

# Entre na fila. Processamento distribuído usando Gearman

Elton Luís Minetto



5° SoLiSC.gz  
congresso catarinense  
de software livre  
solisc.org.br



# Quem sou eu?

- Graduado e pós-graduado em Ciência da Computação.
- Trabalha com PHP/MySQL desde 2000.
- Trabalha com Linux desde 1998
- Autor do livro Frameworks para Desenvolvimento em PHP - Editora Novatec e co-autor do livro Grid Computing in Research and Education - IBM Redbooks
- Membro do PHPSC
- Diretor de Desenvolvimento do Drimio e professor na Unochapecó(Chapecó/SC)
- Sócio da Coderockr

0 problema



A solução



# O que é

- É um framework genérico para distribuir jobs a serem executados por uma ou mais máquinas
- Permite uma aplicação executar tarefas em paralelo, com balanceamento da carga de processos, e até invocar códigos escritos em outras linguagens
- Pode ser usado em uma variedade enorme de aplicações. Desde sites de alto tráfego até manipulação de imagens e grande bancos de dados

# Quem criou

- A primeira implementação foi criada pela Danga Interactive, desenvolvedores do Memcached e MogileFS, para suportar sites de grande tráfego como LiveJournal
- O nome é um anagrama para "Manager"

# Vantagens

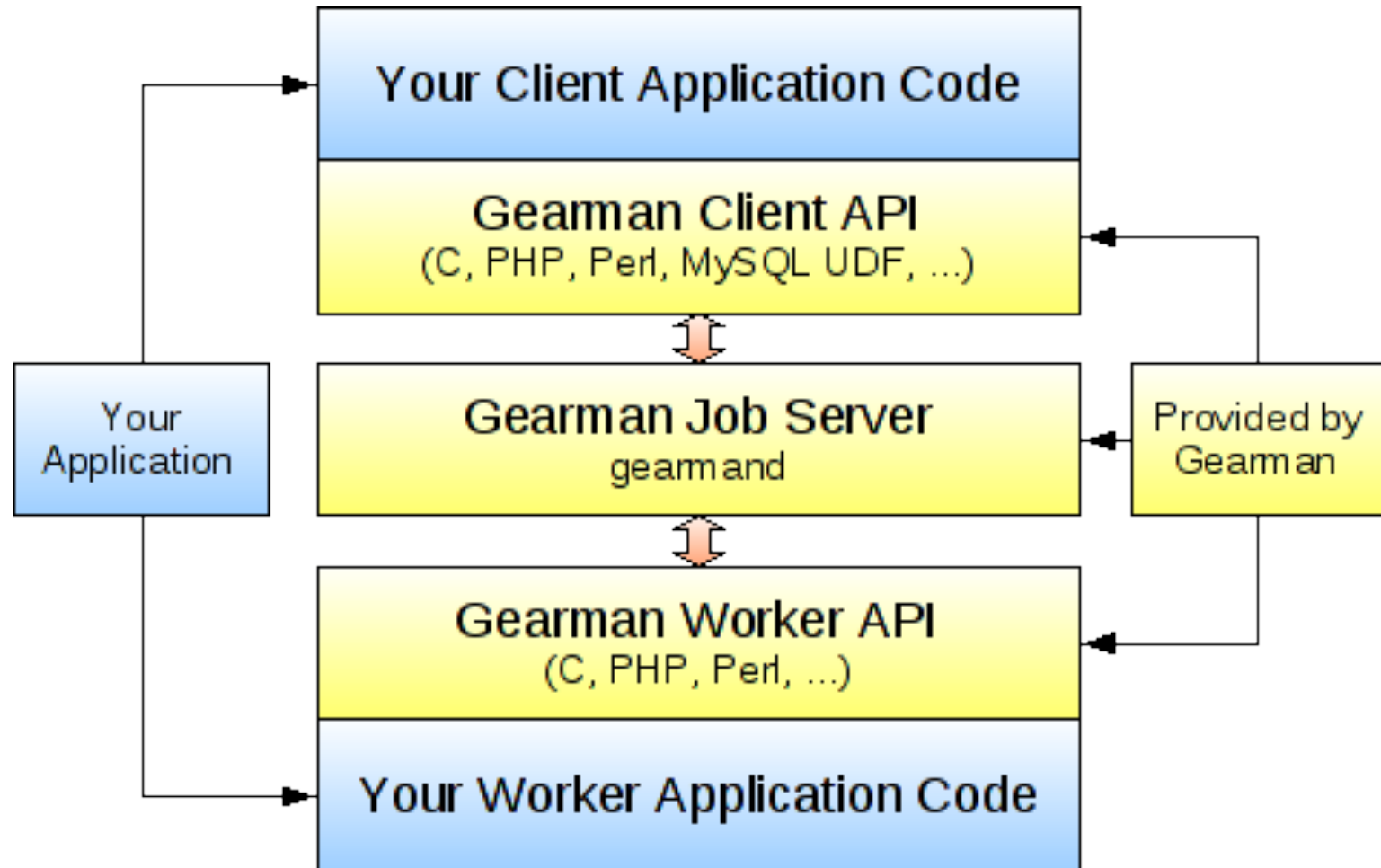
- Open Source - comunidade de desenvolvimento muito ativa
- Multi-language - Existem interfaces para desenvolver em várias linguagens (C, PHP, Perl, MySQL UDF, Python, etc). Clients podem ser em uma linguagem, workers em outra
- Rápido - possui protocolo e interfaces simples, com um servidor desenvolvido em C
- Sem "single point of failure" - é escalável e tolerante a falhas
  - Digg: +45 servers, 400mil jobs/dia
  - Yahoo: +60 servers, 6milhões jobs/dia
- Síncrono ou assíncrono - as tarefas podem ser executadas das duas formas

# Terminologia

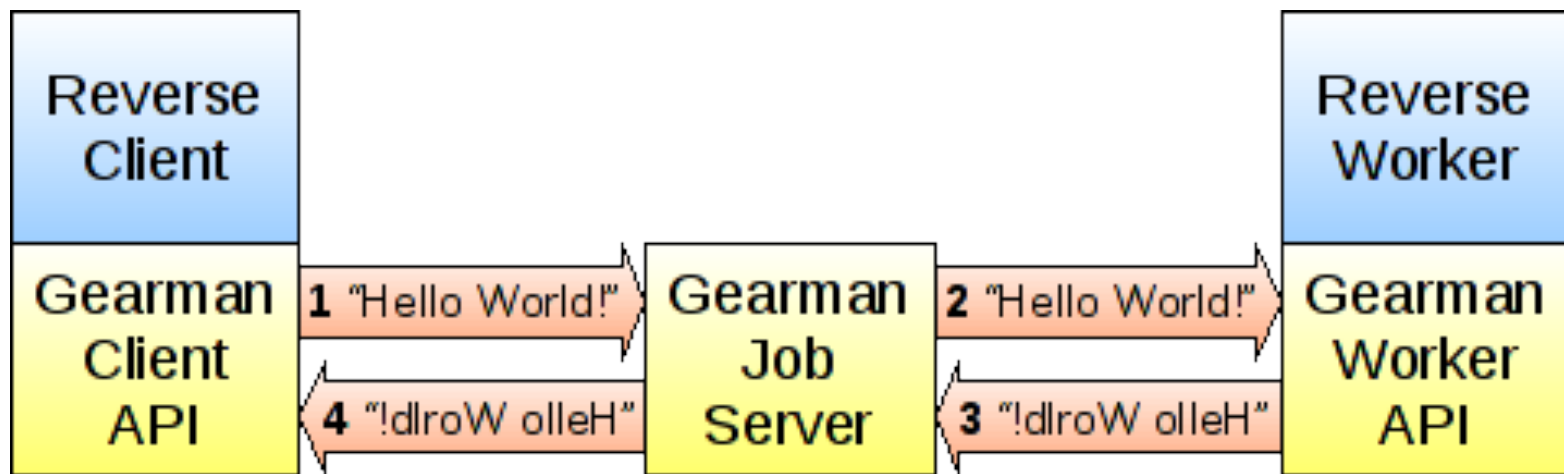
- Client- Cria um job a ser executado e o envia a um servidor de jobs
- Worker - Registra-se em um servidor de jobs e pega jobs para executar
- Job Server - Recebe os jobs e coordena a entrega de um job para um worker executar



# Arquitetura



# Fluxo



# Instalando

```
wget http://launchpad.net/gearmand/trunk/0.13/+download/gearmand-0.13.tar.gz  
tar xfvz gearmand-0.13.tar.gz  
cd gearmand-0.13  
./configure  
make  
sudo make install
```

# Iniciando o server

- **Simple:**

```
/usr/local/sbin/gearmand -d
```

- **Completo:**

```
/usr/local/sbin/gearmand -d -u <user> -L <host> -p 70
```

- **Sendo**

-d	Inicia como um daemon em background
-u <user>	Executa como o usuário específico
-L <host>	Fica ouvindo neste ip
-p <port>	Ouve na porta específica (default 4730)

# PHP

- Instalando a extensão do PHP:

```
wget http://pecl.php.net/get/gearman-0.7.0.tgz
```

```
tar xfvz gearman-0.7.0.tgz
```

```
cd gearman-0.7.0
```

```
phpize
```

```
./configure
```

```
make
```

```
sudo make install
```

- Adicionar no php.ini:

```
extension = gearman.so
```

# Síncrono X Assíncrono

- Síncrono, ou “em primeiro plano” (“foreground”). O cliente fica parado esperando o fim da execução da tarefa
- Assíncrono, ou “em segundo plano” (“background”). O cliente continua sua execução e o trabalho vai ser realizado por algum Worker. O cliente pode ser avisado do status por métodos e callbacks

# PHP - Exemplo: Cliente Síncrono

```
<?php
```

1. `$client= new GearmanClient();`
2. `/*indica em quais servidores o cliente pode enviar jobs. No caso somente o localhost, pois não é passado nada*/`
3. `$client->addServer();`
4. `/*adiciona um job síncrono, para a fila chamada title, enviando uma string como parâmetro*/`
5. `print $client->do("title", "o solisc é muito leGal");`
6. `print "\n";`
7. `?>`

# PHP - Exemplo: Worker Síncrono

```
<?php
1. $worker= new GearmanWorker();
2. $worker->addServer();//servidores q vai atender
3. /*registra a função title_function como responsável por
   atender a fila title*/
4. $worker->addFunction("title", "title_function");
5. while ($worker->work()); //fica em loop trabalhando
6.
7. //função que vai executar o trabalho
8. function title_function($job)
9. {
10.     return ucwords(strtolower($job->workload()));
11. }
12. ?>
```



# PHP - Exemplo: Cliente Assíncrono

```
<?php
1. $client= new GearmanClient();
2. $client->addServer();
3. for($i=0;$i<10;$i++) {
4.     /*adiciona um job assíncrono, para a fila
   chamada title, enviando um array como parâmetro. O
   cliente não fica esperando o final do processamento */
5.     $log['data'] = date('l jS \of F Y h:i:s A');
6.     $log['msg'] = "Teste de log $i";
7.     $client->doBackground("log_queue", serialize
   ($log));
8. }
9. ?>
```

# PHP - Exemplo: Worker Assíncrono

```
<?php
1. $worker= new GearmanWorker();
2. $worker->addServer();
3. //registra a função log_function como responsável por
   atender a fila log_queue
4. $worker->addFunction("log_queue", "log_function");
5. while ($worker->work()); //fica em loop trabalhando
6.
7. function log_function($job){
8.     $log = unserialize($job->workload());
9.     sleep(1); //sleep só para demonstrar executando
10.    echo $log['data'], ' - ', $log['msg'], "\n";
11.}
12. ?>
```

# Python - Exemplo: Worker Assíncrono

```
    from gearman import libgearman
1.
2. def worker_func(job):
3.     workload= job.get_workload()
4.     print workload
5.     return workload
6.
7. worker = libgearman.Worker()
8. worker.set_timeout(5000)
9. worker.add_server('localhost')
10. worker.add_function("log_queue", worker_func)
11. ret= libgearman.GEARMAN_SUCCESS
12. while 1:
13.     ret= worker.work()
```

# PHP - Job API

- `GearmanJob::sendStatus(int $numerator, int $denominator)`
- `GearmanJob::sendWarning(string $warning)`
- `GearmanJob::sendComplete(string $result)`
- `GearmanJob::sendFail(void)`
- `GearmanJob::sendException(string $exception)`

# PHP - Job API - Client

```
<?php
1. $client= new GearmanClient();
2. $client->addServer();
3. $handle = $client->doBackground("title", "o solisc");
4. if($client->returnCode() != GEARMAN_SUCCESS) {
5.     echo "Job com problemas";exit;
6. }
7. $done = false;
8. do {
9.     sleep(2);
10.  $stat = $client->jobStatus($handle);
11.  if(!$stat[0]) { $done = true; }//job terminou
12.  if($stat[1] == true)
13.      echo "Executando:", ' passo ' , $stat[2], ' de ',
        $stat[3], "\n";
14. }while(!$done);
15. ?>
```

# PHP - Job API - Worker

```
<?php
1. $worker= new GearmanWorker();
2. $worker->addServer();
3. $worker->addFunction("title", "title_function");
4. while ($worker->work());
5.
6. function title_function($job){
7.     $job->sendStatus(0, 2);
8.     sleep(3);
9.     $job->sendStatus(1, 2);
10.    sleep(3);
11.    $job->sendStatus(2, 2);
12.    return ucwords(strtolower($job->workload()));
13.}
14.?>
```

# PHP - Client API - Prioridades

```
//sincrono
```

1. `echo $client->do("title", "o solisc é muito leGal");`
2. `echo $client->doHigh("title", "o solisc é muito leGal");`
3. `echo $client->doLow("title", "o solisc é muito leGal");`
- 4.
5. `//assíncrono`
6. `$handle1 = $client->doBackground("title", "o solisc é muito leGal");`
7. `$handle1 = $client->doHighBackground("title", "o solisc é muito leGal");`
8. `$handle1 = $client->doLowBackground("title", "o solisc é muito leGal");`

# PHP - Callbacks

- É possível configurar funções para serem executadas em determinados eventos:
  - `GearmanClient::setDataCallback`
  - `GearmanClient::setCompleteCallback`
  - `GearmanClient::setCreatedCallback`
  - `GearmanClient::setExceptionCallback`
  - `GearmanClient::setFailCallback`
  - `GearmanClient::setStatusCallback`
  - `GearmanClient::setWorkloadCallback`



# Tasks, Jobs e Callbacks

```
<?php
1. $client= new GearmanClient();
2. $client->addServer();
3. //adicionando tarefas. Tmb podem ter prioridades
4. $client->addTask("title", "o solisc é muito leGal",
    null, '1');
5. $client->addTask("title", "tesTanD0", null, '2');
6. //configura o callback a ser executando
7. $client->setCompleteCallback("completo");
8. $client->runTasks(); //execute!!
9. function completo($task) {
10.     echo "Completo:", $task->unique(), ' ', $task->data
        (), "\n";
11. }
12. ?>
```

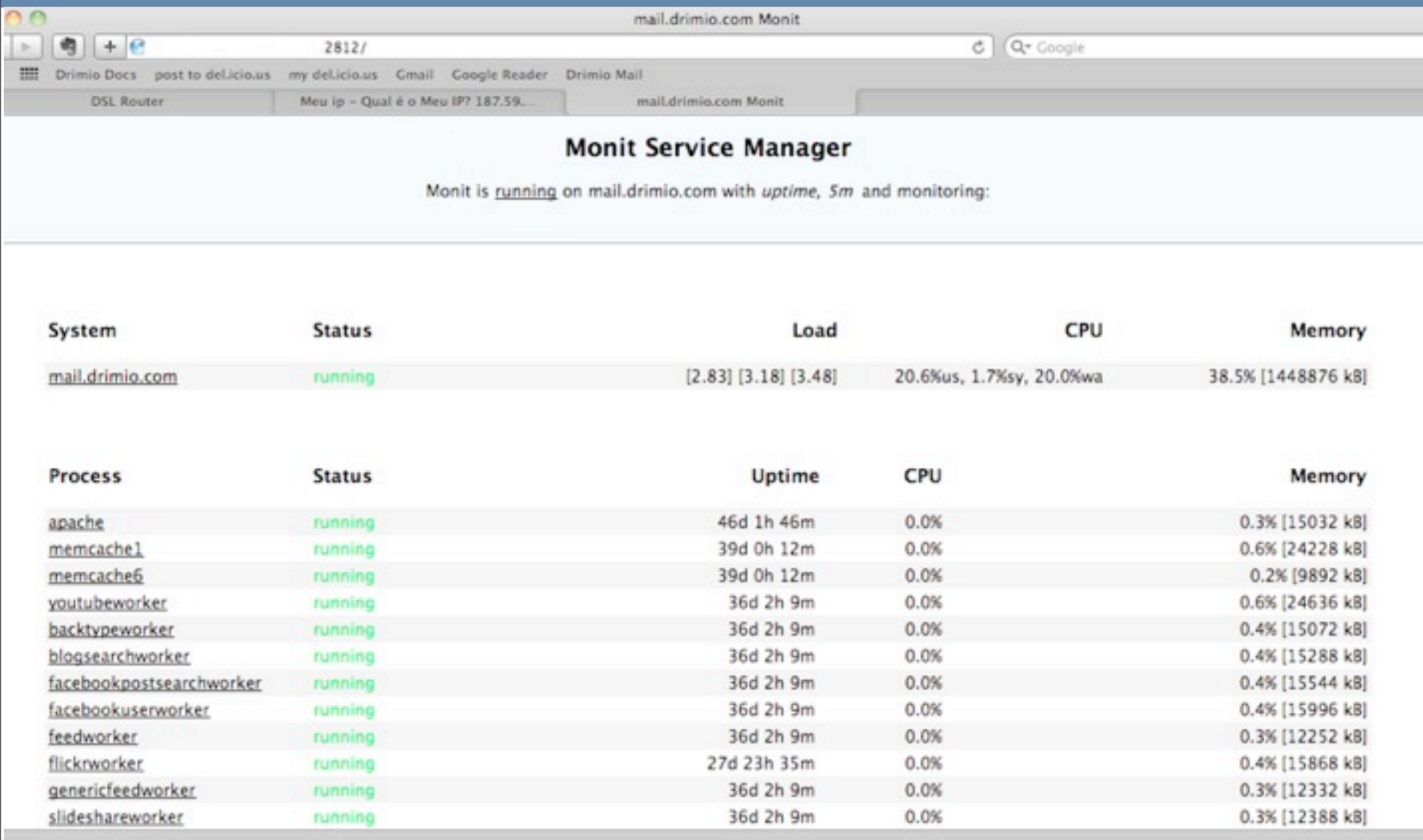
# Filas Persistentes

- Internamente todas as filas de jobs são armazenadas em memória. Se o servidor reinicia ou acontece algum erro todos os jobs pendentes são perdidos
- Pode-se usar filas persistentes para armazenar os jobs pendentes. Mas somente jobs em "background" ou assíncronos podem ser armazenados, pois os jobs síncronos são monitorados pelo cliente que o criou
- É possível armazenar a fila em:
  - MySQL (libdrizzle)
  - Memcached (libmemcached)
  - SQLite3 (libsqlite3)

# Monitorando - Telnet

```
telnet localhost 4730
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
status
jobLog      0      0      1
jobTwitter 0      0      1
jobTwitterUser  0      0      1
jobFacebookPostSearch 0      0      1
jobFeed     0      0      1
jobBlogsearch  0      0      1
jobFacebookUser  2      1      1
jobSlideshareUser 0      0      1
```

# Monitorando - Monit



mail.drimio.com Monit

2812/

Google

Drímio Docs post to del.icio.us my del.icio.us Gmail Google Reader Drímio Mail

DSL Router Meu ip - Qual é o Meu IP? 187.59... mail.drimio.com Monit

## Monit Service Manager

Monit is running on mail.drimio.com with *uptime*, *5m* and monitoring:

System	Status	Load	CPU	Memory
<a href="#">mail.drimio.com</a>	running	[2.83] [3.18] [3.48]	20.6%us, 1.7%sy, 20.0%wa	38.5% [1448876 kB]

Process	Status	Uptime	CPU	Memory
<a href="#">apache</a>	running	46d 1h 46m	0.0%	0.3% [15032 kB]
<a href="#">memcache1</a>	running	39d 0h 12m	0.0%	0.6% [24228 kB]
<a href="#">memcache6</a>	running	39d 0h 12m	0.0%	0.2% [9892 kB]
<a href="#">youtubeworker</a>	running	36d 2h 9m	0.0%	0.6% [24636 kB]
<a href="#">backtypeworker</a>	running	36d 2h 9m	0.0%	0.4% [15072 kB]
<a href="#">blogsearchworker</a>	running	36d 2h 9m	0.0%	0.4% [15288 kB]
<a href="#">facebookpostsearchworker</a>	running	36d 2h 9m	0.0%	0.4% [15544 kB]
<a href="#">facebookuserworker</a>	running	36d 2h 9m	0.0%	0.4% [15996 kB]
<a href="#">feedworker</a>	running	36d 2h 9m	0.0%	0.3% [12252 kB]
<a href="#">flickrworker</a>	running	27d 23h 35m	0.0%	0.4% [15868 kB]
<a href="#">genericfeedworker</a>	running	36d 2h 9m	0.0%	0.3% [12332 kB]
<a href="#">slideshareworker</a>	running	36d 2h 9m	0.0%	0.3% [12388 kB]

# Áreas de Aplicação

- Análise de logs
- Redimensionamento de imagens
- Processamento de URLs
- Atualização de cache
- Jobs de banco de dados
- etc

# Case Drimio

- Processamento de Logs
- Monitoramento de redes sociais (Twitter, Youtube, Flickr, Yahoo, Facebook, Feeds, Blogsearch, Slideshare, etc)
- Envio de e-mails
- Captura de snapshots de URLs
- Envio de tweets/posts Facebook
- Redimensionamento de imagens

# Case Drimio - Buscapé



1. Faz pesquisa

5. Página de resultados



2. Chamada de API



3. Resultado

4. Jobs para processar imagens e para Log de acesso



Imagens processadas e salvas em Memcached e Logs em MySQL



# Referências

- <http://www.phpclasses.org/blog/post/108-Distributing-PHP-processing-with-Gearman.html>
- <http://www.ibm.com/developerworks/br/opensource/library/os-php-gearman/?ca=drs->
- [http://gearman.org/index.php?id=manual:job\\_server#persistent\\_queues](http://gearman.org/index.php?id=manual:job_server#persistent_queues)
- <http://www.slideshare.net/felixdv/high-gear-php-with-gearman>
- <http://toys.lerdorf.com/archives/51-Playing-with-Gearman.html>
- <http://www.oreillyn.com/pub/e/1564>
- <http://weierophinney.net/matthew/archives/240-Writing-Gearman-Workers-in-PHP.html#extended>
- <http://datacharmer.blogspot.com/2009/11/gearman-for-mysql.html>
- <http://www.daviddemartini.com/archives/1405>
- <http://mmonit.com/monit/>



# Perguntas



# Contato

```
<?php
$card = array(
    'nome' => 'Elton Luís Minetto',
    'site' => 'http://www.eltonminetto.net',
    'e-mail' => 'eminetto@coderockr.com',
    'twitter' => '@eminetto'
);
var_dump($card);
?>
```