



The Return of Gearman

Eric Day

`eday@oddments.org`

Percona Performance Conference 2009

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

Overview

- History
- Recent development
- How Gearman works
- Map/Reduce with Gearman
- Simple example
- Use case: URL processing
- Use case: Image processing
- Use case: Log aggregation
- Future plans

History

- Danga – Brad Fitzpatrick & company
- Technology behind LiveJournal
- Related to memcached, MogileFS, Perlbal
- Gearman: Anagram for “manager”
 - Gearman, like managers, assign the tasks but do none of the real work themselves
- Digg: 45+ servers, 400K jobs/day
- Yahoo: 60+ servers, 6M jobs/day
- Other client & worker interfaces came later

Recent Development

- Brian Aker started rewrite in C
- I joined after designing a similar system
- Fully compatible with existing interfaces
- Wrote MySQL UDFs based on C library
- New PHP extension based on C library thanks to James Luedke
- Gearman command line interface
- New protocol additions
- Job server is now threaded!

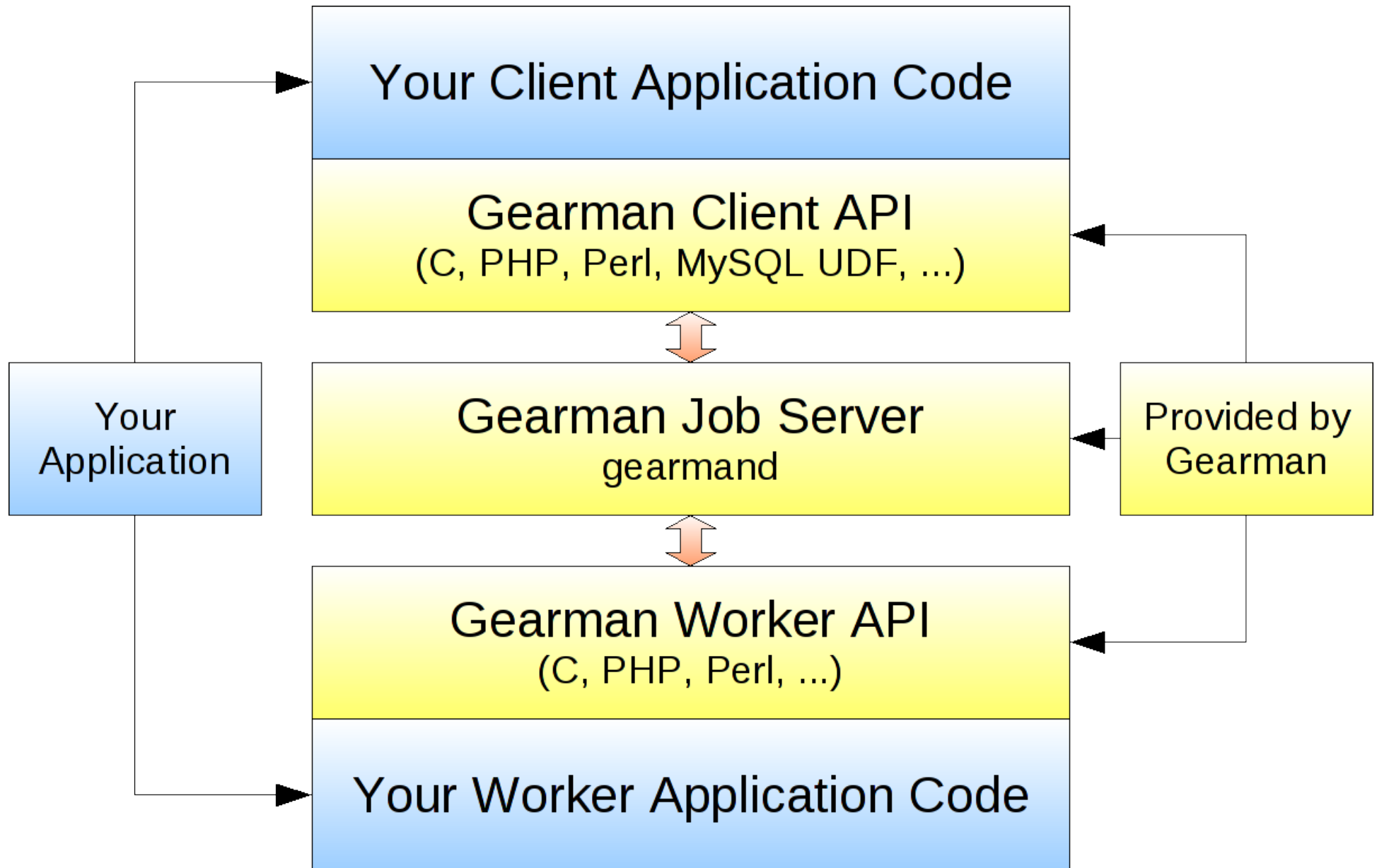
Gearman Basics

- Gearman provides a distributed application framework, does not do any real 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 of jobs if workers go away

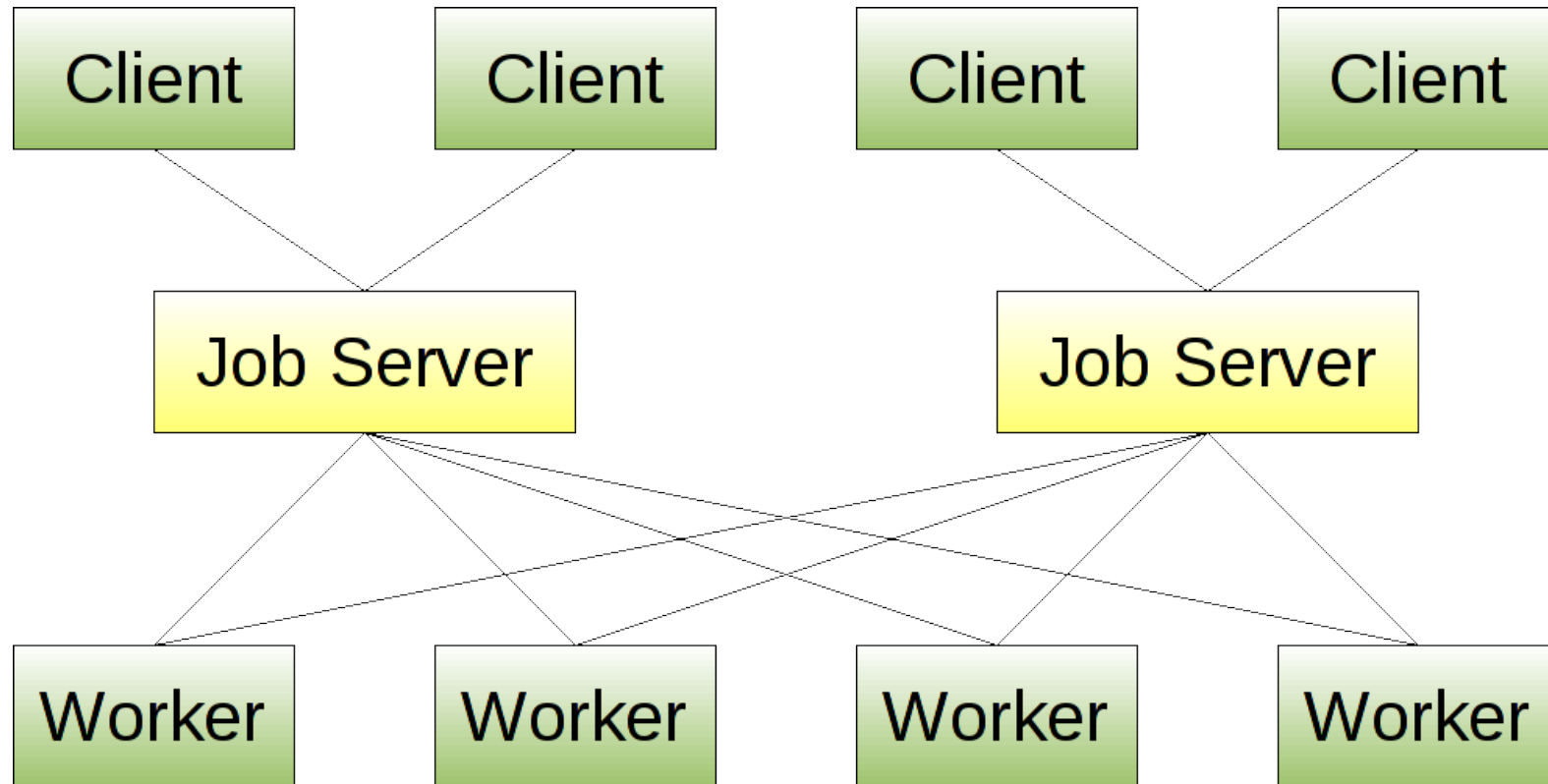
Gearman Benefits

- **Open Source**
- **Multi-language**
 - Mix clients and workers from different APIs
- **Flexible Application Design**
 - Not restricted to a single distributed model
- **Fast**
 - Simple protocol, C implementation
- **Embeddable**
 - Small & lightweight for applications of all sizes
- **No single point of failure**

Gearman Application Stack



Simple Gearman Cluster



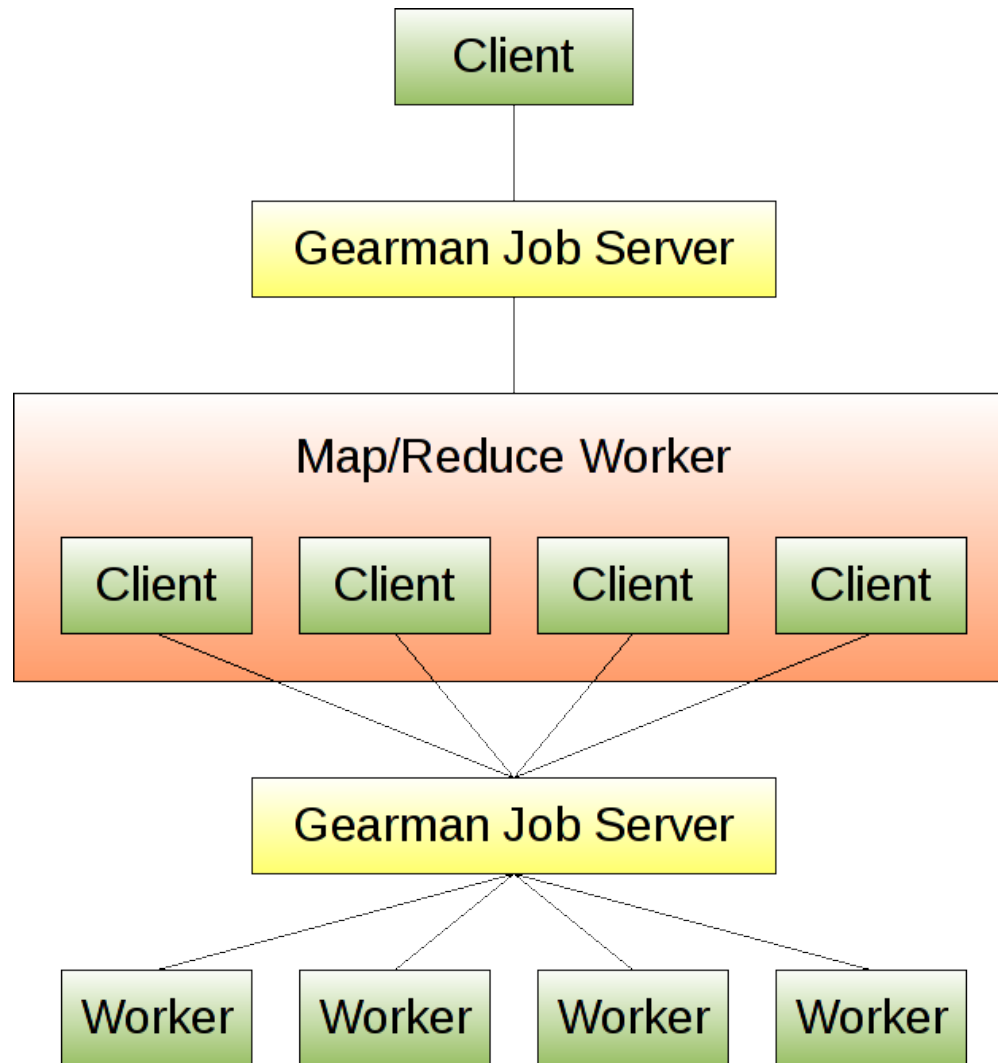
How is this useful?

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

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 worker performs their chunk of work
- Intermediate worker collects results and aggregates them in some way (the “reduce”)
- Client receives completed response from intermediate worker
- Just one way to design such a system

Map/Reduce in Gearman



Simple Example (PHP)

Client:

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

Worker:

```
$worker = new gearman_worker();  
$worker->add_server('127.0.0.1', 4730);  
$worker->add_function('reverse', 'my_reverse_fn');  
while (1) $worker->work();  
  
function my_reverse_fn($job) {  
    return strrev($job->workload());  
}
```

Running the PHP Example

- Gearman PHP extension required

```
shell> gearmand -d

shell> php worker.php &
[1] 17510

shell> php client.php
!dlroW olleH
shell>
```

Simple Example (MySQL)

- Gearman MySQL UDF required

```
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', 'Hello World!') AS result;
+-----+
| result          |
+-----+
| !dlroW olleH  |
+-----+
1 row in set (0.00 sec)
```

Use case: URL processing

- We have a collection of URLs
- Need to cache some information about the
 - RSS aggregating, search indexing, ...
- MySQL for storage
- MySQL triggers
- Gearman for queue and concurrency
- Gearman background jobs
- Scale to more instances easily

Use case: URL processing

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

Use case: URL processing

```
# Setup table

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

# Create Gearman trigger

CREATE TRIGGER url_get
BEFORE INSERT ON url
FOR EACH ROW
  SET @ret=gman_do_background('url_get', NEW.url);
```

Use case: URL processing

```
$worker = new gearman_worker();
$worker->add_server();
$worker->add_function('url_get', 'url_get_fn');
while(1) $worker->work();

function url_get_fn($job)
{
    $url = $job->workload();
    $content = fetch_url($url);

    # Process data in some useful way

    $content = mysql_escape_string($content);
    mysql_connect('127.0.0.1', 'root');
    mysql_select_db('test');
    mysql_query("UPDATE url SET content='$content' " .
                "WHERE url='$url'");
}
```

Use case: URL processing

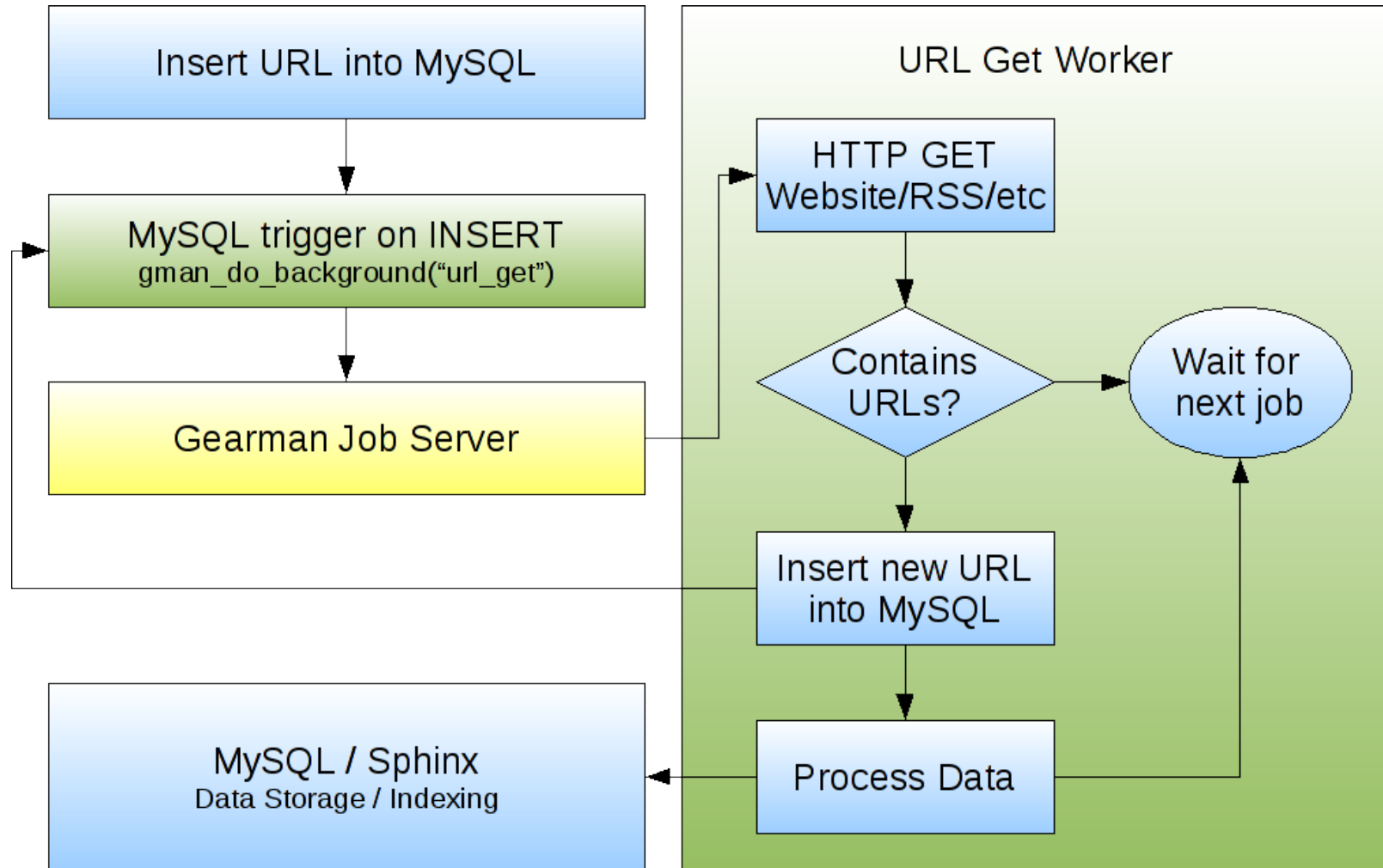
```
# Insert URLs

mysql> INSERT INTO url SET url='http://www.mysql.com/';
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO url SET url='http://www.gearman.org/';
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO url SET url='http://www.drizzle.org/';
Query OK, 1 row affected (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.drizzle.org/ | 45595 |
+----+-----+-----+
3 rows in set (0.00 sec)
```

Use case: URL processing



Use case: Image processing

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

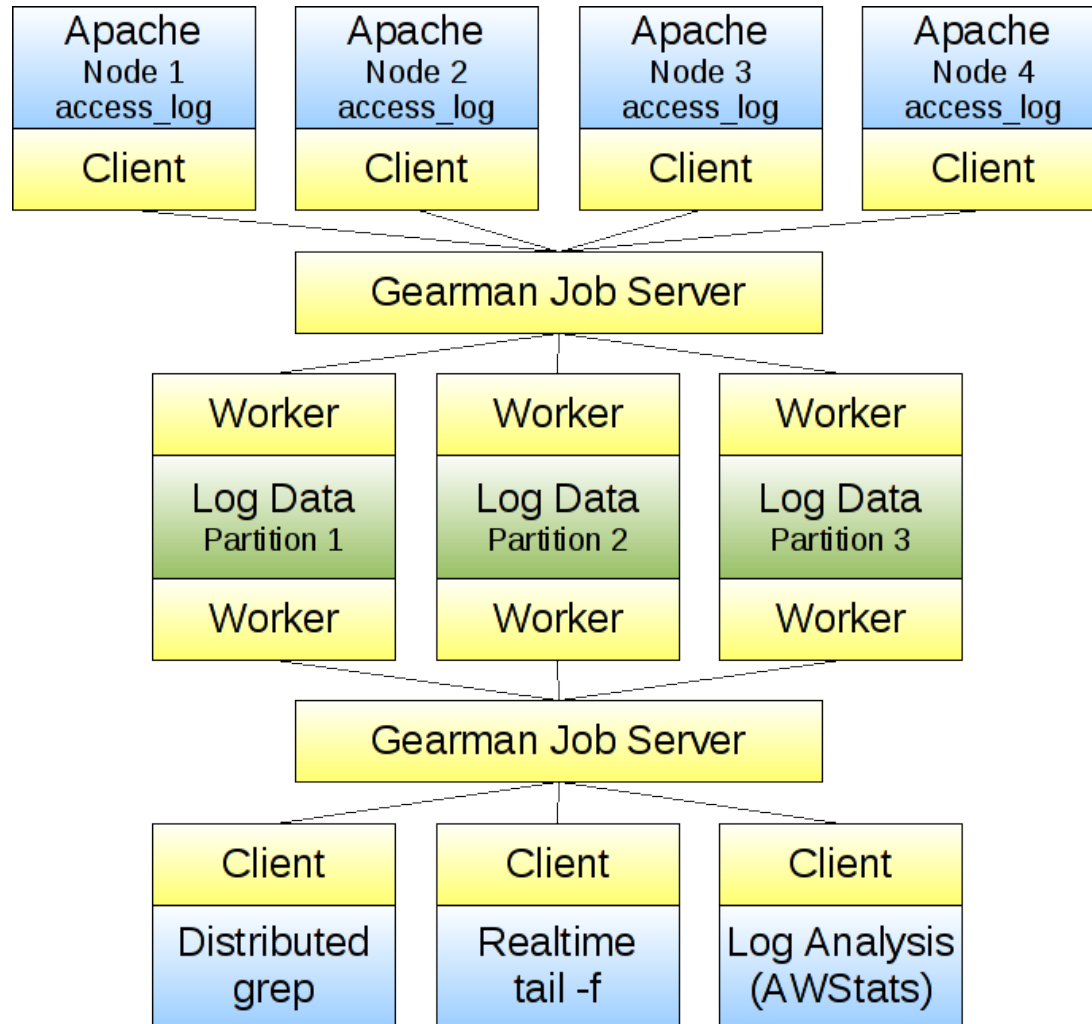
Use case: Log aggregation

- A collection of logs spread across multiple machines
- One consistent view
- Easy way to scan and process these logs
- Map/Reduce-like power for analysis
- Flexibility to push your own code into the log storage nodes
 - Saves on network I/O
- Merge-sort aggregate algorithms

Use case: Log aggregation

- Look at gathering Apache logs
- Gearman plugs into existing environments
 - `tail -f access_log | gearman -n -f logger`
 - CustomLog "|gearman -n -f logger" common
 - Write a simple Gearman Apache logging module
- Multiple Gearman workers
 - Partition logs
 - Good for both writing and reading loads
- Write Gearman clients and workers to analyze the data (distributed grep, summaries, ...)

Use case: Log aggregation



What's next?

- Persistent queues and replication very soon
- More language interfaces based on C library (using SWIG wrappers or native clients), Drizzle UDFs, PostgreSQL functions
- Native Java interface
- Improved event notification, statistics gathering, and reporting
- Drizzle replication and query analyzer
- Dynamic code upgrades in cloud environment
 - “Point & Click” Map/Reduce

Get in touch!

- eday@oddments.org
- <http://www.gearman.org/>
- #gearman on irc.freenode.net
- <http://groups.google.com/group/gearman>
- Gearman booth in the expo hall

- Questions?