-01-01'|
+----------+------------+-----------------------------------+
```

SHOW PROFILE;

```
mysql> SHOW PROFILE CPU FOR QUERY 1;
+--------------------+----------+----------+------------+
| Status             | Duration | CPU_user | CPU_system |
+--------------------+----------+----------+------------+
| starting           | 0.000101 | 0.000089 |   0.000011 |
| Opening tables     | 0.000015 | 0.000014 |   0.000002 |
| System lock        | 0.000006 | 0.000005 |   0.000001 |
| Table lock         | 0.000009 | 0.000008 |   0.000001 |
| init               | 0.000033 | 0.000032 |   0.000001 |
| optimizing         | 0.000016 | 0.000015 |   0.000001 |
| statistics         | 0.000024 | 0.000023 |   0.000001 |
| preparing          | 0.000017 | 0.000015 |   0.000001 |
| executing          | 0.000004 | 0.000003 |   0.000001 |
| Sending data       | 0.337715 | 0.382104 |   0.013387 |
| end                | 0.000014 | 0.000006 |   0.000006 |
| query end          | 0.000005 | 0.000004 |   0.000001 |
| freeing items      | 0.000046 | 0.000024 |   0.000022 |
| logging slow query | 0.000004 | 0.000002 |   0.000001 |
| cleaning up        | 0.000005 | 0.000005 |   0.000001 |
+--------------------+----------+----------+------------+
```

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';
```

Performance Schema

```
mysql> SHOW TABLES FROM performance_schema;
+----------------------------------------------+
| Tables_in_performance_schema                 |
+----------------------------------------------+
| COND_INSTANCES                               |
| EVENTS_WAITS_CURRENT                         |
| EVENTS_WAITS_HISTORY                         |
| EVENTS_WAITS_HISTORY_LONG                    |
| EVENTS_WAITS_SUMMARY_BY_INSTANCE             |
| EVENTS_WAITS_SUMMARY_BY_THREAD_BY_EVENT_NAME |
| EVENTS_WAITS_SUMMARY_GLOBAL_BY_EVENT_NAME    |
| FILE_INSTANCES                               |
| FILE_SUMMARY_BY_EVENT_NAME                   |
| FILE_SUMMARY_BY_INSTANCE                     |
| MUTEX_INSTANCES                              |
| PERFORMANCE_TIMERS                           |
| RWLOCK_INSTANCES                             |
| SETUP_CONSUMERS                              |
| SETUP_INSTRUMENTS                            |
| SETUP_TIMERS                                 |
| THREADS                                      |
+----------------------------------------------+
```

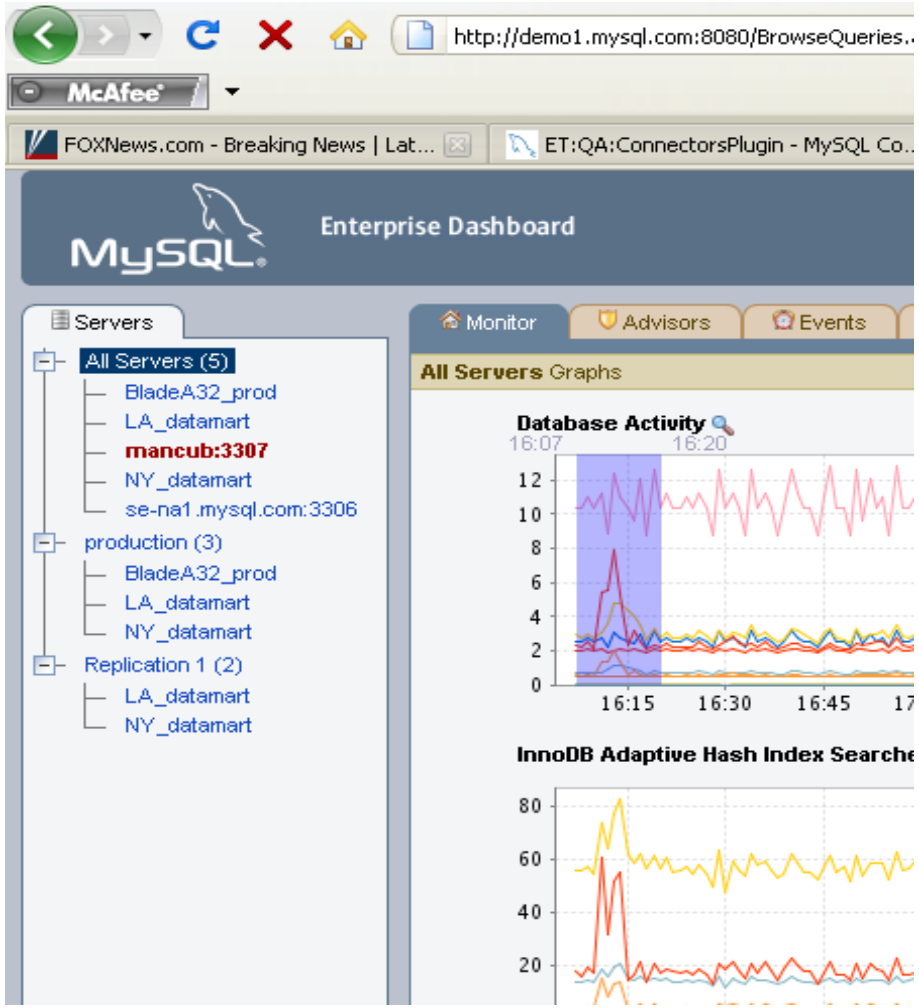
Performance Schema

```
mysql> SELECT EVENT_ID, EVENT_NAME, TIMER_WAIT AS WAIT
    -> FROM EVENTS_WAITS_HISTORY WHERE THREAD_ID = 13
    -> ORDER BY EVENT_ID;
+----------+-----------------------------------------+--------+
| EVENT_ID | EVENT_NAME                              | 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> SELECT * FROM EVENTS_WAITS_CURRENT\G
*************************** 1. row ***************************
            THREAD_ID: 0
             EVENT_ID: 5523
           EVENT_NAME: wait/synch/mutex/mysys/THR_LOCK::mutex
               SOURCE: thr_lock.c:525
          TIMER_START: 201660494489586
            TIMER_END: 201660494576112
           TIMER_WAIT: 86526
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 142270668
     NESTING_EVENT_ID: NULL
            OPERATION: lock
     NUMBER_OF_BYTES: NULL
                FLAGS: 0
```

MySQL Enterprise Monitor

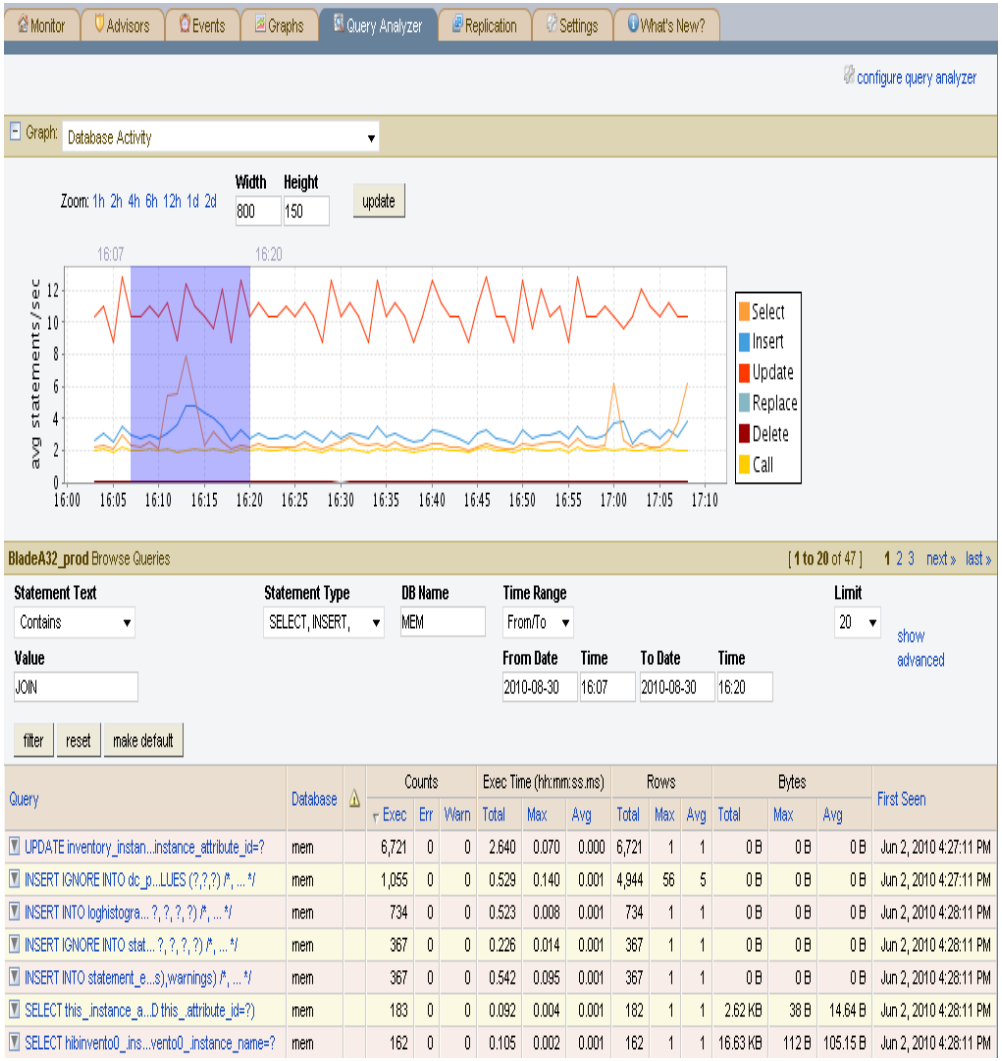
- Single, consolidated view into entire MySQL environment
- Automated, rules-based monitoring and alerts (SMTP, SNMP enabled)
- Query capture, monitoring, analysis and tuning, correlated with Monitor graphs
- Visual monitoring of “hot” applications and servers
- Real-time Replication Monitor with auto-discovery of master-slave topologies
- Integrated with MySQL Support



A Virtual MySQL DBA Assistant!

MySQL Query Analyzer

- Centralized monitoring of Queries across all servers
- No reliance on Slow Query Logs, SHOW PROCESSLIST;, VMSTAT, etc.
- Aggregated view of query execution counts, time, and rows
- Saves time parsing atomic executions for total query expense
- Visual “grab and go” correlation with Monitor graphs



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®

SOFTWARE. HARDWARE. COMPLETE.