

A stylized gear icon with a central circle and three smaller circles around it, positioned to the left of the word "Gearman".

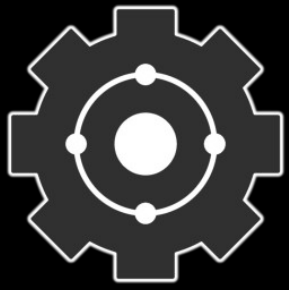
Gearman

Eric Day – <http://oddmments.org/>



Kittens!

(LiveJournal.com Image Processing)



Apache

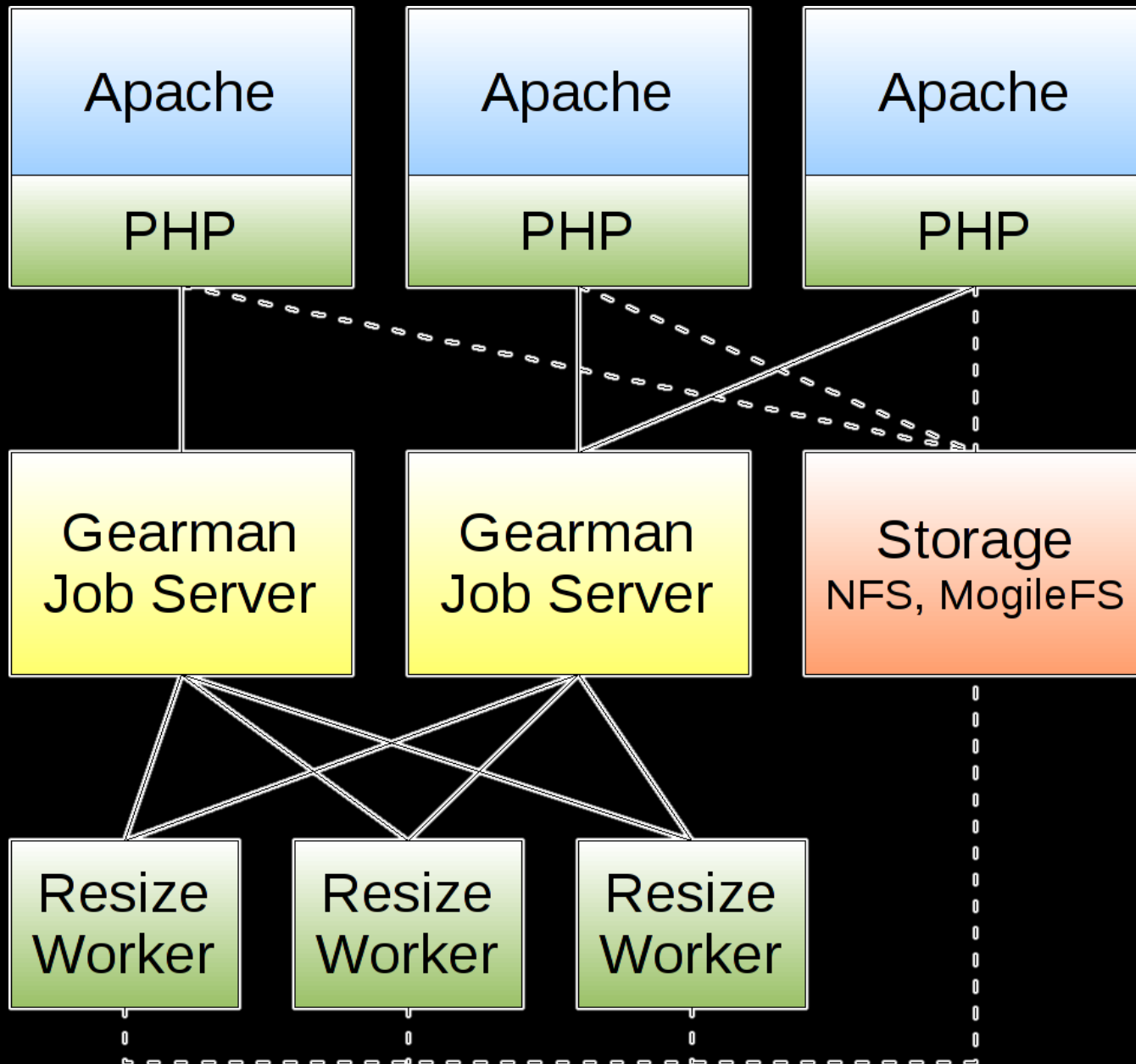
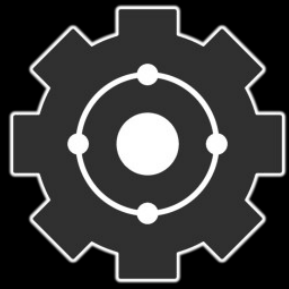
PHP
Resize

Apache

PHP
Resize

Apache

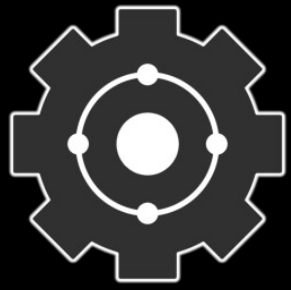
PHP
Resize





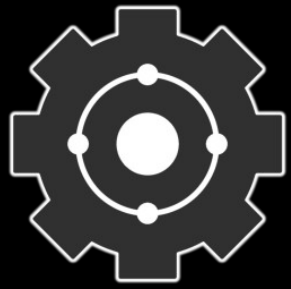
“The way I like to think of Gearman is as a massively distributed, massively fault tolerant fork mechanism.”

- Joe Stump, Digg



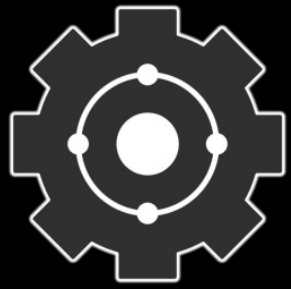
Gearman Overview

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



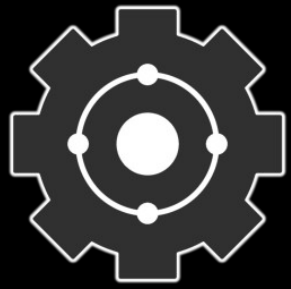
History

- 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
- Protocol Additions
- Multi-threaded (50k jobs/second)
- Persistent Queues
- Pluggable Protocol



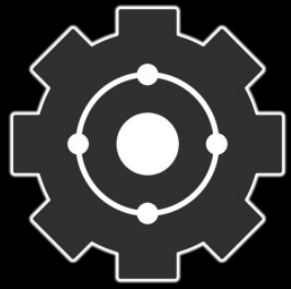
Features

- **Open Source (mostly BSD)**
- **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**

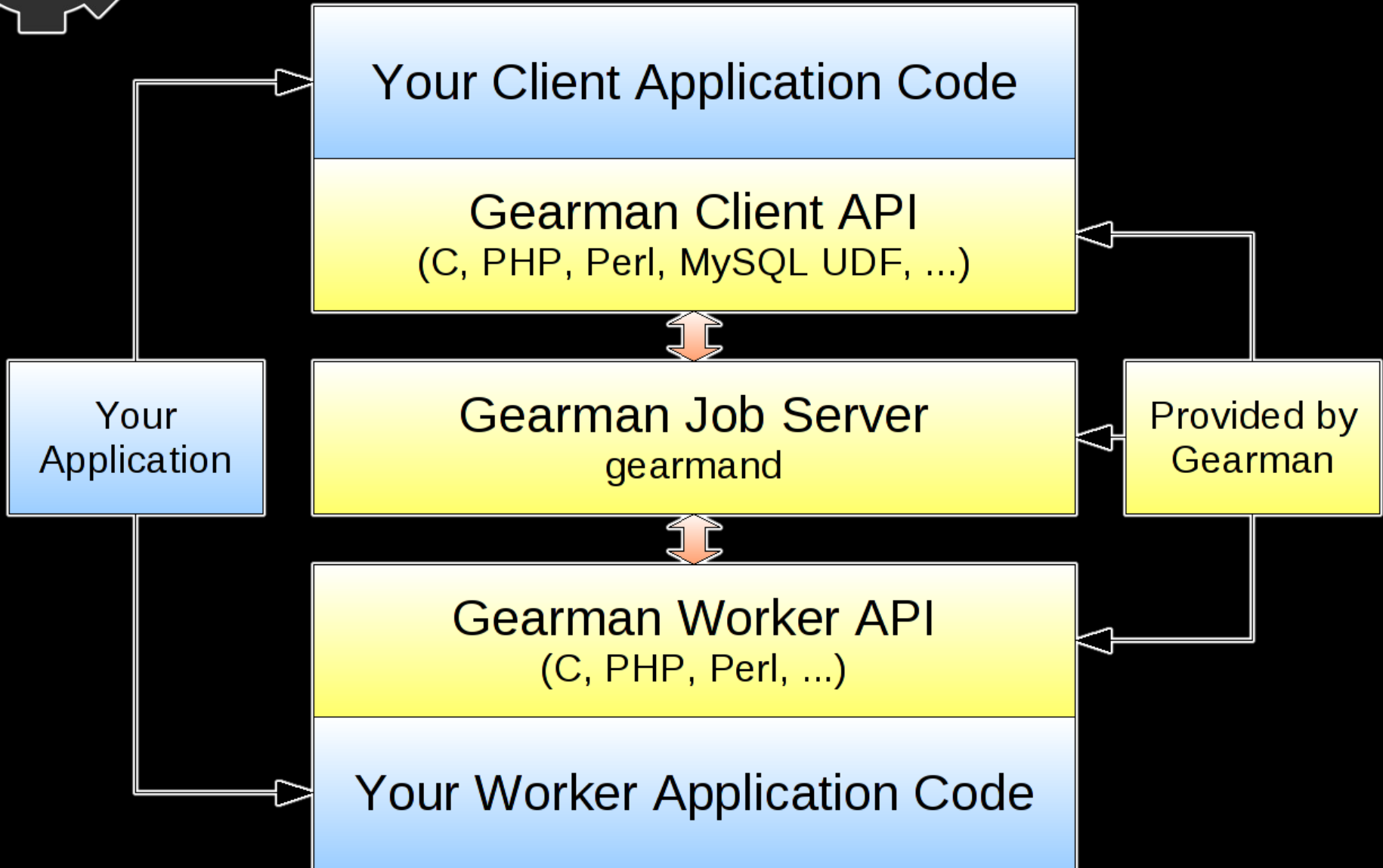


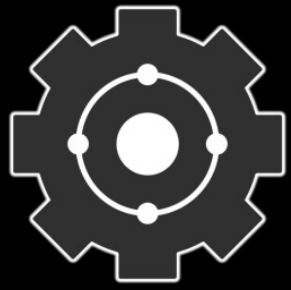
Basics

- 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

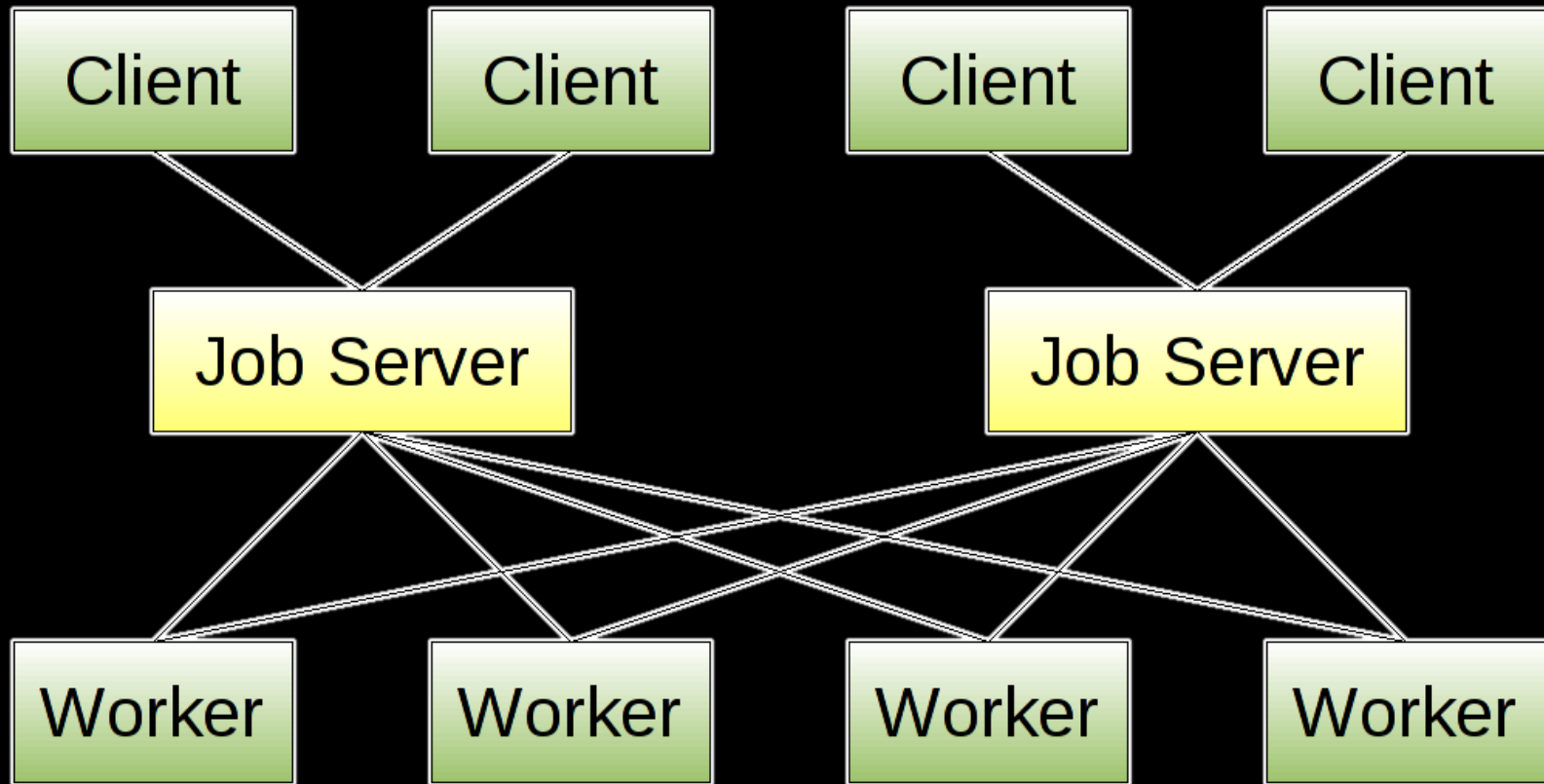


Gearman Stack





No Single Point of Failure

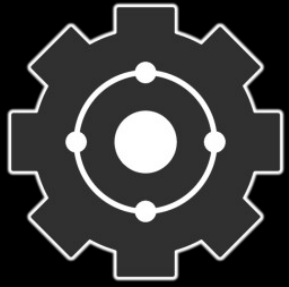




Hello World

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

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

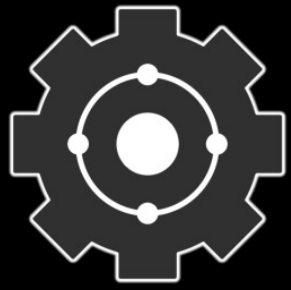


Hello World

```
shell$ gearmand -d
```

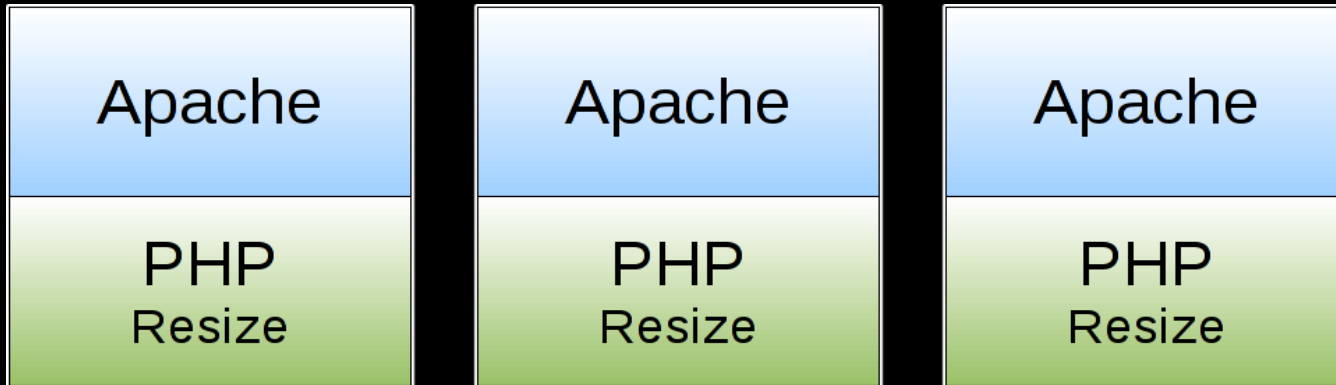
```
shell$ php worker.php &  
[1] 17510
```

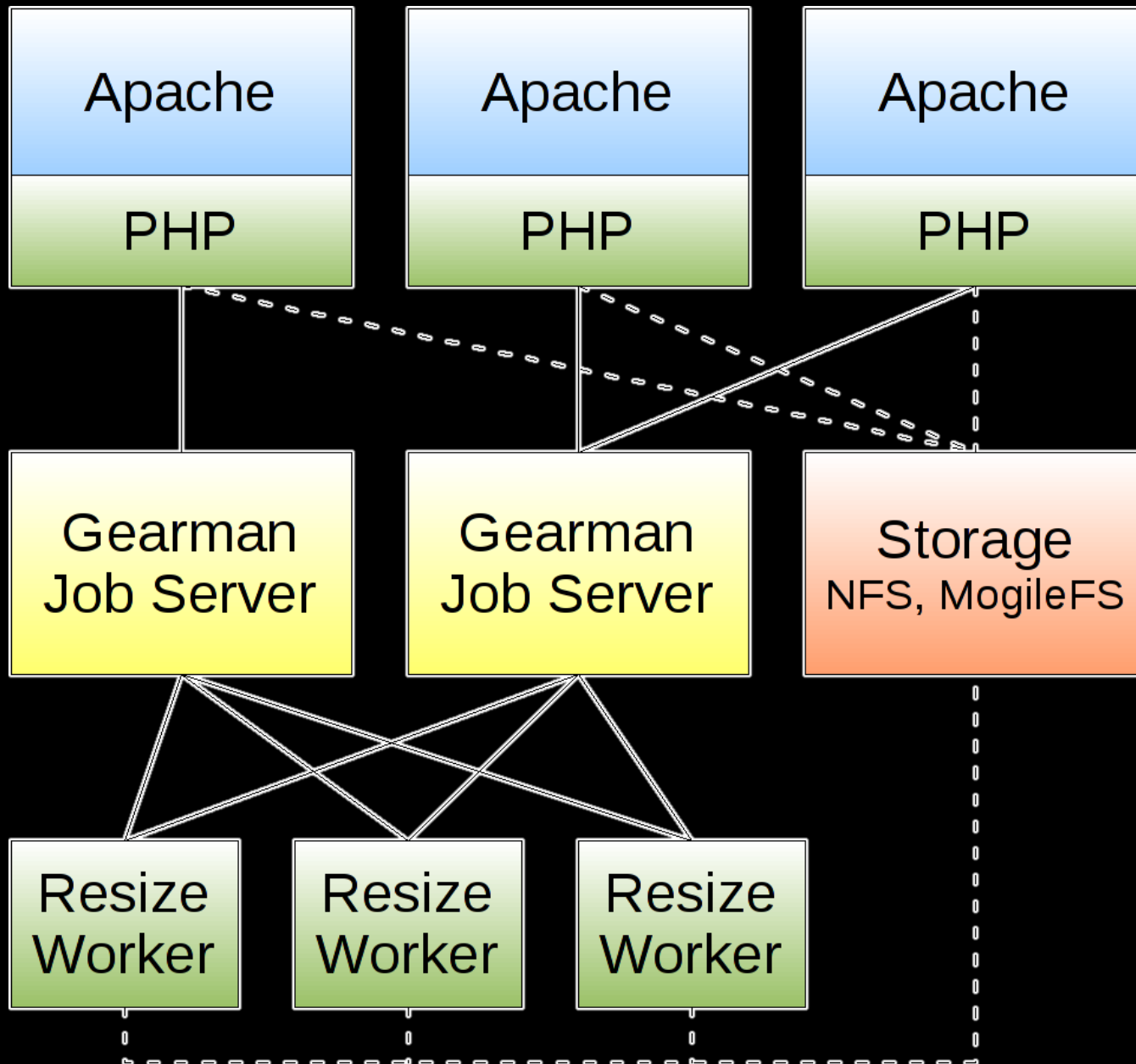
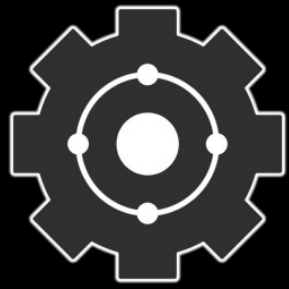
```
shell$ php client.php  
!dlrow olleH
```



Distributed Processing

(Back to the Kittens)





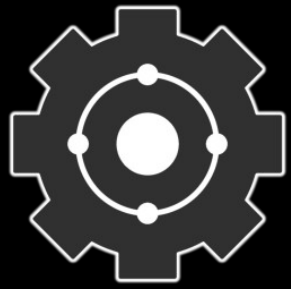


Image Resize Worker

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

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

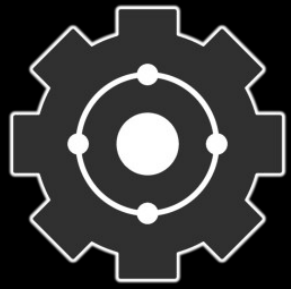


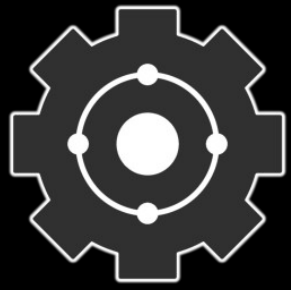
Image Resize Worker

```
shell$ gearmand -d
```

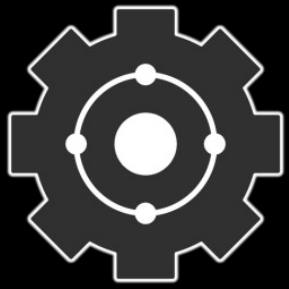
```
shell$ php resize.php &  
[1] 17524
```

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

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

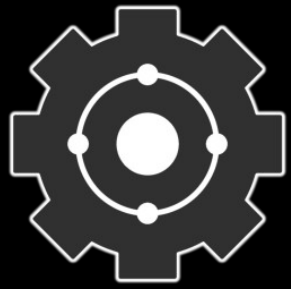


What else?



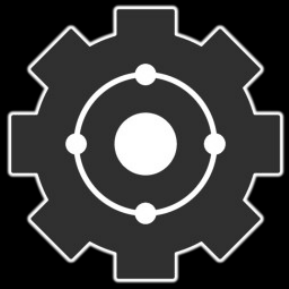
Web Pages

- Reduce page load time with concurrency
- Push expensive operations off
 - Database queries
 - Expensive calculations (ad placement)
 - Data locality
 - Improved caching
- Start sending page back before blocking
 - Improve time to first byte

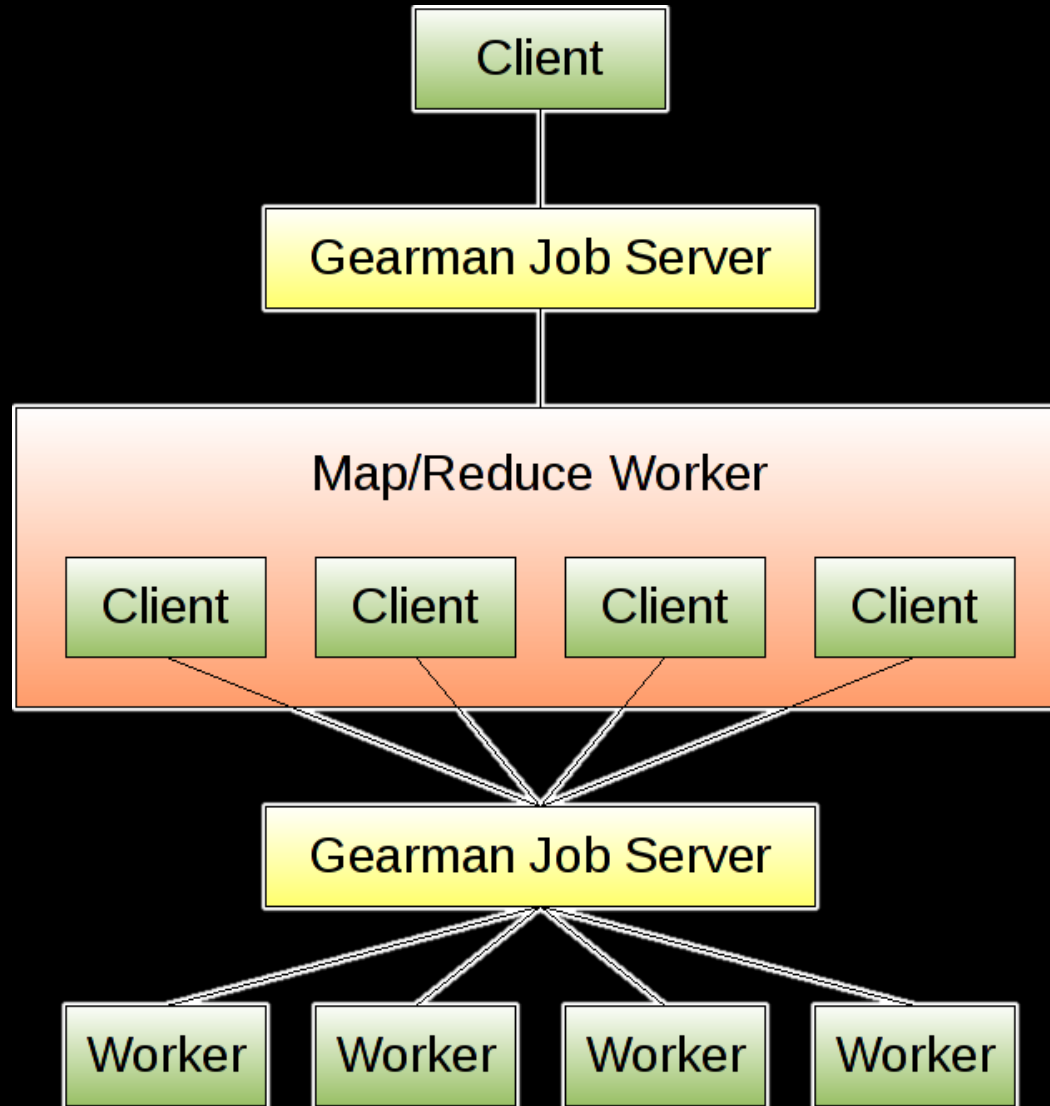


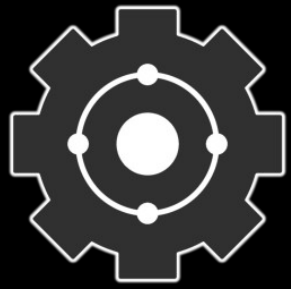
Web Pages

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



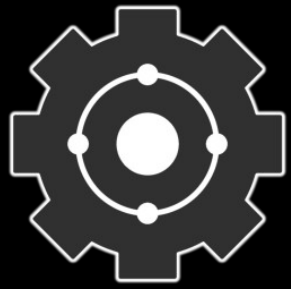
Map/Reduce





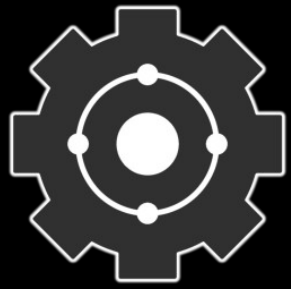
Log Processing

- 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

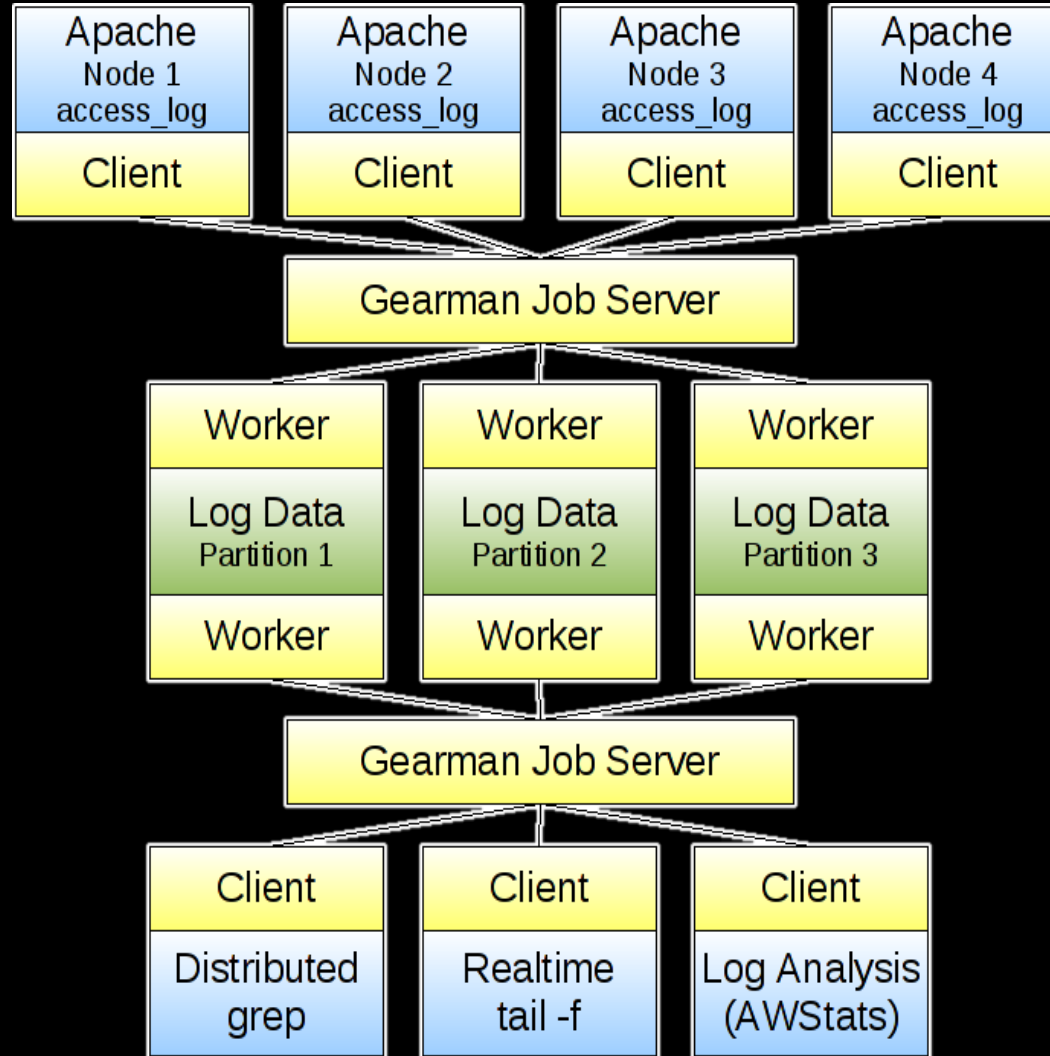


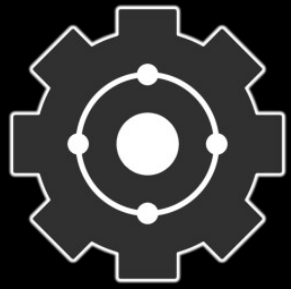
Log Processing

- 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

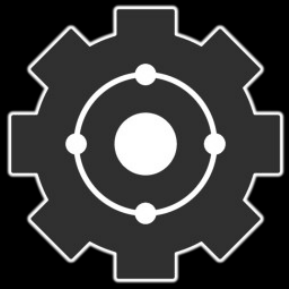




Log Processing

- Simple shell version
- Provide remote storage and distributed grep
- Send logs as mentioned before
 - `tail -f access_log | gearman -h host -f logger`
- On each log collection machine
 - `gearman -w -h host -f logger > log_file`
 - `gearman -w -h host -f logger1 ./dgrep.sh`

```
#!/bin/sh
read pattern
grep $pattern log_file
```



Log Processing

- Query logging machines
 - gearman -h host -f logger1 -f logger2 ... pattern

```
shell$ gearman -h host -f logger1 -f logger2 -f logger3 \  
news_archive | cut -f 4,5,7 -d' '
```

```
[06/Jul/2009:18:22:57 -0700] /bzt/narada/eday-dev/php/in  
dex.php?q=&u=http%3A%2F%2Flocalhost%2Fsrc%2Fnwveg%2Fnews  
_archive.php&s=Submit
```

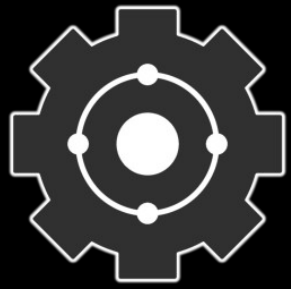
```
[06/Jul/2009:18:23:03 -0700] /src/nwveg/news_archive.php
```

```
[06/Jul/2009:18:23:54 -0700] /src/nwveg/news_archive.php
```

```
[06/Jul/2009:18:24:12 -0700] /src/nwveg/news_archive.php
```

```
[06/Jul/2009:18:31:37 -0700] /bzt/narada/eday-dev/php/in  
dex.php?q=news&s=Search&u=
```

```
...
```



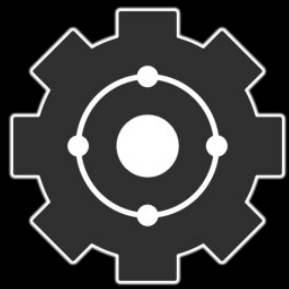
Asynchronous Queues

- Background Tasks
- They help you scale
- Distributed data storage
 - Eventually consistent data models
 - Choose “AP” in “CAP”, make EC work
- Not everything needs immediate action
 - E-Mail notifications, stat counters, indexing, ...
- Allows for batch operations

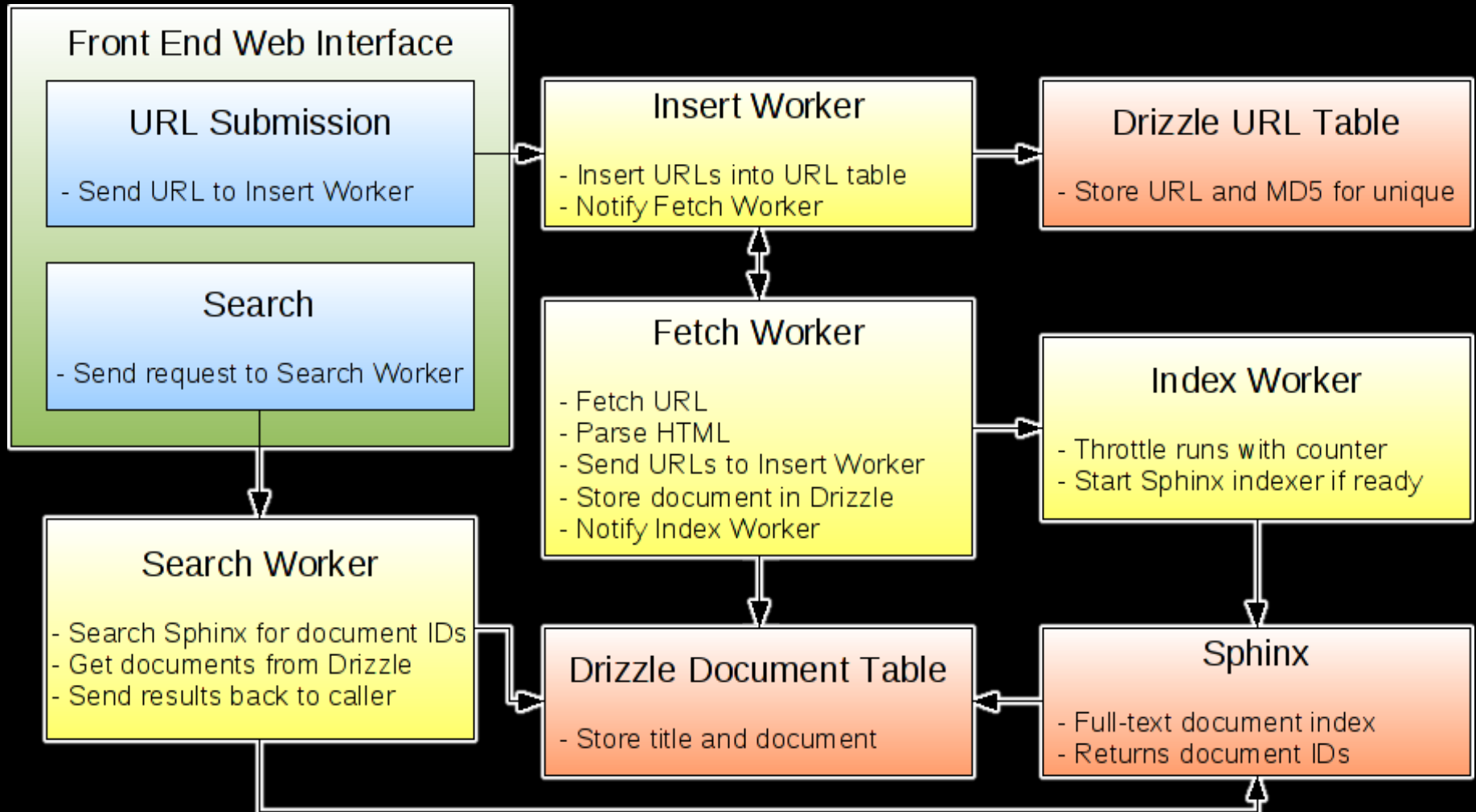


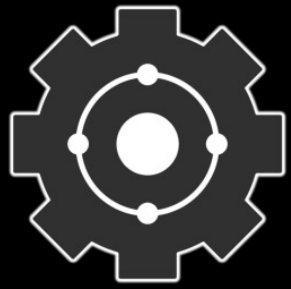
Narada

- Example in Patrick Galbraith's book
- Custom search engine
- PHP, Perl, and Java implementations
- Asynchronous queues
- Drizzle or MySQL
- Easy to integrate into existing projects
- <https://launchpad.net/narada>



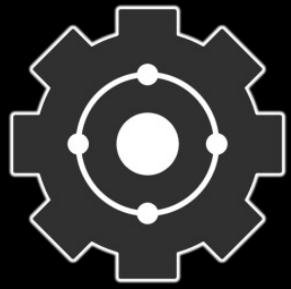
Narada





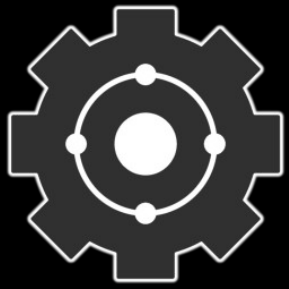
New Applications

- Think of scalable cloud architectures
- Not just LAMP on a virtual machine
- Accept new limitations of cloud
- Utilize power and flexibility cloud
- Elastic servers and services
- New data models



What's Next?

- Job result cache
- More protocols (memcached, XMPP, ...)
- TLS, SASL, multi-tenancy
- Replication
- More language interfaces
 - Suggestions?
- Improved statistics reporting
- Event notification hooks
- Monitor service



Get Involved!

- <http://gearman.org/>
- #gearman on irc.freenode.net
- <http://groups.google.com/group/gearman>
- OpenSQL Camp
 - All things open-source data related
 - <http://opensqlcamp.org/>