

# Acelerando sitios webs con Memcached

César D. Rodas  
*crodas@member.fsf.org*

LATINOWARE  
2008

<http://cesar.la/talks/>

Latinoware 2008, Foz do Iguazu, Brasil

# Quién les habla?

- ⦿ Participante Google Summer of Code 2008
  - Plug-in para WordPress



# Quién les habla?

- ⦿ Participante Google Summer of Code 2008
  - Plug-in para WordPress
- ⦿ Ganador del PHP Innovation award 2007



# Quién les habla?

- ⦿ Participante Google Summer of Code 2008
  - Plug-in para WordPress
- ⦿ Ganador del PHP Innovation award 2007
- ⦿ Centro Nacional de Computacion

LATIN**WARE**  
2008



# Quién les habla?

- ⦿ Participante Google Summer of Code 2008
  - Plug-in para WordPress
- ⦿ Ganador del PHP Innovation award 2007
- ⦿ Centro Nacional de Computacion
- ⦿ [www.CesaRodas.com](http://www.CesaRodas.com)

LATIN**WARE**  
2008



# Quién les habla?

- ⦿ Participante Google Summer of Code 2008
  - Plug-in para WordPress
- ⦿ Ganador del PHP Innovation award 2007
- ⦿ Centro Nacional de Computacion
- ⦿ [www.CesaRodas.com](http://www.CesaRodas.com)

LATIN**WARE**  
2008



# Qué es Cache?

- ⦿ Datos duplicados, almacenados en un medio más rápido.



# Qué es Cache?

- ⦿ Datos duplicados, almacenados en un medio más rápido.
- ⦿ Mejor solución para el cuello de botella.



# Qué es Cache?

- ⦿ Datos duplicados, almacenados en un medio más rápido.
- ⦿ Mejor solución para el cuello de botella.
- ⦿ Cuello de botella de la web (la base de datos).



# Qué es Cache?

- ⦿ Datos duplicados, almacenados en un medio más rápido.
- ⦿ Mejor solución para el cuello de botella.
- ⦿ Cuello de botella de la web (la base de datos).
  - Autenticación.



# Qué es Cache?

- ⦿ Datos duplicados, almacenados en un medio más rápido.
- ⦿ Mejor solución para el cuello de botella.
- ⦿ Cuello de botella de la web (la base de datos).
  - Autenticación.
  - SQL.



# Qué es Cache?

- ⊙ Datos duplicados, almacenados en un medio más rápido.
- ⊙ Mejor solución para el cuello de botella.
- ⊙ Cuello de botella de la web (la base de datos).
  - Autenticación.
  - SQL.
    - ▷ *Compilación.*

LATINOWARE  
2008



# Qué es Cache?

- ⊙ Datos duplicados, almacenados en un medio más rápido.
- ⊙ Mejor solución para el cuello de botella.
- ⊙ Cuello de botella de la web (la base de datos).
  - Autenticación.
  - SQL.
    - ▷ *Compilación.*
    - ▷ *Ejecución del código.*

LATINOWARE  
2008



# Qué es Cache?

- ⊙ Datos duplicados, almacenados en un medio más rápido.
- ⊙ Mejor solución para el cuello de botella.
- ⊙ Cuello de botella de la web (la base de datos).
  - Autenticación.
  - SQL.
    - ▷ *Compilación.*
    - ▷ *Ejecución del código.*
    - ▷ *Bloqueo de IO.*



# Qué es Cache?

- ⊙ Datos duplicados, almacenados en un medio más rápido.
- ⊙ Mejor solución para el cuello de botella.
- ⊙ Cuello de botella de la web (la base de datos).
  - Autenticación.
  - SQL.
    - ▷ *Compilación.*
    - ▷ *Ejecución del código.*
    - ▷ *Bloqueo de IO.*
    - ▷ *Acceso al disco.*



# Qué es Cache?

- ⊙ Datos duplicados, almacenados en un medio más rápido.
- ⊙ Mejor solución para el cuello de botella.
- ⊙ Cuello de botella de la web (la base de datos).
  - Autenticación.
  - SQL.
    - ▷ *Compilación.*
    - ▷ *Ejecución del código.*
    - ▷ *Bloqueo de IO.*
    - ▷ *Acceso al disco.*



# Que no es Memcached!

- ① Base de datos.



# Que no es Memcached!

- ⦿ Base de datos.
- ⦿ El único medio de cacheo.



# Que no es Memcached!

- ⦿ Base de datos.
- ⦿ El único medio de cacheo.
- ⦿ La mejor alternativa para sitios pequeños-medianos.



# Que no es Memcached!

- ⦿ Base de datos.
- ⦿ El único medio de cacheo.
- ⦿ La mejor alternativa para sitios pequeños-medianos.
- ⦿ Solución económica.

LATIN**WARE**  
2008



# Que no es Memcached!

- ⦿ Base de datos.
- ⦿ El único medio de cacheo.
- ⦿ La mejor alternativa para sitios pequeños-medianos.
- ⦿ Solución económica.
- ⦿ Complicada de implementar.

LATINOWARE  
2008



# Que no es Memcached!

- ⦿ Base de datos.
- ⦿ El único medio de cacheo.
- ⦿ La mejor alternativa para sitios pequeños-medianos.
- ⦿ Solución económica.
- ⦿ Complicada de implementar.
- ⦿ Seguro.

LATINOWARE  
2008



# Que no es Memcached!

- ⊙ Base de datos.
- ⊙ El único medio de cacheo.
- ⊙ La mejor alternativa para sitios pequeños-medianos.
- ⊙ Solución económica.
- ⊙ Complicada de implementar.
- ⊙ Seguro.

LATINWARE  
2008



# Que es Memcached!

- ⦿ Cache distribuido en memoria RAM.



# Que es Memcached!

- ⦿ Cache distribuido en memoria RAM.
- ⦿ Creado por y para LiveJournal.



# Que es Memcached!

- ⦿ Cache distribuido en memoria RAM.
- ⦿ Creado por y para LiveJournal.
- ⦿ Escalable.



# Que es Memcached!

- ⊙ Cache distribuido en memoria RAM.
- ⊙ Creado por y para LiveJournal.
- ⊙ Escalable.
- ⊙ Utilizado por los grandes.
  - LiveJournal.
  - Wikipedia.
  - Slashdot.
  - SourceForge.
  - ...



# El servidor

- ⦿ Óptima administración de memoria (Slab Allocation).



# El servidor

- ⦿ Óptima administración de memoria (Slab Allocation).
- ⦿ Utiliza libevent.



# El servidor

- ⦿ Óptima administración de memoria (Slab Allocation).
- ⦿ Utiliza libevent.
- ⦿ Protocolo simple y sencillo.
  - Nada de XML ni nada complicado de parsear.

LATIN**WARE**  
2008



# El servidor

- ⦿ Óptima administración de memoria (Slab Allocation).
- ⦿ Utiliza libevent.
- ⦿ Protocolo simple y sencillo.
  - Nada de XML ni nada complicado de parsear.
- ⦿ No existen bloqueos de IO.

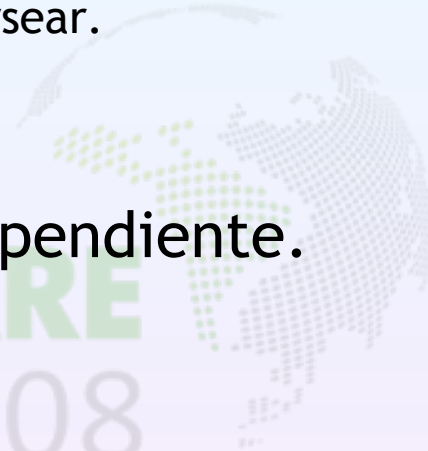
LATIN**WARE**  
2008



# El servidor

- ⦿ Óptima administración de memoria (Slab Allocation).
- ⦿ Utiliza libevent.
- ⦿ Protocolo simple y sencillo.
  - Nada de XML ni nada complicado de parsear.
- ⦿ No existen bloqueos de IO.
- ⦿ Cada servidor es totalmente independiente.
  - No se comparten datos.
  - No hay un punto centralizado.
  - Sencillo, efectivo y rápido.

LATINOWARE  
2008



# El servidor

- ⊙ Óptima administración de memoria (Slab Allocation).
- ⊙ Utiliza libevent.
- ⊙ Protocolo simple y sencillo.
  - Nada de XML ni nada complicado de parsear.
- ⊙ No existen bloqueos de IO.
- ⊙ Cada servidor es totalmente independiente.
  - No se comparten datos.
  - No hay un punto centralizado.
  - Sencillo, efectivo y rápido.
- ⊙ Cuenta con hash interno.

# El cliente

- ⦿ Elige a que servidor conectarse.



# El cliente

- ⦿ Elige a que servidor conectarse.
- ⦿ Serializa los objetos del lenguaje.



# El cliente

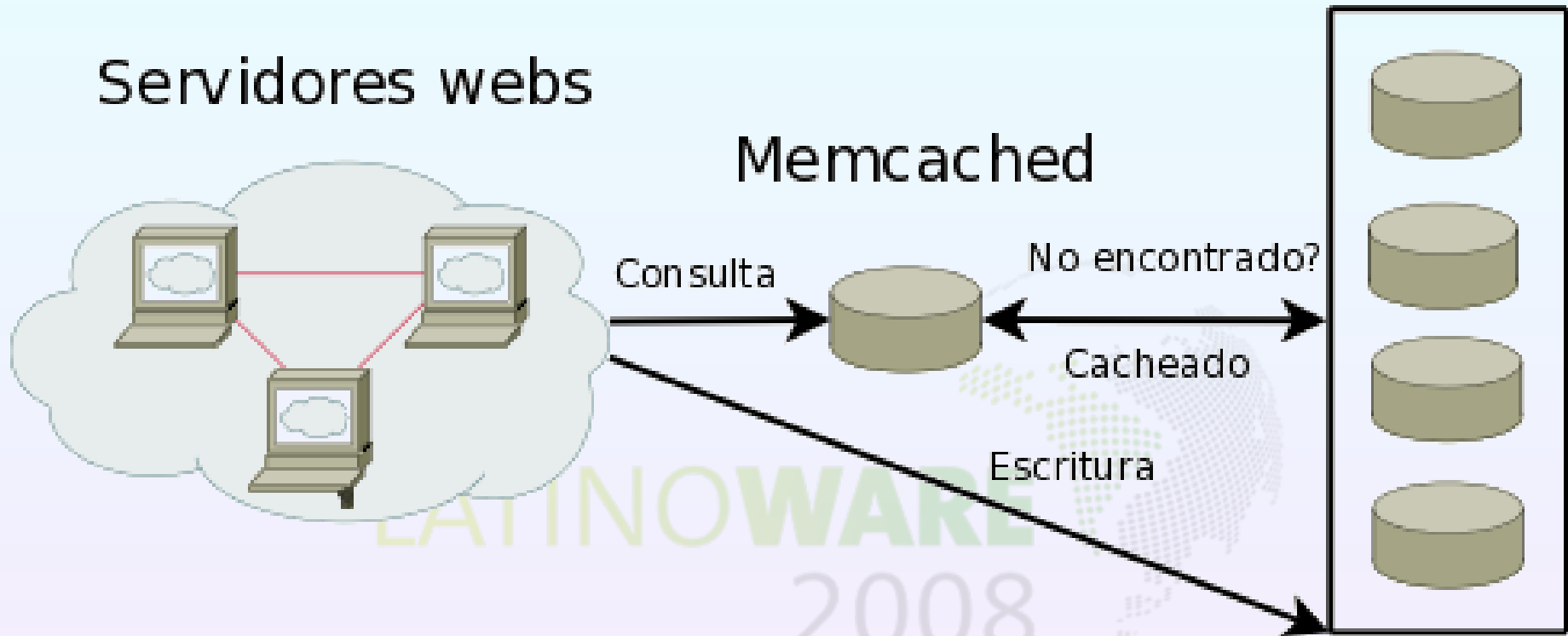
- ⦿ Elige a que servidor conectarse.
- ⦿ Serializa los objetos del lenguaje.
- ⦿ Comprimir datos.



# Base(s) de datos

## Servidores webs

## Memcached



# Otras utilidades de Memcached

- ⊙ Compartir información entre servidores.
  - Sesiones.
  - Información utilizada frecuentemente.
  - ..



# Monitoreando Memcached

## ⊙ Cacti

- <http://dealnews.com/developers/cacti/memcached.html>

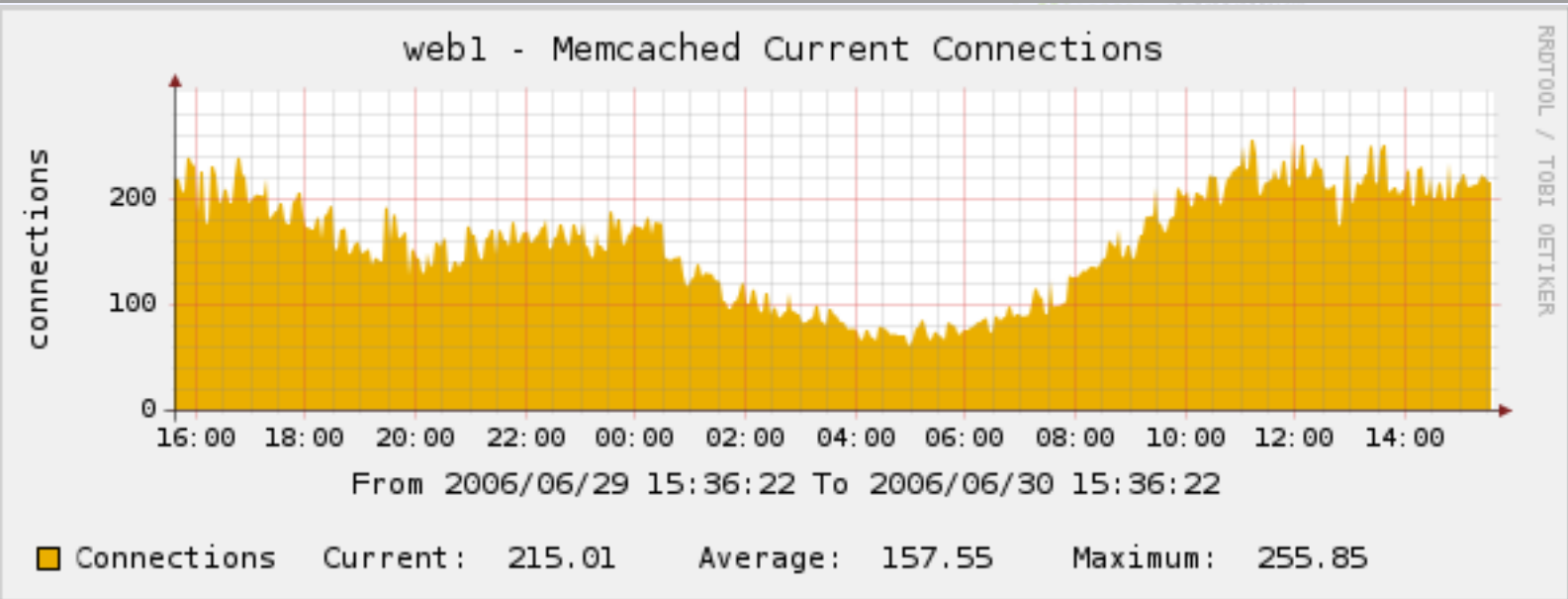
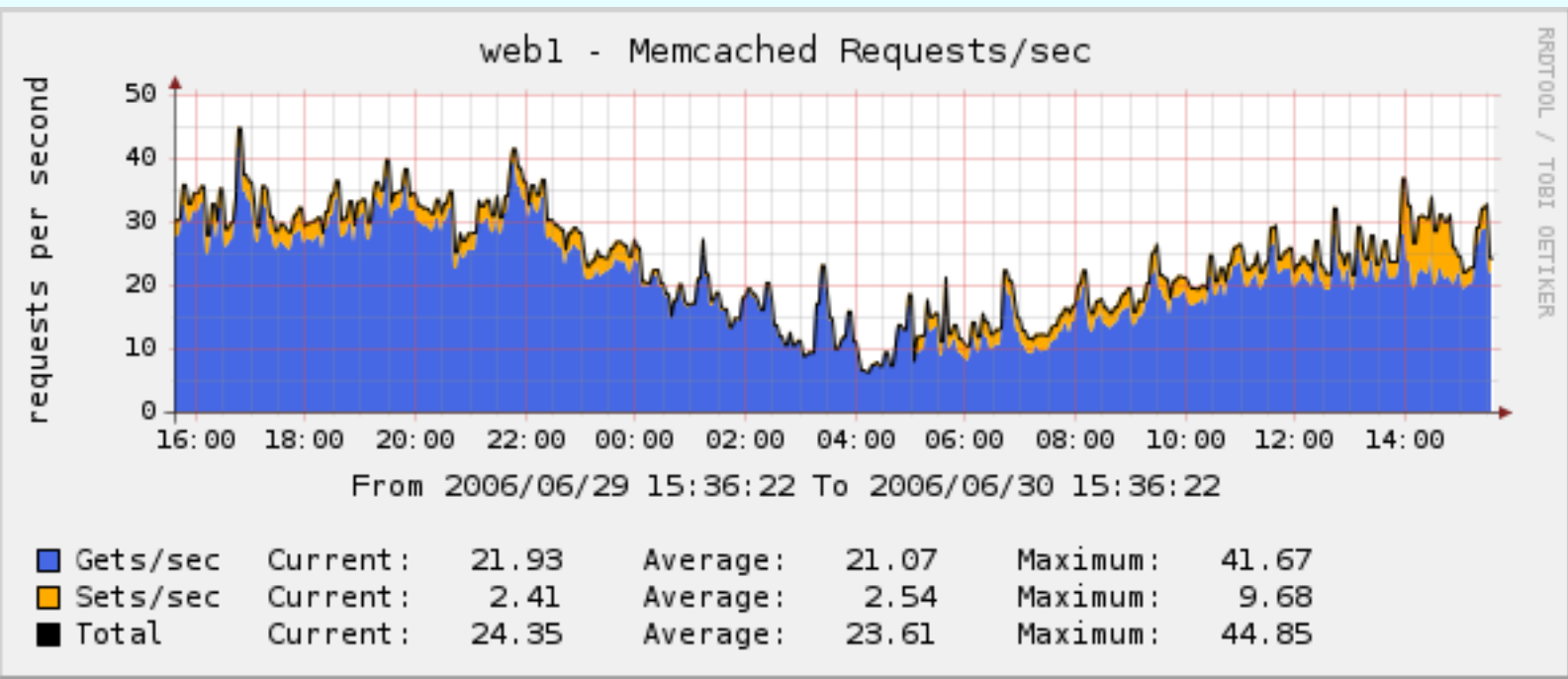
## ⊙ Ganglia

- <http://ganglia.wiki.sourceforge.net/>

## ⊙ ...

LATINWARE  
2008





# Alternativas

- ⦿ MySQL query cache.
  - Útil si la tabla no varía frecuentemente.



# Alternativas

- ⦿ MySQL query cache.
  - Útil si la tabla no varía frecuentemente.
- ⦿ Bases de datos replicadas.
  - Se pueden descentralizar las lecturas pero no las escrituras.

LATIN**WARE**  
2008



# Alternativas

- ⦿ MySQL query cache.
  - Útil si la tabla no varía frecuentemente.
- ⦿ Bases de datos replicadas.
  - Se pueden descentralizar las lecturas pero no las escrituras.
  - Las escrituras bloquean las lecturas.

LATIN**WARE**  
2008



# Alternativas

- ⦿ MySQL query cache.
  - Útil si la tabla no varía frecuentemente.
- ⦿ Bases de datos replicadas.
  - Se pueden descentralizar las lecturas pero no las escrituras.
  - Las escrituras bloquean las lecturas.
- ⦿ Memoria Compartida.
  - El cache no puede ser distribuido.

LATINWARE  
2008



# Alternativas

- ⊙ MySQL query cache.
  - Útil si la tabla no varía frecuentemente.
- ⊙ Bases de datos replicadas.
  - Se pueden descentralizar las lecturas pero no las escrituras.
  - Las escrituras bloquean las lecturas.
- ⊙ Memoria Compartida.
  - El cache no puede ser distribuido.
  - Muy costoso gracias a los bloqueos.

LATINOWARE  
2008



# Alternativas

- ⦿ MySQL query cache.
  - Útil si la tabla no varía frecuentemente.
- ⦿ Bases de datos replicadas.
  - Se pueden descentralizar las lecturas pero no las escrituras.
  - Las escrituras bloquean las lecturas.
- ⦿ Memoria Compartida.
  - El cache no puede ser distribuido.
  - Muy costoso gracias a los bloqueos.
- ⦿ File cache.
  - El cache no puede ser distribuido.

LATINWARE  
2008



# Alternativas

- ⦿ MySQL query cache.
  - Útil si la tabla no varía frecuentemente.
- ⦿ Bases de datos replicadas.
  - Se pueden descentralizar las lecturas pero no las escrituras.
  - Las escrituras bloquean las lecturas.
- ⦿ Memoria Compartida.
  - El cache no puede ser distribuido.
  - Muy costoso gracias a los bloqueos.
- ⦿ File cache.
  - El cache no puede ser distribuido.
  - Mejor cache para sitios pequeños (< 70000 hits por dia).

LATINOWARE  
2008



# Alternativas

- ⊙ MySQL query cache.
  - Útil si la tabla no varía frecuentemente.
- ⊙ Bases de datos replicadas.
  - Se pueden descentralizar las lecturas pero no las escrituras.
  - Las escrituras bloquean las lecturas.
- ⊙ Memoria Compartida.
  - El cache no puede ser distribuido.
  - Muy costoso gracias a los bloqueos.
- ⊙ File cache.
  - El cache no puede ser distribuido.
  - Mejor cache para sitios pequeños (< 70000 hits por dia).



# Alternativas

- ① File cache en "disco RAM".



# Alternativas

- ⦿ File cache en "disco RAM".
  - Alternativa muy rápida y costosa.



# Alternativas

## ⦿ File cache en "disco RAM".

- Alternativa muy rápida y costosa.

- GNU/Linux:

```
/sbin/mke2fs -q -m 0 /dev/ram0
```

```
/bin/mount /dev/ram0 /mnt/cache
```

```
/bin/chown httpd:root /mnt/cache
```

```
/bin/chmod 0750 /mnt/cache
```

LATIN**WARE**  
2008



# Alternativas

## ⦿ File cache en "disco RAM".

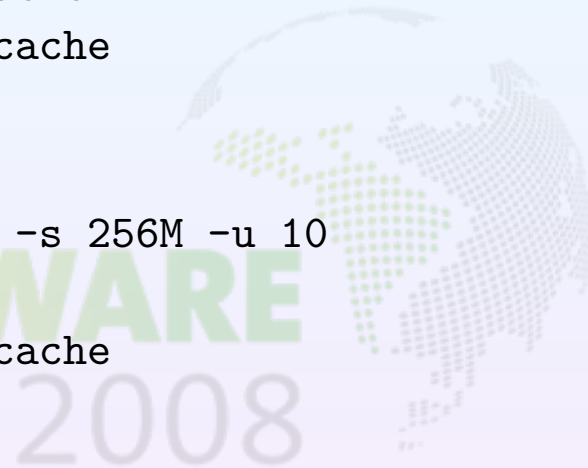
- Alternativa muy rápida y costosa.

- GNU/Linux:

```
/sbin/mke2fs -q -m 0 /dev/ram0  
/bin/mount /dev/ram0 /mnt/cache  
/bin/chown httpd:root /mnt/cache  
/bin/chmod 0750 /mnt/cache
```

- FreeBSD (y posiblemente \*BSD):

```
/sbin/mdconfig -a -t malloc -s 256M -u 10  
/sbin/newfs -U /dev/md10  
/sbin/mount /dev/md10 /mnt/cache
```



# Otros tips utiles

## ⦿ Eliminar los whitespaces.

- Si "ahorramos" 30 bytes por página, en 10,000,000 visitas serían ~286MB.



# Otros tips utiles

## ⦿ Eliminar los whitespaces.

- Si "ahorramos" 30 bytes por página, en 10,000,000 visitas serían ~286MB.

## ⦿ Paginas comprimidas.

- On the fly ( `ob_start("ob_gzhandler")` )
- Cacheado.
- `mod_gzip`.
- ...

LATINOWARE  
2008



# Otros tips utiles

## ⦿ Eliminar los whitespaces.

- Si "ahorramos" 30 bytes por página, en 10,000,000 visitas serían ~286MB.

## ⦿ Paginas comprimidas.

- On the fly ( `ob_start("ob_gzhandler")` )
- Cacheado.
- `mod_gzip`.
- ...

## ⦿ Contenido estático... al servidor multi-threaded.

LATINOWARE  
2008



# Otros tips utiles

## ⦿ Eliminar los whitespaces.

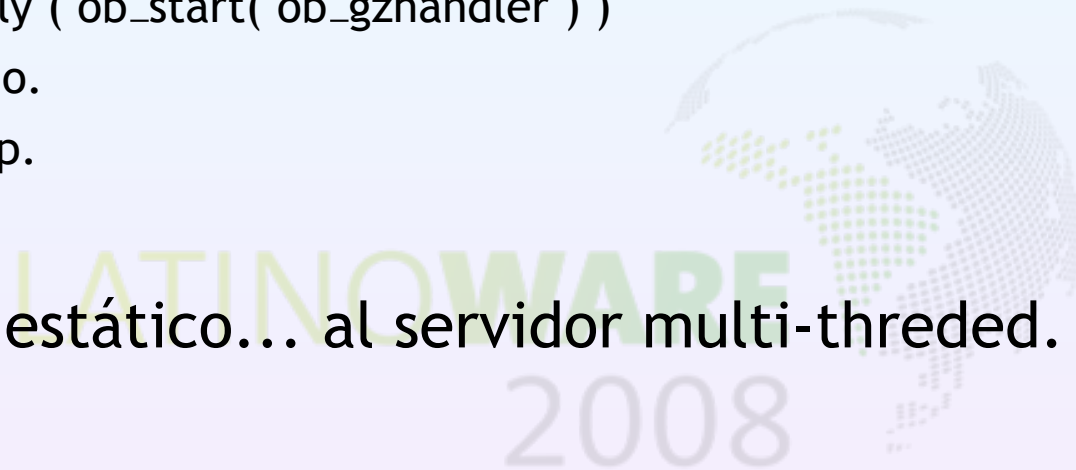
- Si "ahorramos" 30 bytes por página, en 10,000,000 visitas serían ~286MB.

## ⦿ Paginas comprimidas.

- On the fly ( `ob_start("ob_gzhandler")` )
- Cacheado.
- `mod_gzip`.
- ...

## ⦿ Contenido estático... al servidor multi-threaded.

## ⦿ APC.



# Otros tips utiles

- ⊙ Eliminar los whitespaces.
  - Si "ahorramos" 30 bytes por página, en 10,000,000 visitas serían ~286MB.
- ⊙ Paginas comprimidas.
  - On the fly ( `ob_start("ob_gzhandler")` )
  - Cacheado.
  - `mod_gzip`.
  - ...
- ⊙ Contenido estático... al servidor multi-threaded.
- ⊙ APC.
- ⊙ Squid Reverse proxy.

**Talk is cheap  
show me the code!**

LATIN**WARE**  
2008



# PHPCached vs. PHP Memcached

- ⦿ PHPCached es un poco más lento.



# PHPCached vs. PHP Memcached

- ⦿ PHPCached es un poco más lento.
  - Si pensamos así deberíamos programar en C.



# PHPCached vs. PHP Memcached

- ⦿ PHPCached es un poco más lento.
  - Si pensamos así deberíamos programar en C.
  - Se puede solucionar con APC.



# PHPCached vs. PHP Memcached

- ⦿ PHPCached es un poco más lento.
  - Si pensamos así deberíamos programar en C.
  - Se puede solucionar con APC.
- ⦿ PHPCached puede ser modificado fácilmente.

LATIN**WARE**  
2008



# PHPCached vs. PHP Memcached

- ⦿ PHPCached es un poco más lento.
  - Si pensamos así deberíamos programar en C.
  - Se puede solucionar con APC.
- ⦿ PHPCached puede ser modificado fácilmente.
- ⦿ En PHPCached se puede registrar un "callback" para el balanceo.

LATIN**WARE**  
2008



# PHPCached vs. PHP Memcached

- ⊙ PHPCached es un poco más lento.
  - Si pensamos así deberíamos programar en C.
  - Se puede solucionar con APC.
- ⊙ PHPCached puede ser modificado fácilmente.
- ⊙ En PHPCached se puede registrar un "callback" para el balanceo.
- ⊙ PHPCached tiene cacheo en archivos con la misma API.
  - Para sitios pequeños-medianos.
  - Ayuda a la transición (del file-based a memcached).

LATINWARE  
2008

# API sencilla, similar a DBA

```
<?php  
/* Objeto phpcached */  
$cache = new phpcached;
```



# API sencilla, similar a DBA

```
<?php
/* Objeto phpcached */
$cache = new phpcached;
/* Agregar la lista de servidores */
$cache->addServer("10.1.1.1",3128);
$cache->addServer("10.1.1.1",3129);
$cache->addServer("10.1.1.2",3128);
```

LATIN**WARE**  
2008



# API sencilla, similar a DBA

```
<?php
/* Objeto phpcached */
$cache = new phpcached;
/* Agregar la lista de servidores */
$cache->addServer("10.1.1.1",3128);
$cache->addServer("10.1.1.1",3129);
$cache->addServer("10.1.1.2",3128);
/*
** A utilizar Phpcached
** La funcion "do_long_process()" sólo será
** ejecutada cada 10 minutos
*/
if (($value=$cache->get("key")) === false) {
    $value = do_long_process();
    $cache->set("key",$value,600);
}
return $value;
?>
```

# Y si usas PHP5

```
<?php
/* definimos la lista de servidores */
define('PHPCACHED_SERVER1', '10.1.1.1:3128');
define('PHPCACHED_SERVER2', '10.1.1.1:3129');
define('PHPCACHED_SERVER3', '10.1.1.2:3128');

$cache = new arraycached(600);
if (!isset($cache[$key])) {
    $cache[$key] = do_long_process();
}
return $cache[$key];
?>
```



# Jugando con PHPCached

```
<?php  
$cache = new phpcached;  
$cache->addServer("10.1.1.1",3128);  
$cache->addServer("10.1.1.1",3129);  
$cache->addServer("10.1.1.2",3128);
```

LATIN**WARE**  
2008



# Jugando con PHPCached

```
<?php
$cache = new phpcached;
$cache->addServer("10.1.1.1",3128);
$cache->addServer("10.1.1.1",3129);
$cache->addServer("10.1.1.2",3128);
/*
** Definimos una funcion PHP que
** seleccionara que Memcached utilizar
** basado en la longitud del Key.
*/
function balancer($iServer,$zKey) {
return strlen($zKey)%$iServer;
}
```



# Jugando con PHPCached

```
<?php
$cache = new phpcached;
$cache->addServer("10.1.1.1",3128);
$cache->addServer("10.1.1.1",3129);
$cache->addServer("10.1.1.2",3128);
/*
** Definimos una funcion PHP que
** seleccionara que Memcached utilizar
** basado en la longitud del Key.
*/
function balancer($iServer,$zKey) {
return strlen($zKey)%$iServer;
}
/*
** Ahora agregamos nuestra funcion
** como balancer.
*/
$cache->setBalancer("balancer");
?>
```



# Alternativas para sitios pequeños

```
<?php  
/* Objeto phpcached */  
$cache = new phpcached;
```



# Alternativas para sitios pequeños

```
<?php  
/* Objeto phpcached */  
$cache = new phpcached;  
/* Agregar directorio de destino */  
$cache->addFolder("/var/tmp/cached");
```

LATIN**WARE**  
2008



# Alternativas para sitios pequeños

```
<?php
/* Objeto phpcached */
$cache = new phpcached;
/* Agregar directorio de destino */
$cache->addFolder("/var/tmp/cached");
/* A utilizar Phpcached */
/* La funcion "do long process()" sólo será */
/* ejecutada cada 10 minutos */
if (($value=$cache->get("key")) === false) {
    $value = do_long_process();
    $cache->set("key", $value, 600);
}
return $value;
?>
```

# Muchas gracias! Preguntas?

**Cesar Rodas.**

[crodas@member.fsf.org](mailto:crodas@member.fsf.org)

<http://cesarodas.com/>

**Mi blog.**

<http://cesar.la/>

LATINOWARE

2008



# Resources

## ⦿ Memcached

- <http://www.danga.com/memcached>
- <http://tangent.org/552/libmemcached.html>


## ⦿ Mamcached Mail list.

- <http://lists.danga.com/mailman/listinfo/memcached>

## ⦿ PHPCached

- <http://www.phpclasses.org/phpcached>

LATINWARE  
2008



Powered by...



L<sup>A</sup>T<sub>E</sub>X

LATINOWARE

2008