Afinando ...

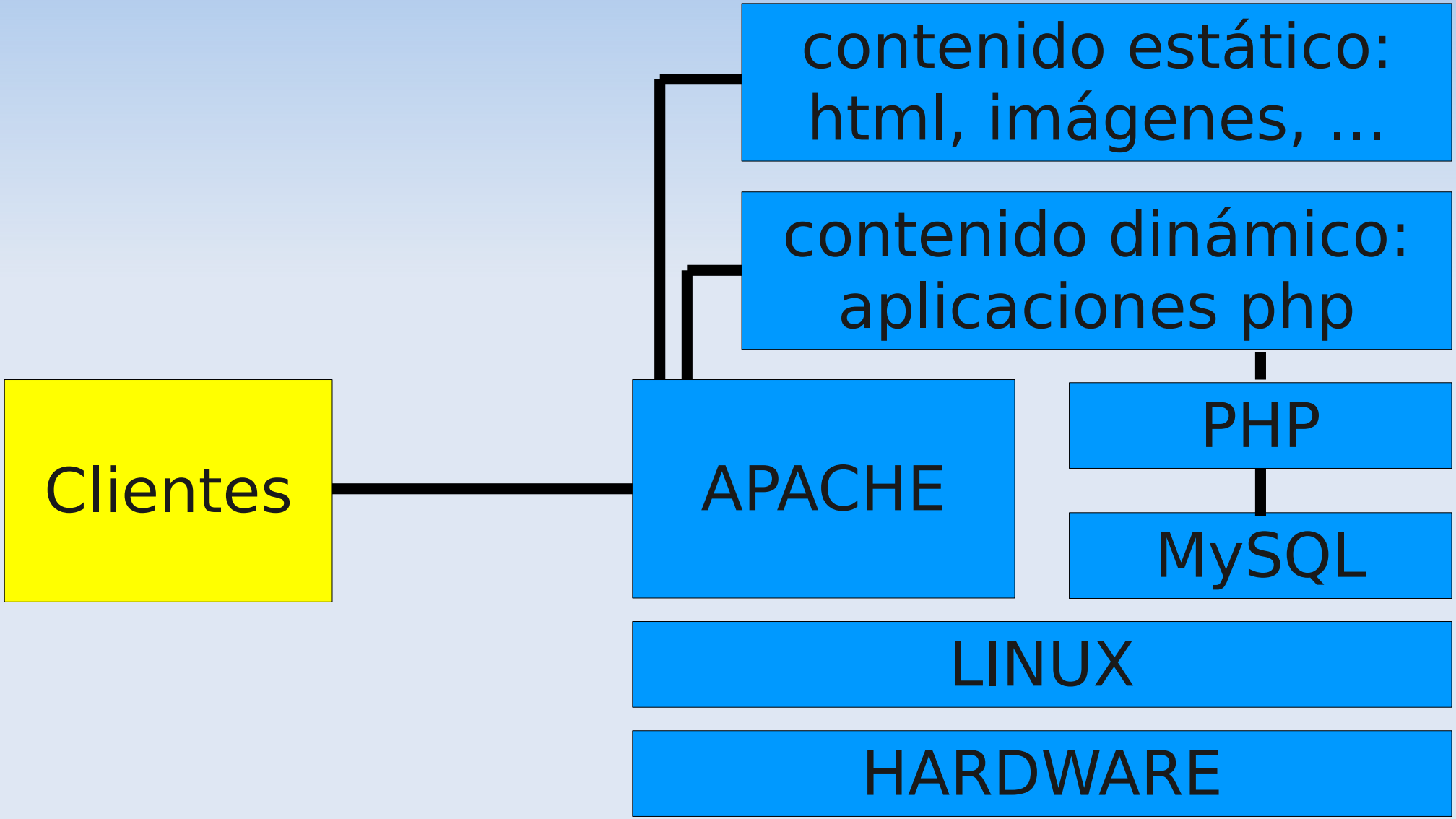
Afinando LAMP

Realizar cambios en la configuración de nuestro servidor para mejorar el rendimiento:

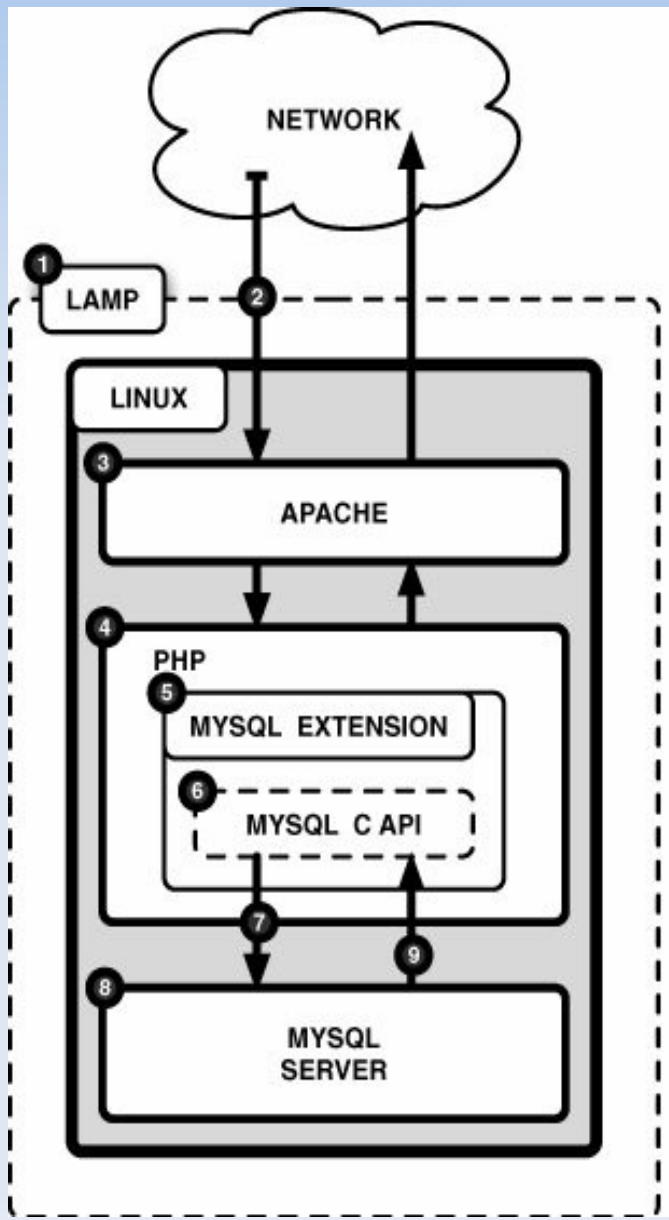
- Cambios en el hardware
- Cambios en el sistema operativo: Linux
- Cambios en los servicios e intérpretes: Apache, PHP, MySQL
- Cambios en las aplicaciones

¿Cómo trabaja LAMP?

¿Cómo trabaja LAMP?



¿Cómo trabaja LAMP?

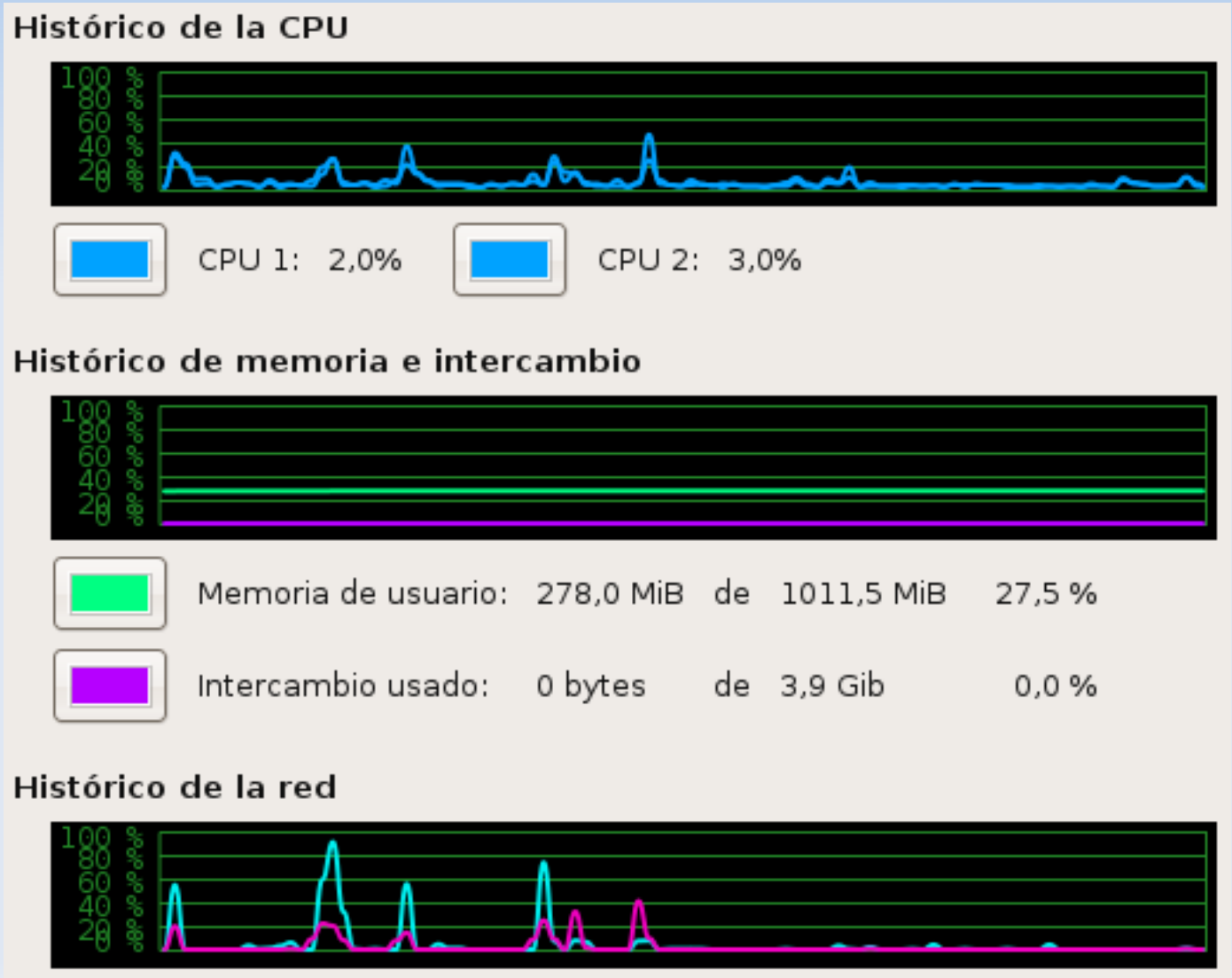


1. Together, Linux, Apache, MySQL and PHP make up the classic *LAMP* stack
2. Linux handles requests from the network
3. Web requests are passed to Apache
4. Requests for PHP pages are passed to the PHP interpreter
5. If the PHP page makes calls to MySQL, they are handled by one of PHP's MySQL extensions
6. The MySQL extensions communicate with the MySQL C API, passing requests betwee PHP and the MySQL C API.
7. The MySQL C API then communicates with a MySQL server.
8. The MySQL server processes the requests and returns data.
9. The results are further processed by PHP and are passed back up the chain to the network

Medir el rendimiento

- Medir para saber cuál es el cuello de botella.
- Medir para hacer predicciones sobre el futuro, cuando el número de peticiones al servidor crezca.
- Medir el rendimiento antes y después de cada cambio, para ver qué ha mejorado. Los cambios se deben medir de uno en uno.

Medir el rendimiento: monitor del sistema, top, ...



Medir el rendimiento: cURL

- curl mide el tiempo de respuesta de un servidor web ante la petición de un único elemento:

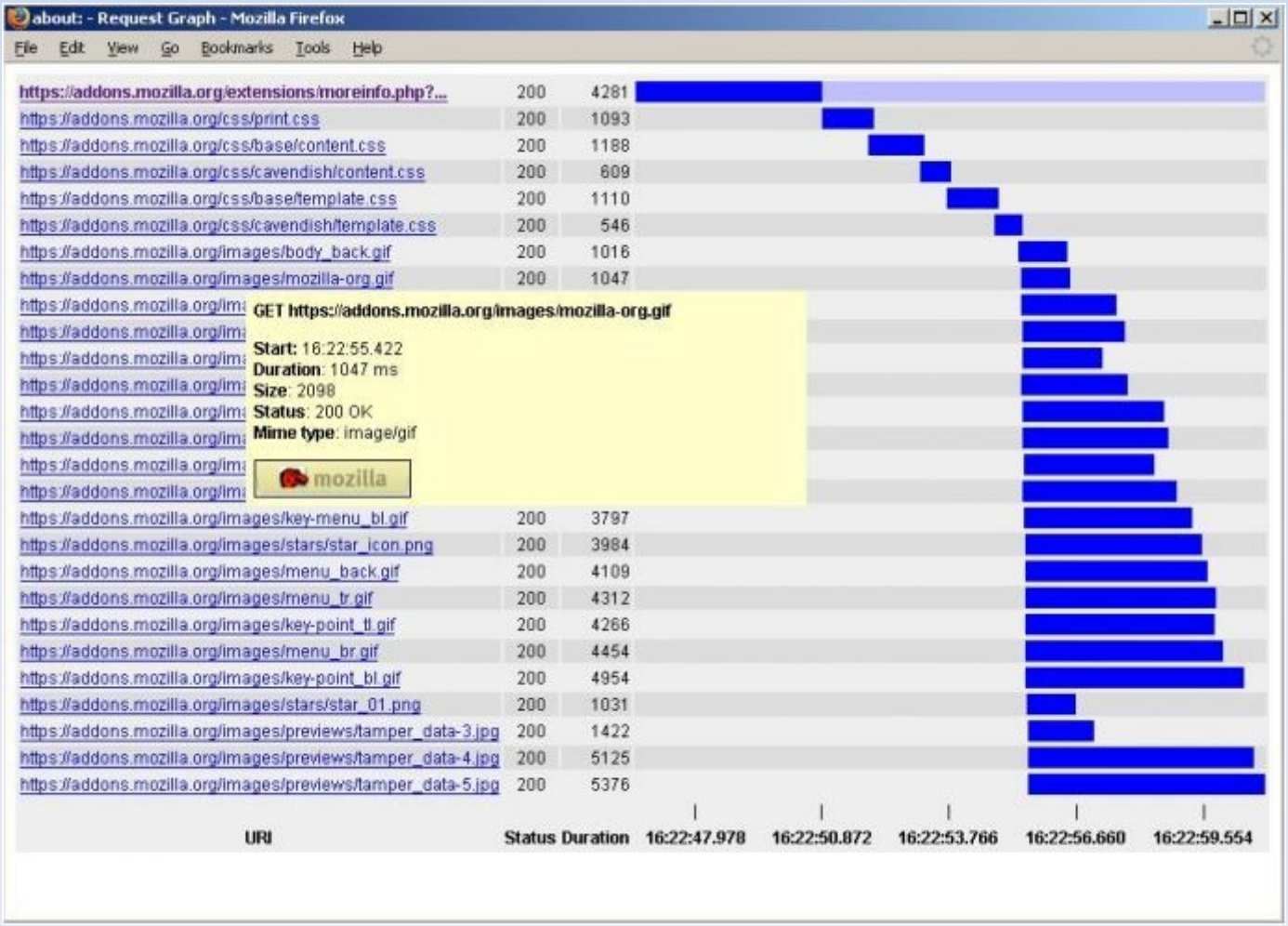
```
$ curl -o /dev/null -s -w %{time_connect}:%{time_starttransfer}:%{time_total}\
  http://www.laquimera.org

0.081:0.272:0.779
```

- Procesar petición y comenzar a enviar datos = 0.272 – 0.081 = 0.191 segundos
- Enviar todos los datos = 0.779 - 0.272 = 0.507 segundos

Medir el rendimiento: Firefox Tamper Data

- https://addons.mozilla.org/es-ES/firefox/addon/966



Medir el rendimiento: Apache

Apache HTTP server benchmarking tool (ab) es una utilidad incluida en Apache con la que hacer pruebas de carga sobre servidores web.

```
$ ab -n 1000 -c 5 http://www.servidor.com/pagina.html

Time taken for tests:   42.907696 seconds
Failed requests:        0
Total transferred:      584000 bytes
Requests per second:    23.31 [#/sec] (mean)
Time per request:       214.538 [ms] (mean)
Time per request:       42.908 [ms] (mean, across concurrent requests)
Transfer rate:          13.28 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       89  105 189.1     92    3089
Processing:    94  107  34.2    101     669
Waiting:       92  105  29.1     99     445
Total:        184  213 191.9    194    3192
```

Medir el rendimiento: MySQL

Toman ficheros con peticiones e interrogan el servidor de base de datos el número de veces que queramos y con el número de conexiones concurrentes que deseemos.

- *MySQL Super Smack*
  http://vegan.net/tony/supersmack/
- *MyBench*
  http://jeremy.zawodny.com/mysql/mybench/

Medir el rendimiento: más ...

- “Easy system monitoring with SAR”
  http://www.ibm.com/developerworks/aix/library/au-unix-perfmonsar.html
- “Expose Web performance problems with the RRDtool”

  http://www.ibm.com/developerworks/edu/dw-esdd-webperfrrd-i.html
- “Monitoring Virtual Memory with vmstat”

  http://www.linuxjournal.com/article/8178

Afinar el hardware

No es el objetivo de esta guía.

Según donde esté el cuello de botella podemos cambiar:

- CPU más rápida, con varios núcleos, 64 bits.
- Aumentar la memoria RAM.
- Discos duros más rápidos, RAID.
- Ancho de banda de conexión al exterior.
- Más máquinas: separar servidor web y servidor BBDD, clúster+balanceo carga.

Afinar las aplicaciones

No es el objetivo de esta guía.

Depende de cada problema. Los programadores pueden:

- Optimizar el código del programa PHP.
- Optimizar la estructura de tablas y las consultas al servidor de BBDD.

Afinar Linux: TCP/IP

```
$ sudo vim /etc/sysctl.conf

# Activa las TCP syncookies contra los ataques de SYN Flooding
net.ipv4.tcp_syncookies = 1

# Aumenta el tamaño de ventana (paquetes enviados antes de un ACK)
net.ipv4.tcp_window_scaling = 1

# Incrementa el tamaño de los buffers de llegada y envío de paquetes.
# Permiten a las aplicaciones del servidor tomar datos más rápidamente,
# y al cliente enviar más datos aunque el servidor esté ocupado
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216

# Incrementa el número de conexiones que pueden ser servidas
net.ipv4.ip_local_port_range = 1024 65000

$ sudo sysctl -p /etc/sysctl.conf
```

Afinar Linux: discos - atime

Cada vez que accedemos a un fichero, aunque sea para lectura, el sistema de ficheros guarda una marca de tiempo.

```
$ sudo vim /etc/fstab

# Dispositivo  Directorio     FS        Opciones                          
/dev/hda1      none            swap      defaults                  0 0
/dev/hda2      /               ext3      defaults,noatime          0 1
/dev/hda3      /home           ext3      defaults,noatime          0 2
none           /proc           proc      defaults                  0 0
/dev/fd0       /mnt/floppy     auto      noauto,user,noexec,rw     0 0
/dev/cdrom     /mnt/cdrom      iso9660   noauto,user,noexec,ro     0 0

$ sudo mount / -o remount
$ sudo mount /home -o remount
```

Afinar Linux: discos - hdparm

Permite afinar el acceso a los discos IDE: 32 bits, tipo de DMA, ... ¡con cuidado!

```
$ sudo hdparm -t /dev/sda
Timing buffered disk reads:  162 MB in  3.00 seconds =  53.99 MB/sec

$ sudo hdparm -vi /dev/sda      características que soporta y las activas

$ sudo hdparm -c 1 /dev/sda activa el modo de transferencia de 32 bits

$ sudo hdparm -m x /dev/sda transfiere x sectores por interrupción

$ sudo hdparm -d 1 -X x /dev/sda   activa las transferencias DMA y
  establece el modo de DMA al especificado por x
```

Guardar en algún script de inicio, como por ejemplo rc.local

Afinar Linux: NFS

Si utilizamos carpetas en red (evitar NFSv2)

- En el cliente:

```
$ sudo vim /etc/fstab
#Dispositivo Directorio   FS    Opciones
host2:/tmp   /mnt/host2   nfs   rsize=32768,wsize=32768,intr,noatime
#bloques de 32Kb, las operaciones se interrumpen si se cuelgan, no atime
...
$ sudo mount / -o remount
```

- En el servidor suficientes threads NFS:

```
$ sudo nfsstat -rc
calls      retrans    authrefrsh
1465903813   0          0
$ sudo rpc.nfsd 64
```

Afinar Apache: ¿Cómo funciona?

Diferentes tipos de Apache:

- Apache compilado estáticamente con las funciones necesarias. Es rápido y ligero, pero no se pueden añadir nuevas funciones sin recompilar.
- Apache compilado dinámicamente. Los módulos se cargan a medida que se necesitan nuevas funciones. El corazón de Apache (Multi Processing Modules) no se puede cambiar sin recompilar.

Afinar Apache: ¿Cómo funciona?

Diferentes Multi Processing Modules:

- Prefork: seguro, varios procesos hijos, un proceso por petición, bueno para 1 o 2 CPUs, uso grande de memoria.
- Worker: menos tolerante a fallos, varios procesos hijos cada uno con varios threads, un thread por petición, bueno para sistemas multiprocesador, menor consumo de memoria.

`$ httpd -l` ó `$ apache2 -l` para saber que MPM utiliza nuestro apache

Afinar Apache: ¿Cómo funciona?

- Al iniciar Apache se crean varios procesos hijo para atender peticiones.
- Un proceso hijo atiende una petición y después se queda en reserva, a la espera de que llegue una petición nueva.
- Las peticiones que no se pueden atender porque hemos llegado al máximo de procesos y ninguno está libre, esperan en una cola.

Afinar Apache: ¿Cómo funciona?

- Como crear un nuevo proceso cuando llega una petición es lento, se intenta que siempre hayan servidores en reserva preparados para atender alguna petición.
- Si todos los procesos están ocupados, se crean un mínimo de procesos en reserva.
- Si hay muchos procesos desocupados, se matan para que sólo haya un máximo de procesos en reserva.

Afinar Apache: ¿Cómo funciona?

- Los procesos nacen pesando 3Mb pero al servir contenido dinámico engordan hasta 20Mb y ya nunca adelgazan:

```
$ ps -ylC apache2   para ver lo que pesan los procesos (columna RSS)
```

- Cuando un proceso ha servido muchas peticiones, se mata y se crea uno nuevo. Así evitamos los errores de memoria solicitada para atender una petición y no liberada.

Afinar Apache: MPM

Los valores por defecto son muy conservadores. Ajustémoslos:

```
$ sudo vim /etc/apache2/apache2.conf

  # Procesos que se crean al arrancar
  # Un buen valor es el promedio de peticiones simultáneas
  StartServers           50

  # Mínimo y máximo de procesos en reserva
  MinSpareServers        15
  MaxSpareServers        30

  # Numero máximo de peticiones simultáneas que se atenderán
  # Un buen valor es el pico diario de peticiones (vigila suficiente RAM)
  MaxClients             225

  # Número de peticiones de un proceso antes de morir
  MaxRequestsPerChild  4000

$ sudo /etc/init.d/apache2 restart
```

Afinar Apache: ¿Cómo funciona?

- Apache permite especificar opciones que se aplicarán a cada directorio y sus correspondientes subdirectorios: contraseñas, restringir IPs, etc.
- 1ª manera: líneas <*Directory ...*> en el fichero de configuración */etc/apache2/apache2.conf*
- 2ª manera: ficheros *.htaccess* en cada directorio. Los puede poner el propietario de esa parte de la web.

Afinar Apache: buscar ficheros

- Cuando Apache sirve contenido, primero busca el fichero *.htaccess* del directorio del contenido **y también de todos los directorios anteriores (hasta llegar a la raíz)**, para poder aplicar las opciones especificadas para el directorio.
- Se puede cambiar este comportamiento con la opción *AllowOverride* a valor *None*, y moviendo la información imprescindible de los *.htaccess* a las líneas <*Directory*> del fichero */etc/apache2/apache2.conf*

Afinar Apache: buscar ficheros

- Si por seguridad desactivamos enlaces simbólicos para no servir contenido que esté fuera de los directorios de la web, Apache comprueba que el fichero a servir y cada uno de los directorios des de la raíz no sean enlaces simbólicos.
- Se puede cambiar este comportamiento con la opción *Options -FollowSymLinks* y activándola sólo en los directorios imprescindibles

Afinar Apache: buscar ficheros

Un ejemplo:

```
$ sudo vim /etc/apache2/apache2.conf

  # Opciones para el directorio raíz y subdirectorios
  # Ignoramos los ficheros .htaccess y no comprobamos enlaces simbólicos
  <Directory />
    AllowOverride    None
    Options    FollowSymLinks
  </Directory>

  # Opciones para los directorios de los usuarios (* es un comodín)
  # Comprobamos que los usuarios no hayan puesto enlaces simbólicos
  <Directory /home/*/public_html>
    Options    -FollowSymLinks
  </Directory>

  # Opciones adicionales para pepito. Antes estaban en un .htaccess
  <Directory /home/pepito/public_html/proyectosecreto>
    AuthUserFile    /home/pepito/.htpassword
  </Directory>

$ sudo /etc/init.d/apache2 restart
```

Afinar Apache: DNS inverso

Apache anota en los logs los nombres de las máquinas que han realizado peticiones, en lugar de su IP. Para ello ha de realizar solicitudes de DNS inverso por cada IP que le solicita algo.

Desactivemos este comportamiento. Luego podemos utilizar la herramienta *logresolve* para encontrar los nombres en los logs:

```
$ sudo vim /etc/apache2/apache2.conf

  HostnameLookups    off

$ sudo /etc/init.d/apache2 restart
```

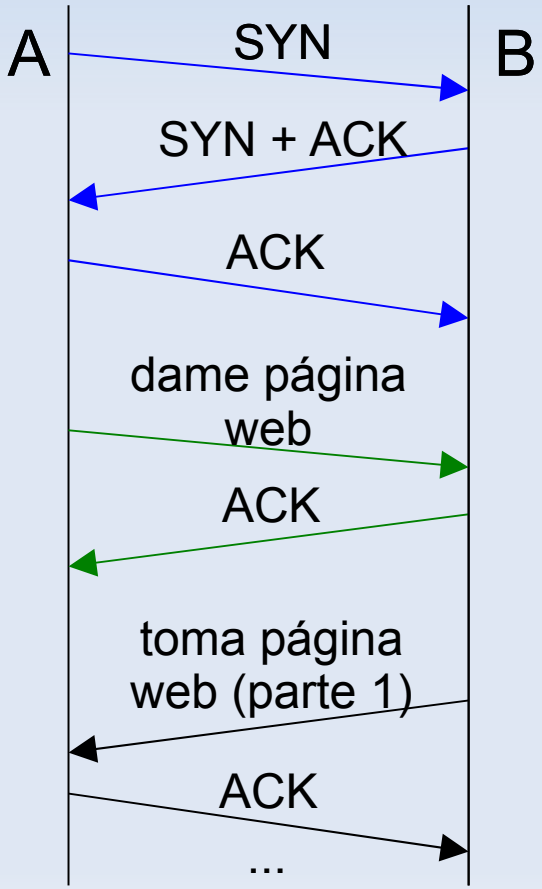
Afinar Apache: DNS inverso

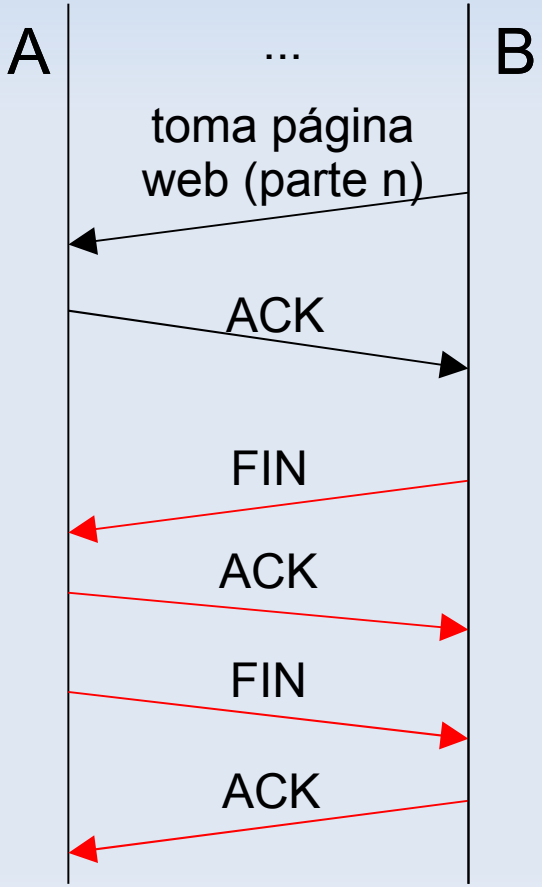
Lo mismo pasa si al usar las directivas “Allow from” o “Deny from” utilizamos nombres de equipos en lugar de Ips.

(Estas directivas son para permitir o denegar a algunas máquinas el acceso a una parte de la web.)

Afinar Apache: persistencia

Navegar = conectar + pedir página web + recibir página web + desconectar





Afinar Apache: persistencia

En lugar de establecer una conexión TCP/IP para cada petición, podemos enviar varias peticiones (por ejemplo: html + imágenes) aprovechando la misma conexión abierta.

```
$ sudo vim /etc/apache2/apache2.conf

  # Máximo de peticiones por conexión
  KeepAlive    5

  # Máximo de segundos de espera a una nueva petición.
  # Si la petición no llega en este tiempo, cerramos la conexión.
  KeepAliveTimeout    2

$ sudo /etc/init.d/apache2 restart
```

Afinar Apache: compresión

El servidor web Apache puede enviar las páginas web comprimidas, utilizando el módulo *mod_deflate* (ver documentación del módulo para aprender cómo).

- Ventaja: ahorro en el ancho de banda y descargas más rápidas.
- Desventaja: mayor uso de CPU en el servidor.

Afinar Apache: servidores estático+dinámico

Mediante los módulos *mod_rewrite* y *mod_proxy* podemos tener dos servidores simultáneamente:

- Un servidor “ligero” (Apache compilado estáticamente con los mínimos módulos, o NginX) sirviendo contenido estático (html, imágenes, etc).
- Un servidor “pesado”: sirviendo contenido dinámico. Recibe del servidor ligero las peticiones que éste no sirve.

Afinar Apache: servidores estático+dinámico

Ejemplo de configuración en el servidor Apache ligero (suponemos el ligero escuchando en el puerto 80 y el pesado escuchando en el puerto 8088):

```
$ sudo vim /etc/apache2/apache2.conf
```

```
<VirtualHost *:80>
ProxyPassReverse / http://%{HTTP_HOST}:8088/
RewriteEngine on
RewriteCond  %{REQUEST_URI} !.*\.(html|css|gif|png|jpg|pdf|gz|zip|tgz)$
RewriteRule ^/(.*) http://%{HTTP_HOST}:8088/$1 [P]
</VirtualHost>
```

```
$ sudo /etc/init.d/apache2 restart
```

- Más sobre hosts virtuales: http://httpd.apache.org/docs/2.2/vhosts/
- *mod_proxy* y *mod_rewrite*: http://httpd.apache.org/docs/2.2/mod/

Afinar Apache: servidores estático+dinámico

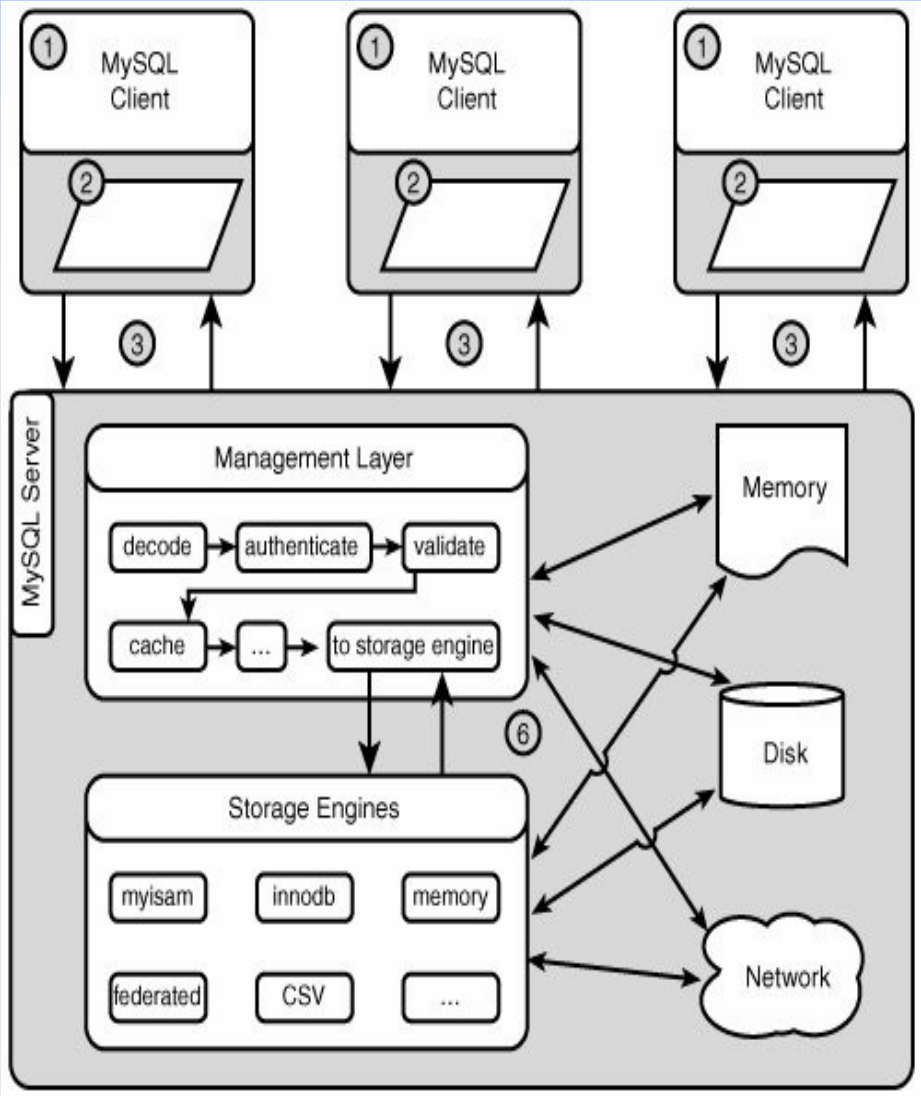
Segundo ejemplo de configuración (suponemos el servidor ligero en el puerto 81 sirviendo imágenes y el pesado en el puerto 80):

```
$ sudo vim /etc/apache2/apache2.conf
```

```
LoadModule proxy_module          libexec/apache2/mod_proxy.so
LoadModule proxy_connect_module libexec/apache2/mod_proxy_connect.so
LoadModule proxy_http_module     libexec/apache2/mod_proxy_http.so
...
<VirtualHost *:80>
  ProxyRequests Off
  ProxyPreserveHost On
  ProxyPass /images http://0.0.0.0:81/
  ProxyPassReverse / http://0.0.0.0:81/
</VirtualHost>
```

```
$ sudo /etc/init.d/apache2 restart
```

Afinar MySQL: ¿Cómo funciona?



1. Los clientes se conectan a servidor.
2. Los clientes inician autentificación, codifican y envían peticiones, comprimen y cifran peticiones, cachean los resultados del servidor, ...
3. El servidor procesa peticiones y devuelve respuestas.
4. Las peticiones son procesadas primero por la capa de manejo, que las desencripta, valida su sintaxis, las busca en la caché, y las envía al correspondiente motor de almacenamiento.
5. Los motores de almacenamiento (MyISAM, InnoDB, Memory, ...) manejan la representación en memoria y disco de bases de datos, tablas e índices, así como generación de estadísticas y algunos logs.
6. La capa de manejo escribe logs a disco, guarda y lee caches en memoria, lee logs binarios de la red, ... Los motores de almacenamiento guardan datos (tablas, logs, ...) en disco y en memoria, envía datos a otros servidores remotos, ...

Afinar MySQL: ¿Cómo funciona?

Table: book

book_id	title	author	cond
1	In the Night Kitchen	Maurice Sendak	mint
2	How to Be a Grouch	Caroll Spinney	poor
3	Jacob Two-two...	Mordecai Richler	good
4	Green Eggs and Ham	Dr. Seuss	fine

Table: person

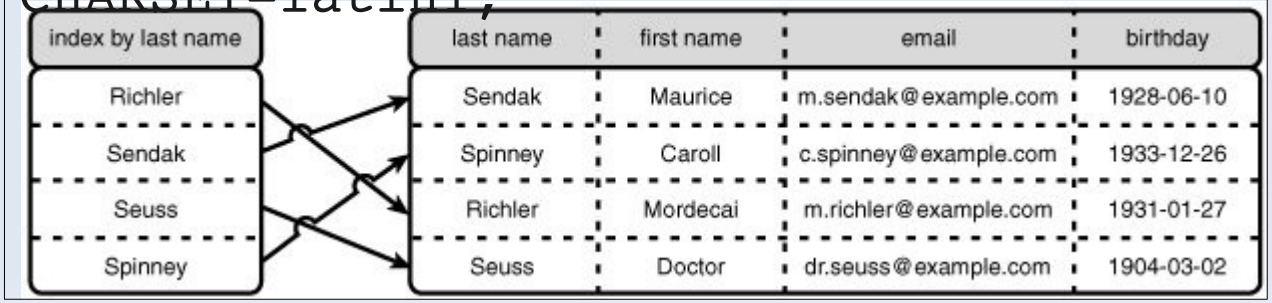
person_id	name	email
1	Yvette	yvette@example.com
2	Lenz	lenz@example.com
3	Thies	thies@example.com
4	Harmony	harmony@example.com

Table: loan

loan_id	book_id	person_id	date_lent
1	2	3	2005-07-17
2	3	3	2005-07-17
3	1	4	2005-09-10
4	4	1	2005-06-05
5	4	3	2005-10-18

- Tablas, índices, claves, claves externas, ...

```
CREATE TABLE `kernelpanic` (
  `last_name` char(30) NOT NULL,
  `first_name` char(30) NOT NULL,
  `email` char(40) NOT NULL,
  `birthday` timestamp default '',
  PRIMARY KEY  (`email`),
  INDEX (`last_name`),
) ENGINE=MyISAM DEFAULT
CHARSET=latin1;
```



Afinar MySQL: ¿Cómo funciona?

http://dev.mysql.com/doc/refman/5.0/es/storage-engines.html

Varios motores de almacenamiento de tablas. Los dos más importantes son:

- MyISAM: no transaccional, muy rápido en lectura y escritura, bajo requerimiento de espacio y memoria.
- InnoDB: transaccional, recuperación de datos, concurrencia más segura en escritura, rollbacks.

```
CREATE TABLE nombre_tabla (definición) ENGINE = nombre_motor;
ALTER TABLE nombre_tabla ENGINE = nombre_motor;
```

Afinar MySQL: fijar límites

Debemos asegurarnos que *mysqld* no deja el sistema sin recursos:

```
$ sudo vim /etc/mysql/my.cnf
...
[mysqld]
; Máximo número de conexiones simultáneas permitidas.
; Para saber el máximo utilizado ejectutar la sentencia SQL:
; SHOW STATUS LIKE 'max_used_connections';
set-variable=max_connections=500

; Máximo tiempo de vida de conexión sin enviar información
set-variable=wait_timeout=10

; Máximas peticiones erroneas antes de bloquear el cliente
max_connect_errors = 100
...

$ sudo /etc/init.d/mysql restart
```

Afinar MySQL: peticiones lentas

- Detectar las peticiones lentas:

```
$ sudo vim /etc/mysql/my.cnf
...
[mysqld]
log-slow-queries = /var/log/mysql/mysql-slow.log
long_query_time = 2
log-queries-not-using-indexes
...

$ sudo /etc/init.d/mysql restart
$ mysqldumpslow
```

- ¿Cómo se ejecuta una sentencia SQL?:
  `EXPLAIN SELECT ...;`
- Soluciones: indexar la tabla, utilizar campos de longitud fija, vigilar joins, ...

Afinar MySQL: buffers

- Ver el tamaño de los buffers:

```
$ mysqladmin variables -u usuario -p | grep buffer
$ mysqladmin extended-status -u usuario -p | grep buffer
$ mysqld --verbose --help | grep buffer

bulk_insert_buffer_size           8388608
join_buffer_size                  131072
key_buffer_size                   16777216
read_buffer_size                  131072
sort_buffer_size                  2097144
...
```

- Aumentar el tamaño de los buffers:

```
$ sudo mysqld_safe   --key_buffer_size=64M   --table_cache=256 \
                     --sort_buffer_size=4M   --read_buffer_size=1M &
```

Afinar MySQL: buffers

- “key_buffer_size”: memoria que guarda los índices de tablas MyISAM. Debería ser suficientemente grande para contener todos los archivos “*.MYI”. (En servidores MySQL MyISAM dedicados entre ¼ y ½ de la memoria total de la máquina).

```
mysql> SHOW STATUS LIKE '%key_read%';

+-------------------+------------+
| Variable_name     | Value      |
+-------------------+------------+
| Key_read_requests | 3606100254 |
| Key_reads         |    2594030 |
+-------------------+------------+
```

La proporción Key_read_requests:Key_read debería ser mayor que 100:1

Afinar MySQL: buffers

- “innodb_buffer_pool_size”: memoria que guarda los índices y datos de tablas InnoDB. (En servidores MySQL InnoDB dedicados el 80% de la memoria total de la máquina).
- “innodb_additional_mem_pool_size”: memoria que guarda los diccionarios de datos de tablas InnoDB. Debería ser suficientemente grande para contener todos los diccionarios de datos.

Afinar MySQL: buffers

- “table_cache”: número máximo de tablas abiertas en memoria por threads *mysqld*.

```
mysql> SHOW STATUS LIKE 'open%tables%';

+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Open_tables   |   98  |
| Opened_tables | 1513  |
+---------------+-------+

Si Open_tables = table_cache (caché al màximo) y Opened_tables >>
  Open_tables entonces se debería incrementar table_cache.
(Open_tables: número de tablas actualmente abiertas)
(Opened_tables: número de tablas que han sido abiertas)
```

Afinar MySQL: cachés

Cada vez que se procesa una petición el servidor debe revisar la sintaxis, planificar la ejecución, y recuperar los datos de disco y devolverlos al cliente.

Podemos establecer caché para peticiones repetidas.

```
$ sudo vim /etc/mysql/my.cnf
[mysqld]
query_cache_limit       = 1M
query_cache_size        = 32M
...

$ sudo /etc/init.d/mysql restart
```

Afinar MySQL: cachés

- Ver y cambiar el tamaño de las caches:

```
$ mysqladmin variables -u usuario -p | grep query_cache
$ mysqladmin extended-status -u usuario -p | grep Qcache
$ mysqld --verbose --help | grep cache
```

```
Qcache_free_blocks              5216 (fragmentación de la caché)
Qcache_free_memory          14640664   (memoria libre en la caché)
Qcache_hits               2581646882   (peticiones servidas por la caché)
Qcache_inserts             360210964   (peticiones metidas en la caché)
Qcache_lowmem_prunes       281680433   (veces limpiar por poca memoria)
Qcache_not_cached           79740667   (peticiones no metidas en caché)
Qcache_queries_in_cache        16927   (peticiones actualmente en caché)
Qcache_total_blocks            47042   (bloques de memoria de la caché)
```

- Coste de mantenimiento: ¡cuidado con caché pequeña (pocos aciertos) o demasiado grande (todo en caché)!

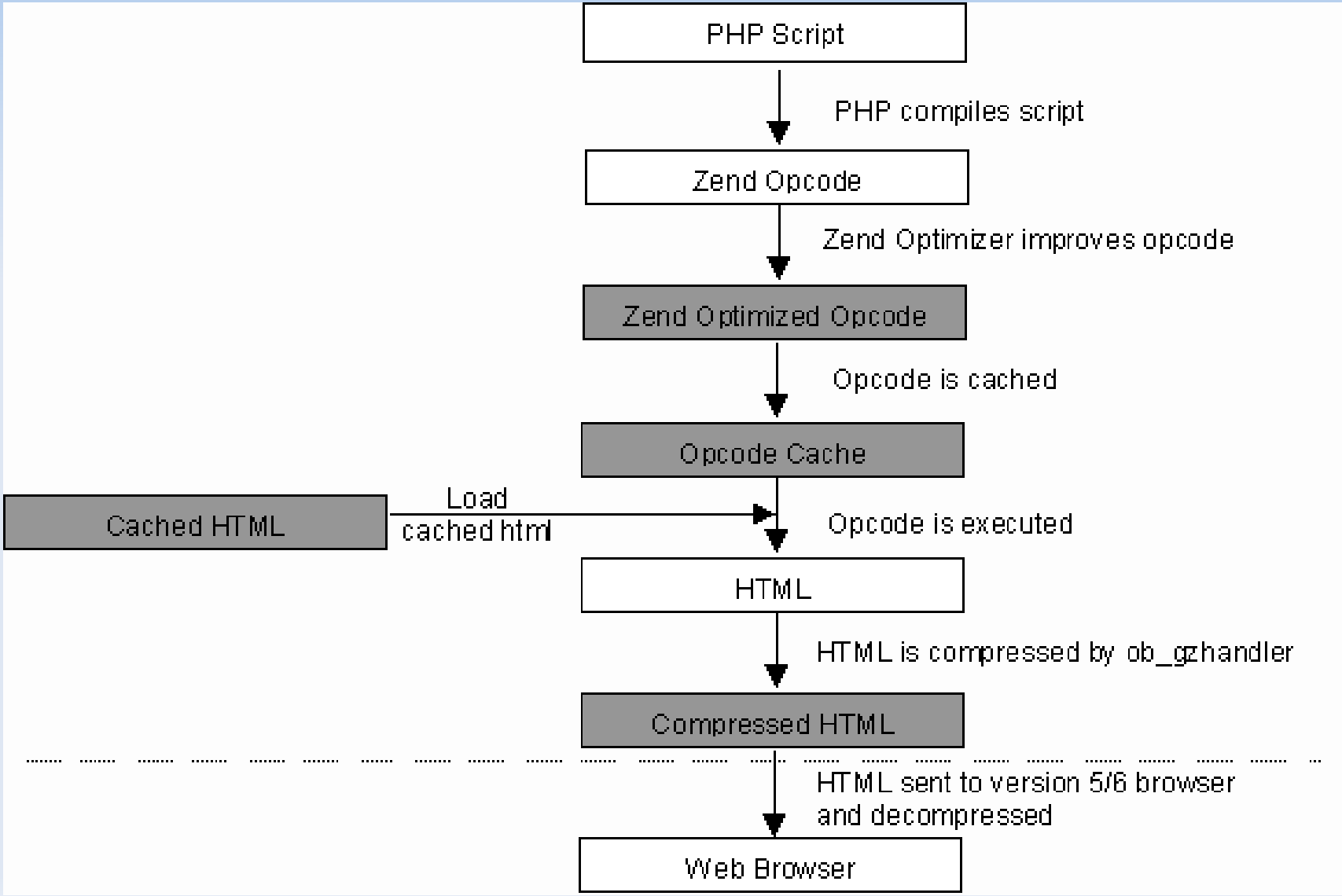
Afinar MySQL: otros ...

- Valor “thread_cache_size”:
  `SHOW STATUS LIKE 'threads%';`
- Valor “tmp_table_size”:
  `SHOW STATUS LIKE 'created_tmp%';`
- Valor “sort_buffer_size”:
  `SHOW STATUS LIKE 'sort%';`
- Indexar las tablas (con índices cortos):
  `ALTER TABLE tabla ADD INDEX (columna,...);`
- Desfragmentar las tablas:
  `OPTIMIZE TABLE tabla;`

Afinar MySQL: herramientas

- mytop: informa qué está pasando en el servidor (conexiones activas, peticiones, estadísticas, ...)
  http://jeremy.zawodny.com/mysql/mytop/
- mysqlard: gráficas a largo término de uso de la caché, eficiencia de las claves, ...
  http://gert.sos.be/en/
- mysqlreport: analiza las variables de estado y aconseja mejoras
  http://hackmysql.com/mysqlreport

Afinar PHP: ¿Cómo funciona?



Afinar PHP: cachear opcode

Cuando se solicita una página dinámica PHP: (1) lee el script, (2) lo compila a “opcode”, (3) lo ejecuta, y (4) **lo olvida**.

Instalar y configurar una caché de opcode:

- APC: http://pecl.php.net/package/APC
- eAccelerator: http://eaccelerator.net/
- Xcache: http://trac.lighttpd.net/xcache/

Afinar PHP: pregenerar html

Generar repetidamente contenido dinámico es costoso. Podemos pregenerar las páginas html de salida y así servir el máximo de contenido estático.

La pregeneración: se puede hacer una vez al día, o bien cuando el contenido asociado a una página cambie, etc.

Todos los gestores de contenido (Wordpress, Drupal, Joomla, MediaWiki, ...) tienen caches que guardan en html las páginas solicitadas.

Afinar PHP: aumentar recursos

Valores recomendados en el fichero php.ini

```
$ sudo vim /etc/php5/apache2/php.ini

;Cuantos segundos de CPU puede consumir un script
max_execution_time   30

;Cuantos segundos puede esperar datos de entrada un script
max_input_time       60

;Cuantos bytes de memoria puede consumir un script sin ser eliminado
memory_limit     32M

;Cuantos bytes de datos se guardan en buffer antes de enviar al cliente
output_buffering 4096

;Loggear lo mínimo imprescindible
error_reporting = E_COMPILE_ERROR|E_ERROR|E_CORE_ERROR

$ sudo /etc/init.d/apache2 restart
```

Referencias

Nivel básico:

- Tuning LAMP systems, Part 1: Understanding the LAMP architecture
  http://www.ibm.com/developerworks/linux/library/l-tune-lamp-1/
- Tuning LAMP systems, Part 2: Optimizing Apache and PHP
  http://www.ibm.com/developerworks/linux/library/l-tune-lamp-2.html
- Tuning LAMP systems, Part 3: Tuning your MySQL server
  http://www.ibm.com/developerworks/linux/library/l-tune-lamp-3.html

Nivel avanzado:

- Linux Performance and Tuning Guidelines
  http://www.redbooks.ibm.com/redpieces/pdfs/redp4285.pdf
- Apache Performance Tuning
  http://httpd.apache.org/docs/2.2/misc/perf-tuning.html
- MySQL 5.1 documentation, Chapter 7: Optimization
  http://dev.mysql.com/doc/refman/5.1/en/optimization.html