

Ao infinito e além com PHP, Memcached e Gearman

Elton Luís Minetto



Quem sou eu?

- Graduado e pós-graduado em Ciência da Computação.
- Trabalha com PHP/MySQL/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
- Professor na Unochapecó(Chapecó/SC)
- Sócio da Coderockr

O problema



O problema



O problema



O problema



O problema

- Performance: a habilidade que uma aplicação tem de atingir um objetivo, como por exemplo responder no menor tempo possível
- Capacidade: a carga total que uma aplicação pode suportar
- Escalabilidade: a habilidade de uma aplicação manter a performance quando a carga de trabalho aumenta. É a junção da capacidade e da performance
 - “Performance is a problem. Scaling your performance is a bigger problem”

Distribua!

Distribua!

Memória



Distribua!

Trabalho



Distribua!




GLAMMP

- Gearman
- Linux
- Apache
- MySQL
- Memcached
- Php (ou Perl ou Python)



O que é?

- Sistema distribuído e de alta performance para fazer cache de objetos em memória RAM.
- Genérico por natureza mas muito usado para acelerar aplicações web dinâmicas, reduzindo a carga de bases de dados, sessões de usuários, arquivos CSS.
- Criado pela  DANGA
INTERACTIVE

Por que usar?

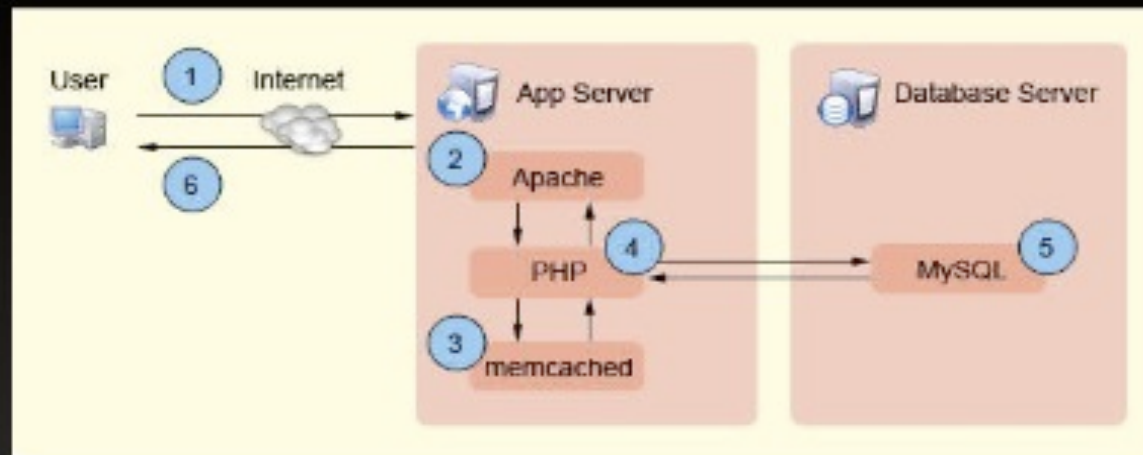
- Escalável: fácil adicionar máquinas e instâncias
- RAM é muito mais rápido que I/O em disco
- Alivia a carga do banco de dados
- Muito flexível: pode armazenar qualquer dado, desde que não ultrapasse 1 MB
- Bibliotecas client disponíveis em diversas linguagens (PHP, C, Java, Ruby, Python)
- Open Source

Como funciona?

- Armazena qualquer coisa que pode ser serializado
- Armazena com uma chave única (255 caracteres) e tempo de validade
- O programador usa da seguinte forma :
 - Encontrou no cache? Retorna os dados
 - Não encontrou no cache? Processa, armazena no cache e retorna o dado

Como funciona?

How memcached Works



1. User requests web page
2. Web server passes request to PHP
3. PHP checks if object is in memcached
4. If yes, build page with object
5. If no, PHP gets object from DB & stores in memcached
6. Send page to user

O que armazenar?

- Resultados de consultas SQL
- Páginas inteiras
- Sessões de usuários
- Fragmentos de HTML gerados
- Imagens (thumbnails)
- Arquivos css
- Arquivos js
- “Qualquer coisa que demore um pouco para ser gerada”

Quem usa?

- Facebook
 - 805 servidores 8-core. 28TB de RAM. 300K req/seg (Set. 2008)
 - Desenvolveu fork com suporte a UDP
 - <http://github.com/fbmarc/facebook-memcached/tree/master>
- Flickr
 - 38K req/seg. 12 milhões de objetos (Set. 2008)
- Digg
- Drimio
 - Sessões de usuários, imagens, arquivos CSS, arquivos JS, resultados de consultas SQL

Como usar?

Baixar código em <http://www.danga.com/memcached/>
e...

```
tar -xvzf memcached-X.Y.Z.tar.gz  
cd memcached-X.Y.Z
```

```
./configure  
make  
sudo make install
```



Santíssima Trindade

Como usar?

- No exemplo abaixo iniciamos o daemon com 128MB RAM, ouvindo na porta 11211 no ip 127.0.0.1

```
/usr/local/bin/memcached -d -m 128 -l 127.0.0.1 -p 11211
```

- Pode-se criar quantas instâncias forem necessárias, mudando a porta. Assim aumenta-se o espaço total do cache

Como usar com PHP?

- Download do módulo em <http://pecl.php.net/package/memcache>.
- e..

```
tar -xvzf memcache-2.2.5.tgz  
cd memcache-2.2.5  
phpize  
./configure  
make  
sudo make install (executar como root)
```
- Adicionar a linha abaixo no `php.ini` e reiniciar o Apache
`extension=memcache.so`

Como usar com PHP?

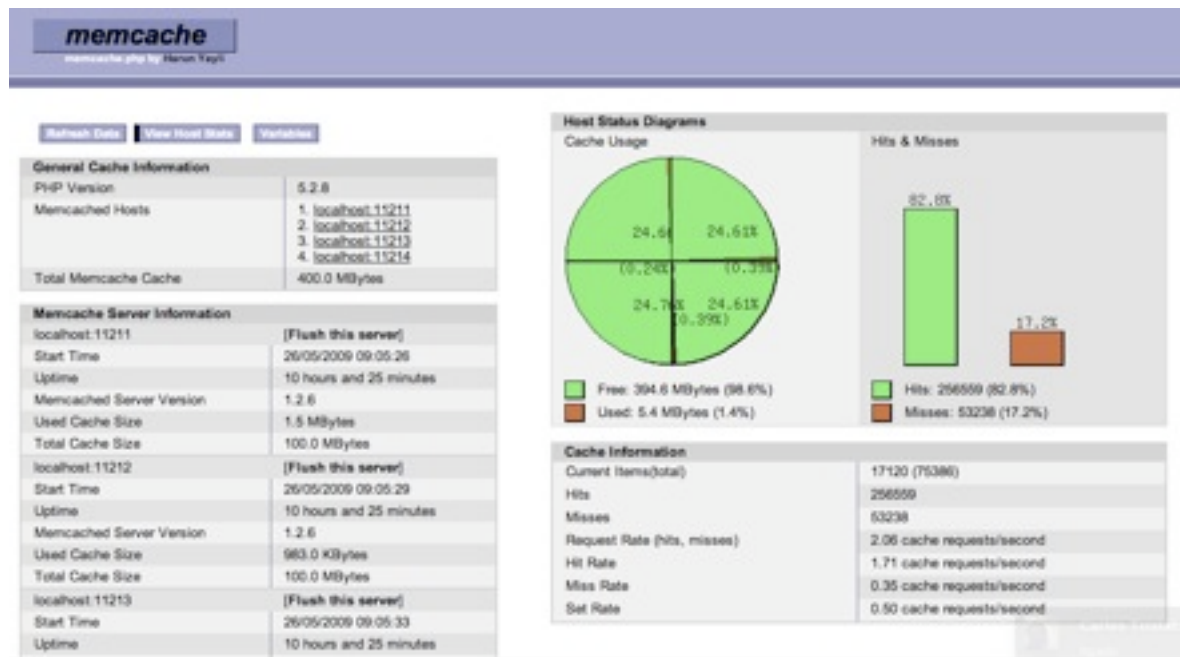
```
<?php
$memcache = new Memcache;
$memcache->connect('localhost', 11211);
$memcache->connect('localhost', 11212);
$memcache->connect('192.168.0.10', 11212);
$conteudo = $memcache->get('estados');
if($conteudo === false) {
    $uf = array('SC', 'RS', 'SP');
    $memcache->set('estados', $uf, false, 100) ;
    $conteudo = $uf;
}
var_dump($conteudo);
?>
```

API

- Comandos de armazenamento:
 - set, add, replace, append
- Comandos de recuperação:
 - get
- Comandos de exclusão:
 - delete
- Outros comandos:
 - stats, flush_all, version, verbosity, quit

Monitorando

- Plugin para o Nagios
- memcache.php









gearman

Aproximadamente 1.600.000 resultados (

Você quis dizer: [german](#)

[Gearman](#) ☆ - 3 visitas - 4 mai
5 Apr 2010 ... Gearman provides a
machines or processes that are be
Download - Documentation - Prese
[gearman.org/](#) - Em cache - Similar

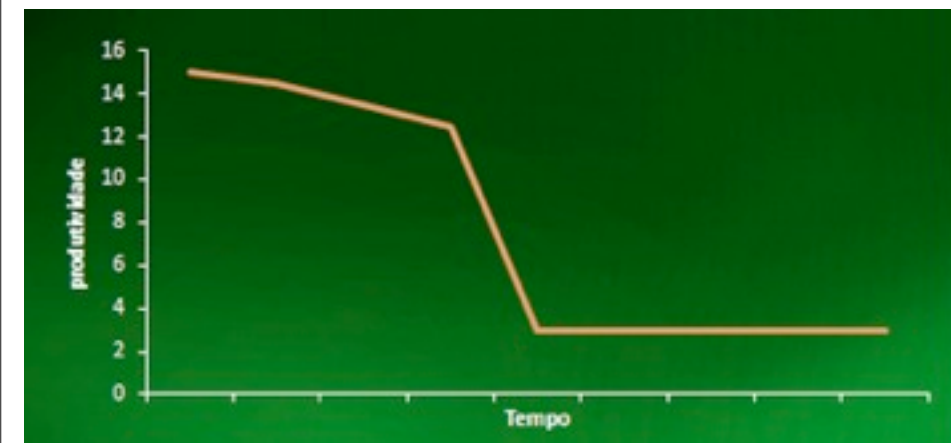
Tudo

Mais


A Web

Páginas em português

Páginas de Brasil



O que é?

- 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
- O nome é um anagrama para "Manager"
- Criado pela  DANGA INTERACTIVE

Áreas de Aplicação

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

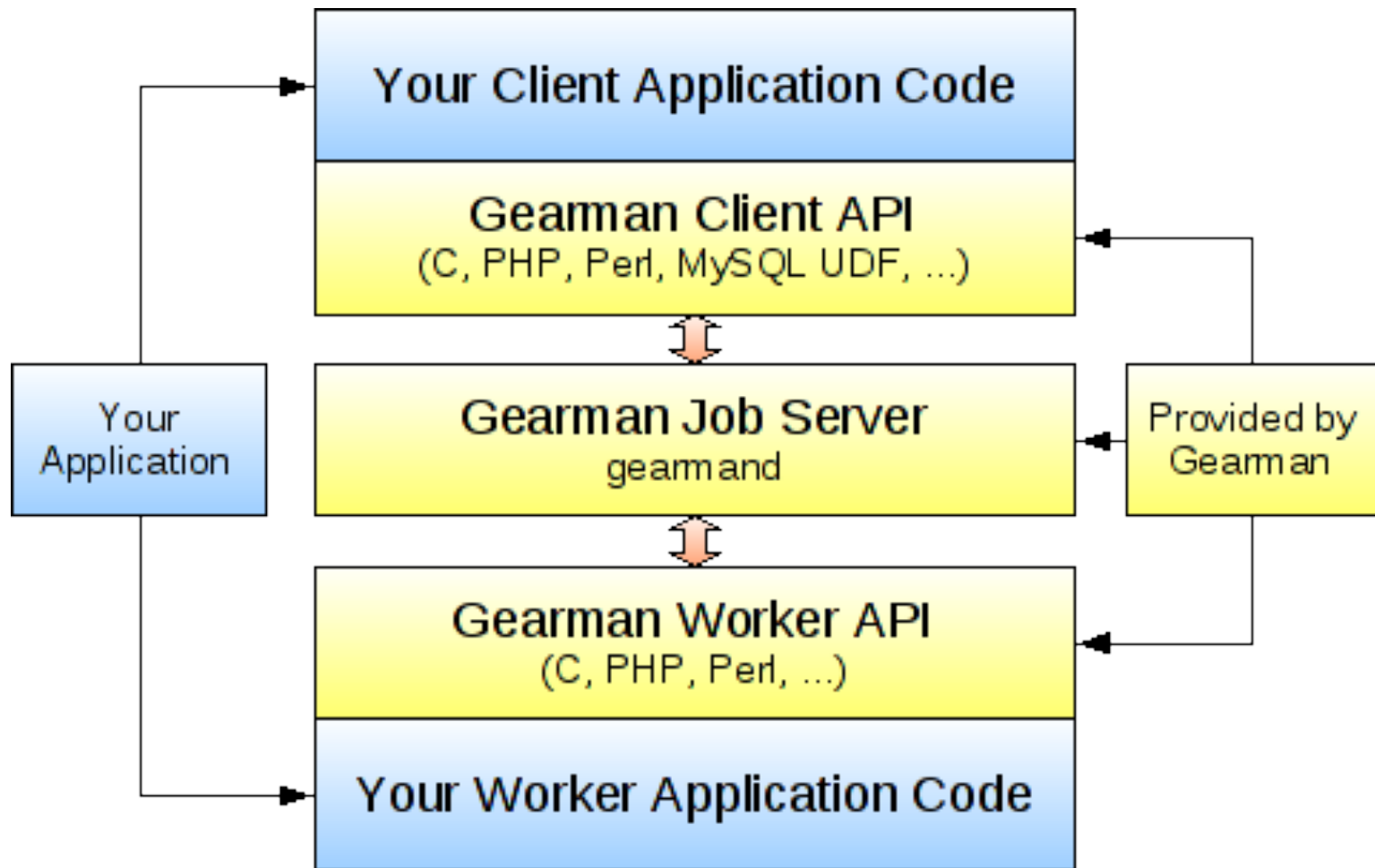
Por que usar?

- Open Source com 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
- Escalável e tolerante a falhas
 - Digg: +45 servers, 400mil jobs/dia (em 2009)
 - Yahoo: +60 servers, 6milhões jobs/dia
- Síncrono ou assíncrono

Como funciona?

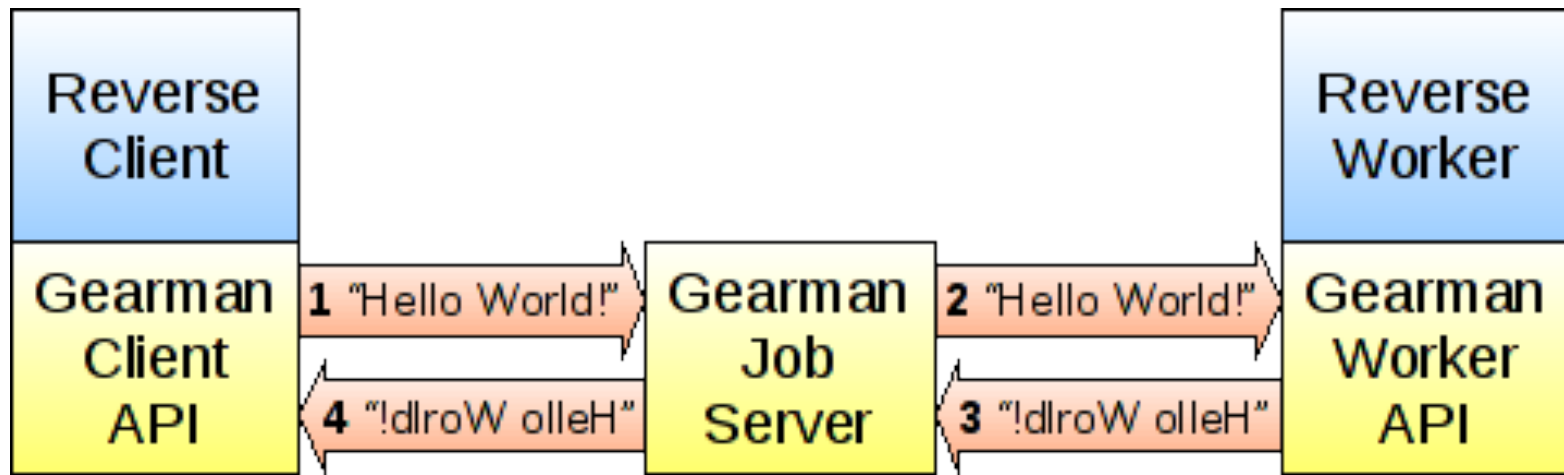
- Client- Cria um job a ser executado e o envia a um servidor de jobs
- Worker - Registra-se em um servidor de jobs e pega jobs para executar
- Job Server - Recebe os jobs e coordena a entrega de um job para um worker executar

Como funciona?



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

Como funciona?



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

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

Instalando

```
wget http://launchpad.net/gearmand/trunk/0.13/+download/gearmand-X.Y.tar.gz
```

```
tar xfvz gearmand-X.Y.tar.gz
```

```
cd gearmand-X.Y
```

```
./configure
```

```
make
```

```
sudo make install
```

Lembra?



Santíssima Trindade

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-X.Y.Z.tgz
```

```
tar xfvz gearman-X.Y.Z.tgz
```

```
cd gearman-X.Y.Z
```

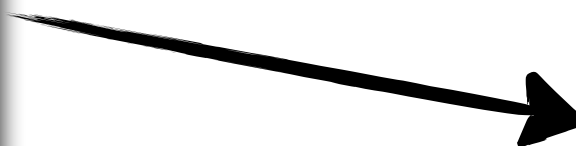
```
phpize
```

```
./configure
```

```
make
```

```
sudo make install
```

- Adicionar no php.ini:
extension = gearman.so



Acabou a
criativade :(
Mas você
entendeu né?

PHP - Exemplo: Cliente Síncrono

```
<?php
$client= new GearmanClient();
/*indica em quais servidores o cliente pode enviar jobs.
No caso somente o localhost, pois não é passado nada*/
$client->addServer();
/*adiciona um job síncrono, para a fila chamada title,
enviando uma string como parâmetro*/
print $client->do("title", "o Ivo andava de PATINS!!!");
print "\n";
?>
```

PHP - Exemplo: Worker Síncrono

```
<?php
$worker= new GearmanWorker();
$worker->addServer();//servidores q vai atender
/*registra a função title_function como responsável por
atender a fila title*/
$worker->addFunction("title", "title_function");
while ($worker->work()); //fica em loop trabalhando

//função que vai executar o trabalho
function title_function($job)
{
    return ucwords(strtolower($job->workload()));
}
?>
```


PHP - Exemplo: Cliente Assíncrono

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

PHP - Exemplo: Worker Assíncrono

```
<?php
$worker= new GearmanWorker();
$worker->addServer();
/*registra a função log_function como responsável por
atender a fila log_queue*/
$worker->addFunction("log_queue", "log_function");
while ($worker->work()); //fica em loop trabalhando

function log_function($job){
    $log = unserialize($job->workload());
    sleep(1); //sleep só para demonstrar executando
    echo $log['data'], ' - ', $log['msg'], "\n";
}
?>
```



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 = $client->doBackground("title", "o PHp");
4. if($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

- **Síncrono**

```
echo $client->do("title", "uma frase qualquer");  
echo $client->doHigh("title", "uma frase qualquer");  
echo $client->doLow("title", "uma frase qualquer");
```

- **Assíncrono**

```
$handle1 = $client->doBackground("title", "uma frase qualquer");  
$handle1 = $client->doHighBackground("title", "uma frase  
qualquer");  
$handle1 = $client->doLowBackground("title", "uma frase  
qualquer");
```


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 (somente jobs assíncronos)
- É possível armazenar a fila em:
 - MySQL (libdrizzle)
 - Memcached (libmemcached)
 - SQLite3 (libsqlite3)

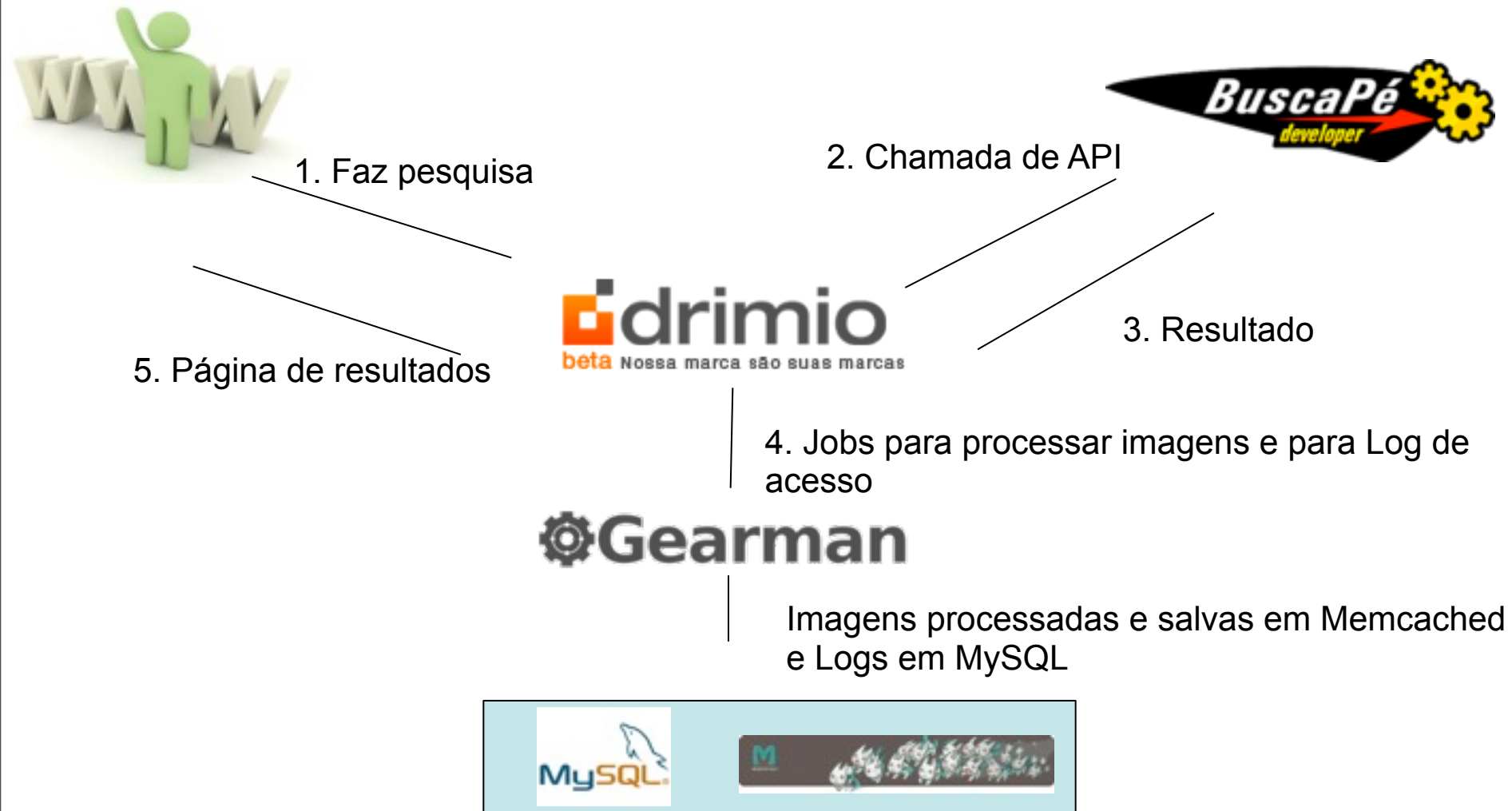
Monitorando

```
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
```

Gearman no 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
- Média de 55 mil jobs/dia

“Tudo junto reunido!”



Perguntas?



Contato

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

Referências

- <http://www.danga.com>
- <http://www.slideshare.net/andreizm/all-the-little-pieces-1573862>
- <http://www.slideshare.net/acme/scaling-with-memcached>
- <http://www.slideshare.net/oemebamo/introduction-to-memcached>
- <http://www.slideshare.net/felixdv/high-gear-php-with-gearman-3404242>
- <http://www.eltonminetto.net/material-da-palestra-no-phpsc-conf-2009.htm>
- <http://www.eltonminetto.net/material-de-minha-palestra-sobre-gearman.htm>