Getting started with Gearman for MySQL

Giuseppe Maxia
Eric Day
A short (I promise!) history of computing
Mainframe

Operating system
Hardware
Application

Terminal
Terminal
Terminal

User friendliness

0 - 100

Tuesday, 13 April 2010
Mini computers

- Mainframe
- Mini
- terminal
- terminal
- terminal
- terminal

operating system
hardware
application

USER FRIENDLINESS

0

Tuesday, 13 April 2010
Some actors

• memcached
• gearman

Used in production by Yahoo!, LiveJournal and CraigList
>= GERMAN
GEARMAN?
MANAGER
server
worker
client

http://gearman.org
Gearman: a technology for distributed computing
Distributed
Multiple operating systems
multiple languages
redundancy
USING GEARMAN

- Server: gearmand
- Client libraries:
  - C/C++
  - Java
  - Perl
  - PHP
  - Python
Simple usage

GEARMAN

- Command line client and worker
starting the server

/usr/local/sbin/gearmand -d

# started as daemon.
# No feedback given on the command line
/usr/local/sbin/gearmand -v -v
INFO Starting up
INFO Listening on :::4730 (5)
INFO Listening on 0.0.0.0:4730 (6)

# started as normal application
# verbose output requested
starting the worker

gearman -w -h hostname -p 4730 \ -f conta wc

# -w = act as worker
# -f = function
# conta = function name
# wc = command to execute when function 'conta' is called
what the server says

/usr/local/sbin/gearmand -v -v
INFO Starting up
INFO Listening on ::::4730 (5)
INFO Listening on 0.0.0.0:4730 (6)
...
INFO Accepted connection from 127.0.0.1:4994
INFO [ 0] 127.0.0.1:4994 Connected
starting the client

gearman -h hostname -p 4730 \
  -f conta < ~/.bashrc
  57 135 2149  # <- output
  # from worker

# -f = function
# conta = function name
# < ~/.bashrc = input data
what the server says

/usr/local/sbin/gearmand -v -v
INFO Starting up
INFO Listening on :::4730 (5)
INFO Listening on 0.0.0.0:4730 (6)
...
INFO Accepted connection from 127.0.0.1:4994
INFO [  0] 127.0.0.1:4994 Connected
...
INFO Accepted connection from 127.0.0.1:5181
INFO [  0] 127.0.0.1:5181 Connected
INFO [  0] 127.0.0.1:5181 Disconnected

Tuesday, 13 April 2010
What happened

1. server start

   listen to port 4730
What happened

2 worker starts

registers function 'conta' to server
What happened

3

client starts

requires function 'conta' from server

provides input data
What happened

4

server sends client request to worker

passes all input data to worker
What happened

5

worker receives request and data

processes input

Tuesday, 13 April 2010
What happened

worker returns processed data

server passes it to client
What happened

7 client receives processed data

client displays result
A simple Perl worker
A simple worker

1. add server
2. add function
3. loop
4. function definition
use strict;
use warnings;
use Gearman::XS qw(:constants);
use Gearman::XS::Worker;

my $host = '127.0.0.1';
my $port = 4730;

my $worker = new Gearman::XS::Worker;

my $ret = $worker->add_server($host, $port);
if ($ret != GEARMAN_SUCCESS) {
    printf(STDERR "\%s\n", $worker->error());
    exit(1);
}
my $options = ''; 

$ret = $worker->add_function(
    "reverse",  # public function name
    0,           # timeout
    &myreverse,  # reference to function
    $options);   # function arguments
if ($ret != GEARMAN_SUCCESS) {
    printf(STDERR "%s\n", $worker->error());
}
while (1) {
    my $ret = $worker-&gt;\texttt{work}();
    if ($ret != GEARMAN\_SUCCESS) {
        printf(STDERR "\texttt{\%s\n}", $worker-&gt;error());
    }
}


sub myreverse {
  my ($job) = @_;

  my $workload = $job->workload();
  my $result   = reverse($workload);

  printf("Job=%s F_Name=%s Workload=%s Result=%s\n",
      $job->handle(),
      $job->function_name(),
      $job->workload(),
      $result);
  return $result;
}
A simple client
A simple perl client

- add server
- run a task
use strict;
use warnings;
use Gearman::XS qw(:constants);
use Gearman::XS::Client;

my $client = new Gearman::XS::Client;

my $host = '127.0.0.1';
my $port = 4730;
my $ret = $client->add_server($host, $port);
if ($ret != GEARMAN_SUCCESS) {
    printf(STDERR "%s\n", $client->error());
    exit(1);
}
my $input = shift || 'teststring';

my ($return, $result) =
   $client->do("reverse", $input);
if ($return == GEARMAN_SUCCESS) {
   printf("Result=%s\n", $result);
}
A sample run
host 1

perl worker.pl
perl client.pl
Result=gnirtstset
host 1

perl worker.pl
Job=H:gmac3.local:4  F_Name=reverse
Workload=teststring  Result=gnirtstset
more client functions

- `do_background`
- `add_task`
- `run_tasks`
Some advanced usage

- DBIx::SQLCrosstab
- Data cubes
- Perl only
See more hacks!

- Gearman hacks with MySQL
- Thursday at 2pm
Image processing

• CPU intensive
• Large storage needed
• Application is OS specific
See more performance!

- Boosting database performance with Gearman
- tomorrow, at 3:05pm
Gearman for the MySQL server
More power to MySQL

• Perl/PHP/Python functions
• Shell access (you can send and receive email!)
• filesystem access
• advanced monitoring through Gearman features
WARNING!
You can easily shoot yourself in the foot
Using MySQL UDF

• Install Gearman
• Get MySQL server source (needed to compile the UDF)
• Get the UDF source
  • https://launchpad.net/gearman-mysql-udf
• Follow the instructions (good luck) 😊
my @functions = ( ['reverse', \&myreverse], ['count', \&mycount], ['shell', \&myshell], ['eval', \&myeval], ['store', \&mystore] );

# see the full code here:
Create a worker (2)

```perl
for my $func (@functions) {
    $ret = $worker->add_function(
        $func->[0], 0, $func->[1], $options);
    if ($ret != GEARMAN_SUCCESS) {
        printf(STDERR
               "error with function %s - %s\n",
               $func->[0],
               $worker->error());
    }
}
```
sub myshell {
    my $job = shift;
    my $workload = $job->workload();
    my $result   = qx($workload);
    return $result;
}

# WARNING!
# You can shoot yourself
# in the foot!
sub myshell {
    my $job = shift;
    my $workload = $job->workload();
    my $result   = qx($workload);
    return $result;
}

# WARNING!
# You can shoot yourself
# in the foot!
Create a worker (4)

sub myeval {
    my $job = shift;
    my $workload = $job->workload();
    my $result   = eval $workload;
    return $result;
}

# WARNING!
# You can machine gun yourself
# in the foot!
sub myeval {
    my $job = shift;
    my $workload = $job->workload();
    my $result = eval $workload;
    return $result;
}

# WARNING!
# You can machine gun yourself
# in the foot!
Using the UDF

```sql
mysql> select gman_do('reverse','abcd') as test;
+------+
| test |
+------+
| dcba |
+------+
```
Using the UDF

mysql> SELECT gman_do('shell',
                   concat('ls -lh ',
                   (select variable_value from
                    information_schema.global_variables
                   where variable_name = "datadir" ))))\G

total 40976
-rw-rw---- 1 gmax staff  5.0M Nov 11 13:34 ib_logfile0
-rw-rw---- 1 gmax staff  5.0M Nov 11 13:34 ib_logfile1
-rw-rw---- 1 gmax staff  10M Nov 11 13:34 ibdata1
-rw-rw---- 1 gmax staff  1.2K Nov 11 13:34 msandbox.err
drwx------- 2 gmax staff  2.4K Nov 11 13:34 mysql
-rw-rw---- 1 gmax staff  6B Nov 11 13:34 mysql_sandbox5140.pid
drwx------- 2 gmax staff  68B Nov 11 13:34 test

Tuesday, 13 April 2010
Using the UDF

```sql
mysql> select gman_do('eval','2 * 3') ;
+-------------------------+
| gman_do('eval','2 * 3') |
| 6                       |
+-------------------------+
```
Using the UDF

```sql
mysql> select gman_do('eval',
    concat('$_="",host,"";tr/a-z/b-za/; $_')
) as test from mysql.user;
```

```
+-----+
| test |
| %    |
| mpdbmiptu |
+-----+
```
Replication scenario

- Master
- Slave
- Slave

Tuesday, 13 April 2010
Replication scenario

<table>
<thead>
<tr>
<th>slaves status</th>
</tr>
</thead>
<tbody>
<tr>
<td>slave 1</td>
</tr>
<tr>
<td>slave 2</td>
</tr>
</tbody>
</table>

Tuesday, 13 April 2010
THANKS

Let's talk!