Verteilte und asynchrone Berechnung mit Gearman

Dennis Schön
Principal Software Architect
Berlin, 12/11/09
Why?
Kittens!
(LiveJournal.com Image Processing)
Kittens!
(LiveJournal.com Image Processing)
• Danga - Brad Fitzpatrick & company
  - Related to memcached, MogileFS, ...

• Anagram for “manager”
  - Gearman, like managers, assign the tasks but do none of the real work themselves

• Digg: 45+ Server, 400K jobs/day
• Yahoo: 60+ Server, 6M jobs/day
• XING: 45+ Server, 15M jobs/day
• LiveJournal, SixApart, DealNews, ...
Recent Development

- Rewrite in C
- New language APIs
  - PHP/C, Perl/C Drizzle, MySQL, Java, Python/C, JMS, Erlang, OCaml, ...
- Command line tool
- Protocol additions
- Multi-threaded (50k jobs/second)
- Persistent queues
- Pluggable protocol
Features

- Open Source (mostly BSD)
- Multi-language
  - Mix clients and workers from different APIs
- Flexible Application Design
  - Not restricted to a single distributed model
- Simple & Fast
- Embeddable
- Small & lightweight for applications of all sizes
- No Single Point of Failure
Gearman provides a distributed application framework
Uses TCP port 4730 (was port 7003)

**Client** - Create jobs to be run and send them to a job server

**Worker** - Register with a job server and grab jobs to run

**Job Server** - Coordinate the assignment from clients to workers, handle restarts
Gearman Stack

- Client Application
- Gearman Client API (C, Perl, PHP, MySQL UDF, ...)
- Gearman Job Server
- Gearman Worker API (C, Perl, PHP, ...)
- Worker Application

Provided by Gearman

Your Application
No Single Point of Failure

Verteilte und asynchrone Berechnung mit Gearman  |  Dennis Schön  | Berlin, 12/11/09
$client = new Gearman::XS::Client;
$client->add_server();
print $client->do('reverse', 'Hello World!');

$worker = new Gearman::XS::Worker;
$worker->add_server();
$worker->add_function("reverse", 0, \&reverse, 0);

while (1) {
    $worker->work();
}

sub reverse {
    my ($job) = @_;
    return reverse($job->workload());
}
$ gearmand -d

$ perl worker.pl &
[1] 7929

$ perl client.pl
!dlroW olleH
How Is This Useful?

- Natural load balancing
  - Workers are notified and ask for work, not forced
- Multi-language integration
- Distributed parallel processing
- Possibly closer to the data
- Synchronous and Asynchronous queues
Back to the Kittens
Kittens!
(LiveJournal.com Image Processing)
Image Resize Worker

```perl
$worker = new Gearman::XS::Worker;
$worker->add_server();
$worker->add_function("resize", 0, \&resize, 0);

while (1) {
    $worker->work();
}

sub resize {
    my ($job) = @_;
    my $image = Image::Magick->new();
    $image->BlobToImage($job->workload());
    $image->Scale(width => 200, height => 150);
    return $image->ImageToBlob();
}
```
$ gearmand -d

$ perl resize.pl &
[1] 17529

$ gearman -f resize < large.jpg > thumb.jpg

$ ls -sh large.jpg thumb.jpg
3.0M large.jpg 32K thumb.jpg
Gearman @ XING
• 8 million members

• 15 million dynamic page impressions per day

• 300+ servers

• 2 data centers

• 3 main development languages
• 2 job servers

• 20 dedicated worker servers

• 300 Gearman jobs / second avg.

• 700+ Gearman jobs / second peak
Admin Interface

Verteilte und asynchrone Berechnung mit Gearman | Dennis Schön | Berlin, 12/11/09
<table>
<thead>
<tr>
<th>Status Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Status</td>
</tr>
<tr>
<td>Status information</td>
</tr>
<tr>
<td>Performance Data</td>
</tr>
<tr>
<td>Current Attempt</td>
</tr>
<tr>
<td>State Type</td>
</tr>
<tr>
<td>Last Check Type</td>
</tr>
<tr>
<td>Last Check</td>
</tr>
<tr>
<td>Next Scheduled Active Check</td>
</tr>
<tr>
<td>Latency</td>
</tr>
<tr>
<td>Check Duration</td>
</tr>
<tr>
<td>Last State Change</td>
</tr>
<tr>
<td>Current State Duration</td>
</tr>
<tr>
<td>Last Service Notification</td>
</tr>
<tr>
<td>Current Notification Number</td>
</tr>
<tr>
<td>Is This Service Flapping?</td>
</tr>
<tr>
<td>Percent State Change</td>
</tr>
<tr>
<td>In Scheduled Downtime?</td>
</tr>
<tr>
<td>Last Update</td>
</tr>
</tbody>
</table>
Image Upload

- Scale
- Upload
- Synchronous
  - return success/failure

- ImageMagick
- Security record
• Import contacts from webmail
• Proprietary software (PHP)
• Synchronous
  - return list of contacts
Mark E-Mail addresses bounced

- Flag E-Mail addresses
- Command line client
- Synchronous
  - return success/failure
• check all form parameters
• CGI::IDS
• Asynchronous
  - log malicious requests

• +700 jobs/second peak
Dear Dennis,

Here's the XING newsletter you requested, with the latest news from your network, new features and your personal statistics:

**Discover companies on XING**

Now companies can draw even more attention to themselves on XING. This gives you, the XING member, more information at your fingertips: contact persons, statistics, updates and connection paths. The best part is you can connect with companies, and stay informed automatically.

Have a look at some of our launch partners’ profiles: Bertelsmann, Telekom, Accenture, Continental, Lufthansa & Hubert Burda Media. [Go to XING Company Profiles](#)

**Your personal statistics**

You have no unread messages.
You have no unconfirmed contact requests.

**Last week:**

- Contact requests received: 1
- Messages received: 2

**Your current network:**

- Direct contacts: 217
- Contacts of your contacts: 45,551

---

**XING Best Offers**

Send unique Christmas wishes with **PLUSBRIEF** from Deutsche Post, and save up to €10.

This is an exclusive offer for XING Premium members

[› All Best Offers](#)

---

**XING Jobs**

**Job offers for you**

- [Developer Ruby on Rails (…)](#)
  XING AG
- [Software-Architekt](#)
  EliteMedianet GmbH/ TOMORROW FOCUS AG
- [Teamleiter Customer Intel […]](#)
  XING AG

[› More job offers](#)

[› To XING Jobs](#)
• ~1 million newsletter per night

• 3 step process
  - cronjob creates one NL job per user
    ‣ sends 2000 user batched to job server
  
  - NL job triggers one render job per box
    ‣ concurrent interface waits until all boxes are rendered
  
  - NL job assembles newsletter and triggers send job
APIs
• C (libgearman)
  - PHP, Perl::XS, Drizzle, MySQL, PostgreSQL
• SWIG (Python, Ruby, ...)
• Perl
• PHP
• Java
• Python
• Ruby
• Erlang
• ...

APIs
• gearman
  - Included in C server and library package
  - Command line and shell script interface

• Client mode
  - ls | gearman -f function
  - gearman -f function < file
  - gearman -f function “some data”

• Worker mode
  - gearman -w -f function -- wc -l
  - gearman -w -f function ./script.sh
$ gearmand -d

$ gearman -w -f test --grep lib &
[1] 7622

$ ls / | gearman -f test
lib
lib32
lib64
Client API
• Create client object
  - Can have multiple objects in an application
  - Server lists, options, ...

```php
$client = new Gearman::XS::Client;
$client->add_server();

$client = new Gearman::XS::Client;
$client->add_server("10.0.0.1");
$client->add_server("10.0.0.2", 7003);

$client = new Gearman::XS::Client;
$client->add_servers("10.0.0.1, 10.0.0.2:7003");
```
• One job at a time
  -*do* functions and methods

```php
$client = new Gearman::XS::Client;
$client->add_server();
print $client->do("reverse", "Hello World!");
print $client->do_high("reverse", "Hello World!");
print $client->do_low("reverse", "Hello World!");
```
Multiple jobs at a time
- "*task*" functions and methods

```
$client = new Gearman::XS::Client;
$client->add_server();

$client->add_task("reverse", "Hello World!");
$client->add_task_high("reverse", "Hello High!");
$client->add_task_low("reverse", "Hello Low!");

$client->set_complete_fn(&completed_cb);
$client->run_tasks();  # blocks

sub completed_cb {
  my ($task) = @_;  
  print $task->data();
}
```
Tasks vs Jobs

• A job is a task
• A task is not always a job

• Check status of background job
  - Not a job, but is a task
  - Send job handle, receive status
    - $client->job_status()
    - $client->add_task_status()

• Clients deal tasks
• Workers deal with jobs
• Foreground (Synchronous)
  - $client->do()

• Background (Asynchronous)
  - $client->do_background()

• High and Low priority
  - $client->do_high()
  - $client->do_low()

• Mix and match
  - $client->do_high_background()
Job Attributes

- Application data/state for tasks
- Optional Unique Key
  - UUID generated if none given
  - Coalesce jobs in job server
  - Only one worker per key at any time
  - All clients get the same response
  - Resizing an image
    ‣ Multiple clients submit jobs at the same time
    ‣ Workers runs once, all clients get thumbnail
Concurrent Task API

• Queue jobs to run

• Register callbacks for interesting events

• Run tasks
  - $client->run_tasks()
  - Returns when all tasks are complete

• No guarantee on response order
• Available Callbacks
  - `set_workload_fn()` - Streaming job workload
  - `set_created_fn()` - Job created in server
  - `set_data_fn()` - Partial data response
  - `set_warning_fn()` - Partial warning response
  - `set_status_fn()` - Status (X/Y done)
  - `set_complete_fn()` - Job complete
  - `set_exception_fn()` - Exception caught (deprecated Perl)
  - `set_fail_fn()` - Job failed
  - `clear_fn()` - Clear all set callbacks
Worker API
• Create worker object
  - Can have multiple objects in an application
  - Server lists, options, ...

```php
$worker = new Gearman::XS::Worker;
$worker->add_server();
```

```php
$worker = new Gearman::XS::Worker;
$worker->add_server("10.0.0.1");
$worker->add_server("10.0.0.2", 7003);
```

```php
$worker = new Gearman::XS::Worker;
$worker->add_servers("10.0.0.1, 10.0.0.2:7003");
```
Worker API

• Register function abilities
  - Function name
  - Local callback function
• Wait for jobs on those functions
• Will only run one job at a time

```php
$worker = new Gearman::XS::Worker;
$worker->add_server();

$worker->add_function("function1", &first_callback);
$worker->add_function("function2", &common_callback);
$worker->add_function("function3", &common_callback);

while (1) {
    $worker->work();
}
```
• Have access to $job object

• Get job information
  - $job->workload() - Get job workload
  - $job->function_name() - Get function name
  - $job->unique() - Get unique key
  - $job->handle() - Get job handle
Worker Callback Function

• Send intermediate job responses
  - \$job->send_data(...) - Send partial result
  - \$job->send_warning(...) - Send warning
  - \$job->send_status(X, Y) - Send X/Y status

• Return value sent back to caller

• Don’t bother if it’s a background job
• Except \$job->status(X, Y)
• Can also return failure

```perl
$worker = new Gearman::XS::Worker;
$worker->add_server();
$worker->add_function("reverse", 0, \&always_fail, 0);

sub always_fail {
    my ($job) = @_;  
    return GEARMAN_WORK_FAIL;
}
```
Job Server
Job Server

- Listens on port 4730
- Clients and workers connect
- Handle job assignment
- Restart jobs on worker failure
- Advanced features
  - Pluggable persistent queue
  - Pluggable protocols
• gearmand --help
• Common options

- d, --daemon
- h, --help
- l, --log-file=FILE
- p, --port=PORT
- P, --pid-file=FILE
- t, --threads=THREADS
- u, --user=USER
- v, --verbose
• Administrative Protocol
  - Telnet port 4730

• workers
  - Lists connected clients

• status
  - Lists registered functions

• maxqueue
  - Sets the maximum queue size for a function

• shutdown
  - Shutdown the server
  - Optional “graceful” argument

• version
• Administrative Protocol
  - "workers" command
  - IP address
  - Client ID
  - List of functions

```
$ telnet obc-grm42-1 4730
Escape character is '^]'.
workers
949 10.43.12.6 - : resize_image
948 10.43.12.7 - : ids_check
940 10.43.12.5 - : mark_email ids_check
...```
Administrative Protocol
- "status" command
- Function
- Total
- Running
- Available

$ telnet obc-grm42-1 4730
Escape character is '^[].'
status
resize_image 0 0 10
ids_check 0 0 10
mark_email 0 0 10
...

Verteilte und asynchrone Berechnung mit Gearman | Dennis Schön | Berlin, 12/11/09
Persistent Queues

- Only for background jobs
- Specify -q <queue> option

- libdrizzle module for Drizzle and MySQL
- libmemcached
- PostgreSQL
- sqlite3
- Flat file
• Handle packet parsing and packing
• Optionally handle raw socket I/O

• HTTP protocol
• Others coming soon
  - memcached
  - XMPP
• GET and POST requests
  - Send workload with POST, Content-Length
• Function name from URL
• Optional headers
  - X-Gearman-Unique: <unique key>
  - X-Gearman-Background: true
  - X-Gearman-Priority: <high/low>
• Other headers ignored for now
$ gearmand -r http -d

<start reverse worker>

$ nc localhost 8080
POST /reverse HTTP/1.1
Content-Length: 12
Hello World!

HTTP/1.0 200 OK
X-Gearman-Job-Handle: H:lap:1
Content-Length: 12
Server: Gearman/0.9

!dlroW olleH
Get involved

- http://gearman.org
- #gearman on irc.freenode.net
- http://groups.google.com/group/gearman
- http://xing.com/career/  :-)

Verteilte und asynchrone Berechnung mit Gearman  |  Dennis Schön  |  Berlin, 12/11/09
Thank you for your kind attention!