Monitoring and logging a MySQL database server

Giuseppe Maxia
QA Developer - MySQL AB
Agenda

- Polling
- Monitoring
- Logging
- How to
- Advanced techniques
- Q&A
Polling

- Tracking down things over time
- Also called statistics
- Works on known events
Monitoring

- Tracking down things that your server is doing NOW
- Works on known, selected events
Logging

- Tracking down things now for future use
- Mostly unknown events
Polling example

- Measuring the number of queries per minute
- Counting how many queries per connection
Monitoring example

- How many connections are on the server now?
- Which query is taking longer than 5 seconds to execute?
- What's the CPU usage now?
Logging example

- Write down all server requests
  - no matter what they do
  - no matter if there are mistakes
- Write down the system conditions
  - disk, memory, CPU load
  - processes, threads
  - room temperature, DBA mood, everything
How to

- Go commercial
- Do it yourself
How to - $$$
How to - $$$
How to - DIY

- `vmstat`
- `mysqladmin extended-status`
- `mysqladmin variables`
- ...
- Perl + bash or PHP + Apache
- glue, tape, scissors, ingenuity
## How to - DIY Monitoring

```bash
vmstat 1
```

```
procs -----------memory---------- --swap-- -----io----- --system-- -----cpu-----
 r  b  swpd   free  buff  cache si  so  bi  bo in  cs us  sy  id  wa  st
1  0   160   240376 161576  2166232  0  0  0  272 1145 280  1  2 97  0  0
0  0   160   240376 161596  2166240  0  0  0  0 1012 243  1  7 92  0  0
0  0   160   240376 161620  2166216  0  0  0  0 120 1050 280  1  3 97  0  0
0  0   160   241740 161792  2166364  0  0  0  4 8784 2862 3207 3 12 78  7  0
0  0   160   241120 161816  2166292  0  0  0  716 1089 397  1  3 96  0  0
1  0   160   239384 161860  2166224  0  0  0  0 628 1138 635  1  4 92  3  0
0  0   160   239384 161876  2166216  0  0  0  0 1029 259  1  7 92  0  0
0  0   160   239384 161892  2166284  0  0  0  0 1011 225  1  2 97  0  0
0  0   160   240376 161916  2166260  0  0  0  4 432 1128 555  1  7 92  0  0
0  0   160   240376 161916  2166276  0  0  0  0 12 1008 202  0  0 100  0  0
0  0   160   240376 161916  2166296  0  0  0  0 160 1075 333  0  5 95  0  0
0  0   160   240392 161920  2166296  0  0  0  0 52 1027 310  0  0 100  0  0
0  0   160   252048 161920  2166296  0  0  0  0 64 1047 256  1  5 94  0  0
0  0   160   252420 161920  2166292  0  0  0  0 1005 192  0  0 100  0  0
0  0   160   252420 161920  2166292  0  0  0  0 1022 192  0  5 95  0  0
0  0   160   252436 161920  2166292  0  0  0  0 1004 220  0  0 100  0  0
0  0   160   252436 161920  2166292  0  0  0  0 1023 186  1  5 94  0  0
0  0   160   252436 161920  2166292  0  0  0  0 72 1021 212  0  0 100  0  0
0  0   160   242764 162008  2166456  0  0  0  4 2944 1556 1373 3 9 84  5  0
0  0   160   242028 162016  2166224  0  0  0  0 668 1060 255  1  1 98  0  0
0  0   160   240912 162032  2166372  0  0  0  0 1068 1218 685  1  6 91  2  0
```
How to - DIY

monitoring

```
mysqladmin -r -i 1 \ 
extended-status | grep -v '| 0'
```

<table>
<thead>
<tr>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes_received</td>
<td>35</td>
</tr>
<tr>
<td>Bytes_sent</td>
<td>6303</td>
</tr>
<tr>
<td>Com_show_status</td>
<td>1</td>
</tr>
<tr>
<td>Created_tmp_disk_tables</td>
<td>1</td>
</tr>
<tr>
<td>Created_tmp_tables</td>
<td>1</td>
</tr>
<tr>
<td>Handler_read_rnd_next</td>
<td>273</td>
</tr>
<tr>
<td>Handler_write</td>
<td>272</td>
</tr>
<tr>
<td>Questions</td>
<td>1</td>
</tr>
<tr>
<td>Select_scan</td>
<td>1</td>
</tr>
<tr>
<td>Uptime</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes_received</td>
<td>35</td>
</tr>
<tr>
<td>Bytes_sent</td>
<td>6303</td>
</tr>
<tr>
<td>Com_show_status</td>
<td>1</td>
</tr>
<tr>
<td>Created_tmp_disk_tables</td>
<td>1</td>
</tr>
<tr>
<td>Created_tmp_tables</td>
<td>1</td>
</tr>
<tr>
<td>Handler_read_rnd_next</td>
<td>273</td>
</tr>
<tr>
<td>Handler_write</td>
<td>272</td>
</tr>
<tr>
<td>Questions</td>
<td>1</td>
</tr>
</tbody>
</table>
Trouble

- No timestamps
- No synchronization between system calls and DBMS variables
What to do?

- Store everything in a (different) database
- Use database timestamp for logging
DB polling - how

- Tables
  - Status (every N minutes)
  - Variables (store only changes)
  - Processlist (every N minutes)
  - disk/memory load (every N minutes)
- separate server for logging
DB polling - how
Logging DB events

- General logs
  - PRO: logs every query
  - PRO: human readable
  - CON: needs a server restart
  - CON: can grow enormously
  - CON: does not store result info
  - CON: only sequential search
Logging DB events

- Binary logs
  - PRO: compact format
  - PRO: logs only successful queries
  - CON: only DML queries
  - CON: not human readable
  - CON: does not store result info
Logging DB events

- Slow logs
  - PRO: logs result info (time, retrieved rows)
  - PRO: human readable
  - CON: only slow queries
  - CON: only sequential search
Logging DB events

- Table based logs (5.1)
  - PRO: logs every query
  - PRO: human readable
  - PRO: can do indexed search
  - PRO: no server restart
  - CON: can grow enormously (*)
  - CON: does not store result info
  - CON: no filtering

(*) log rotation is the answer
Logging DB events

- Table based logs hacks

Diagram showing a server (server ID 10) sending data to a MySQL database labeled `general_log` and a MyISAM table. The data is sent to servers with server IDs 10 and 20, each with a `general_log` table and a federated table.
Logging DB events

- Table based logs hacks

![Diagram showing logging DB events with tables and triggers.](image)
Hacking log tables

(1)
DROP TABLE
  IF EXISTS gl0, gl1,
CREATE TABLE
  mysql.gl1
like mysql.general_log;
Hacking log tables

(2)

ALTER TABLE gl1
    ENGINE=MyISAM,
    KEY (user_host);
Hacking log tables

(3)

RENAME TABLE

general_log to gl0,
gl1 to general_log;
Hacking log tables

(2.1)
DROP TABLE
  IF EXISTS gl0, gl1,
CREATE TABLE
  gl1 like general_log;
Hacking log tables

(2.2)
ALTER TABLE gl1
  ENGINE=ARCHIVE;
RENAME TABLE
general_log to gl0,
gl1 to general_log;
Hacking log tables

(3.1)
# remote server
CREATE SCHEMA logs;
USE logs;
CREATE TABLE mylog
  LIKE mysql.general_log;
ALTER TABLE mylog
  ENGINE=MyISAM,
  KEY (user_host);
(3.2)
# local server
create server logserver
foreign data wrapper mysql
options (host 'remote_server.net',
database 'logs',
port 3306,
user 'remote_user_name',
password 'remote_secret');
Hacking log tables

(3.3)

```sql
# local server
use mysql;
DROP TABLE gl1, gl0;
CREATE TABLE gl1 (
# columns like general_log
) ENGINE = FEDERATED
CONNECTION = 'logserver/mylog';
```
Hacking log tables

(3.4)

# local server
RENAME TABLE
  general_log to gl0,
  gl1 to general_log;
Logging DB events

- Table based logs hacks
Logging DB events

- Logging through Proxy *(any version)*
  - **PRO:** can log every query
  - **PRO:** human readable
  - **PRO:** can store in any format
  - **PRO:** no server restart
  - **PRO:** can filter
  - **PRO:** can store result info
  - **CON:** can grow enormously (*)
  - **CON:** overhead

(*) so you need to rotate logs
Logging DB events

- Logging through Proxy (any version)

MySQL Proxy

MySQL Server

1. query

2. result

3. query injection
   ✔ query injection
   ✔ filtering
   ✔ rewriting
   ✔ macro expansion

4. query

Client
logging via Proxy

# client (1)
mysql> drop table t1;
Query OK, 0 rows affected (0.05 sec)

mysql> create table t1 (i int);
Query OK, 0 rows affected (0.02 sec)
logging via Proxy

# proxy (1)
2007-08-24 11:37:28 296 --
drop table t1 {0}

2007-08-24 11:37:35 296 --
create table t1 (i int) {0}
logging via Proxy

# client (2)
mysql> insert into t1;
ERROR 1064 (42000): You have an error in your SQL syntax;
logging via Proxy

# proxy (2)

2007-08-24 11:37:43 296 --
insert into t1 {0} [ERR]
logging via Proxy

# client (3)
mysql> insert into t1 values (1), (2);
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from t1;
+----+
| i  |
+----+
| 1  |
| 2  |
+----+
2 rows in set (0.00 sec)
logging via Proxy

# proxy (3)

2007-08-24 11:38:00 296 -- insert into t1 values (1),(2) {2}

2007-08-24 11:38:03 296 -- select * from t1 {2}
Logging DB events

Client → MySQL Server via 3306/tcp

Client → MySQL Proxy via 3306/tcp (redirect by iptables from 3306 to 4040)

MySQL Proxy → (logging?) via 4040/tcp
logging via Proxy

(1) do

```
sudo iptables -t nat \ 
-I PREROUTING \ 
-s ! 127.0.0.1 -p tcp \ 
--dport 3306 -j \ 
REDIRECT --to-ports 4040
```
logging via Proxy

(1) undo

```bash
sudo iptables -t nat \ 
-D PREROUTING \ 
-s ! 127.0.0.1 -p tcp \ 
--dport 3306 -j \ 
REDIRECT --to-ports 4040
```
Logging DB events

Client ➤ 3306/tcp ➤ MySQL Server

Client ➤ 3306/tcp ➤ MySQL Proxy (logging?) ➤ 4040/tcp ➤ MySQL Server

redirect by iptables from 3306 to 4040
Live examples

Some live examples now

MONITORING via Proxy
Q&A

Any questions?

slides at http://datacharmer.org