

Gearman & MySQL

<http://www.gearman.org/>



Eric Day

Software Engineer at Concentric
<http://www.concentric.com/>

Blog: <http://www.oddments.org/>
eday@oddments.org

Overview

- History and recent development
- How Gearman works
- Simple example
- Use case: URL processing
- Use case: Image processing
- Use case: E-mail storage
- Map/Reduce
- Future plans
- Related projects

History

- Danga – Brad Fitzpatrick
- Technology behind LiveJournal
- memcached, MogileFS, Gearman
- Gearman: Anagram for “manager” because managers assign tasks, but do nothing themselves.
- Digg: 45+ servers, 400K jobs/day
- Yahoo: 60+ servers, 6M jobs/day
- Other client/worker interfaces
- All Open Source!

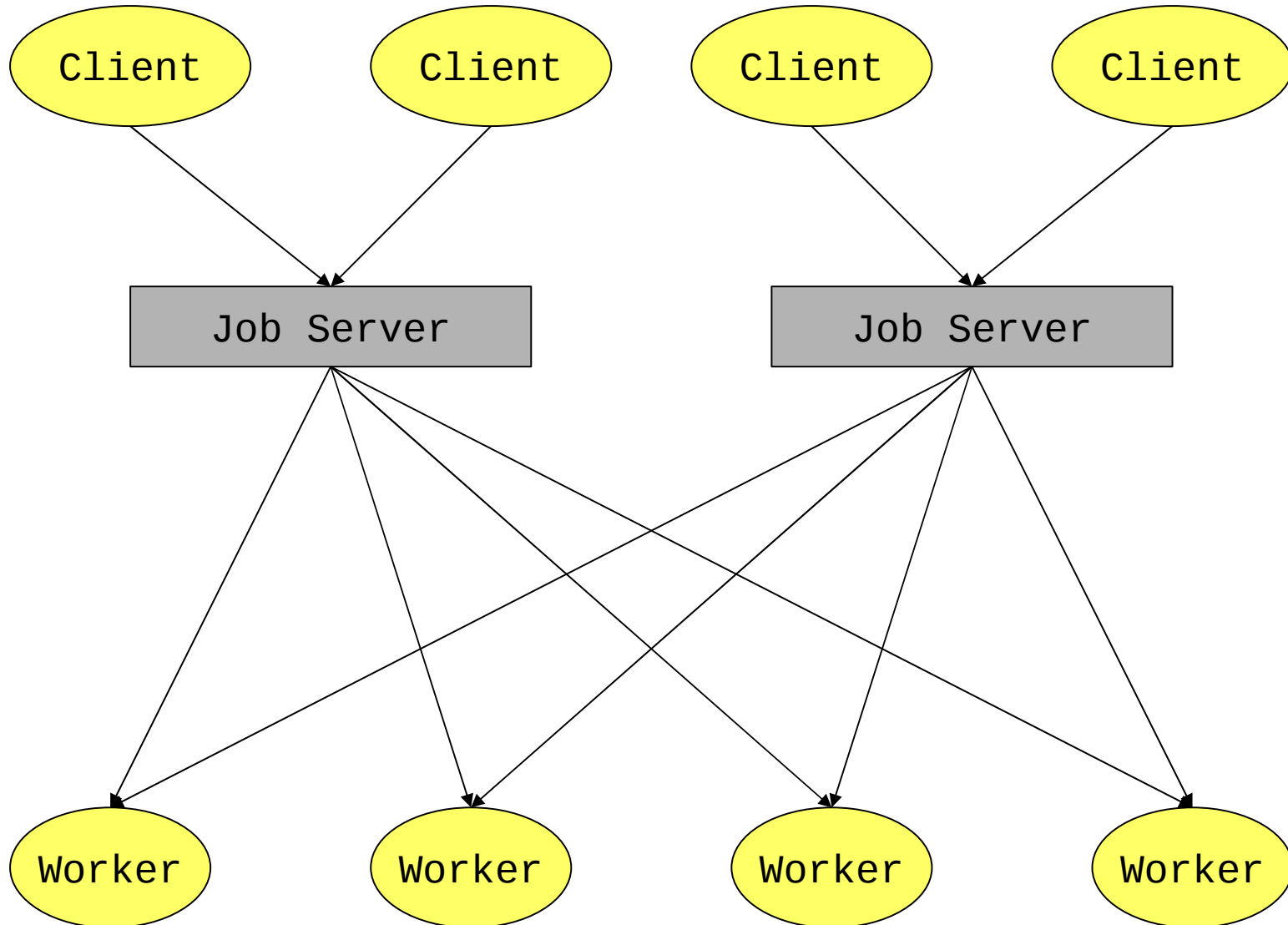
Recent Development

- Brian Aker started rewrite in C
- I joined after designing a similar system
- Helped with the rewrite in C
- Fully compatible with existing interfaces
- Wrote MySQL UDFs based on C library
- New PHP extension based on C library with James Luedke
- Persistent worker queues, replication, e-mail storage

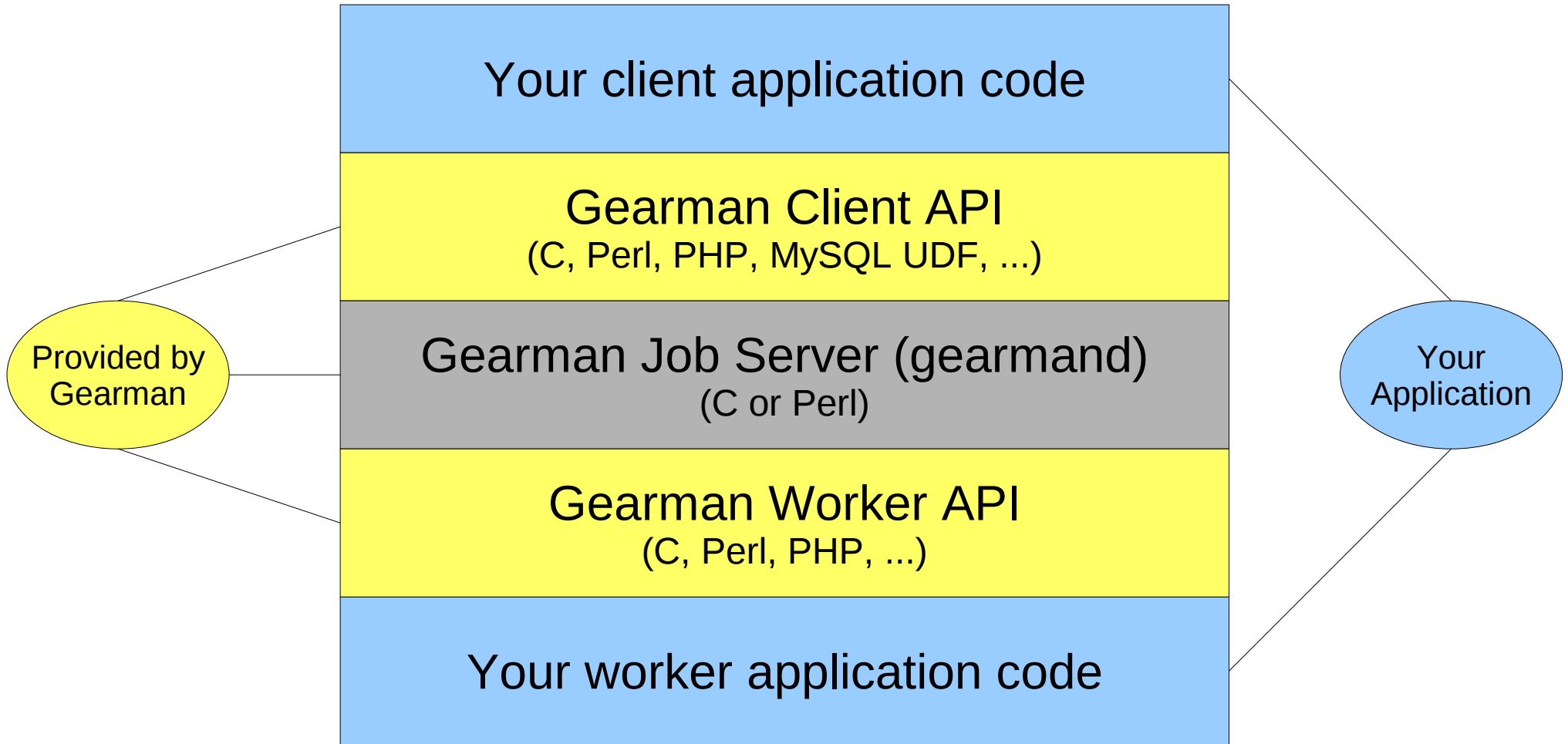
Gearman Basics

- Gearman provides a job distribution framework, does not do any work itself
- Uses TCP, port 4730 (was port 7003)
- **Client** – Create jobs to be run and then send them to a job server
- **Worker** – Register with a job server and grab jobs as they come in
- **Job Server** – Coordinate the assignment of jobs from clients to workers. Handle restarting jobs if workers go away.

Gearman Componentnets



Gearman Application Stack



How is this useful?

- Natural load distribution, easy to scale out
- Push custom application code closer to the data, or into “the cloud”
- For MySQL, it provides an extended UDF interface for multiple languages and/or distributed processing
- It is the nervous system for how distributed processing communicates
- Building your own Map/Reduce cluster

Simple Example (Perl)

Client:

```
use Gearman::Client;
my $client = Gearman::Client->new;
$client->job_servers('127.0.0.1:4730');
$ref= $client->do_task('reverse', 'MySQL Webinar');
print "$$ref\n";
```

Worker:

```
use Gearman::Worker;
sub my_reverse_fn {
    reverse $_[0]->arg;
}

my $worker = Gearman::Worker->new();
$worker->job_servers('127.0.0.1:4730');
$worker->register_function('reverse',
                           \&my_reverse_fn);
$worker->work() while 1;
```

Running the Perl Example

- Using CPAN, install Gearman::Client, then:

```
shell> gearmand -p 4730 -d
```

```
shell> ./worker.perl &  
[1] 17510
```

```
shell> ./client.perl  
ranibeW LQSyM  
shell>
```

Simple Example (PHP)

Client:

```
$client = new gearman_client();
$client->add_server('127.0.0.1', 4730);
list($ret, $result)= $client->do('reverse',
                                'MySQL Webinar');
print "$result\n";
```

Worker:

```
function my_reverse_fn($job) {
    return strrev($job->workload());
}

$worker = new gearman_worker();
$worker->add_server('127.0.0.1', 4730);
$worker->add_function('reverse',
                    'my_reverse_fn');
while (1) $worker->work();
```

Requires Gearman PHP extension and php-cli for worker

Simple Example (MySQL)

- Install Gearman MySQL UDF, then:

```
mysql> SELECT gman_servers_set("127.0.0.1:4730") AS result;
+-----+
| result |
+-----+
| NULL   |
+-----+
1 row in set (0.00 sec)

mysql> SELECT gman_do('reverse', 'MySQL Webinar') AS result;
+-----+
| result          |
+-----+
| ranibeW LQSyM |
+-----+
1 row in set (0.00 sec)
```

Use case: URL processing

- We have a collection of URLs
- Need to cache some information about them
- RSS aggregating, search indexing, ...
- Use MySQL for storage, Gearman for concurrency and load distribution
- Allows you to scale to more instances easily
- Use Gearman background jobs
- LWP Perl module (Library for WWW in Perl)
- MySQL DBD driver for Perl

Use case: URL processing

```
# Setup table
```

```
CREATE TABLE url (  
  id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
  url VARCHAR(255) NOT NULL,  
  content LONGBLOB  
);
```

```
# Insert a few URLs
```

```
mysql> SELECT * FROM url;
```

id	url	content
1	http://www.mysql.com/	NULL
2	http://www.gearman.org/	NULL
3	http://www.oddments.org/	NULL

```
3 rows in set (0.00 sec)
```

Use case: URL processing

- Run `SELECT` statement to start Gearman jobs
- Gearman UDF will queue all URLs that need to be fetched in the job server
- Perl worker will:
 - Grab job from job server
 - Fetch content of URL passed in from job using LWP
 - Connect to MySQL database
 - Insert the content into the 'content' column
 - Return nothing (since it's a background job)

Use case: URL processing

```
use Gearman::Worker;
use LWP::Simple;
use DBI;

my $worker = Gearman::Worker->new();
$worker->job_servers('127.0.0.1:4730');
$worker->register_function('url_get', \&url_get);
$worker->work while 1;

sub url_get
{
    my $content = get $_[0]->arg;
    my $dbh = DBI->connect("DBI:mysql:test:127.0.0.1", "root");
    my $sth = $dbh->prepare("UPDATE url SET content=? WHERE url=?");
    $sth->execute($content, $_[0]->arg);
    $sth->finish();
    $dbh->disconnect();
    "";
}
```

Use case: URL processing

```
mysql> SELECT gman_do_background('url_get', url) FROM url;
+-----+
| gman_do_background('url_get', url) |
+-----+
| H:lap:6                             |
| H:lap:7                             |
| H:lap:8                             |
+-----+
3 rows in set (0.00 sec)

# Wait a moment while workers get the URLs and update table

mysql> SELECT id,url,LENGTH(content) AS length FROM url;
+----+-----+-----+
| id | url                | length |
+----+-----+-----+
|  1 | http://www.mysql.com/ | 17665 |
|  2 | http://www.gearman.org/ | 16291 |
|  3 | http://www.oddments.org/ | 45595 |
+----+-----+-----+
3 rows in set (0.00 sec)
```

Use case: Image processing

- Need to generate thumbnails, perform image recognition, or apply various filters
- Create your own image processing farm
- Building off of URL processing example, use image URLs, filenames, or BLOBs in MySQL
- Write a Perl or PHP worker that uses GD library or ImageMagick
- Store result into database or filesystem
- Run multiple instances of the worker on as many machines as you need

Use case: E-mail storage

- The Problem:
 - Need to handle many message inserts/second
 - Need fast mailbox access, so preferably partitioned
 - Need replication for load balancing and reliability
 - Want real multi-master (2+, no heartbeat failover)
 - Custom application code close to data (filtering, ...)
- MySQL, with proper tuning, replication, and sharding will get you the first three
- What about real multi-master and custom code in the data nodes?

Use case: E-mail storage

- Use Gearman workers to keep persistent & replicated queues for all write operations
- Gearman workers still store data in MySQL
- Create a Gearman worker name for each shard (mail1, mail2, mail3, ...)
- Mail servers use Gearman client interface to query each shard, aggregate results for complete view
- Data nodes are disposable, they can and will fail (using principals of Map/Reduce)

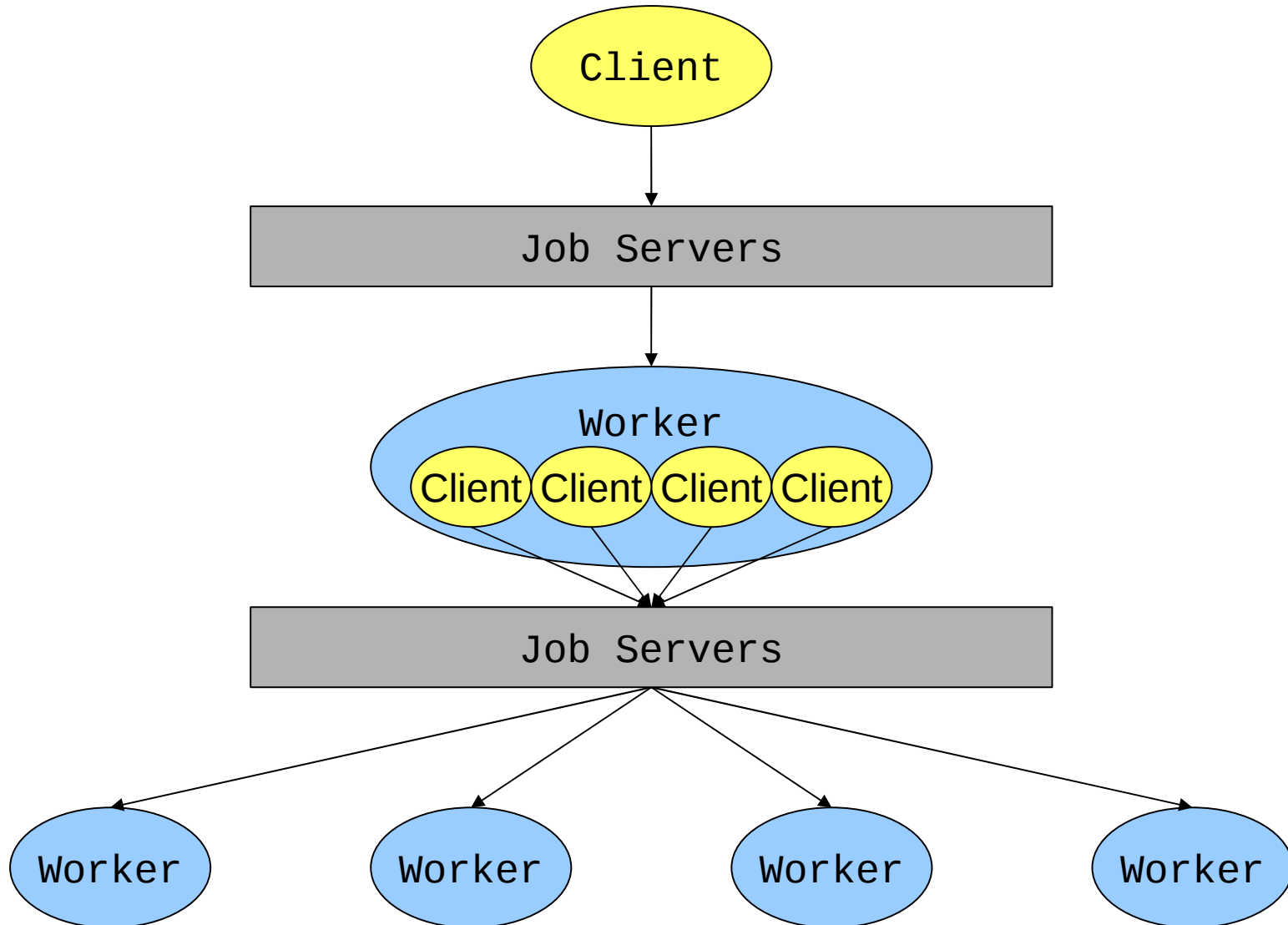
Use case: E-mail storage

- Multi-master requires an eventual consistency data model
- Think quantum mechanics: data is in flux until you observe it, application needs to handle this
- Can apply write operations in any order, unique event IDs resolve differences
- When resolving differences, always err on the side of preserving data
- Model can apply to many other applications

Map/Reduce in Gearman

- Top level client requests some work to be done
- Intermediate worker splits the work up and sends a chunk to each leaf worker (the “map”)
- Each leaf workers perform their chunk of work
- Intermediate worker waits for results, and aggregates them in some meaningful way (the “reduce”)
- Client receives completed response from intermediate worker
- Just one way to design such a system

Map/Reduce in Gearman



What's missing?

- Generic worker for handling persistent queues and replication almost ready
- More language interfaces based on C library (using SWIG wrappers or native clients), Drizzle UDFs, PostgreSQL functions
- Scheduled job queue (think cron)
- Improved event notification, statistics gathering, and reporting
- Dynamic code upgrades in cloud environment

Related Projects

- Gearman C Server and Library:
<https://launchpad.net/gearmand>
- Gearman MySQL UDF:
<https://launchpad.net/gearman-mysql-udf>
- Perl modules:
<http://search.cpan.org/~bradfitz/>
- New Gearman PHP extension:
<https://launchpad.net/gearman-php-ext>
- Gearman Persistent/Replicated Queue:
<https://launchpad.net/gearman-repq>
- E-Mail Storage Project:
<https://launchpad.net/gearman-mail-db>

Get in touch!

- #gearman on irc.freenode.net
- <http://groups.google.com/group/gearman>
- Questions?