



# Gearman

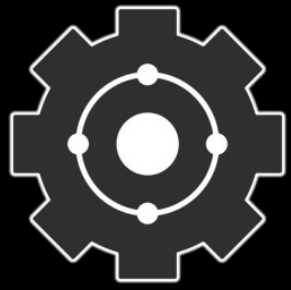
## **Introduction to Gearman**

O'Reilly Webcast

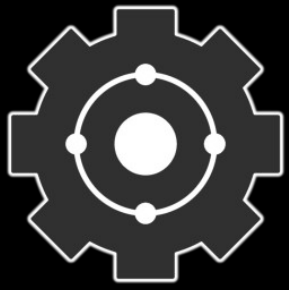
**Eric Day**

Senior Software Architect @ Rackspace

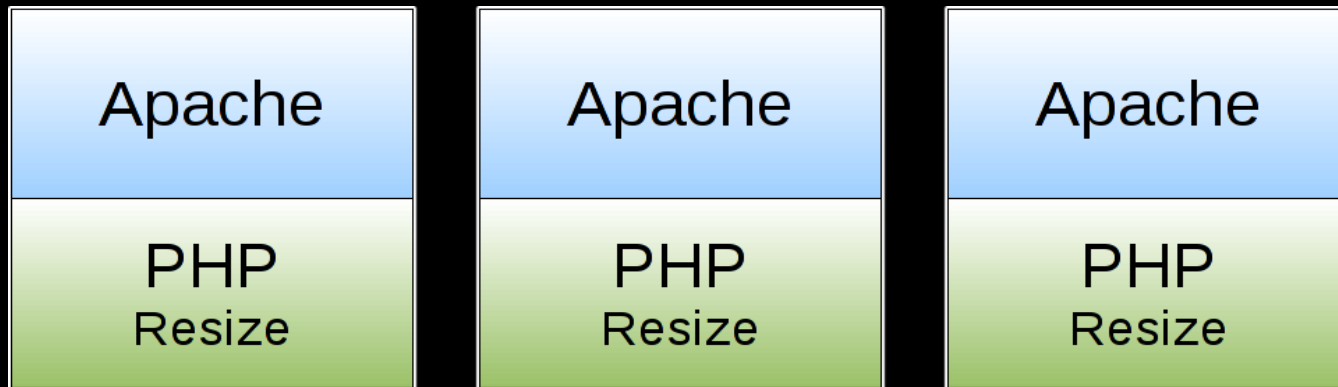
<http://oddmments.org/>

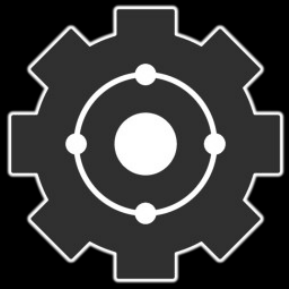


# Distributed image processing at LiveJournal.com

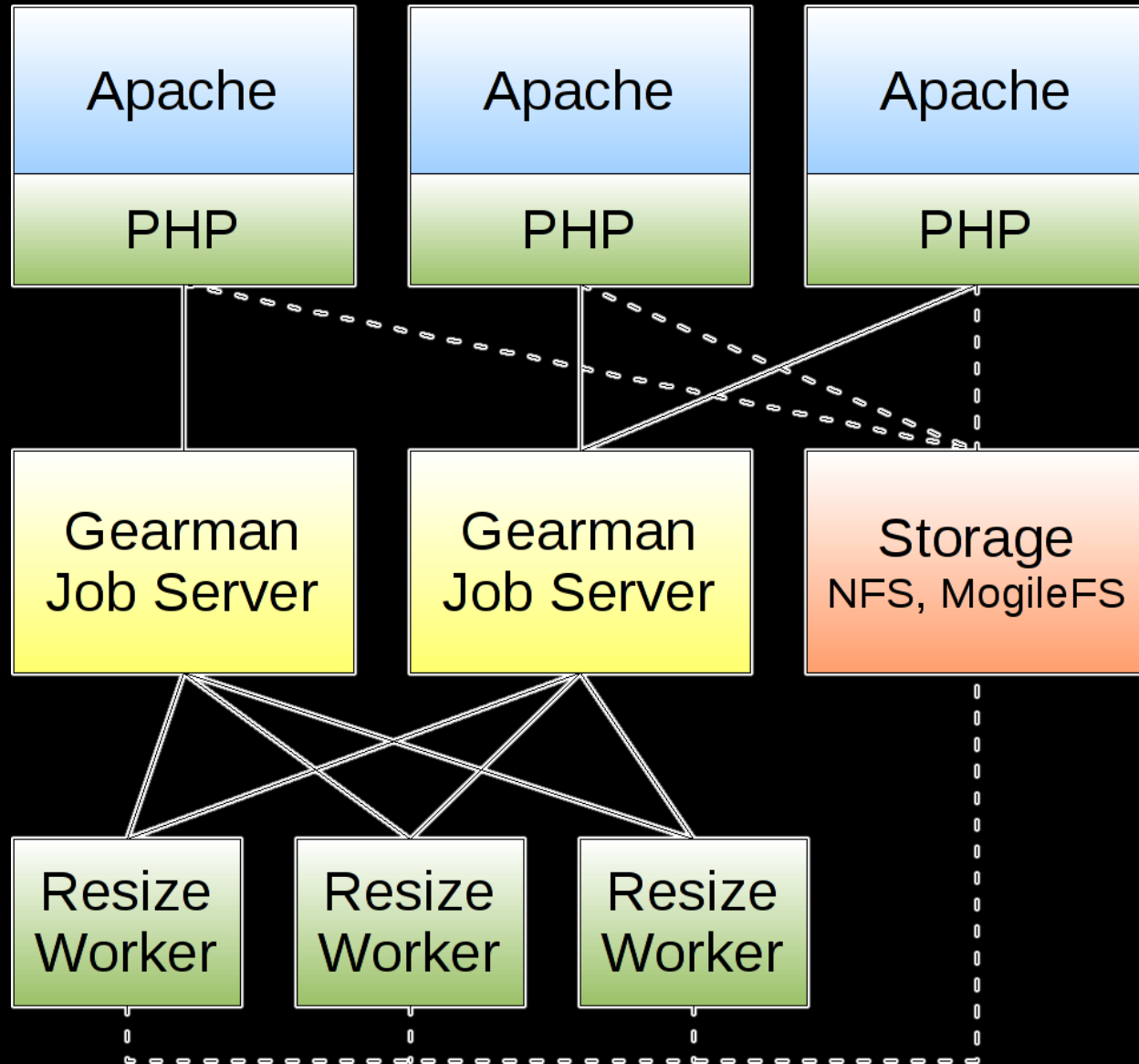


# Before





# After





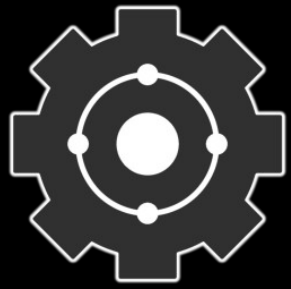
“The way I like to think of Gearman is as a massively distributed, massively fault tolerant fork mechanism.”

- Joe Stump, previously at Digg



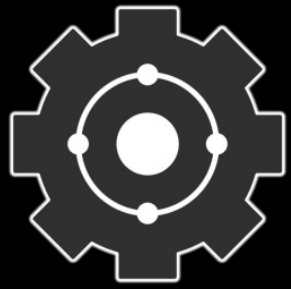
# Gearman Overview

- History
- Basics
- Distributed Processing
- Map/Reduce
- Asynchronous Queues
- Roadmap



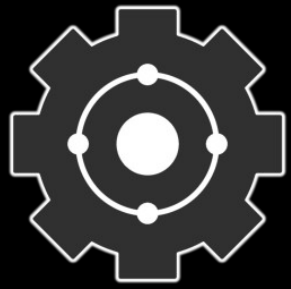
# History

- 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+ servers, 400K jobs/day
- Yahoo: 120+ servers, 12M jobs/day
- LiveJournal, SixApart, DealNews, xing.com, ...



# Recent Development

- Rewrite in C
- New Language Bindings
  - PHP/C, Perl/C, Drizzle, MySQL, PostgreSQL, Java, JMS, Python/C, Twisted Python, OCaml, ...
- Command line tool
- Multi-threaded (50k jobs/second)
- Persistent Queues
- Pluggable Protocol (HTTP)



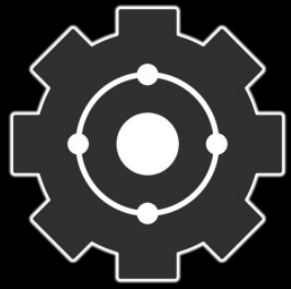
# Features

- **Open Source**
- **Simple & Fast**
- **Multi-language**
  - Mix clients and workers from different APIs
- **Flexible Application Design**
  - Not restricted to a single distributed model
- **Embeddable**
  - Small & lightweight for applications of all sizes
- **No Single Point of Failure**

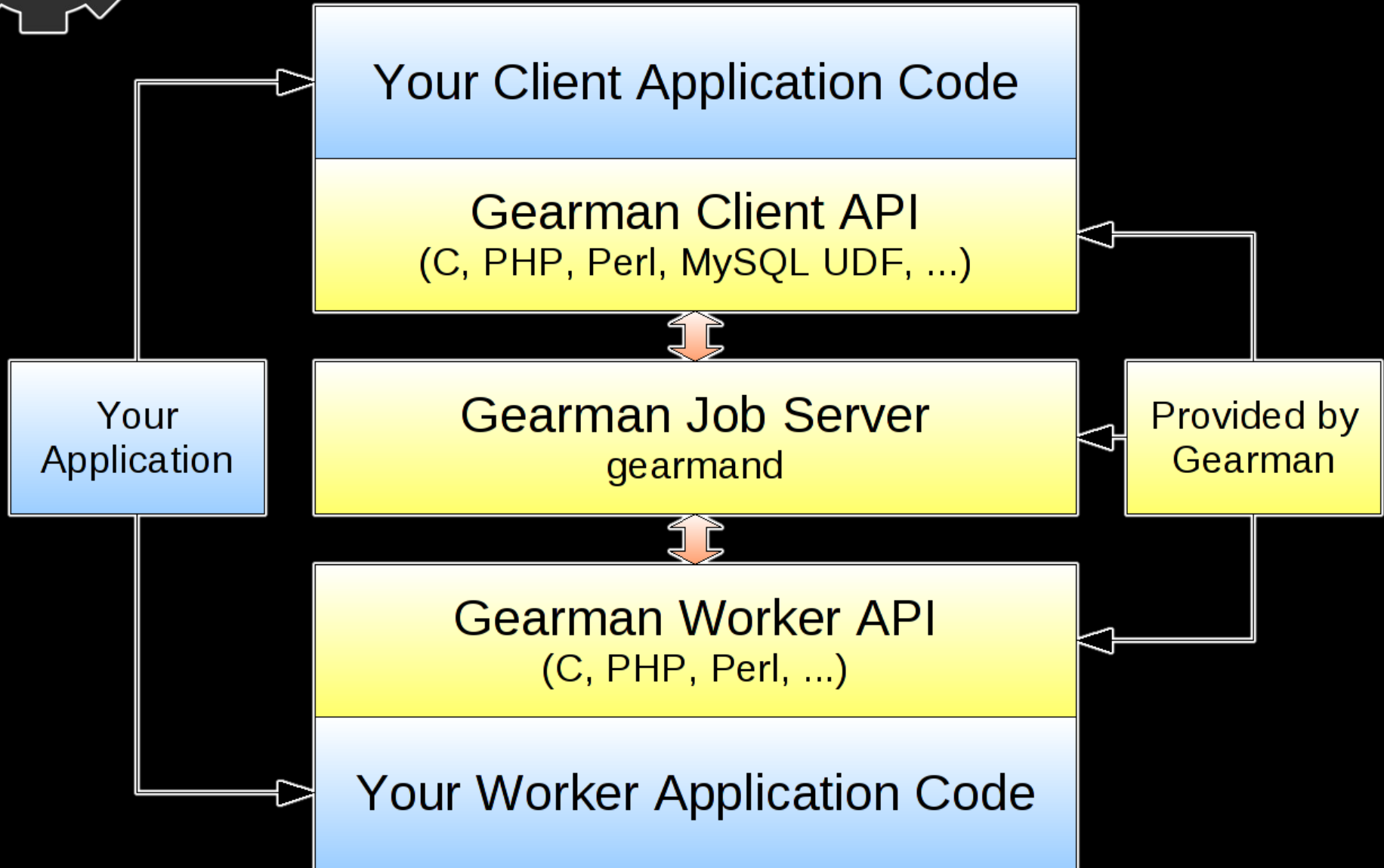


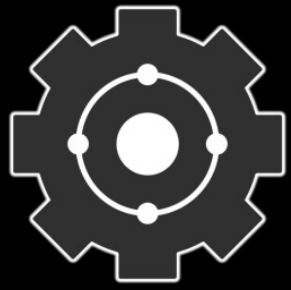
# Basics

- 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

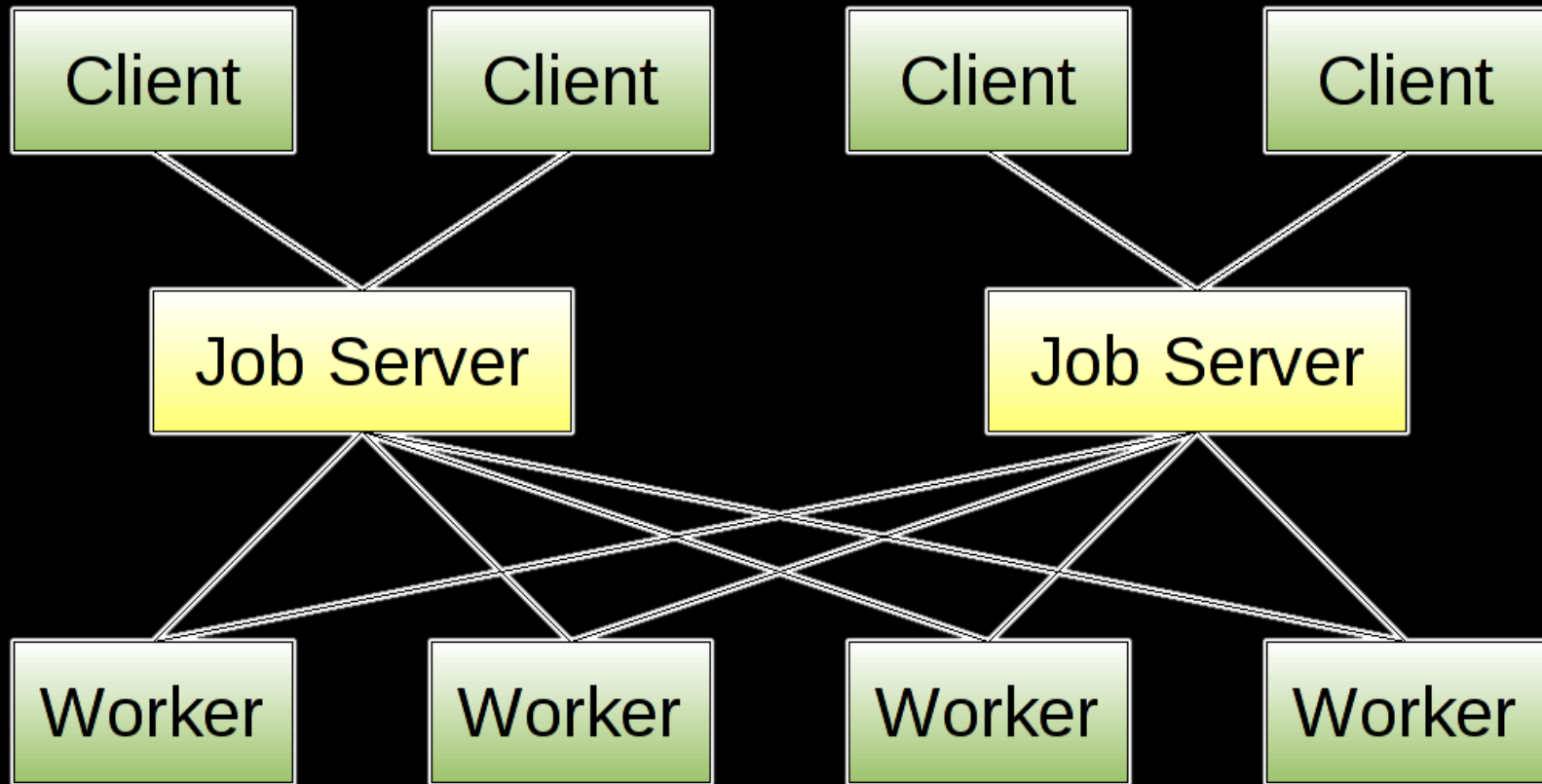


# Gearman Stack





# No Single Point of Failure

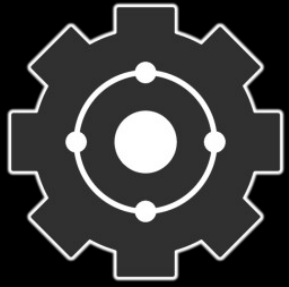




# Hello World

```
$client= new GearmanClient();  
$client->addServer('127.0.0.1');  
print $client->do('reverse', 'Hello World!');
```

```
$worker= new GearmanWorker();  
$worker->addServer('127.0.0.1');  
$worker->addFunction('reverse', 'my_reverse_function');  
while ($worker->work());  
  
function my_reverse_function($job)  
{  
    return strrev($job->workload());  
}
```



# Hello World

```
shell$ gearmand -d
```

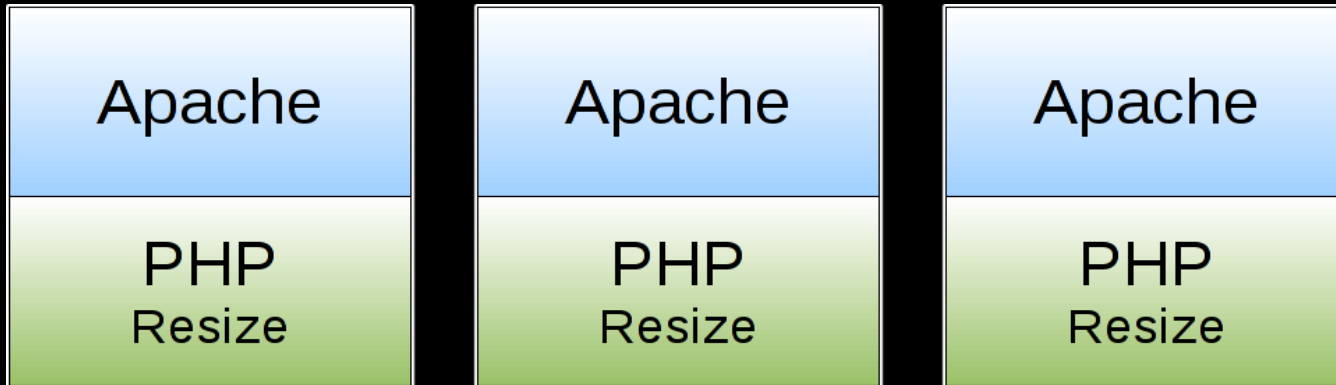
```
shell$ php worker.php &  
[1] 17510
```

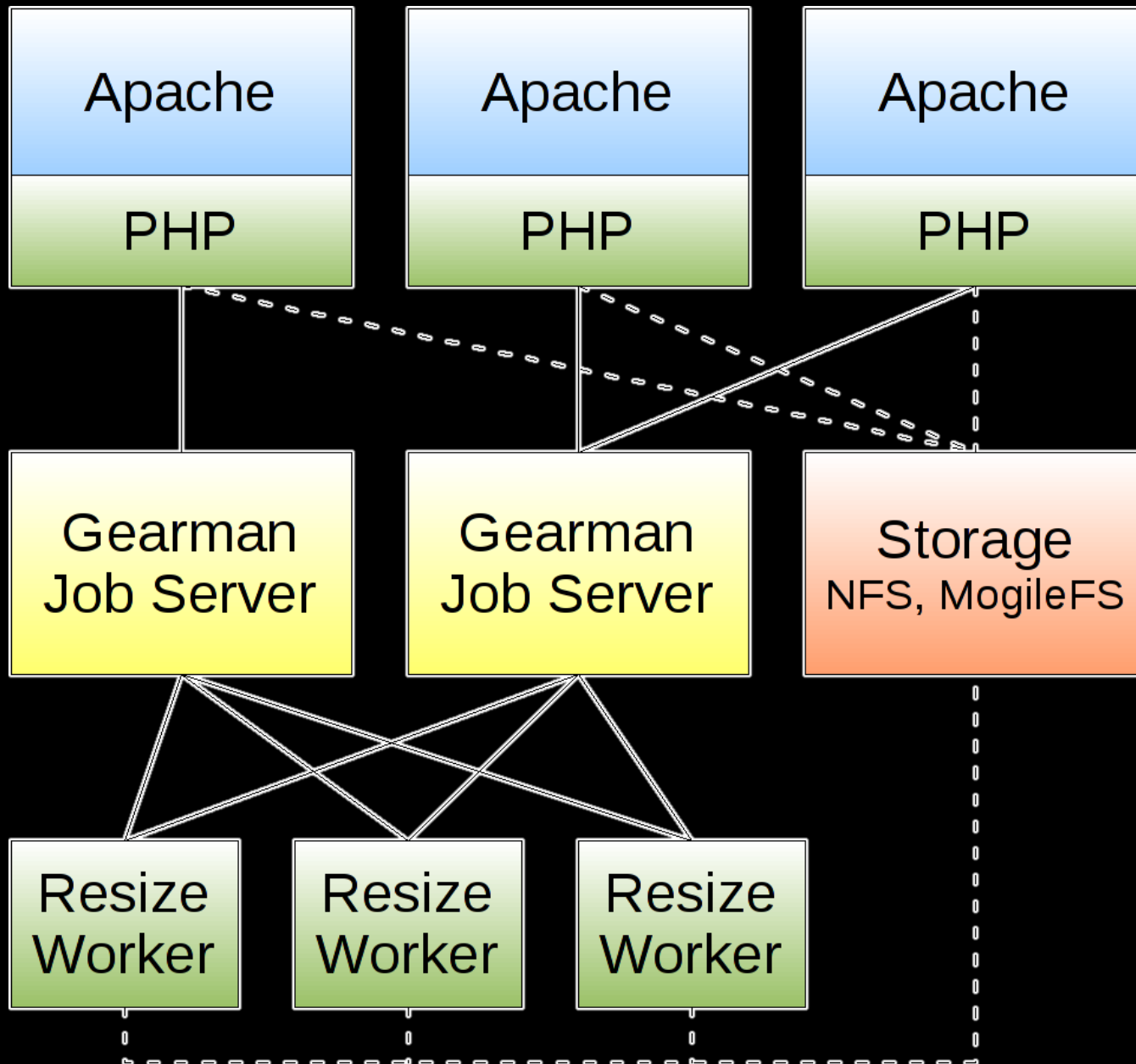
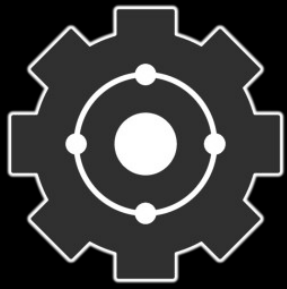
```
shell$ php client.php  
!dlrow olleH
```

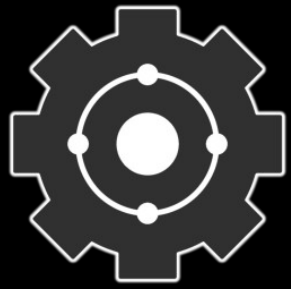


# Distributed Processing

(Back to image processing)



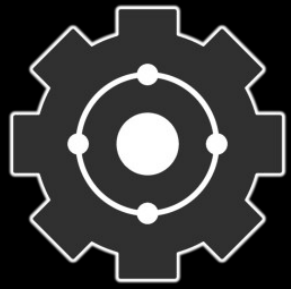




# Image Resize Worker

```
$worker= new GearmanWorker();
$worker->addServer('127.0.0.1');
$worker->addFunction('resize', 'my_resize_function');
while ($worker->work());

function my_resize_function($job)
{
    $thumb = new Imagick();
    $thumb->readImageBlob($job->workload());
    $thumb->scaleImage(200, 150);
    return $thumb->getImageBlob();
}
```



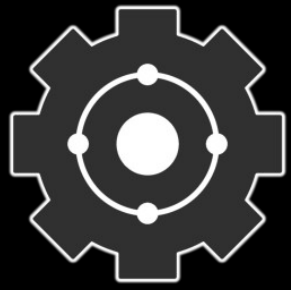
# Image Resize Worker

```
shell$ gearmand -d
```

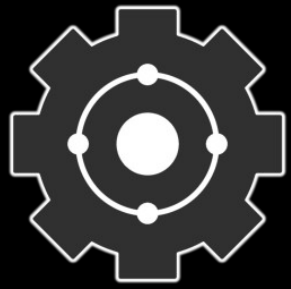
```
shell$ php resize.php &  
[1] 17524
```

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

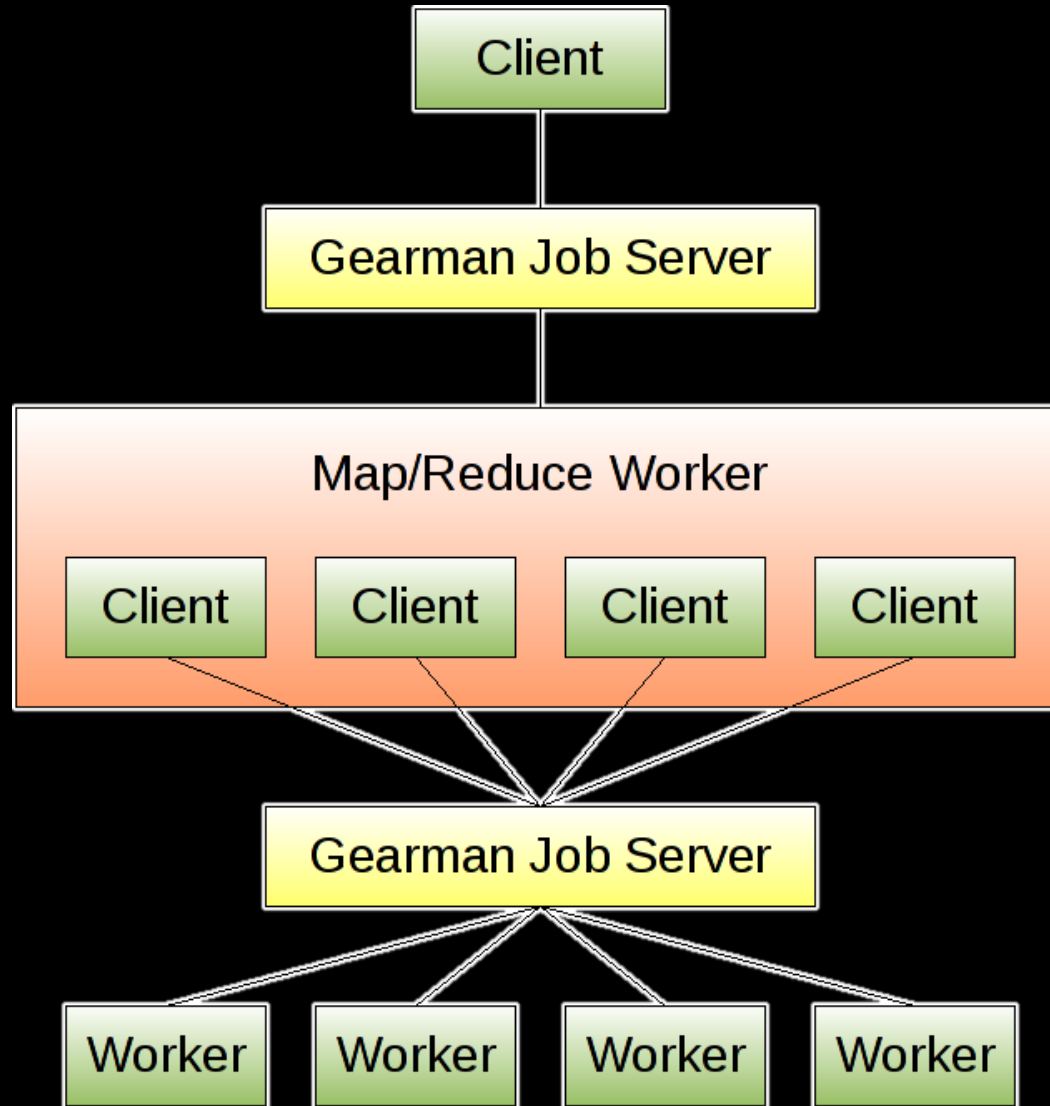
```
shell$ ls -sh large.jpg thumb.jpg  
3.0M large.jpg    32K thumb.jpg
```

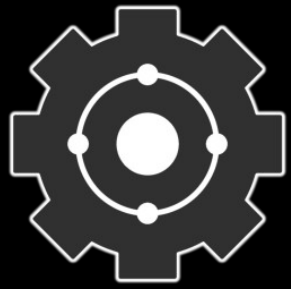


What else?



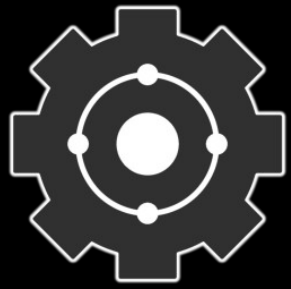
# Map/Reduce





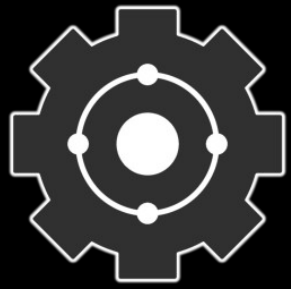
# Map/Reduce for Web

- Reduce page load time with concurrency
  - Gearman clients allow for concurrent jobs
- Push expensive operations off
  - Database queries
  - Expensive calculations (custom ad generation)
  - Data locality
  - Improved caching
- Start sending page back before blocking
  - Improve time to first byte



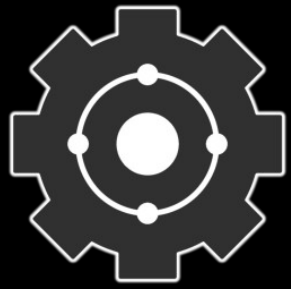
# Map/Reduce for Web

- Start request
  - Start non-blocking Gearman jobs
- Send first byte
- Block for required job result
  - Only block for results you need
  - Cache others until needed
  - Repeat
- Finish request



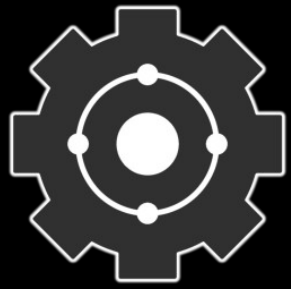
# Log Processing

- Bring Map/Reduce to Apache logs
- Get log storage off Apache nodes
- Push processing to log storage nodes
- Combine data in some meaningful way
  - Summary
  - Distributed merge-sort algorithms

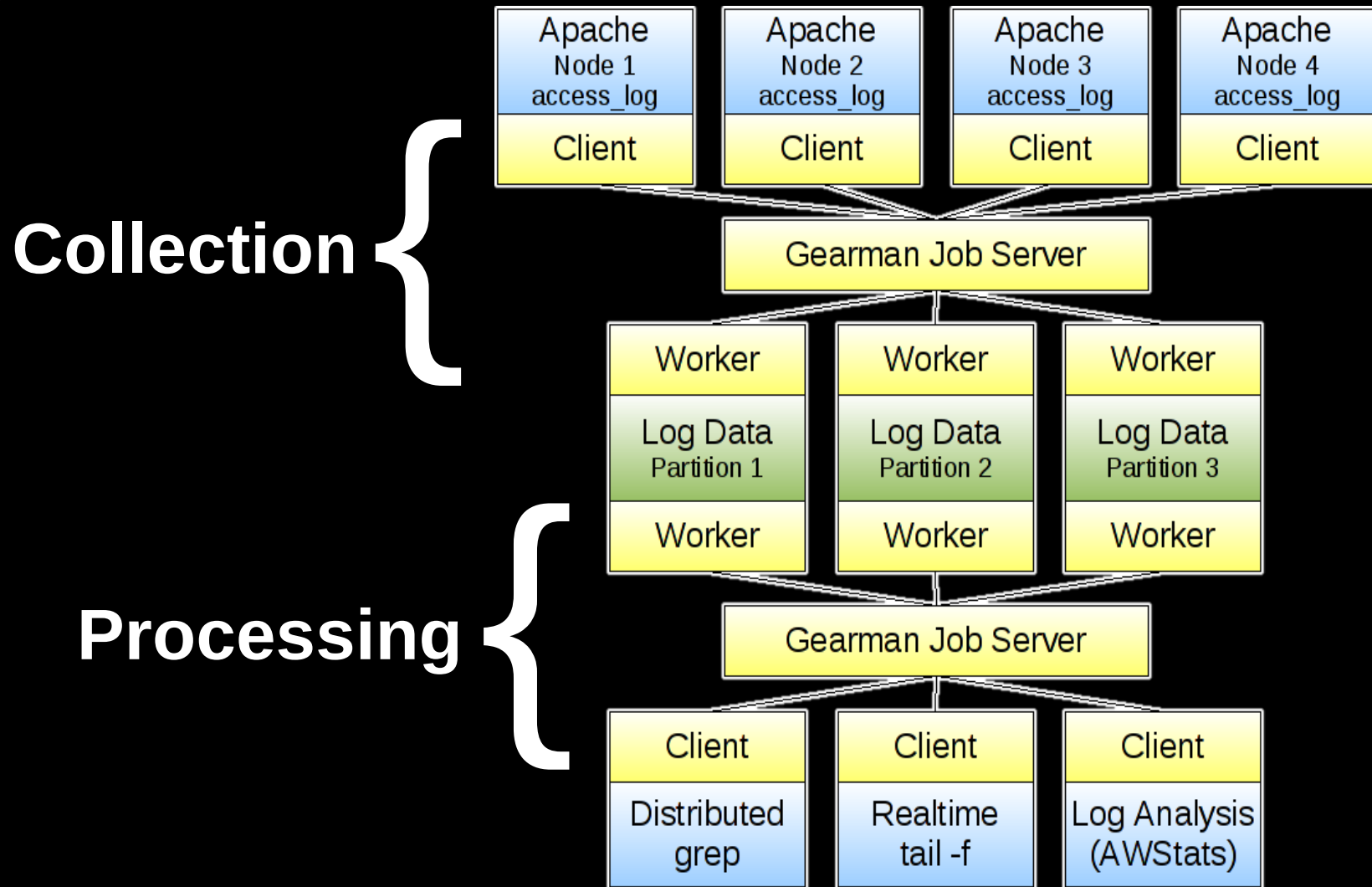


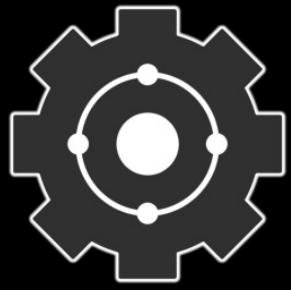
# Log Processing

- Collection
  - `tail -f access_log | gearman -n -f logger`
  - CustomLog "| gearman -n -f logger" common
  - Write a Gearman Apache logging module
- Processing
  - Distributed/parallel grep
  - Log Analysis (AWStats, Webalizer, ...)
  - Custom data mining & click analysis



# Log Processing



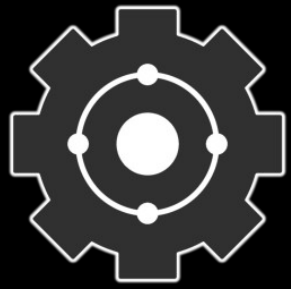


But wait,  
there's more!



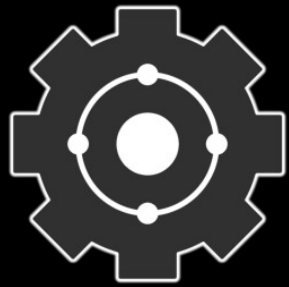
# Asynchronous Queues

- In Gearman, know as background tasks
- They help you scale
- Not everything needs immediate action
  - E-Mail notifications
  - Stat counters
  - Indexing
- Allows for batch operations

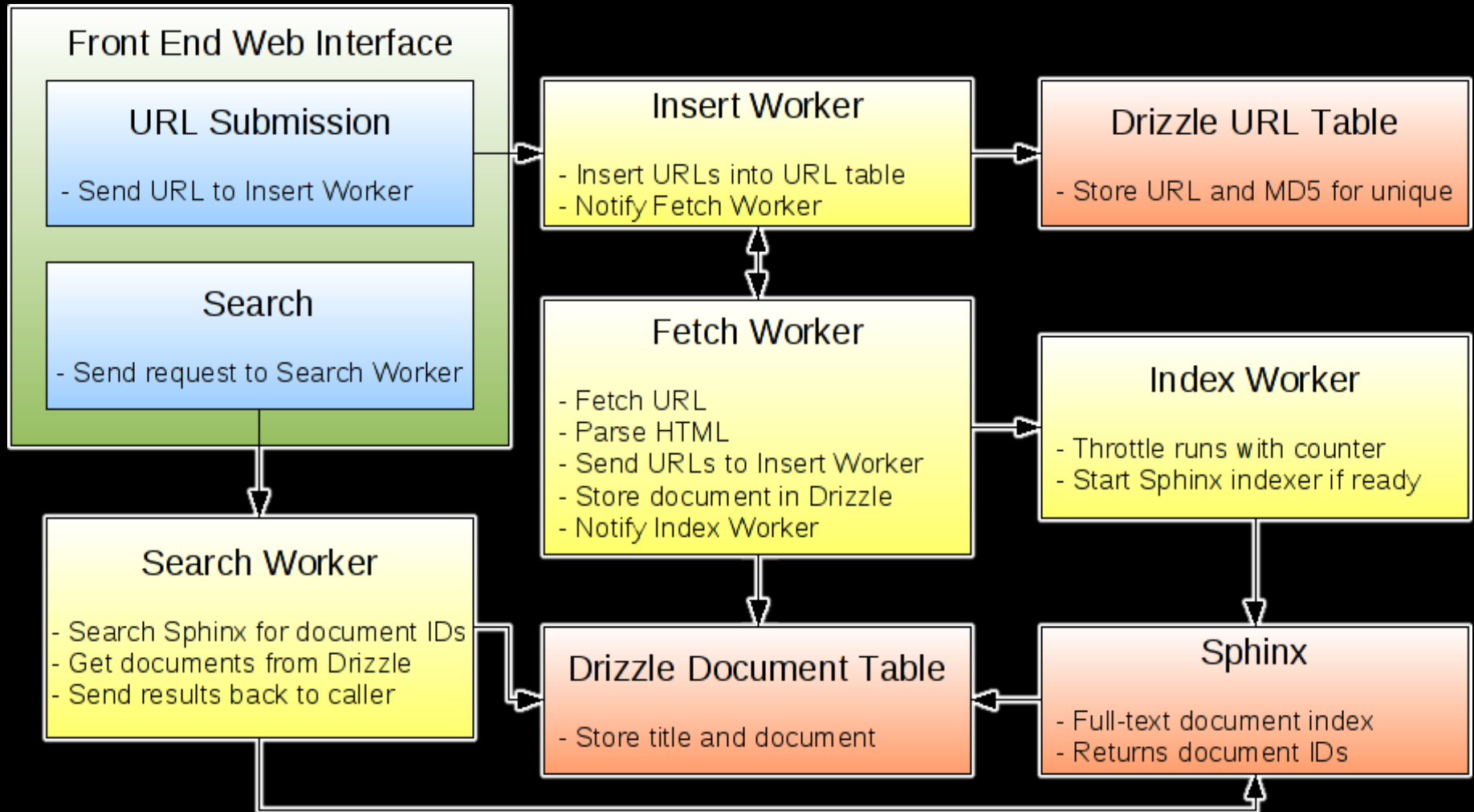


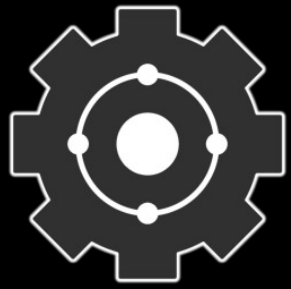
# Narada

- Example in book by Patrick Galbraith
- Custom search engine
- PHP and Perl implementations
- Pipeline of asynchronous queues
- Drizzle or MySQL
- Easy to integrate into existing projects
- <https://launchpad.net/narada>



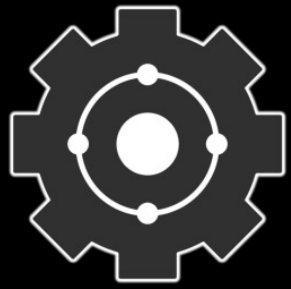
# Narada





# What's Next?

- Server rewrite in C++ for improved modularity
- Job result cache
- More protocols (memcached, XMPP, ...)
- TLS, SASL, multi-tenancy
- Replication
- Improved statistics reporting
- Event notification hooks



# Get Involved!

- <http://gearman.org/>
- #gearman on irc.freenode.net
- <http://groups.google.com/group/gearman>
- Gearman @ O'Reilly MySQL Conference
  - <http://en.oreilly.com/mysql2010/>
  - Visit the booth! (Gearman & Drizzle)