Facultad: Ingeniería
Escuela: Computación
Asignatura: PHP

Tema: Conceptos de Optimización PHP

Contenido

PHP es un lenguaje de programación veloz y eficiente, sin embargo existen muchas tareas de optimización complementarias, que no precisamente tienen que ver con el código y pueden mejorar su ejecución.
En esta guía de laboratorio se explica el porqué la optimización de PHP, involucra muchos factores que no están relacionados directamente con aspectos de programación y porque la optimización de PHP requiere un entendimiento de como PHP interactúa con otros procesos en el servidor (arquitectura). Luego identificar los cuellos de botellas causados por esos procesos y como eliminarlos. Así mismo se realizaran tareas básicas de optimización de programas en PHP.

Objetivo Especifico

- Conocer conceptos de optimización de servidor PHP.
- Ejecutar tareas básicas de administración del servidor de PHP/apache.
- Realizar tareas de optimización de código.

Material y Equipo

- Windows 7
- Stack WAMP
- Notepad

Introducción Teórica

Dado que existe una gran variedad de hardware y diferentes tipos de contenido que pueden servirse en internet, ajustar la configuración de Apache para extraer el mayor rendimiento posible de un servidor es algo necesario. Es por ello que se deben de realizar todas las tareas necesarias, a nivel de código y de configuración para garantizar un buen rendimiento de las aplicaciones Web.

Al hablar de buen rendimiento de una aplicación, no se hace referencia solamente a la velocidad de ejecución de un script. El rendimiento hace referencia a un conjunto de "Tradeoffs" entre la rapidez, exactitud y escalabilidad. Un ejemplo de rapidez versus exactitud puede ser un script que deba de ser optimizado para ejecutarse rápidamente por medio de almacenamiento en caché; Sin embargo los datos crecerán desactualizados y menos exactos.

PHP es un lenguaje de programación interpretado. Esto significa que está diseñado para ser ejecutado por medio de un intérprete, en contraste con los lenguajes compilados que son analizados una sola vez por el compilador. Es decir, en lugar de compilarse el programa una vez,

PHP se compila sobre la marcha cada vez que va a ejecutarse. La gran ventaja es que cualquier aplicación codificada en PHP no necesita ser recompilada si se cambia de sistema operativo, mientras que su desventaja principal es que cada vez que se ejecuta un archivo .php tiene que ser compilado en tiempo de ejecución nuevamente. Este problema puede solventarse mediante un framework que almacene en cache los resultados de esta compilación.

Procedimiento

1. Identificación de cuellos de Botella

Los cuellos de botella hace referencia a una serie de condiciones que ralentizan (disminuyen) el proceso de ejecución global. Para una aplicación que realiza alguna técnica de alojamiento en memoria temporal (caché) un cuello de botella podría ser las dimensiones de memoria RAM, sin embargo existen muchos más factores a tomar en cuenta:

a) Red (Networking)

El acceso a red es el cuello de botella más importante a tomar en cuenta en la administración de servidores web. Dimensione de cuanto es el acceso a Internet en su equipo con http://www.speedtest.net/

¿Cuánto es la velocidad de Carga (upload)?_____ mbps

Si se tiene un promedio de 30K por pagina en su servidor, ¿cuántas paginas podrán saturar el enlace?_____

¿Que otros servicios de red se encuentran activos en su máquina?:

¿Pueden causar impacto en su aplicación?_____

b) Procesamiento (CPU)

Para un servidor enviar HTML no causa problemas de procesamiento, sin embargo PHP obliga al procesador realizar acciones adicionales que pueden causar lentitud en las aplicaciones. Dimensione las características importantes del procesador de su computadora:

_____
_____
_____
_____

C) Sistemas de Archivos:

Verifique el estado del disco duro:

Fragmentación: _____
Espacio en disco (%): _____
Sistema de Archivos: _____

e) Administración de procesos (Hilos).

Evitar saturar el servidor WEB con procesos innecesarios. Identificar y Listar Procesos, tareas y programas que se ejecutan y son innecesarios:

2. Optimización de APACHE.

a) Configurar el archivo httpd-mpm.conf, con la directiva ThreadsPerChild con valores superiores a 512 y múltiplos de 1024

```
ThreadsPerChild      150 // valor por default
```

b) Configurar el archivo httpd.conf con las siguientes directivas

```
SendBufferSize 512000
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15
MinSpareServers 5
MaxSpareServers 10
StartServers 5
MaxClients 150
MaxRequestsPerChild 0
```

c) Si no se requiere tener activo DNS y no se accede al archivo htaccess se puede configurar lo siguiente:

```
# PHP tomara solamente las ips

HostnameLookups off

# desabilitar acceso al archivo htaccess

<Directory />

AllowOverride none

</Directory>
```

3. Optimización de Código PHP.

a) Evite copiar nombres de variables

Cuando una variable se utilizará solamente una vez, evalué eliminar la variable.

```
$Texto = strip_tags($_POST['description']);
echo $Texto;
```

Código Optimizado:

```

```

b) Elimine parámetros de referencias innecesarios

PHP no provee ventajas a strings, integers y otros tipos de datos básicos, el tenerlos como referencia en. Se debe evitar su uso si no tiene una razón.

```
function VarRef(&$a)
{
    $b = $a;
    $c = $a;
}
$Var1 = 1;
VarRef($Var1);
```

Código Optimizado:

```

```

c) Elimine funciones set y get innecesarias:

Usualmente se utilizan funciones que solamente establecen y devuelven valores en la definición de clases. Si no realizan algún tipo de validación deben de eliminarse.

	Código Optimizado

`class Perro {`
`  public $Nombre = '';`
 	
`  public function setNombre($Nombre) {`
`  $this->name = $name;`
`  }`
 	
`  public function getNombre() {`
`  return $this->Nombre;`
`  }`
`}`
 	
`$ranger = new Perro();`
`$ranger ->setNombre('Katinka');`
`echo $rover->getNombre();`	

d) Evite ejecutar consultas a Bases de Datos dentro de loops

Esto produce lentitud en el proceso global de ejecución de scripts php y puede ocasionar bloqueos a la Base de datos.

```
foreach ($userList as $user) {
  $query = 'INSERT INTO users (first_name,last_name) VALUES("' .
$user['first_name'] . '", "' . $user['last_name'] . '")';
  mysql_query($query);
  }
```

Código Optimizado:

```

```

Investigación complementaria

- Investigue sobre ab (Apache HTTP server benchmarking tool, incluida con Apache) y que tipos de pruebas se pueden realizar con esta herramienta.
- Investigue sobre Memcache y otras utilidades para utilizar cache.
- Investigue mas normas de desarrollo para optimizar código PHP

Hoja de cotejo: 12

Guía 12: Conceptos de Optimización PHP

Alumno:

Maquina No:

Docente:

GL: Fecha:

EVALUACION					
	%	**1-4**	**5-7**	**8-10**	**Nota**
CONOCIMIENTO	Del 20 al 30%	Conocimiento deficiente de los fundamentos teóricos	Conocimiento y explicación incompleta de los fundamentos teóricos	Conocimiento completo y explicación clara de los fundamentos teóricos	
APLICACIÓN DEL CONOCIMIENTO	Del 40% al 60%				
ACTITUD					
	Del 15% al 30%	No tiene actitud proactiva.	Actitud propositiva y con propuestas no aplicables al contenido de la guía.	Tiene actitud proactiva y sus propuestas son concretas.	
TOTAL	100%				