MySQL Proxy
From architecture to implementation

Giuseppe Maxia
MySQL Community Team Lead
Sun Microsystems

Ronald Bradford
Independent consultant

This work is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported License.
Raise your hand
Raise your hand

- Who has heard of MySQL Proxy?
Raise your hand

- Who has heard of MySQL Proxy?
- Who has tried out MySQL Proxy?
Raise your hand

- Who has heard of MySQL Proxy?
- Who has tried out MySQL Proxy?
- Who runs MySQL Proxy in production?
Raise your hand

- Who has heard of MySQL Proxy?
- Who has tried out MySQL Proxy?
- Who runs MySQL Proxy in production?
- Who wrote scripts for MySQL Proxy?
Raise your hand

- Who has heard of MySQL Proxy?
- Who has tried out MySQL Proxy?
- Who runs MySQL Proxy in production?
- Who wrote scripts for MySQL Proxy?
- Who runs MySQL Proxy from the source tree?
Raise your hand

- Who has heard of MySQL Proxy?
- Who has tried out MySQL Proxy?
- Who runs MySQL Proxy in production?
- Who wrote scripts for MySQL Proxy?
- Who runs MySQL Proxy from the source tree?
- Who runs another MySQL Proxy like application?
Agenda

- Proxy concepts
- MySQL Proxy architecture
- Proxy with a single backend
- Uses in the field
- Proxy with multiple backends
Solving database problems

- broken?
- missing feature?
- not flexible?
Solving database problems - traditional way
Solving database problems - traditional way

1. file a bug report
Solving database problems - traditional way

1. file a bug report
2. wait
Solving database problems - traditional way

1. file a bug report
2. wait
Solving database problems
open source way

MySQL Server

source code

modify

MySQL Server

compile

new source code
Solving database problems
open source way

MySQL Server

source code
modify

new source code
compile

not suitable for everyone
Solving database problems
Creative (shortsighted) way

bring the logic at application level
Solving database problems
Creative (shortsighted) way

bring the logic at application level

NOT flexible - Unaccessible from other applications
database handling
database handling
too many languages
Solving database problems
Creative (enlightened) way

set the logic at server level (stored routines)
Solving database problems
Creative (enlightened) way

language limitations - privilege issues - performance
database handling

- C/C++
- Java
- Perl
- PHP
- Python
- .NET

MySQL Server

database

stored routines library

Client
Solving database problems
Creative (more enlightened) way

set the logic at protocol level (proxy)
Solving database problems
Creative (more enlightened) way

set the logic at protocol level (proxy)

flexible and suitable for many
database handling

- C/C++
- Java
- Perl
- PHP
- Python
- .NET
- Proxy library
- MySQL Server
- Database
what can you do with MySQL Proxy
what can you do with MySQL Proxy

• create new commands
what can you do with MySQL Proxy

• create new commands
• filter queries (deny specific queries)
what can you do with MySQL Proxy

- create new commands
- filter queries (deny specific queries)
- collect statistics on usage
what can you do with MySQL Proxy

• create new commands
• filter queries (deny specific queries)
• collect statistics on usage
• implement usage quotas
what can you do with MySQL Proxy

• create new commands
• filter queries (deny specific queries)
• collect statistics on usage
• implement usage quotas
• execute shell commands
what can you do with MySQL Proxy

• create new commands
• filter queries (deny specific queries)
• collect statistics on usage
• implement usage quotas
• execute shell commands
• create customized logs
what can you do with MySQL Proxy

- create new commands
- filter queries (deny specific queries)
- collect statistics on usage
- implement usage quotas
- execute shell commands
- create customized logs
- implement server-side pivot tables
what can you do with MySQL Proxy

- create new commands
- filter queries (deny specific queries)
- collect statistics on usage
- implement usage quotas
- execute shell commands
- create customized logs
- implement server-side pivot tables
- start/stop a MySQL server remotely
what can you do with MySQL Proxy

- create new commands
- filter queries (deny specific queries)
- collect statistics on usage
- implement usage quotas
- execute shell commands
- create customized logs
- implement server-side pivot tables
- start/stop a MySQL server remotely
- play movies (seriously!)
what can you do with MySQL Proxy

- create new commands
- filter queries (deny specific queries)
- collect statistics on usage
- implement usage quotas
- execute shell commands
- create customized logs
- implement server-side pivot tables
- start/stop a MySQL server remotely
- play movies (seriously!)
- make coffee (really?)
what can you do with MySQL Proxy

- create new commands
- filter queries (deny specific queries)
- collect statistics on usage
- implement usage quotas
- execute shell commands
- create customized logs
- implement server-side pivot tables
- start/stop a MySQL server remotely
- play movies (seriously!)
- make coffee (really?)
- sharding
what can you do with MySQL Proxy

- create new commands
- filter queries (deny specific queries)
- collect statistics on usage
- implement usage quotas
- execute shell commands
- create customized logs
- implement server-side pivot tables
- start/stop a MySQL server remotely
- play movies (seriously!)
- make coffee (really?)
- sharding
- load balancing servers
Basic principles

MySQL Proxy

MySQL Server

query injection
✓ filtering
✓ rewriting
✓ macro expansion

Client
Basic principles
Basic principles

PROXY

CORE

- connection hook
- read query hook
- read result hook
Basic principles

**PROXY**

**CORE**

- connection hook
- read query hook
- read result hook

**Lua script**

- function
- function
- function
Basic principles

PROXY CORE

- connection hook
- read query hook
- read result hook

Lua script

function

function

function
Lua

Why not ... Perl? PHP? Javascript? [whatever]?
Lua
Lua

• SMALL ( < 200 KB)
Lua

- SMALL ( < 200 KB)
- DESIGNED for EMBEDDED systems
Lua

- SMALL (< 200 KB)
- DESIGNED for EMBEDDED systems
- Widely used (lighttpd)
Lua

- SMALL ( < 200 KB)
- DESIGNED for EMBEDDED systems
- Widely used (lighttpd)

lighttpd, like MySQL Proxy, was created by Jan Kneschke
Lua

Very popular among game writers
Lua

Very popular among game writers
Lua

Very popular among game writers
Proxy overview

- Global context
- Session context

Lua script:
- connect_server
- read_handshake
- read_auth
- read_auth_result
- read_query
- read_query_result
- disconnect_client
Proxy overview

/usr/local/sbin/mysql-proxy \ 
--proxy-lua-script=/path/name.lua

IMPORTANT!
THE SCRIPT DOES NOT START UNTIL THE FIRST CLIENT CONNECTION
read_query and read_query_result

If a query is passed directly to the server, its result is **NOT** evaluated by read_query_result.
read_query and read_query_result

Only if a query is added to the query queue, its result is evaluated by read_query_result.
debugging

server

proxy

client

diagnostics
text
debugging scripts

server

proxy

diagnostics text

diagnostics text

proxy

client
chained proxy: double features

server

proxy

pivot tables

loops

proxy

client
testing

server

proxy

fake packets

client

e.g. connectors
An example: user quotas

- You want to limit user access
- When the user reaches a given amount of data, further queries should be rejected
- the quota must work across sessions
  (code online)
User quotas

-- some global variables

-- proxy.global.bandwidth
-- will sum up the used bytes

proxy.global.bandwidth
    = proxy.global.bandwidth or {};

-- session_user will identify the user
-- throughout the session
local session_user
User quotas

-- session_user needs to be initialized
-- when the user information is passed
-- (during the authentication)

function read_auth( auth )
    session_user = auth.username
    proxy.global.bandwidth[session_user] = proxy.global.bandwidth[session_user] or 0
end
BREAK (a handy Lua idiom)

-- simple assignment

\[
a = a \text{ or } 0
\]

-- corresponds to:

\[
\begin{align*}
\text{if } a & = \text{ nil} \\
\text{then} & \\
\quad a & = 0 \\
\text{else} & \\
\quad a & = a \\
\text{end}
\end{align*}
\]

\[
\begin{align*}
\text{if } a & \neq \text{ nil} \\
\text{then} & \\
\quad a & = a \\
\text{else} & \\
\quad a & = 0 \\
\text{end}
\end{align*}
\]
BREAK (a handy Proxy Lua function)

-- returns an error to the client

function error_result (msg)
    proxy.response = {
        type = proxy.MYSQLD_PACKET_ERR,
        errmsg = msg,
        errcode = 7777,
        sqlstate = 'X7777',
    }
    return proxy.PROXY_SEND_RESULT
end
function read_query (packet )
    if proxy.global.bandwidth[session_user] > 10000
        and session_user ~= 'root'
    then
        return error_result( 'you have exceeded your query quota')
    end
-- ...
User quotas

-- read_query (2)
-- adding to the totalizer

-- ...

proxy.global.bandwidth[session_user] =
proxy.global.bandwidth[session_user] + packet:len()
proxy.queries:append(1, packet)
return proxy.PROXY_SEND_QUERY
end
User quotas

-- read_query_result (1)
-- adding row headers to the totalizer

function read_query_result(inj)
    local fields = inj.resultset.fields
    local rows = inj.resultset.rows
    if fields then
        for i = 1, #fields do
            proxy.global.bandwidth[session_user] =
            proxy.global.bandwidth[session_user] +
            (fields[i] and fields[i].name:len() or 0)
        end
    end
User quotas

-- read_query_result (2)
-- adding rows contents to the totalizer

if rows then
  for row in rows do
    for i = 1, #fields do
      proxy.global.bandwidth[session_user] =
      proxy.global.bandwidth[session_user] +
      (row[i] and row[i]:len() or 0)
  end
end
end
User quotas

-- read_query_result (3)
-- displaying the current bandwidth

    print (session_user .. ' -> ' ..
           proxy.global.bandwidth[session_user])

end
User quotas (VERY advanced)

-- you can create another module
-- to be loaded at run time
-- (this is really advanced)
-- and in such module you define
-- a SHOW QUOTAS command
User quotas (VERY advanced)

-- load_multi
User quotas (VERY advanced)

```
mysql> pload show_quotas.lua;
+--------------------------------+
| info                           |
+--------------------------------+
| module "show_quota.lua" loaded |
+--------------------------------+

mysql> show quotas;
+----------+-------+
| name     | quota |
+----------+-------+
| simple   | 3578  |
| root     | 2111  |
| msandbox | 102   |
+----------+-------+
```
Uses in the field

- Analyzing Queries
- Better Instrumentation
- Connection Management
Analyzing Queries

Currently

- EXPLAIN Plan
- Session Status
  - Get Before
  - Get After
  - Do diff
  - Remove act of measurement
Analyzing Queries

With Proxy

$mysql-proxy --with-lua-script=analyze_query.lua
Analyzing Queries

Norm_Query: "SELECT `device` FROM `schema` . `l_device` WHERE `id` = ? "
Exec_time: 11.044 ms
.. Bytes_received = 47
.. Com_select = 1
.. Handler_read_key = 1
.. Handler_read_next = 1
.. Key_read_requests = 3
.. Last_query_cost = 1.199
.. Questions = 1
.. Table_locks_immediate = 1
Analyzing Queries

Norm_Query: "SELECT * FROM ( SELECT `i` . `iv` , `dl` . * FROM `schema` . `vri` `i` , `schema` . `vd1` `dl` WHERE `i` . `c_key` = `dl` . `c_key` AND `i` . `m_type` IN ( ? ) ORDER BY `i` . `index_value` DESC LIMIT ? ) `foo` ORDER BY RAND( ) LIMIT ? "

Exec_time: 19548 us
.. Bytes_received = 328
.. Bytes_sent = 96297
.. Com_select = 1
.. **Created_tmp_tables = 2**
.. Handler_read_key = 528
.. Handler_read_prev = 524
.. Handler_read_rnd = 262
.. Handler_read_rnd_next = 1047
.. Handler_write = 1045
.. Innodb_buffer_pool_read_requests = 817
.. Innodb_rows_read = 1050
.. Last_query_cost = 141.249
.. Qcache_not_cached = 2
.. Questions = 1
.. **Select_scan = 2**
.. Slow_queries = 1
.. Sort_rows = 262
.. **Sort_scan = 1**
.. Table_locks_immediate = 2
Analyzing Queries

Further Uses

- Determine key status’s you wish to monitor and write SQL, EXPLAIN Plan, and Status output based on these rules
- e.g.
  - Select_scan
  - Sort_merge_passes
  - Com_alter_table (Watching Admin Commands)
Better Instrumentation

Currently

- `SHOW GLOBAL STATUS`
- `COM_???” e.g. SELECT, INSERT, UPDATE etc`
- `$ mysqladmin -r -i 1 -n 60 extended-status | grep -v " | 0 "`
Better Instrumentation

With Proxy

$mysql-proxy --with-lua-script=histogram.lua

- Gives Read/Write breakdown per table
- Shows top executing queries by frequency
- Shows max and avg execution time
### Better Instrumentation

```sql
mysql> select * from histogram.tables;
```

<table>
<thead>
<tr>
<th>table</th>
<th>reads</th>
<th>writes</th>
</tr>
</thead>
<tbody>
<tr>
<td>schema1.dotCommLookup</td>
<td>991</td>
<td>0</td>
</tr>
<tr>
<td>schema2xxxxxx.dotCommClient</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>schema2xxxxxx.ReleaseNumbers</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>schema2xxxxxx.PAYCYCLE</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>schema1.dotCommClient</td>
<td>1505</td>
<td>0</td>
</tr>
<tr>
<td>schema1.Job</td>
<td>131</td>
<td>0</td>
</tr>
<tr>
<td>schema2xxxxxx.CondMod</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>schema2xxxxxx.SPINFO</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>schema1.Poll</td>
<td>1184</td>
<td>633</td>
</tr>
<tr>
<td>schema2xxxxxx.LaborSchedules</td>
<td>0</td>
<td>96</td>
</tr>
<tr>
<td>schema1.JobRequest</td>
<td>192</td>
<td>0</td>
</tr>
<tr>
<td>schema1.PollTasks</td>
<td>454</td>
<td>1048</td>
</tr>
<tr>
<td>schema1.Tasks</td>
<td>1048</td>
<td>0</td>
</tr>
<tr>
<td>schema1.auth</td>
<td>141</td>
<td>59</td>
</tr>
<tr>
<td>schema2xxxxxx.dotComm</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>schema2xxxxxx.MenuItem</td>
<td>0</td>
<td>2446</td>
</tr>
<tr>
<td>schema1.Location</td>
<td>19</td>
<td>118</td>
</tr>
<tr>
<td>schema2xxxxxx.CONCASH</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>schema1.dotComm</td>
<td>156</td>
<td>0</td>
</tr>
<tr>
<td>schema1.Client</td>
<td>47</td>
<td>0</td>
</tr>
</tbody>
</table>

20 rows in set (0.00 sec)
Better Instrumentation

mysql> `select * from histogram.queries;`

8. row

query: `SELECT `type` , COUNT( * ) AS `count` FROM `schema1` . `table1` `a` WHERE `clientID` = ? AND `bocDate` = ? AND `locationID` IN( ? ) GROUP BY ? ORDER BY ?

count: 8
max_query_time: 12932
avg_query_time: 4154.125

35. row

query: `SELECT `empl_first_nam` , `empl_last_nam` , `empl_mid_initial` FROM `schema2` . `table2` WHERE `clientID` = ? AND `empl_ssn` = ? AND `locationID` = ? LIMIT ?

count: 84
max_query_time: 61974
avg_query_time: 2501.3095238095

305. row

query: `INSERT INTO `schema3` . `table3` SET `eventTime` = NOW( ) , `event` = ? , `userID` = ? , `details` = ?

count: 59
max_query_time: 433610
avg_query_time: 38760.983050847

302. row

query: `SELECT * FROM `table4` WHERE `jobID` = ? AND `taskID` = ?

count: 1056
max_query_time: 17564
avg_query_time: 672.67708333333
Proxy and multiple backends ...

For many of you, this is probably the main reason for attending this talk
Proxy and multiple backends ...

You may be disappointed.

There is no silver bullet (yet). Let us explain
Proxy options for multiple backends

--proxy-address
    =<host:port>

--proxy-read-only-backend-addresses
    =<host:port>

--proxy-backend-addresses
    =<host:port>
Proxy functions for multiple backends

`connect_server()`  
(backend host)

`read_auth()`  
(username and password)

`read_auth_result()`  
(accepted/rejected)
Proxy sample scripts for multiple backends

rw-splitting.lua
(masked master/slave behind a proxy)

auditing.lua
(captures the info passing)

tutorial-packets.lua
(shows really all the info passing between client and server)
Why rw-splitting.lua is a challenge

**PRO**
- split queries automatically
- writes go to the master
- reads go to the slaves
- transactions go to the master
Why rw-splitting.lua is a challenge

CON
- Just a proof-of-concept
- not stable
- user variables aren't handled nicely
- functions with updates aren't handled

function xxx () returns int
BEGIN
    INSERT INTO x VALUES (1);
    return 0;
END
Community
<table>
<thead>
<tr>
<th>Name/Title</th>
<th>Type</th>
<th>Added On</th>
<th>Author</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL-Proxy connection pooler</td>
<td>code snippet</td>
<td>Jul 9, 2008 @ 00:34</td>
<td>John Loehrer</td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy tutorial - Crosstabs</td>
<td>code snippet</td>
<td>Jul 3, 2008 @ 12:49</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>quick wrapper for MySQL Proxy</td>
<td>code snippet</td>
<td>May 27, 2008 @ 18:56</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>Run two chained Proxies in protected environments</td>
<td>code snippet</td>
<td>May 27, 2008 @ 17:20</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy - Improved CREATE TABLE ... SELECT</td>
<td>code snippet</td>
<td>May 1, 2008 @ 03:10</td>
<td>Diego Medina</td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy - Log warnings and errors</td>
<td>code snippet</td>
<td>Apr 12, 2008 @ 03:45</td>
<td>Diego Medina</td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy Tutorial - Return a full dataset</td>
<td>code snippet</td>
<td>Mar 29, 2008 @ 21:47</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy Tutorial - Return a simple dataset</td>
<td>code snippet</td>
<td>Mar 29, 2008 @ 21:43</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy Tutorial - Return an error</td>
<td>code snippet</td>
<td>Mar 29, 2008 @ 21:40</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy tutorial - tracking transaction status</td>
<td>code snippet</td>
<td>Dec 2, 2007 @ 11:11</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy tutorial - Measuring bandwidth by user</td>
<td>code snippet</td>
<td>Dec 2, 2007 @ 11:01</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy tutorial - Measuring bandwidth by session</td>
<td>code snippet</td>
<td>Dec 2, 2007 @ 10:55</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy tutorial - Show all hooks</td>
<td>code snippet</td>
<td>Dec 2, 2007 @ 10:51</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>MySQL Proxy tutorial - Blocking unwanted queries</td>
<td>code snippet</td>
<td>Nov 19, 2007 @ 14:10</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>mysql-proxy tutorial - loops</td>
<td>code snippet</td>
<td>Aug 12, 2007 @ 11:42</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>mysql-proxy - Query statistics</td>
<td>code snippet</td>
<td>Jul 2, 2007 @ 20:42</td>
<td>Stephane Varoqui</td>
<td>Not rated</td>
</tr>
<tr>
<td>Using mysql-proxy without changing port on Linux</td>
<td>code snippet</td>
<td>Jun 30, 2007 @ 12:20</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
<tr>
<td>mysql-proxy tutorial - query and basic result logging</td>
<td>code snippet</td>
<td>Jun 29, 2007 @ 09:21</td>
<td>Giuseppe Maxia</td>
<td></td>
</tr>
</tbody>
</table>
Community Contribution - Gaia Online

- online hangout for teens
- 5 million unique visitor
- 1.3 billion forum entries
Community Contribution - Gaia Online

Existing Architecture
• Data is sharded
• Large number of different db server connections per page
• Analysis shows 75% of time in authorization of connection
Community Contribution - Gaia Online

Forum Page → Session DB → User DB → Forum DB

- start session
- get userdata
- get topic
- get related users
- save session
Community Contribution - Gaia Online

- Connect: 75%
- Read challenge: 19%
- Write auth: 19%
- Read auth OK: 6%
- Write query: 6%
- Read result: 6%
- Close: 6%
Use of Proxy

- Connection multiplexer
- Recycling back-end connections
- App connections are transparent
- Per connection authentication is removed from normal connections
Community Contribution - Gaia Online

Client -> Proxy -> Back-End Pool

- Query
- Allocate
- Process query
- Free back-end

MySQL Server
Community Contribution - Gaia Online

Use of Proxy

• turning 7,500 connections
• into 100 connections (in pools)
• large reduction in per query time
The future of MySQL Proxy
MySQL Load Balancer

MySQL software which can route database queries to the appropriate database in a scale-out environment.

- Intelligently route reads across slaves
- Use database least behind, least loaded
- Remove latent slaves from read rotation
- Other distribution algorithms

- Improves/scales throughput of reads for online applications
- Helps customers reduce cost of adding slaves
MySQL Query Analyzer

MySQL Enterprise Monitor feature that allows users to trace, monitor, and analyze MySQL query activity for specific servers, users, and applications.

- Adaptive “Evil” query collection/tracing
- Historical browsing/analysis
  - “Needle in a haystack” identification of worst queries
  - Worst execution times, # of execs, etc.

SQL code is the #2 cause of performance issues
97% of those surveyed will use this
Next Steps

MySQL Proxy Forum


Documentation – MySQL Proxy


Documentation – MySQL Load Balancer


Beta Testing MySQL Proxy-Enabled Products

enterprise-beta@mysql.com

MySQL Proxy Online Poll

http://dev.mysql.com/tech-resources/quickpolls/mysql-proxy.html
THANKS

Let's talk!