



*TFC: ARQUITECTURA DE COMPUTADORES  
Y SISTEMAS OPERATIVOS*

# **MEMORIA**

## **GESTIÓN DE PROCESOS EN LOS SISTEMAS OPERATIVOS**

**ALUMNO: Francisco Javier Serrano Castaño**

**CONSULTOR: Francesc Guim Bernat**

## Resumen

En los últimos años se puede apreciar cómo, dentro de los tres Sistemas Operativos más utilizados para computadoras personales, hay una especie de batalla por presentar cual ofrece mejores prestaciones. Estos tres sistemas mencionados son Windows 7 de Microsoft [8], Ubuntu de Linux [6] y Mac OS X de Apple [7].

Por ello en este Trabajo Fin de Carrera (TFC) se realizará una comparación de los tres sistemas operativos mencionados respecto al rendimiento que ofrecen sobre el servidor web Apache instalado en cada uno de ellos. Antes de realizar dichas pruebas, se describirán cómo cada uno de los sistemas operativos maneja la gestión de procesos, gestión de memoria, etc. Se realizará una descripción de cada uno de ellos respecto sobre cómo realizan dichas funciones. Con ello se pretende tener una base sobre la cual fundamentar posteriormente los resultados obtenidos de las pruebas con los procesos que genere el servidor web Apache [11].

Cabe mencionar que la instalación de los sistemas operativos será llevada a cabo a través de máquinas virtuales, utilizando para ello como software de creación y gestión de las mismas VMware [13]. Se utilizará dicho software por la experiencia y robustez para albergar a los tres sistemas operativos a evaluar. Se ha elegido este tipo de instalación debido al creciente auge de la instalación de equipos virtuales, por lo que se podrá comprobar también el rendimiento de cada uno de los sistemas instalados en dichas máquinas virtuales.

La realización de la práctica se utilizará en concreto los siguientes Sistemas Operativos: Windows 7 Profesional, Ubuntu 10.10 y Mac OS X 10.6. En cada uno de los sistemas se configurará el servidor Web Apache, al que se le realizarán una serie de test con los que se podrá comprobar las diferencias de rendimiento de unos sistemas a otros.

## *Índice General*

### 1- Introducción

1.1 Objetivos del TFC

1.2 Enfoque del TFC

1.3 Planificación del TFC

1.3.1 Recursos de software y hardware

1.3.2 Tareas y coste del proyecto

### 2- Sistemas Operativos

2.1 ¿Qué es un Sistema Operativo?

2.2 Clases de Sistemas Operativos

2.3 Máquina Virtual

### 3- Gestión de Procesos

3.1 Definición

3.2 Estados de un proceso

3.3 Planificación de un proceso

3.4 Creación y destrucción de procesos

### 4- Gestión de Memoria

4.1 Memoria principal

4.2 Memoria virtual

4.2.1 Traducción de página virtual a física

4.2.2 Paginación

4.2.3 Segmentación

4.3 Gestión de Memoria en UNIX

4.4 Gestión de Memoria en Windows

## 5- Implementación

### 5.1 Hypervisor

#### 5.1.1 Concepto de Hypervisor

#### 5.1.2 Instalación del Hypervisor

#### 5.1.3 Creación y configuración de máquinas virtuales

#### 5.1.4 Instalación del Sistema Operativo

### 5.2 Sistemas Operativos

#### 5.2.1 Configuración

#### 5.2.2 Servidor Web apache

##### 5.2.2.1 Conceptos

##### 5.2.2.2 Instalación y configuración

### 5.3 Benchmarking

#### 5.3.1 Descripción y tipos

#### 5.3.2 Instalación y configuración

## 6- Evaluación de los Sistemas Operativos

### 6.1 Test inicial

### 6.2 Cambios de configuración

### 6.3 Test final

## 7- Análisis de resultados

## 8- Conclusiones

## 9- Bibliografía y enlaces

# 1. Introducción

## 1.1. Objetivos del TFC

El objetivo principal del trabajo es la de realizar un estudio comparativo de los diferentes Sistemas Operativos a estudiar respecto al rendimiento que ofrecen respecto a un mismo problema. En este caso se trata de los sistemas; Windows 7 Profesional, Ubuntu 10.10 y Mac OS X 10.6.

El problema a tratar será ver la respuesta del servidor web apache ante un *benchmarking* [24], es decir, utilizar una aplicación que mediante técnicas de fuerza pueda medir el rendimiento de ciertas aplicaciones o hardware, en este caso de la aplicación Apache, y compararlo entre los tres sistemas operativos mencionados. Las aplicaciones que se utilizarán para realizar el *benchmarking* son: *JMeter* [25] y *ab* [26].

## 1.2. Enfoque del TFC

Este proyecto se inicia con una fase teórica en la que se realizará una breve descripción de distintos aspectos de dicho trabajo, entre los cuales nos encontramos la definición de sistema operativo y las clases de sistemas que vamos a tratar en el trabajo. Además se habla de qué es una máquina virtual, ya que en este trabajo los sistemas a utilizar serán instalados sobre máquinas virtuales.

Otros aspectos a destacar, son los relativos a la gestión de procesos y gestión de memoria por parte de los sistemas a tratar. Se explicarán en qué consiste esta gestión y cómo realiza la misma cada uno de los sistemas operativos, por lo que se puede ir realizando una primera comparación de los sistemas respecto al tratamiento que cada uno realiza en su funcionamiento o procesamiento de tareas.

En una segunda fase más práctica, se realizará la instalación de un Hypervisor o monitor de máquina virtual, en este caso del Hypervisor VMware, en una máquina para posteriormente crear las máquinas virtuales donde se instalarán cada uno de los sistemas operativos a analizar.

En estos sistemas operativos tendremos que instalar el servidor web apache, que será el proceso que medirá el rendimiento de los mismos. Este servidor será objeto de distintas configuraciones para ver los cambios de rendimiento producidos en el mismo.

Para medir el rendimiento del servidor web Apache se utilizarán herramientas de benchmarking [24]. Estas herramientas serán instaladas y configuradas en un equipo distinto al utilizado anteriormente para realizar las mediciones de rendimiento de los mismos.

En una última fase, se analizarán los resultados obtenidos en los test, y se realizarán las conclusiones pertinentes.

### 1.3. Planificación del TFC

#### 1.3.1 Recursos de hardware y software

En este apartado se detallan los recursos necesarios para llevar a cabo el trabajo. Respecto al hardware, necesitamos lo siguiente:

- Dos PCs con un equipamiento similar al siguiente: procesador Core 2 Duo o similar, 4 GB memoria RAM, 160 GB de HD (Disco duro), tarjeta de red mínimo a 100 MB/s. El coste de dichos equipos rondan los 450 €
- Conexión entre los dos PC, ya sea mediante un cable cruzado conectado directamente sobre las tarjeta de red de los dos equipos, es la manera más económica, o bien a través de un conmutador o switch.

Respecto al software necesario:

- El PC que realizará el test tendrá instalado Windows 7 Profesional, con un coste de 278 €.
- El PC con las máquinas virtuales tendrá instalado un Hypervisor sobre Windows 7 Profesional (coste 278 €), se va a utilizar VMware WorkStation 7 [16], debido a su potencial, compatibilidad con los sistemas operativos a analizar. Coste 153€.
- Los sistemas operativos que se analizarán; Windows 7 Profesional, Ubuntu 10.10 y Mac OS X 10.6. Excepto Ubuntu los otros dos sistemas son propietarios, con un coste aproximado de 278 € en el caso de Windows y 29€ en el caso de MAC OS X 10.6.
- Servidor web Apache [11], que será el proceso a analizar en cada uno de los sistemas, es una aplicación gratuita.

- Las herramientas de Benchmarking; *JMeter* [25] y *ab* [26]. Todas las elegidas son gratuitas y relacionadas con Apache.

### 1.3.2 Tareas y coste del proyecto

En este apartado se va a exponer las tareas en las que consta el proyecto, así como el coste en tiempo empleado en cada una. Se ha estimado que el coste por hora de trabajo es de 50 €/h, en el cual se ha incluido las licencias, consumo eléctrico y coste de la persona/h. En el mismo no se ha contemplado el coste de hardware de los equipos, ya que este puede variar según modelos.

- *Creación del plan de trabajo* (total 10 horas). Esta fase considerada como una de las más importantes del mismo, ya que una vez elaborada correctamente nos servirá como guía para el resto del trabajo. En la misma se realiza una búsqueda superficial de información para tener claro el contenido que va a tener dicho trabajo. A esta fase corresponde todo el apartado Introducción del proyecto.
- *Documentación* (Total 5 horas). En esta fase se realiza la búsqueda de documentación necesaria para el proyecto, la cual quedará reflejada en el apartado de bibliografía y enlaces.
- *Comparación teórica de los Sistemas Operativos* (Total 20 horas). En esta fase se hará mención a todos los aspectos técnico-teóricos que expliquen qué es un sistema operativo, y como cada uno de ellos gestionan los procesos, memoria, archivos de sistema, etc.
- *Instalación y configuración del PC con Hypervisor*. (Total 6 horas). Esta tarea consta de la instalación del equipo donde se instalarán las máquinas virtuales. Una vez se instala el sistema operativo Windows 7 Profesional, se instalará el Hypervisor VMware WorkStation 7, tras el cual se crearán 3 máquinas virtuales en cada una de las cuales se instalará uno de los sistemas a evaluar.
- *Configuración de Apache y testeo de los sistemas* (Total 30 horas). Esta fase consta de la instalación de la aplicación Apache en cada una de las máquinas virtuales y la configuración de las mismas para poder ser evaluadas desde otro equipo. Antes de comenzar a realizar las pruebas se ha de configurar el equipo con las herramientas de benchmarking. A lo largo de

esta fase se realizarán distintas pruebas, ya que se harán cambios en la configuración del servidor web apache para ver como repercuten dichos cambios en el rendimiento del mismo.

- *Estudio y análisis de las pruebas* (Total 10 horas). Una vez obtenidos los resultados de los test realizados en cada uno de los sistemas, se compararán y se analizarán, concretando quién ha tenido mejores resultados y en qué condiciones.
- *Elaboración de la memoria* (Total 10 horas). Se recopilarán todos los datos y documentos realizados hasta la fecha, revisando la escritura y presentación de los mismos, de manera que quede maquetada para su entrega.
- *Creación de la presentación* (Total 10 horas). Se realizará el documento de presentación del TFC, donde se plasmarán las ideas más importantes del trabajo.
- *Revisión y entrega*. Durante la última semana, antes de la entrega del TFC, se realizará una última revisión en busca de fallos o mejoras en el mismo, y se procederá a su entrega final.





## 2. Sistemas Operativos

### 2.1. ¿Qué es un Sistema Operativo?

Un sistema operativo se puede definir como la capa de software que equipa a las computadoras, cuya labor es la de administrar y gestionar todo el hardware que interactúa en la misma y proporcionar una interfaz sencilla a los programas para comunicarse con dicho hardware, entendido éste como los procesadores, memoria, discos duros, teclado, pantalla, impresoras, interfaz de red y todo tipo de dispositivo de E/S.

Se puede decir por tanto que el sistema operativo se encuentra entre el hardware y las aplicaciones de usuario (navegadores web, programas de ofimática, etc.) proporcionado así a los programadores de dichas aplicaciones un conjunto de instrucciones conocidas como llamadas al sistema, para trabajar con más comodidad, de lo contrario los programadores tendrían que realizar dichas tareas en un lenguaje de programación que se comunicara directamente con los dispositivos, siendo éste conocido como lenguaje de máquina, lo que complicaría enormemente la labor de estos.

Respecto a la gestión del hardware que realiza el sistema operativo, consiste en administrar los recursos de una computadora cuando dos o más programas que se ejecutan simultáneamente requieren usar el mismo recurso (tiempo de CPU, memoria, etc.), por lo que será el sistema operativo el encargado de ordenar y compartir dichos elementos en dos formas: en el tiempo y en el espacio. Cuando un recurso se multiplexa (comparte) en el tiempo, diferentes programas o usuarios se turnan en su uso, un ejemplo de ello es la cola de impresión que genera el sistema operativo indicando quien imprime primero y quien a continuación. El otro tipo de multiplexación es en el espacio, en este caso lo que ocurre es que el sistema reparte el recurso entre los usuarios y programas. Un ejemplo típico es el uso de la memoria principal, donde, dependiendo del tamaño de la misma, pueden residir distintos programas a la vez a la espera del uso de la CPU.

## 2.2. Clases de Sistemas Operativos

En este punto se realizará una mención por los distintos tipos de Sistemas Operativos, dependiendo del número de aplicaciones que ejecutan, del número de usuarios que interactúan, del número de procesadores que manejan o dependiendo de la estructura. A continuación se enumeran los sistemas mencionados:

- Según la utilización de los recursos. Si un sistema operativo se ejecuta en una computadora con un solo procesador se clasifican en sistemas monoprogramados y multiprogramados. Sin embargo si tiene varios procesadores se habla de sistemas multiprocesamiento.
  - Sistema monoprogramado. Sólo se permite la ejecución de un programa en el sistema, por lo que el programa se carga en memoria y permanece en ella hasta su finalización.
  - Sistema multiprogramado o multitarea. Estos sistemas pueden ejecutar simultánea varias tareas y de varios usuario. El sistema operativo mantiene varios trabajos en la memoria, y mientras uno está a la espera de realizar alguna tarea, por ejemplo la terminación de una operación de entrada/salida (E/S), ya sea el presionar una tecla u otra tarea, el sistema operativo selecciona otro trabajo para que la CPU lo ejecute y esta última no quede vacía de trabajo mientras estos esperan su ejecución. Hoy en día todos los sistemas operativos son multitarea.
  - Sistema multiprocesador. Estos sistemas son aquellos que pueden manejar varios procesadores, con lo que se alcanza un mayor rendimiento ya que al tener más procesadores podemos realizar más tareas al mismo tiempo, aunque hay que decir que este aumento no es proporcional al aumento del número de procesadores. A este tipo de sistema también se le llama computadoras paralelas o multicomputadoras, ya que cada procesador tiene una copia del sistema operativo, y éstas se comunican entre sí para el manejo de las tareas, repartiéndose las mismas entre los procesadores, es decir, una tarea es ejecutada en proporción por todos los procesadores.
- Según la interactividad. En este caso se habla de los sistemas dependiendo del tipo de trabajo y los servicios que se presta a los usuarios se habla de sistemas de procesos por lotes, sistemas de tiempo compartido y sistemas de tiempo real.

- Sistemas de procesamiento por lotes (Batch). En este tipo de sistemas no existe intervención del usuario durante la ejecución de los trabajos. Cada trabajo consiste en una relación de pasos secuenciales, que al juntarse forman un lote. Estos procesamientos son largos y no tienen límite de tiempo en su ejecución.
- Sistemas de tiempo compartido. En este tipo de sistemas los procesos se ejecutan de manera simultánea mientras la CPU conmuta entre ellos, de manera que el usuario es ajeno a la misma. En estos sistemas los usuarios sí pueden interactuar con los programas durante su ejecución.
- Sistemas de tiempo real. Se caracterizan porque maximizan el tiempo en el que se manda la información, por ello se utilizan para procesos delicados, como por ejemplo la ejecución por parte del brazo de un robot en una planta de ensamblaje de vehículos. Dicho robot debe obtener la información de ejecución dentro de unos márgenes estrictos de tiempo, ni antes ni después.
- Según el número de usuarios. En este caso se habla de los sistemas dependiendo del tipo de trabajo y los servicios que se presta a los usuarios se habla de sistemas de procesos por lotes, sistemas de tiempo compartido y sistemas de tiempo real.
  - Sistema operativo de computadora personal. Se puede también denominar sistema monousuario. Atiende al número de usuarios que utilizan dicho sistema al mismo tiempo. En el caso de los monousuario hasta que la persona que está utilizando el equipo no termina, éste no puede ser utilizado por otra. El sistema operativo de computadora personal tiende a ser monousuario, ya que es un sistema enfocado a maximizar la comodidad del usuario y a responder con rapidez a las necesidades de éste. Como ejemplo actual de este sistema podemos mencionar a Windows 7, que excepto en algunas versiones mediante la herramienta de terminal server se puede convertir en multiusuario, la mayoría es monousuario.
  - Sistema operativo servidor o multiusuario. Al contrario que el anterior, en este sistema concurren más de un usuario de forma simultánea desde diferentes ubicaciones. Por ejemplo el caso de los servidores web, los cuales sirven las páginas alojadas en los mismos a través de internet a todo aquel que las visite. Entre los sistemas operativos de servidor encontramos a UNIX, Windows Server y Linux.

- Según el número de computadoras. Dependiendo del número de computadoras que formen parte del sistema, se puede hablar de sistemas centralizados o sistemas distribuidos.
  - Sistemas centralizados. Este tipo de sistemas es aquel que utiliza los recursos de una sola computadora, como es la memoria, CPU, discos, periféricos. Este tipo de sistema planteaba un problema, y es que cuando se interconectaban ordenadores a uno principal, cuando este último requería un cambio de hardware para aumentar su capacidad, ello era más costoso que añadir nuevos computadores.
  - Sistemas distribuidos. Como lo define Tanenbaum [3], *“Un sistema distribuido es una colección de computadoras independientes que aparecen ante los usuarios del sistema como una única computadora”*.  
 En este tipo de sistemas se utiliza al mismo tiempo los recursos de los distintos equipos, ya bien sea el hardware o el software de los mismos, por lo que las computadoras se comunican entre sí a través de distintos medios de comunicación, como pueden ser líneas de alta velocidad o líneas telefónicas. Este tipo de sistemas puede abarcar desde unas pocas computadoras hasta miles o millones de ellas, enlazadas a través de internet. Estos recursos que son compartidos son manejados por un gestor de recursos, que es un software que se encarga de controlar dichos recursos.
- Según la estructura del sistema operativo. En este caso se encuentra los distintos sistemas dependiendo del diseño interno de los mismos. Entre ellos encontramos los sistemas monolíticos, los sistemas en capas, las máquinas virtuales y el sistema cliente-servidor.
  - Sistemas monolíticos. Son sistemas pequeños, sencillos y limitados. No tienen una estructura definida, sino que todos sus componentes se encuentran agrupados en un único programa. Cada conjunto de procedimientos puede invocar a cualquiera de los otros procedimientos. Un ejemplo de este tipo de sistemas es el sistema MS-Dos.
  - Sistema por capas. El sistema se organiza como una jerarquía de capas, donde cada capa ofrece una interfaz a la capa superior, y sólo utiliza los servicios que ofrece la capa inferior. Cada capa tiene funciones específicas, así cada capa se encarga de una parte del sistema operativo.

- Sistema de máquina virtual. Este sistema será tratado de una manera más amplia en el siguiente punto 2.3.
- Sistema cliente-servidor. El sistema consiste en un conjunto de módulos autónomos, los cuales ponen a disposición de los demás una serie de servicios o competencias. Es decir, cada uno de los módulos actúan como servidores que atienden las peticiones de otros módulos que actuarían como clientes. Este tipo de sistema es muy apropiado para los sistemas distribuidos.

### 2.3. Máquina Virtual

Este TFC, como ya se ha descrito, versa sobre las diferencias de gestión entre sistemas operativos, pero estos van a ser instalados en máquinas virtuales creadas al efecto, por lo que cabe hacer una reseña del funcionamiento de dichas máquinas virtuales.

El concepto de máquina virtual surge con el sistema VM/370<sup>1</sup> de IBM, en el que se separa las dos funciones que proporciona un sistema de tiempo compartido o multiusuario, como son la multiprogramación y la abstracción del hardware.

El corazón del sistema de creación de las máquinas virtuales es conocido como monitor de máquina virtual, y se ejecuta sobre el hardware proporcionando varias máquinas virtuales a la siguiente capa de software. Se puede decir que estas máquinas virtuales son copias exactas del hardware desnudo donde se incluyen los recursos principales de una computadora, como es el kernel, dispositivos E/S, interrupciones, memoria principal, etc. A continuación se muestra la estructura de dicho sistema<sup>2</sup>.

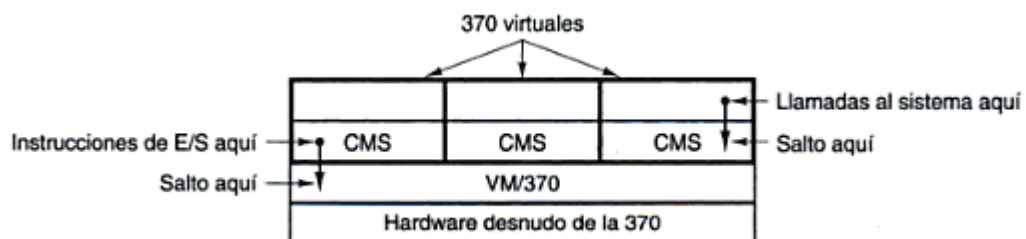


Figura 1 Estructura del VM/370

Cada máquina virtual puede ejecutar cualquier sistema operativo debido a la circunstancia comentada de que cada una es una copia exacta del hardware. En el caso de la figura anterior se puede apreciar como distintos sistemas operativos

<sup>1</sup> Se puede encontrar más información detallada en la web de IBM <http://www.vm.ibm.com/>

<sup>2</sup> Esta figura ha sido extraída del libro **Tanenbaum Andrew S.:** *Sistemas Operativos Modernos*. Ed. Prentice Hall

monousuarios (CMS, *Sistema Monitor de Conversaciones*), cuando realizan una llamada al sistema por medio de algún programa, ésta salta al sistema operativo de la propia máquina virtual, no a la VM/370 como hubiera ocurrido en una máquina real, entendiendo que esta sería la capa de software por encima de la capa de hardware de la computadora. A continuación el CMS emite las instrucciones de E/S necesarias para ejecutar la llamada, el sistema VM/370 recoge estas instrucciones y las ejecuta como parte de su simulación de hardware real.

Esta idea de la virtualización también se encuentra en la ejecución de programas, en lo que se conoce como máquinas virtuales de proceso o máquinas virtuales de aplicación. Fue Sun Microsystem [21] quién al inventar el lenguaje de programación Java, también inventó la máquina virtual java (JVM) [22]. Consiste en que al ejecutarse una aplicación en este lenguaje, las instrucciones no son ejecutadas por el procesador ni por el hardware de la máquina, sino que son ejecutadas por un software, en este caso por la JVM que es una máquina virtual, que las interpreta, y lleva a cabo una acción sobre el hardware a partir de estas. Esta característica de ser portable es fundamental, ya que se puede ejecutar en cualquier máquina que tenga un intérprete de JVM sin necesidad de cambio del código, aun ejecutándose en distintos sistemas operativos, e incluso acceder vía internet a aplicaciones remotas y ejecutarse.

Otra máquina virtual de proceso muy utilizada es la proporcionada por el entorno .Net de Microsoft [23], que recibe el nombre de *Common Language Runtime*.

Respecto a las máquinas virtuales de sistema podemos decir que hay dos tipos o clases que son las más comunes; las que se ejecutan directamente sobre el hardware o las que se ejecutan sobre un sistema operativo.

La diferencia entre una u otra es que en la primera no tenemos un sistema operativo entre las máquinas virtuales y el hardware, en cambio en las segundas, las máquinas virtuales se sitúan en un nivel superior al del sistema operativo que alberga el monitor de máquinas virtuales (VMM). Vemos dichas diferencias en las siguientes figuras<sup>3</sup> 2 y 3.

---

<sup>3</sup> La fuente de los gráficos es del TechNet Magazine: <http://64.4.11.252/es-es/magazine/2008.10.hyperv.aspx>



Figura 2 Monitor sin Sistema Operativo



Figura 3 Monitor con Sistema Operativo

Dentro de los distintos VMM o Hypervisor los más usuales son: VMware en las versiones ESXi [15] y Workstation [16], XEN de Citrix [17], Virtual Box de Sun [18], Virtual PC 2007 de Microsoft [19], Hyper V de Microsoft [20].

Estas máquinas virtuales se utilizan hoy en día para diferentes entornos, los más utilizados son los entornos de desarrollo o pruebas, pero también se están utilizando cada vez más para entornos en producción, así se puede encontrar por internet como muchos ISP ofrecen sus servicios de hosting mediante servidores virtuales. También se utiliza la virtualización para consolidar servidores en una sola máquina con suficientes recursos para ello, así tendremos las máquinas aglutinadas en una misma computadora, a las cuales se podrá hacer una copia de seguridad del sistema completo, ya que este se compone de un solo fichero. Este último punto junto con la posibilidad que nos ofrecen las máquinas virtuales de ser traspasadas de un host a otro con el mismo tipo de VMM, nos ofrecen un entorno en el que la pérdida de un servicio gestionado por una máquina virtual puede ser mínimo, al poderse restaurar dichas máquinas virtuales de una manera fácil entre los distintos hosts. Otro de los beneficios que proporcionan las máquinas virtuales y por lo que son elegidas es el ahorro de energía. No consume lo mismo una máquina física con 10 máquinas virtuales que 10 máquinas físicas.

Otro tipo de virtualización es la llamada virtualización a nivel de sistema operativo. Esta técnica consiste en dividir una computadora física en varias con compartimentos independientes en los cuales se puede instalar un sistema operativo diferente. Así el sistema operativo de la máquina virtual comparte el mismo kernel que el sistema usado para implementar dichas máquinas virtuales. Las aplicaciones de un sistema operativo que corren en una máquina virtual, lo ven como un sistema autónomo. Dentro de esta categoría nos encontramos distintos ejemplos: Virtuozzo [28], Linux-VServer [29], Solaris containers [30], OpenVZ [31] y FreeBSD Jails [32].

Podemos decir por tanto que la virtualización es cada vez más utilizada y más extendida en una amplia variedad de ámbitos, desde un usuario que quiere probar distintos sistemas



operativos hasta llegar a algo muy de moda como es el *cloud computing* (computación en la nube), cuyos servidores están cada vez más virtualizados.

Debido a este interés por la virtualización de servidores, se ha decidido utilizar máquinas virtuales para ver no sólo el rendimiento de los sistemas operativos ante un problema dado, sino también como estos responden siendo instalados en máquinas virtuales.

## 3. Gestión de procesos

### 3.1. Definición

Utilizando la definición dada en el material de la asignatura Sistemas Operativos se va a explicar qué es un proceso. La definición es la siguiente: *“Un proceso es básicamente un entorno formado por todos los recursos necesarios para ejecutar programas. Desde el punto de vista del SO, un proceso es un objeto más que hay que gestionar y al cual hay que dar servicio”*<sup>4</sup>.

Podemos decir que un programa es una entidad pasiva, en tanto en cuanto es un conjunto de instrucciones de código máquina y datos almacenados en un ejecutable. Mientras que un proceso sería la ejecución de ese programa, es decir, el programa en acción. Esto indica que los procesos son dinámicos, están en constante cambio debido a estos recursos necesarios, ya que al intentar realizar algún tipo de acción puede ser que tenga que permanecer a la espera de que dicho recurso esté disponible, por ejemplo una petición de lectura del disco duro, y que el brazo lector del disco duro lo esté utilizando otro proceso.

Al igual que las instrucciones de programa, los procesos incluyen los contadores de programa que indican la dirección de la siguiente instrucción que se ejecutará de ese procesos y los registros de CPU, así como las pilas que contienen datos temporales, como son los parámetros de subrutina, las direcciones de retorno y variables locales. Los procesos también contienen una sección de datos con variables globales y memoria dinámica. Todo ello permite gestionar de una manera más eficaz los procesos en los sistemas operativos multiprocesos, ya que cada proceso es independiente, por lo que el bloqueo de uno no debe de hacer que otro proceso en el sistema se bloquee.

En estos sistemas operativos multiproceso se intenta maximizar la utilización de la CPU, por lo que los procesos se ejecutan simultáneamente en la CPU y sólo quedan a la espera de ejecución si requieren de algún recurso del sistema que esté ocupado en ese momento, en cuanto obtiene dicho recurso podrá ejecutarse de nuevo. Todo este proceso de gestión lo realiza el sistema operativo, por lo que es el que decide si un proceso es más prioritario que

---

<sup>4</sup> Definición dada en el punto 1 del módulo 6 (La gestión de procesos) dentro de la asignatura *Sistemas Operativos*.

otros. Es el programador el que decide esta prioridad, por ejemplo en el caso de Linux se utiliza un número en la programación de estrategias para garantizar la equidad de los procesos.

Cada proceso se representa en el sistema operativo con un bloque de control de proceso (PCB, Process Control Block). En este PCB se guardan una serie de elementos de información de los mismos. Estos elementos son: el identificador del proceso, el estado del proceso, registros de CPU (acumuladores, punteros de la pila, registros índice y registros generales), información de planificación de CPU (prioridad de proceso, punteros a colas de planificación, etc.), información de gestión de memoria, información de contabilidad (tiempo de uso de CPU, números de procesos, etc.), información de estado de dispositivos E/S (lista de archivos abiertos, etc.).

### 3.2. Estados de un proceso

En un sistema multiprogramado o multitarea donde existen muchos procesos y un procesador, puede ocurrir que en un momento dado sólo se ejecute un proceso o varios y los demás estén esperando a ser procesado o esperen la finalización de una operación de E/S. Los pasos por los que puede pasar un proceso se pueden representar con un diagrama de estado como el de la figura<sup>5</sup> 4. Así se puede apreciar que a medida que un proceso se ejecuta va cambiando de estado dependiendo de las preferencias que cada uno tengan asignadas, por lo que será el procesador el que se encargue de ejecutar unos u otros.

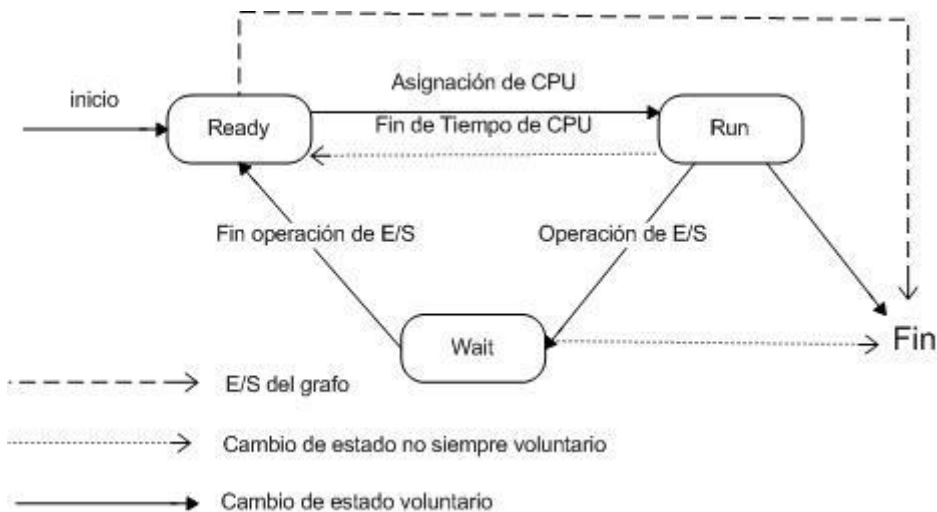


Figura 4 Diagrama de Estados de un proceso

Cómo se aprecia en la imagen 4 los estados por los que puede pasar un proceso son los siguientes:

<sup>5</sup> Diagrama obtenido del material de la asignatura Sistemas Operativos (módulo 6. Gestión de procesos)

- Nuevo. En los sistemas operativos hay varias razones por las que se crea un proceso. Entre éstas se pueden destacar; la inicialización del sistema, cuando se arranca el sistema se generan una serie de procesos ya bien sean para interactuar con el usuario o procesos en segundo plano con una función específica, como por ejemplo el aceptar la solicitud de una página web que está en dicha máquina; ejecución de una llamada al sistema por parte de otro proceso, un proceso puede requerir la descarga de ficheros, por lo que serán otros procesos los que se encarguen de ubicar el archivo o archivos en la ubicación específica; por medio de la acción de un usuario, por ejemplo al hacer doble clic en un icono; mediante el inicio de un trabajo por lotes.

Una vez el proceso ha sido creado queda a la espera de ser admitido, por lo que si es así pasaría a estado preparado o en caso contrario terminaría dicho proceso.

- Preparado (ready o listo). Un proceso en este estado está esperando a que se le asigne un procesador. Como se puede apreciar en la figura 4, un proceso en este estado puede, o bien finalizar, lo que ocurriría por la acción de otro proceso o por algún acontecimiento externo, o bien el proceso pasa al estado ejecución, ya que el gestor de procesos le asigna una CPU para ser ejecutado.
- Ejecución (run). El proceso en este estado está en la CPU ejecutando instrucciones. Puede ocurrir tres situaciones; que el proceso ejecute todas las instrucciones hasta su última línea de código y finaliza; pasa a estado bloqueado (wait) por que espera una acción externa como la entrada de información por teclado; o bien el proceso pasa a estado preparado debido a que ha agotado su tiempo de ejecución, por lo que cede su tiempo de ejecución.
- Bloqueado (wait, en espera). El proceso está esperando a que se produzca un evento externo, como una señal de E/S, y pasaría a estado ejecución. Al igual que el estado preparado, el proceso puede finalizar debido a un acontecimiento externo.

El sistema operativo utiliza varias colas para gestionar los estados, cada cola puede tener una política diferente. Así, podemos encontrar una cola para los estados preparados y una cola para los estados en espera. El planificador del procesador al examinar estas colas asigna el procesador al proceso más conveniente.

### 3.3. Planificación de un proceso

El sistema operativo es el encargado de decidir qué procesos entran en la CPU cuando ésta queda libre, y en qué momento sale de la CPU el proceso que está en ejecución. Todo ello se lleva a cabo a través de una política de planificación de procesos.

Se pueden definir múltiples políticas de planificación de procesos: por orden de llegada, primero la tarea más breve, por orden de prioridad, etc. En definitiva, lo que una política de planificación debe conseguir es que los procesos obtengan adecuadamente sus turnos de ejecución por lo que son tratados de la misma forma, que no se produzca sobrecarga, es decir, el planificador debe responder rápidamente ante cargas de trabajo ligera y responder de la misma forma ante cargas de trabajo similares. Y obtener un buen rendimiento, por lo se debe lograr finalizar el mayor número de procesos y maximizar el tiempo de respuesta.

No existe una política de planificación óptima para todas las computadoras, sino que depende de las características de los procesos. Así se puede ver cómo una política obtiene unos resultados excelentes en un sistema, sin embargo en otro sistema el rendimiento es mucho menor. Ello se debe a las características de los procesos, donde cada uno puede tener una cantidad de operaciones de E/S enorme cómo es el caso de las bases de datos, otros usan mayormente la CPU, otros realizan una mayor lectura de datos frente a otros, hay procesos que requieren una prioridad máxima en los turnos de ejecución, es el caso de los procesos de tiempo real, y hay procesos que requieren más tiempo de ejecución que otros, por lo que habrá que valorar si terminar primero los cortos o no.

Existen diferentes planificadores en el sistema. Primero nos encontramos el planificador a largo plazo, el cual es el encargado de controlar el grado de multiprogramación en el sistema, intentando conseguir una mezcla adecuada de trabajos en CPU y E/S. Es por tanto el encargado de suministrar los procesos a la cola de planificación a corto plazo.

Existe también un planificador a medio plazo. Es el encargado de suspender y posteriormente restaurar procesos de poco interés, realizando el intercambio de los mismos entre la memoria principal y el disco o memoria secundaria. Dicho proceso es conocido como swapping, y se ejecuta cuando hay escasez de recursos.

El planificador a corto a plazo es el encargado de asignar y desasignar la CPU. Su trabajo es coger un proceso de la cola de procesos preparados y asignarle una CPU. Hay dos tipos de planificadores a corto plazo; no expulsivos, el proceso abandona la CPU cuando termina o a la espera de un suceso externo; y expulsivos, el proceso que se está ejecutando puede pasar a estado listo enviado por parte del sistema operativo, es el caso de los sistemas de tiempo compartido y tiempo real, como UNIX, Windows NT/XP o superior, MAC OS X. Las políticas de planificación de procesos más comunes son las siguientes:

- Primero en llegar, primero en salir (FCFS). El primero proceso que llega a la cola de preparados será el primero en ser planificado y pasado a la CPU. Es no expulsiva, por lo que no es adecuada para los sistemas de tiempo compartido, por lo que provocan una especie de convoy con los procesos de E/S.
- Primero el proceso más corto (SJF). Cuando el proceso que está en la CPU cesa su ejecución se elige de la cola de procesos preparados aquel cuya ráfaga de CPU sea menor, es decir, que su tiempo de ejecución sea menor. Es del tipo no expulsiva, aunque existe una versión expulsiva (SRTF) que cuando llega un proceso más pequeño que el que se está ejecutando a la cola de preparados, éste es bloqueado y pasa a estado preparado mientras se ejecuta el nuevo proceso.
- Prioridades. Cada proceso tiene asignada una prioridad ya sea por medio del sistema operativo o por el usuario. Los procesos se dividen en distintas colas dependiendo de la prioridad, por lo que el planificador elegirá primero los procesos de la primera cola mediante FIFO, y cuando quede vacía elegirá los de la segunda cola. Esta política puede ser expulsiva o no. Esto puede provocar que haya procesos que se queden sin ejecutar debido a esta prioridad, la solución pasaría por aumentar la prioridad progresivamente a los procesos en espera.
- Turno rotatorio (Round-Robin). Adecuado para los sistemas de tiempo compartido. Consiste en generar periódicamente una interrupción de reloj, donde cada proceso dispone de un cuanto de tiempo máximo (*quantum*), por lo que cuando termina este tiempo, el proceso en ejecución pasa a preparado y pasa a ejecutarse el siguiente proceso en la cola de preparados según FIFO.
- Retroalimentación. Otro tipo de planificación es trabajar con diferentes colas de preparados cada una con una política diferente. Así si un proceso que ha pasado del estado ejecución al estado preparado, primero estaba en la cola 1 y ahora

pasa a la cola 2, y así sucesivamente hasta llegar a la última cola hasta que termina.

Los procesos en Linux pueden ser divididos en tres categorías: interactivos, tiempo real o por lotes. Los procesos de tiempo real son manejados bien por un algoritmo FIFO o turno rotatorio (Round-Robin), ya que al ser procesos considerados como prioritarios deben de ser ejecutados antes que los demás. Los demás procesos son manejados utilizando una planificación Round-Robin con un sistema de envejecimiento utilizado en la planificación de prioridades mencionada

Mientras en Windows, la planificación de procesos se basa en la utilización de colas múltiples de prioridades. Posee 23 niveles de colas clasificadas de la 31-16 en clase de tiempo real y las demás en clase variable. Cada cola es manejada mediante Round-Robin, pero si llega un proceso con mayor prioridad, se le es asignado el procesador.

El caso de Mac OS X es parecido al de Windows, utilizando varias colas de procesos cada una con un nivel de prioridad. Un hilo puede pasar de una cola a otra dependiendo de los requerimientos. Estos niveles se pueden manejar mediante llamadas al sistema. Los niveles son: normal, alta, Kernel y tiempo real.

### **3.4. Creación y destrucción de procesos**

Los sistemas operativos que son objeto de estudio tienen diferentes formas en la gestión de procesos, por lo que en este apartado haremos una distinción entre el sistema UNIX y el sistema Windows.

El por qué se hace el estudio del sistema UNIX es debido a que tanto el sistema Linux como el sistema Mac OS X están basados en dicho sistema, por lo que es más razonable hacer la mención a UNIX y no a estos dos sistemas. Las únicas diferencias fundamentales detectadas entre UNIX y Mac OS X se basan en el sistema de ficheros, donde Mac OS X utiliza HFS+ frente al UFS de UNIX y el estándar ext3 de Linux.

Respecto al Kernel, ambos sistemas están basados en el Kernel de UNIX, pero actualmente Mac OS X utiliza un híbrido del núcleo de UNIX Mach llamado XNU, mientras que Linux utiliza un núcleo monolítico llamado Linux.

- **UNIX.**

Los procesos en los sistemas UNIX están identificados por un número que es único (PID), además de que cada proceso el espacio de memoria utilizado, formado por tres segmentos: el código, los datos y la pila. También contiene la información de control del proceso, que indica la planificación y estado del proceso, la estructuración de datos y la comunicación entre procesos. Otra información importante es el número de identificación de usuario (UID) y el número de identificación del grupo de usuarios al que pertenece el proceso (GID). Todo ello pertenece al bloque de control de proceso (PCB).

La creación y destrucción de procesos en UNIX se ajusta a la filosofía de la manera más sencilla posible, así las llamadas al sistema tienen el mínimo número de parámetros. Las llamadas correspondientes a la creación, destrucción y bloque o espera de un proceso son respectivamente: fork, exit y wait.

La llamada fork crea un nuevo proceso hijo idéntico al proceso padre. Tienen la misma imagen de memoria, el mismo bloque de control de proceso y los mismos archivos abiertos, aunque situados en distintos espacios de memoria. Para poder distinguir a ambos procesos, la llamada fork devuelve distintos valores, el hijo recibe un valor 0 y el padre el PID del hijo.

La llamada exit finaliza un proceso. Cuando se produce esta finalización del proceso hijo, se manda al sistema la llamada wait, para que el padre se bloquee a la espera de la finalización del hijo. Si el proceso hijo finalizara antes de que el padre recibiera esta llamada, el proceso hijo se convertiría en un proceso en estado zombie, y hasta que no se ejecute esta llamada wait el proceso no se eliminará. Para evitar la acumulación de procesos UNIX prevé un límite de números de procesos zombie y aquellos procesos hijos que se destruyen más tarde que sus procesos padres, al quedar huérfanos sería el primer proceso del sistema (init) el que se encarga de recoger su estado de finalización.

La llamada wait bloquea el proceso que lo ha llamado hasta que uno de sus procesos hijos es destruido, por lo que si el proceso no tiene hijos wait regresa y el valor devuelto es igual al Pid de dicho proceso hijo.

En UNIX se puede utilizar un comando para eliminar un proceso que se está ejecutando por la razón que estimemos oportuno; gran consumo de recursos, ya no es necesario, etc. Esta orden es “kill *Pid proceso a eliminar*”, por lo que para eliminar un proceso primero se debe conocer su Pid.

- **Windows.**

Un programa en Windows es controlado por eventos. Así el programa principal espera la llegada de un evento como puede ser al presionar una tecla, y posteriormente invoca un procedimiento para procesar dicho evento, actualización de pantalla, del programa, etc.

Windows también tiene llamadas al sistema al igual que UNIX, de hecho el número de llamadas es extremadamente grande. En Windows encontramos que por cada llamada al sistema existe un procedimiento de biblioteca que los invoca. Por ello Windows ha creado un conjunto de procedimientos, llamado API Win32 [5], que se ha de utilizar para solicitar servicios al sistema operativo.

La creación de procesos en Windows se genera mediante la llamada `CreateProcess`, que tanto crea el proceso como carga el programa en el nuevo proceso. Esta llamada tiene 10 parámetros: el programa a ejecutar, atributos de seguridad, bits de control de archivos abiertos, prioridad, especificación de la ventana a crear y un apuntador a la estructura a la que al invocador se le devuelve información del proceso recién creado. `CreateProcess` tiene 100 funciones más para administrar y sincronizar procesos.

Al igual que en UNIX, en Windows se crea un nuevo proceso hijo a partir del padre, el cual tiene su propio espacio de direcciones como en UNIX, pero mientras en UNIX el espacio de direcciones del hijo era una copia del padre, en Windows el espacio de direcciones del hijo es completamente diferente al del padre desde el principio.

La llamada en Windows relativa a la destrucción de un proceso es `ExitProcess`, y al igual que ocurría con UNIX cuando se especifica la llamada `WaitForSingleObject` junto con un parámetro que especifica un proceso, quien lo invoca espera hasta que se produce la finalización del proceso.

En Windows encontramos multitud de llamadas al sistema igual que en UNIX (Figura 25), aunque hay algunas en UNIX que no se pueden utilizar en Windows, como es el manejo de enlaces, manejo de montaje de unidades, etc. Si en UNIX utilizamos la orden `kill Pid` para eliminar un proceso, en Windows existe la función `TerminateProcess`.



UNIX	WIN32	Descripción
Fork	CreateProcess	Crea un proceso nuevo
Waitpid	WaitForSingleObject	Puede esperar a que un proceso termine
Execve	(ninguna)	CreateProcess= fork + execve
Exit	ExitProcess	Termina la ejecución
Open	CreateFile	Crea un archivo o abre uno existente
Close	CloseHandle	Cierra un archivo
Read	ReadFile	Lee datos de un archivo
Write	WriteFile	Escribe datos en un archivo
Lseek	SetFilePointer	Mueve el apuntador de archivo
Stat	GetFileAttributesEx	Obtiene diversos atributos de archivo
Mkdir	CreateDirectory	Crea un directorio nuevo
Rmdir	RemoveDirectory	Elimina un directorio vacío
Link	(ninguna)	Win32 no maneja enlaces
Mount	(ninguna)	Win32 no maneja montajes
unmount	(ninguna)	Win32 no maneja montajes
Chdir	SetCurrentDirectory	Cambio el directorio de trabajo actual
Kill	TerminateProcess	Elimina un proceso
Time	GetLocalTime	Obtiene la hora actual

Figura 5. Llamadas de la API Win32 que corresponden con llamadas de Unix

## 4. Gestión de memoria

La gestión de memoria se encarga de asignar la memoria física del sistema a los programas, éstos se expanden hasta llenar la memoria con que se cuenta.

Todas las computadoras tienen una jerarquía de memoria, con una pequeña cantidad de memoria caché, una cantidad mucho mayor de memoria principal (RAM) y decenas o centenas de gigabyte de almacenamiento en disco.

El administrador de memoria es el encargado de administrar la jerarquía de memoria. Es el encargado de saber qué partes de la memoria están en uso o no, asignar y liberar la memoria principal a los procesos que la requieren, y administrar los intercambios entre la memoria principal y el disco.

Se puede decir que los objetivos principales de un sistema de gestión de memoria pasan por ofrecer a cada proceso un espacio lógico propio proporcionando una protección entre los procesos, permitir que los procesos compartan la memoria. Además se debe maximizar el rendimiento del sistema y proporcionar a los procesos mapas de memoria grandes.

En un sistema de multiprogramación cada programa debe contener dentro del código referencias al espacio de memoria a utilizar, ya que el mismo no siempre será el mismo, por tanto el sistema tendrá que realizar una reubicación de las direcciones

de memoria a las que hacen referencia las instrucciones de los programas para que se correspondan con las direcciones de memoria principal asignadas al mismo. Esto se logra a través de la unidad de manejo de memoria (MMU), que como se ha comentado se encarga de convertir las direcciones lógicas (memoria virtual) emitidas por los procesos en direcciones físicas. Además se encarga de que la conversión se realiza con éxito y que el proceso que intenta acceder a las direcciones de memoria correspondientes tiene permiso para ello.

En caso de fallo se envía una excepción que será tratada por el kernel. Éste último está siempre en la memoria principal, ya que si estuviera en la memoria secundaria nadie podría llevarlo a la memoria principal en caso de fallo de acceso a memoria.

Dentro de la MMU existe una pequeña memoria asociativa denominada *TLB* (*Translation Lookaside Buffer*), que mantiene la información de las últimas páginas accedidas. Se puede decir que se trata de una memoria tipo caché. Con ello se evita que el procesador esté siempre leyendo la tabla de páginas directamente, con el consecuente decremento de rendimiento.

El funcionamiento del TLB consiste en verificar si la dirección requerida se encuentra en el mismo, si es así automáticamente la MMU traducirá la dirección lógica en su respectiva dirección física para ser utilizada.

En cambio, si la dirección no está presente en el TLB se produce un fallo de página, por lo que se procederá a buscar en la tabla de dirección mediante un proceso llamado *page walk*. Este proceso es costoso, ya que requiere de la lectura de múltiples ubicaciones de memoria. Una vez determinada la dirección física a través del *page walk*, el mapeo de la dirección virtual a la física es ingresado en el TLB para su posterior uso.

#### **4.1. Memoria principal**

La memoria principal constituye el dispositivo de almacenamiento primario de los computadores. De carácter volátil, en ella se ubican los datos que precisan los programas para su ejecución. Esta memoria es de mayor costo que la memoria secundaria, pero el acceso a su información es mucho mayor.

La memoria principal se comunica con la CPU mediante el bus de direcciones, y es el ancho de este bus el que determina la capacidad del microprocesador para el direccionamiento de direcciones de memoria.

Se puede decir de manera coloquial que la memoria principal está formada como un conjunto de casillas, cada una con una dirección que las identifica, donde se almacenan los datos y las instrucciones correspondientes a los programas. Cada casilla está enumerada con lo que se denomina dirección de memoria.

## 4.2. Memoria virtual

El tamaño del programa, datos y pila puede exceder la cantidad de memoria física disponible, por lo que el sistema guarda en la memoria secundaria aquellas partes del programa que no están en ejecución o que no tienen espacio para ser cargadas en la memoria principal.

La memoria virtual utiliza estos dos niveles de jerarquía de memoria: principal y secundaria o de respaldo que suele ser el disco. Con ello podemos obtener una cantidad de memoria mayor que la memoria física disponible, y por tanto ejecutar procesos con una memoria lógica mayor que la memoria física disponible.

Hay que resaltar que el uso de la memoria virtual no acelera la ejecución de un programa, sino que puede incluso que la ralentice debido a la sobrecarga asociada a las transferencias entre la memoria principal y la secundaria. Esto hace que esta técnica no sea apropiada para sistemas de tiempo real.

Sobre la memoria de respaldo se establece un mapa uniforme de memoria virtual, y son a estas direcciones del mapa a las que se refiere el procesador en la ejecución del programa, pero los accesos reales se realizan sobre la memoria principal. La memoria virtual requiere por tanto de una gestión automática de la jerarquía de memoria formada por la memoria principal y el disco.

El sistema operativo gestiona la transferencia de bloques desde la memoria principal a la memoria secundaria que sirve de respaldo a la memoria virtual. Las direcciones generadas por las instrucciones máquina están referidas al espacio virtual. En este sentido se habla de que el procesador genera direcciones virtuales.

El mapa virtual asociado a un programa en ejecución está soportado físicamente por una zona del disco denominada de intercambio o swap, y por una zona de la memoria principal.

La memoria virtual se implementa sobre un esquema de paginación. A este dispositivo se le denomina dispositivo de paginación. Se denominan páginas virtuales a las páginas de espacio virtual, páginas de intercambio a las páginas residentes en el disco y marcos de página a los espacios en los que se divide la memoria principal.

Cada marco de página es capaz de albergar una página virtual cualquiera. Se puede decir que la unidad de hardware denominada MMU, traduce el número de página virtual en el correspondiente número de marco de página.

Cuando las páginas de memoria de un programa necesarias para su ejecución no se encuentren en memoria principal, la MMU producirá la respectiva excepción.

Los fallos de página son atendidos por el sistema operativo que se encarga de realizar la adecuada migración de páginas, para traer la página requerida por el programa a un marco de página. Se denomina paginación por demanda al proceso de migración necesario para atender los fallos de página. En el siguiente punto se profundizará más en este proceso de paginación.

También cabe reseñar que el tamaño del espacio virtual suele ser muy grande, y dado que los programas requieren en general mucho menos espacio, una de las funciones que realizar el sistema operativo es la asignación de espacio virtual a los programas en ejecución. El programa no podrá utilizar todo el espacio virtual sino que se tiene que restringir a la zona o zonas que le asigne el sistema operativo. Estas zonas se denominan segmentos, y a dicho proceso segmentación que se profundizará más adelante.

#### **4.2.1. Traducción de página virtual a física**

La memoria física de un ordenador es una secuencia de bytes donde cada byte tiene una dirección que es la posición que ocupa en la propia memoria.

Las direcciones de memoria que utiliza un programa no son directamente las direcciones físicas, sino que son direcciones lógicas que son traducidas o mapeadas en direcciones físicas.

Para realizar la traducción es necesaria la utilización de hardware con ayuda del sistema operativo. Respecto a este hardware hablamos del gestor de memoria (MMU) junto con el búfer de traducción anticipada de instrucciones (TLB).

El gestor de memoria será el encargado de traducir las direcciones virtuales a direcciones de memoria principal, de tal forma que cada una corresponda con su respectivo marco de página.

#### 4.2.2. Paginación

La paginación surge como intento de paliar los problemas de asignación de memoria contigua, sofisticando el hardware de gestión de memoria (MMU) del procesador y aumentando considerablemente la cantidad de información de traducción que se almacena por cada proceso.

La MMU usa dos tablas de páginas. Una tabla de páginas de usuario para traducir las direcciones lógicas del espacio de usuario y una tabla de páginas del sistema para las direcciones lógicas del espacio del sistema.

La paginación no proporciona un aprovechamiento óptimo de la memoria como lo haría un esquema que permitiese que cada palabra del mapa de memoria de un proceso pudiera corresponder con cualquier dirección de memoria, pero esto implicaría una enorme cantidad de información de traducción. En la paginación cada página del mapa de un proceso puede corresponder con cualquier marco de memoria. Esto reduce drásticamente el tamaño de la tabla de traducción.

Cada entrada de la tabla de páginas, además del número de marco que corresponde con esa página, contiene información adicional como la protección de la información y la indicación de página válida mediante un bit de confirmación.

Un aspecto importante en el rendimiento de un sistema de paginación es el tamaño de la página. Un tamaño pequeño reduce la fragmentación y permite ajustarse mejor al conjunto de trabajo del proceso. Sin embargo, un tamaño grande implica tablas más pequeñas y un mejor rendimiento en los accesos a disco.

#### 4.2.3. Segmentación

El sistema operativo debe guardar para cada proceso una tabla de regiones que especifiquen qué páginas pertenecen a cada región. Esto implica que hay que rellenar las entradas de las páginas pertenecientes a la región con las

mismas características y que para compartir la región, las entradas correspondientes de dos procesos deben apuntar al mismo marco.

La segmentación es una técnica que intenta dar soporte a las regiones. Para ello, considera el mapa de memoria de un proceso compuesto de múltiples segmentos. Cada región se almacenará en un segmento que tendrá un tamaño variable.

Dado que cada segmento se almacena en memoria de forma contigua, este esquema presenta fragmentación externa, por lo que hace disminuir el rendimiento de dicho sistema.

### 4.3. Gestión de Memoria en UNIX

Al igual que en el apartado de procesos se realizó una comparación de la forma en que los dos sistemas operativos; UNIX y Windows, realizan dicha tarea, en este punto se realizará unas reseñas de cómo el sistema UNIX realiza la gestión de memoria, y en el punto 4.4 se realizará sobre el sistema Windows.

Sólo se realizará este estudio de estos dos sistemas, porque cómo ya se ha comentado, los sistemas que van a ser objeto de estudio, Linux y Mac OS X, están diseñados a partir de UNIX, de ahí que se elija dicho sistema. El otro sistema a estudiar será Windows 7, de ahí que sea también objeto de estudio.

En el sistema operativo UNIX, la gestión de memoria ha variado de las versiones antiguas a las actuales. Antes, UNIX se basaba sólo en el intercambio (swapping) donde se empleaban particiones variables sin ningún tipo de esquema de memoria virtual. Las versiones actuales se basan en la memoria virtual paginada, utilizando para ello la paginación combinado con el intercambio.

Las direcciones lógicas generadas por un programa se dividen en un número de páginas con el mismo tamaño. Este tamaño varía dependiendo de la versión, utilizándose en versiones anteriores tamaños de 512 bits o 1024 bits. Hoy en día, con las CPUs actuales el tamaño de página en los equipos a 32 bits es de 4 KB, por lo que el espacio de direcciones es de  $2$  elevado a 32 bits (4 GB).

Respecto a la paginación, ya se ha tratado en el punto 4.2.1., UNIX utiliza las tablas de páginas, el descriptor del bloque donde se almacena la información para acceder a la página en memoria secundaria y los marcos de página que guarda

información acerca del estado de la página, el número de procesos que referencia al marco y el dispositivo que contiene la copia de la página.

Los marcos no asignables a un proceso son marcos libres asignables a cualquier proceso. El sistema define el número mínimo de marcos libres, que comprueba periódicamente. Si en un instante no se alcanza dicho límite, un proceso paginador envejece las páginas. Toda referencia a una página pone a cero la edad de la página. Si la página pasa una edad, la página pasa a estado libre. La página de un marco libre puede ser rescatada en una referencia si el marco no hubiera sido asignado antes.

En situaciones donde hay mucha demanda por parte de los procesos, puede ocurrir que el paginador no sea capaz de conseguir marcos libres a la velocidad necesaria. Entonces es cuando se produce el swapping, mediante el cual se saca algún proceso de la memoria principal y se pasa a la memoria secundaria (swap out). En cambio cuando existe espacio en la memoria principal, se intercambian los procesos a memoria copiando desde el área de intercambio (swap in). Los criterios de elección se basan en el estado del proceso, su prioridad, el tamaño del programa y el tiempo que lleva en memoria.

Para soportar swapping se requiere un espacio de intercambio en almacenamiento secundario, normalmente en un disco. Se puede utilizar un dispositivo específico, una partición del disco o incluso compartir la misma partición del sistema de ficheros. En los sistemas UNIX se suele utilizar una partición del disco para ello.

En los sistemas UNIX se puede observar como en memoria se carga todo lo que se pueda. Es decir, se utiliza la memoria como caché de datos. Esto se utiliza para optimizar el sistema y hacerlo más rápido. En memoria se carga todos los datos a los que tenga que acceder la CPU, por lo que si ésta tiene los datos en memoria en vez de en el disco, la rapidez de acceso a los mismos será mayor, por lo que el rendimiento aumenta. Ello hace que se optimice la mayor cantidad de memoria principal disponible, ya que el uso de la misma es del 100%. En cuanto, un proceso requiera de memoria se utilizará los mecanismos de swapping ya descritos liberando la memoria utilizada para caché. Ello no ocurre en el sistema Windows donde como se verá en el siguiente punto no utiliza el total de la memoria para caché de datos de la CPU.

En UNIX las páginas pueden estar en diferentes estados. El valor de este estado se refleja en el campo *flags* de la página. A continuación se describen los estados que puede estar una página:

CONSTANTE (FLAGS)	VALOR	SIGNIFICADO
PG_locked	0	La página está bloqueada en memoria
PG_error	1	Se ha producido un error en la carga de la página en memoria
PG_referenced	2	La página ha sido accedida
PG_uptodate	3	El contenido de la página está actualizado
PG_dirty	4	Indica si el contenido de la página se ha modificado
PG_lru	5	La página está en la lista de páginas activas e inactivas
PG_active	6	La página está en la lista de páginas activas
PG_slab	7	El marco de página está incluido en un slab
PG_owner_priv_1	8	
PG_arch_1	9	
PG_reserved	10	La página está reservada para un uso futuro, no es posible acceder a ella.
PG_private	11	Contiene datos privados
PG_writeback	12	Se indica que la pagina usa el método writeback
PG_compound	14	
PG_swapcache	15	Indica si la página esta intercambiada
PG_mappedtodisk	16	Indica si la página está intercambiada
PG_reclaim	17	Página marcada para escribirse a disco
PG_buddy	19	Página libre en listas buddy

Figura 6. Estados de las páginas de memoria en UNIX

#### 4.4. Gestión de Memoria en Windows

Los procesos dentro de Windows utilizan un espacio de direcciones de 32 bits que permite habilitar un espacio de direcciones de hasta 4 gigabytes de direcciones, o 8 terabytes en el caso de los sistemas de 64 bits.

Cuando todos los procesos que se ejecutan en el sistema requieren de más memoria que la disponible, Windows utiliza la alternancia entre la memoria virtual (determinado por un archivo de paginación) y la memoria física, que al igual que en UNIX recibe el nombre de swapping.

Este intercambio se realiza no de bytes a bytes, sino de página a página, que en Windows son bloques de 4 KB donde se almacenan los datos de los procesos. De ahí que toda la memoria esté paginada, tanto la física como la virtual. La física en forma de marcos de página y la virtual en forma de páginas.

Además, Windows dispone de tablas de páginas que son las que apuntan a las propias páginas, y los punteros de estas tablas se almacenan a su vez en un directorio de páginas. Cada proceso dispone sólo de un directorio.



El espacio de direcciones para un proceso es el conjunto de direcciones de memoria virtual que puede utilizar. El espacio de direcciones para cada proceso es privado y no se puede acceder por otros procedimientos, a menos que se comparta.

Una dirección virtual no representa la ubicación física real de un objeto en la memoria, sino que el sistema mantiene una tabla de páginas para cada proceso. El sistema traduce la dirección virtual a una dirección física mediante diferentes algoritmos, lo que hace que no se pueda acceder a las posiciones de otros procesos porque no es posible hacer referencia a dichas posiciones.

A diferencia de los otros sistemas, las páginas pueden estar en tres estados; *libre*, donde no puede ser accedida por ningún proceso pero sí reservada o encargada; *reservada*, es un espacio de dirección virtual fijada para usos futuros; *encargada*, aquella que ha sido asignada a un espacio físico, ya sea en memoria física o virtual.

## 5. Implementación

A lo largo de los siguientes apartados se realizará una descripción de los pasos seguidos en la instalación y configuración de los equipos para realizar los respectivos test.

De esta forma, tendremos una idea de los pasos necesarios para poder abordar el problema del análisis de los distintos sistemas. Se detallarán los problemas encontrados en la instalación de dichos sistemas, de su posterior instalación dentro de un Hypervisor y de la instalación y configuración de Apache como servidor Web.

Además se hará una breve explicación de los puntos relativos a los programas de *benchmarking* utilizados, es decir, cómo se realizarán los test, qué configuración de dichos programas será necesaria para dicha acción, etc.

En definitiva, se mencionarán todos los pasos necesarios para implantar el pequeño laboratorio de test.

### 5.1. Hypervisor

#### 5.1.1. Concepto de Hypervisor

La virtualización es una tecnología compuesta por una capa de software permitiendo la instalación y utilización de varios sistemas operativos sobre la misma máquina física. Esta capa de software es el Hypervisor o Monitor de Máquina Virtual (Virtual Machine Monitor, VMM).

El VMM crea una capa de abstracción entre el hardware de la computadora central, o física (host), y el sistema o sistemas operativos de las máquinas virtuales. De manera que presenta un hardware compatible con los sistemas operativos que se vayan a instalar. La máquina virtual es un sistema operativo que corre como si estuviera instalado en una plataforma de hardware autónoma.

La VMM maneja y gestiona los cuatro recursos principales de la computadora: CPU, Memoria, Almacenamiento y Red. La VMM puede distribuir dinámicamente dichos recursos entre las distintas máquinas virtuales.

Hay varios tipos de virtualización como ya se ha comentado.

- Tipo I. Denominada virtualización nativa o bare metal (metal desnudo). Consiste en la instalación del software Hypervisor sobre el hardware de la máquina física.
- Tipo II. Denominada hosted, donde el software Hypervisor se instala sobre el sistema operativo que contenga el host.

### **5.1.2. Instalación del Hypervisor**

Como se ha comentado en la introducción de este trabajo, en concreto en el punto 1.3 Planificación del proyecto, el software de virtualización que vamos a utilizar es VMWare WorkStation 7.

En un principio la opción elegida era VMWare Server 2, debido a que este tipo de virtualización era del tipo I, y se podrían ver mejores resultados. Pero debido a la incompatibilidad del hardware de la máquina que se está utilizando no ha sido posible instalar dicho software.

La instalación de VMWare WorkStation 7 se ha realizado sobre el sistema operativo Windows 7 Profesional 64 bits. La instalación ha sido estándar por lo que no ha requerido de ninguna configuración especial.

### **5.1.3. Creación y configuración de máquinas virtuales**

Para la creación de las máquinas virtuales se han utilizados los mismos parámetros para cada uno de los tres sistemas a instalar. Para que cada uno tenga el mismo hardware.

Dicho hardware se compone de los siguientes parámetros:

- La primera opción que aparece es la especificación del número de procesadores. Se ha elegido utilizar un procesador con 2 cores por procesador, por lo que en total tendrá dos procesadores. Como ilustra la siguiente imagen.

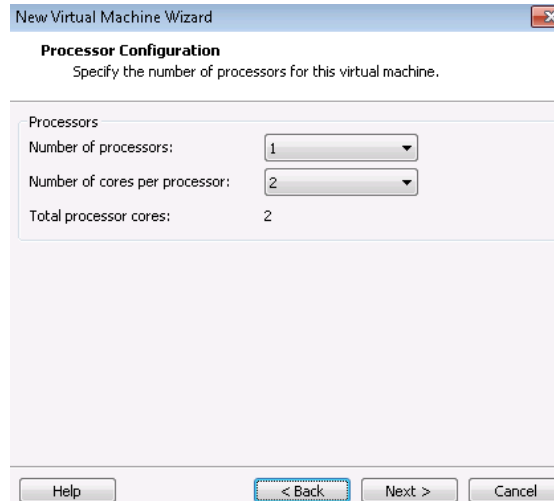


Figura 7. Número de procesadores

- Memoria. El tamaño de la memoria será el mismo para todos. En este caso se va a proporcionar 2 GB de memoria RAM a cada equipo. Aunque el equipo físico sólo dispone de 4 GB, se tendrá que iniciar cada equipo de forma individual para proceder a su análisis. Nunca tendremos iniciados los tres equipos a la vez. A continuación se muestra la imagen con la opción de memoria elegida (Figura 8).

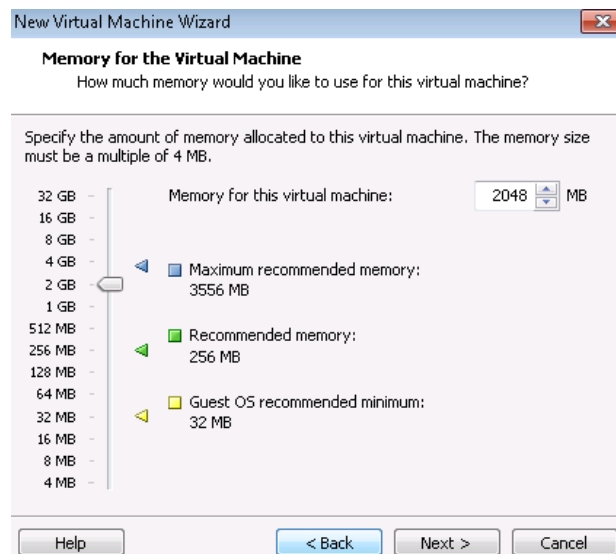


Figura 8. Tamaño de memoria elegida para las máquinas virtuales

- La siguiente opción a elegir en la creación de la máquina virtual es la conexión de red. En este caso la opción ha de ser “Use bridged networking” (Figura 9). De este modo cada máquina podrá tener su propia dirección ip.

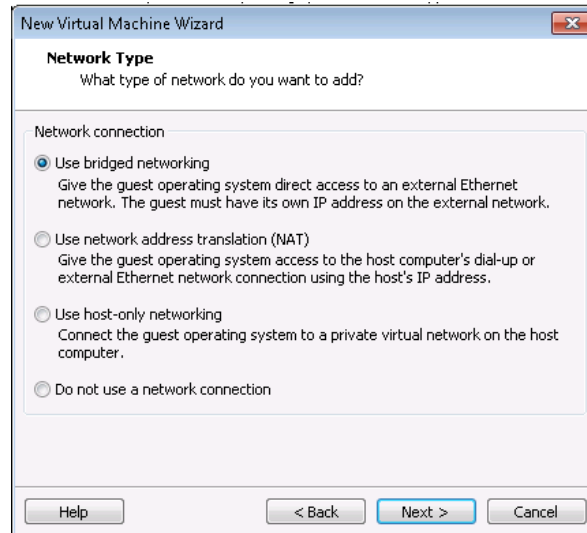


Figura 9. Modo de conexión de red

- Seleccionamos el controlador SCSI por defecto por ser más compatibles que otros. Al utilizar distintos sistemas operativos tan dispares, si se quiere tener las máquinas lo más parecida posible se tiene que optar por los componentes estándar en detrimento de componentes que puedan dar mayor rendimiento.

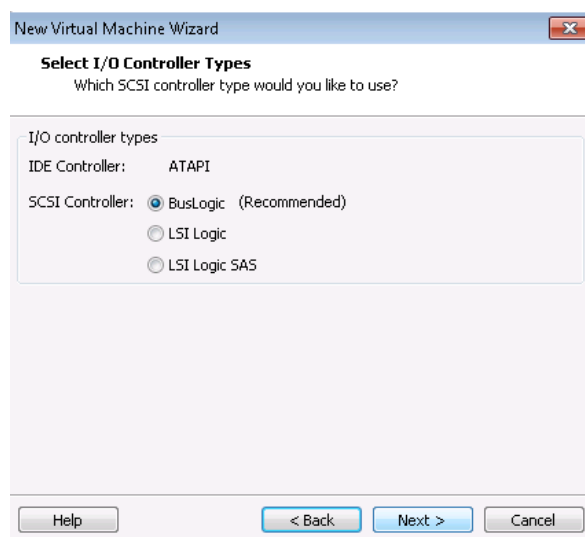


Figura 10. Modo de conexión de red

- Otra de las opciones a configurar será la capacidad del disco que será de 20 GB. Además se configura que dicho sistema sea guardado en un fichero lo que permite mayor seguridad a la hora de hacer copias de seguridad.



Figura 11. Capacidad del disco

Se puede ver en la siguiente imagen (Figura 12) las tres máquinas virtuales creadas, y como está ejecutándose la máquina virtual con el sistema operativo Mac OS X 10.6.4

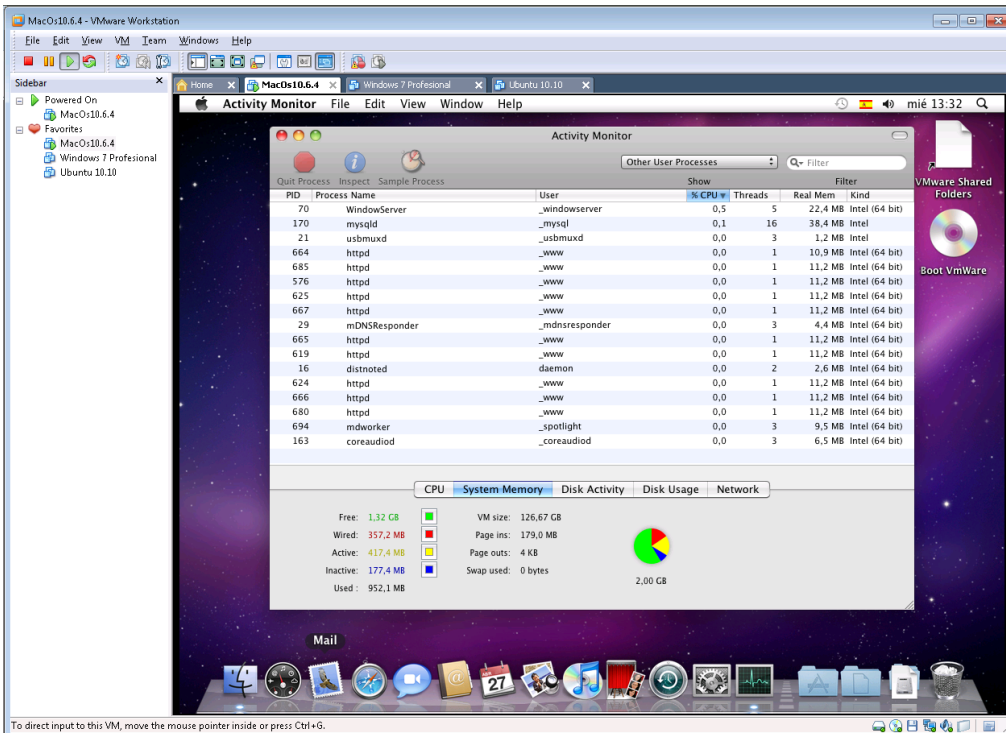


Figura 12. Máquinas Virtuales en VMware

### 5.1.4. Instalación de los Sistemas Operativos

La instalación de los sistemas operativos se lleva a cabo igual que en una computadora normal. En este caso VMWare permite elegir como origen de la instalación una imagen iso del sistema operativo, por lo que no sería necesario tener acumulados los CD o DVD de instalación de dichos sistemas.

Respecto a la instalación de los sistemas operativos, se realizará una instalación estándar en cada uno de ellos, configurando sólo los aspectos de zona horaria, usuario y demás campos estándar solicitados.

No se instalará ningún software que no sea los necesarios para la instalación de *Wordpress*, que será la aplicación web que proporcionará las páginas que serán testeadas por los benchmarking. Dicho requerimiento consta del servidor web Apache, del entorno de programación PHP y del gestor de base de datos Mysql.

A continuación se muestra una imagen con el comienzo de la instalación de uno de los sistemas operativos. En concreto se trata de la instalación del sistema operativo Ubuntu, el cual se ha instalado desde una imagen iso descargada desde la página oficial de dicho sistema [6].

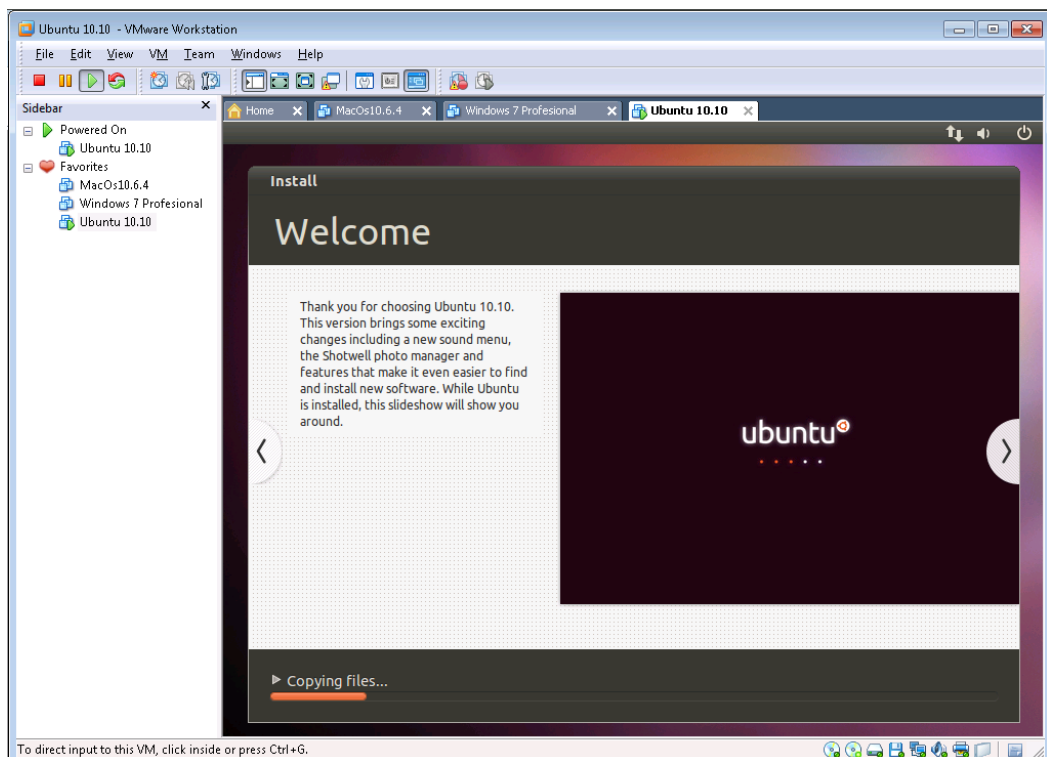


Figura 13. Instalación de Ubuntu en máquina virtual

Dentro de los problemas encontrados en la instalación de los sistemas, cabe mencionar la utilización de la EFI (*Extensible Firmware Interface*) a la hora de instalar Mac OS X. Esta interface es necesaria para el arranque del sistema Mac OS X y hace las funciones de la BIOS de las computadoras personales (PC).

No se han encontrado ningún tipo de problema de otra índole, como por ejemplo en la creación de particiones. Al dejar todo por defecto no se ha encontrado ninguna diferencia entre dichos sistemas operativos.

## 5.2. Sistemas Operativos

### 5.2.1. Configuración

La configuración de los sistemas operativos se va a basar simplemente en la configuración de una ip diferente en cada uno de los sistemas, para diferenciar claramente el acceso a cada uno de ellos. De esta manera los sistemas tendrán las siguientes direcciones de red.

Respecto a la configuración que haya que realizar para poder instalar las aplicaciones Apache, PHP y Mysql, se realizarán en el punto 5.2.2 y sus apartados.

- Sistema Ubuntu. Tendrá como dirección de red la siguiente:

Dirección IP	192.168.1.210
Máscara de subred	255.255.255.0
Puerta de enlace	192.168.1.100
Servidores DNS	192.168.1.13 192.168.1.11

- Sistema Windows. Tendrá la siguiente dirección de red:

Dirección IP	192.168.1.211
Máscara de subred	255.255.255.0
Puerta de enlace	192.168.1.100
Servidores DNS	192.168.1.13 192.168.1.11

- Sistema Mac OS X. tendrá como dirección de red:

Dirección IP	192.168.1.212
Máscara de subred	255.255.255.0
Puerta de enlace	192.168.1.100
Servidores DNS	192.168.1.13 192.168.1.11

## 5.2.2. Servidor Web Apache

### 5.2.2.1. Concepto

Apache es un software código abierto que pone a disposición de todo aquel que quiera utilizarlo como servidor web. Este software es fruto de la colaboración de un grupo de voluntarios en todo el mundo que planean y desarrollan las mejoras necesarias para conseguir una implementación de código robusto.

Apache es un servidor modular, por lo que se irán añadiendo aquellos módulos que sean necesarios. También es multiplataforma, ya que se puede instalar en sistemas tan diversos como UNIX, Windows, Mac, etc. como se podrá comprobar a lo largo de este trabajo.

Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación, etc. pero la falta de una interfaz gráfica que ayude a su configuración hace que reciba ciertas críticas. A pesar de todo es el servidor HTTP más utilizado en los sitios web de todo el mundo con un 66.62 % [33].

### 5.2.2.2. Instalación y configuración

Dependiendo del sistema operativo la instalación será de manera diferente. Por lo que veremos cada sistema por separado.

También se comentarán los pasos a seguir para la instalación de las herramientas del entorno PHP y el gestor de bases de datos Mysql para cada uno de los sistemas.

Respecto a la configuración de apache en los distintos sistemas será muy similar en el comienzo, ya que todas las instalaciones se hacen por defecto y suelen tener la misma configuración. Sólo cabe mencionar que se realizará el cambio de ubicación del DocumentRoot o directorio principal donde



residirán las páginas web que servirá el servidor. También habrá que indicar a Apache que utilice como página índice *index.php*. Y por último se añadirán aquellas líneas necesarias para que apache sepa que el entorno PHP está disponible.

A continuación se realizará una descripción de los pasos realizados para la instalación y puesta en marcha de apache.

- Ubuntu.

Para la instalación de las aplicaciones se ha utilizado el gestor de paquetes Synaptics. Mediante dicho gestor se han ido buscando los módulos necesarios. En concreto se ha buscado apache2, php5 y Mysql. A la hora de instalar estos paquetes, el propio gestor informa de los módulos que también serán instalados por ser necesarios.

Otra forma de instalar estas aplicaciones habría sido a través de un terminal, y con la orden “*aptitude install paquete*”, donde paquete sería en nombre del paquete a instalar, o mediante la orden “*apt-get install paquete*”.

No existe diferencia entre uno u otra a la hora de realizar la instalación del software que queramos. La diferencia la encontramos en el modo de gestionar los paquetes, ya sea buscándolos, borrándolos, actualizándolos, etc. Mientras apt tiene diferentes herramientas con sus respectivos argumentos para realizar dichas tareas, mientras aptitude sólo utiliza una herramienta, el propio aptitude, con diferentes argumentos para realizar las mismas tareas que apt. También cabe mencionar que aptitude ha manejado siempre mejor la eliminación de paquetes junto con sus dependencias huérfanas. Algo que *apt* ha mejorado en las versiones recientes con “*apt-get autoremove*”.

Respecto a los cambios de configuración en las distintas aplicaciones, no se ha realizado ninguno al respecto. Se ha dejado la configuración tal cual, ya que para que funcione la aplicación *Wordpress* no es necesario realizar ningún cambio.

En la siguiente imagen se puede ver que se ha logrado correctamente la instalación de *Wordpress* en *Ubuntu* para su posterior testeo.

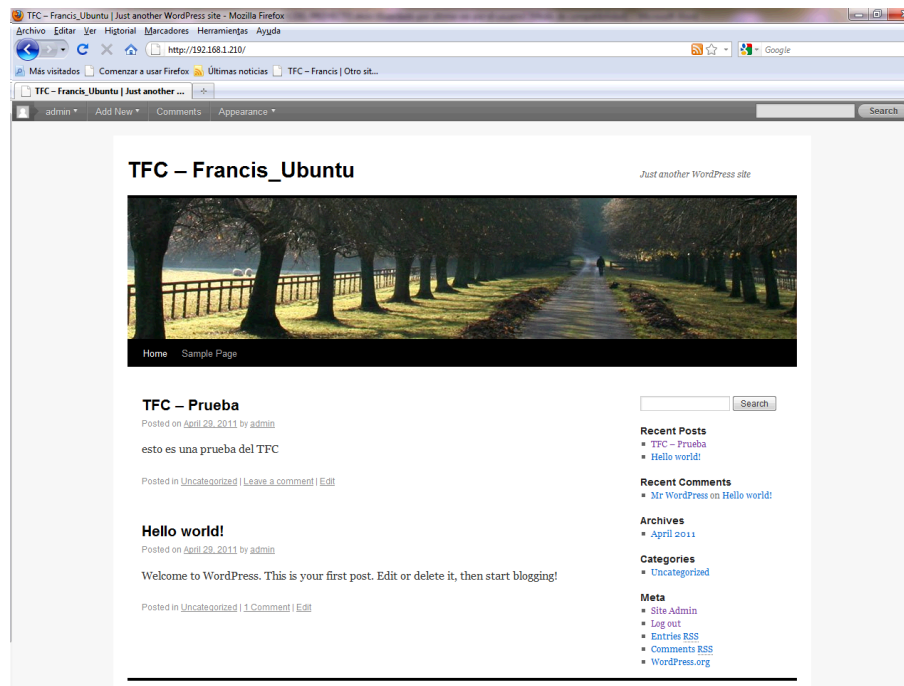


Figura 14. Instalación correcta de *Wordpress* en Ubuntu.

- Windows 7.

En la instalación de los programas en Windows, se ha realizado la instalación de cada uno de ellos por separado, realizando primero la descarga de los mismos desde sus páginas oficiales.

Una vez han sido instalados, hay que realizar unos cambios en la configuración de Apache para que el mismo tenga conocimiento de que el entorno de PHP está disponible. Para ello hay que cargar el módulo de PHP. “`LoadModule php5_module C:/Servidor/PHP/php5apache2.dll`”, y posteriormente se añaden unas líneas de código necesarias para que Apache pueda interpretar las extensiones de PHP.

Se puede observar como se ha logrado la correcta instalación de *Wordpress* en dicho sistema operativo.

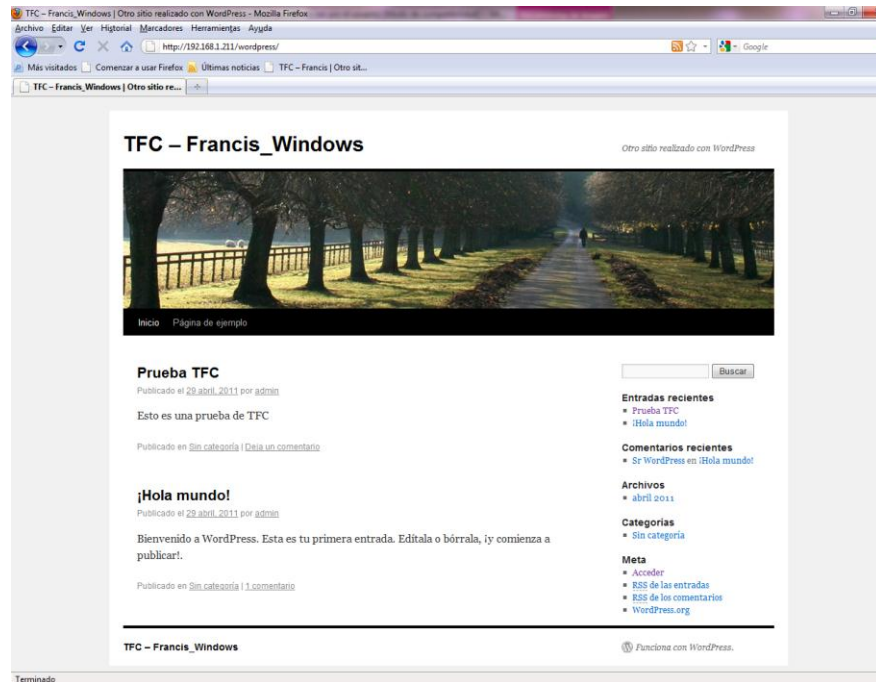


Figura 15. Instalación correcta de *Wordpress* en Windows.

- Mac OS X.

En este sistema operativo ha sido muy sencilla la puesta en marcha del servidor web Apache junto con PHP, y la instalación de Mysql.

Este sistema ya viene con la instalación de Apache y PHP en el mismo, así lo único que hay que realizar es la activación de ambas herramientas. En el caso de Apache se encuentra dentro de las herramientas de sistema >sharing>Web Sharing. En el caso de PHP sólo hay que asegurarse que Apache conozca que PHP está instalado. Para ello se añaden varias líneas de código dentro del fichero de configuración de Apache (httpd.conf), y con ello ya tenemos php funcionando. Estas líneas son:

- *AddType application/x-httpd-php .php*
- *AddType application/x-httpd-php-source .php5*

En el caso de Mysql lo único que tenemos que hacer es descargarnos el paquete .dmg desde la web de Mysql [27]. Una vez descargado la instalamos y simplemente tenemos que iniciar el servicio de Mysql.

Una vez creada la base de datos para *Wordpress*, ya podemos realizar la instalación de éste para su posterior utilización (Figura 16).

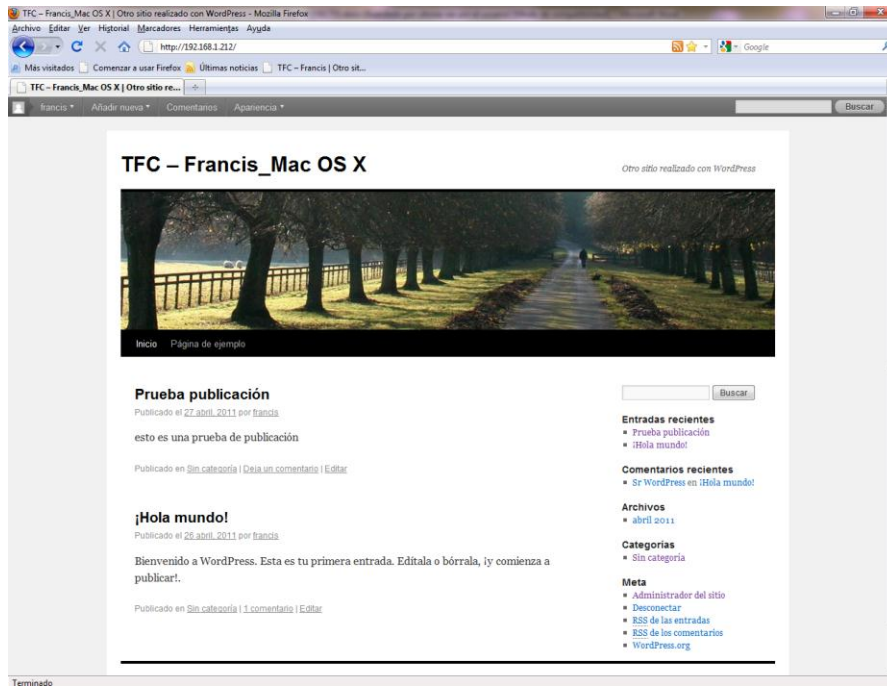


Figura 16. Instalación correcta de *WordPress* en Mac OS X.

## 5.3. Benchmarking

### 5.3.1. Descripción y tipos

El benchmarking es una forma de determinar qué tan bien funciona un producto o empresa respecto a otras. El benchmarking utiliza la evaluación continua y sistemática de los productos, servicios, procesos, etc. para obtener una representación de entre los analizados de cuáles forman una referencia o modelo a seguir.

Este proceso de benchmarking es utilizado por las empresas para ver su competitividad respecto a otras del mismo sector. En nuestro caso lo utilizaremos para ver la competitividad de los tres sistemas operativos respecto a un mismo problema como es el de servir páginas web mediante el servidor web Apache.

Las herramientas a utilizar son las ya mencionadas; *JMeter* [25] y *ab* [26]. Ambas herramientas están creadas para testear el rendimiento de Apache, sobre todo *ab*, la cual está integrada dentro de la instalación del propio Apache, por lo que no es necesaria su instalación, de ahí que sólo es posible utilizarla desde la propia máquina donde está instalado el servidor Apache. En cambio *JMeter*,

aunque en un principio fue diseñada para testear aplicaciones web, actualmente puede testear otras funciones como bases de datos, servidores de correo, servidores LDAP, etc. *JMeter* a diferencia de *ab*, sí puede ser ejecutada desde un equipo distinto al que tiene instalado la aplicación web, pero evidentemente tienen que estar en la misma red, o el equipo a testear ser visible desde *JMeter*.

### 5.3.2. Instalación y configuración

Cómo ya se ha comentado la aplicación *ab* no requiere de ninguna instalación, ya que forma parte de Apache. Respecto a los parámetros que se utilizarán serán como los siguientes: “*ab -c 100 -n 20 <http://localhost/>*”

Con estos parámetros le indicamos a la aplicación que realice un test de 100 solicitudes (-c 100) con 20 peticiones concurrentes (-n 20) al servidor localhost. En cada uno de los test se especificará el número de peticiones y solicitudes concurrentes.

Respecto a los datos que se van a obtener con este *benchmark*, nos centraremos principalmente en los valores “*Request per second*” (páginas servidas por segundo) y “*Time per Request*” (tiempo por respuesta). Con ello se podrá realizar un seguimiento de cómo cada uno de los sistemas es capaz de ejecutar más procesos y poder servir cuanto antes las páginas solicitadas.

En la siguiente imagen (Figura 17) tenemos un ejemplo de una captura de un test realizado con *ab*, en el cual se puede apreciar cómo se han completado 500 peticiones con 50 usuarios concurrentes. En dicho test se ha obtenido una media de 3,79 páginas por segundo, con un tiempo medio de 13 segundos por petición.

```

Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Finished 500 requests

Server Software:      Apache/2.2.14
Server Hostname:     localhost
Server Port:         80

Document Path:       /
Document Length:     7853 bytes

Concurrency Level:   50
Time taken for tests: 131.979 seconds
Complete requests:   500
Failed requests:     0
Write errors:        0
Total transferred:   4068000 bytes
HTML transferred:    3926500 bytes
Requests per second: 3.79 [#/sec] (mean)
Time per request:    13197.890 [ms] (mean)
Time per request:    263.958 [ms] (mean, across all concurrent requests)
Transfer rate:       30.10 [Kbytes/sec] received

Connection Times (ms)
      min  mean[+/-sd] median  max
Connect:    0    4  25.3      0   522
Processing: 6171 13014 1390.6 12957 20882
Waiting:    6171 12990 1392.0 12925 20880
Total:      6171 13018 1390.8 12963 20887

Percentage of the requests served within a certain time (ms)
 50% 12963
 66% 13201
 75% 13394
 80% 13512
 90% 13819
 95% 14607
 98% 18288
 99% 19436
100% 20887 (longest request)
Mac-user:~ user$ 5~

```

Figura 17. Captura test apache

En cuanto a *JMeter*, no requiere una instalación como tal, ya que se trata de una aplicación diseñada en Java 100%, por lo que puede ser ejecutada en cualquier plataforma que tenga instalada la máquina virtual java 1.5 o superior.

Una vez descargada lo único que tenemos hacer es ejecutar el archivo bin>jmeter.bat (en el caso de Windows) y se mostrará la ventana de la aplicación. A partir de ahí habrá que realizar los distintos cambios y configuraciones que creamos oportunos para realizar el test.

Sin querer hacer un tutorial de *JMeter*, se realizará una breve explicación de los pasos a seguir en la configuración del mismo para llevar a cabo el test.

Con *JMeter* podemos hacer que grabe todas las peticiones que hagamos desde un navegador web hacia el servidor web, por lo que nos ahorramos la comunicación entre el cliente web y la aplicación. Para llevar a cabo dicho trabajo tendremos que realizar una serie de configuraciones tanto en la herramienta *JMeter* como en el navegador donde reside dicha herramienta.

Respecto a la secuencia que *JMeter* va a grabar y llevará a cabo contra la aplicación web, será siempre la misma respecto a los distintos sistemas operativos. Esta secuencia será la siguiente:

- Acceder a la página principal de *Wordpress* en dicho servidor
- Ingresar con usuario y contraseña a la aplicación web
- Crear una nueva entrada dentro *Wordpress*
- Publicar dicha entrada
- Ver la entrada nueva
- Salir de *Wordpress*

Esta secuencia se puede ver en la Figura 20

Una vez que ya sabemos la secuencia que se va a realizar, pasamos a describir la configuración necesaria en *JMeter* para que lleve a cabo la grabación de esta secuencia. Para ello cuando iniciamos *JMeter* se nos presenta una interfaz con dos partes (figura 18); en la izquierda aparecerá el *Plan de Pruebas* junto con el *Banco de Trabajo*, y en la parte de la derecha todas las configuraciones posibles de los anteriores dependiendo de cuál elegimos.

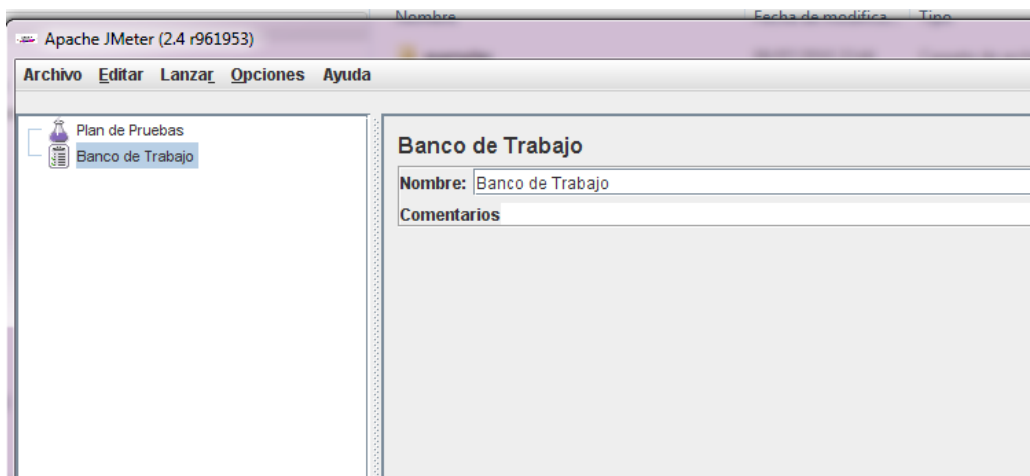


Figura 18. Interfaz de *JMeter*

Lo primero que haremos será crear un nuevo Grupo de Hilos dentro del Plan de Pruebas, para ello nos situamos sobre el mismo, y en la opción Editar>Añadir>Threads(Users)>Grupo de Hilos. Una vez creado podemos cambiar dicho nombre y configurar las propiedades, que como se puede ver comprende los siguientes parámetros (Figura 19):

- Número de hilos. Es el número de usuarios que *JMeter* va a simular. En nuestro caso utilizaremos el valor 10 y 40, por lo que simulará 10 o 40 usuarios accediendo a la web y realizando las mismas acciones ya descritas.
- Periodo de subida. Es el tiempo en segundos que se quiere que transcurra entre cada grupo de hilos. Utilizaremos el valor 10.
- Contador del bucle. Es el número de ciclos que se van a repetir. En este caso realizaremos 1 ciclo. Tenemos la opción de “Sin fin”, que realizaría el test indefinidamente.

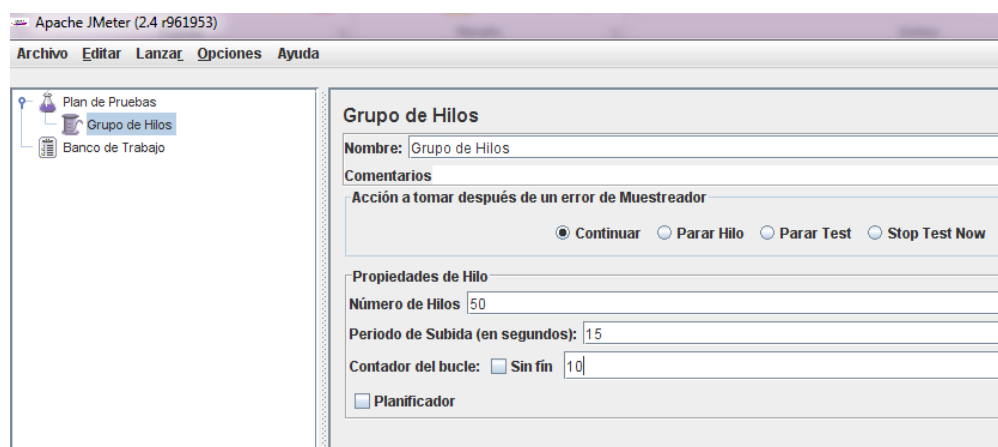


Figura 19. Parámetros del Grupo de Hilos

Si ya tenemos creado el Grupo de Hilos, procedemos a crear dentro del Banco de Pruebas un elemento *NoDePrueba*, concretamente “*Servidor Proxy HTTP*”. Con ello estamos configurando *JMeter* como servidor Proxy, así lo único que tenemos que hacer es indicarle al navegador web que utilice un servidor proxy en sus conexiones y le indicamos el puerto configurado en el mismo que por defecto será el 8080. Una vez configurado el servidor proxy pulsamos en el botón *Arrancar*, y ya tenemos funcionando el servidor proxy. A partir de este momento y una vez configurado el navegador web correspondiente, todas las páginas consultadas por dicho navegador quedarán grabadas en el plan de trabajo (Figura 20).



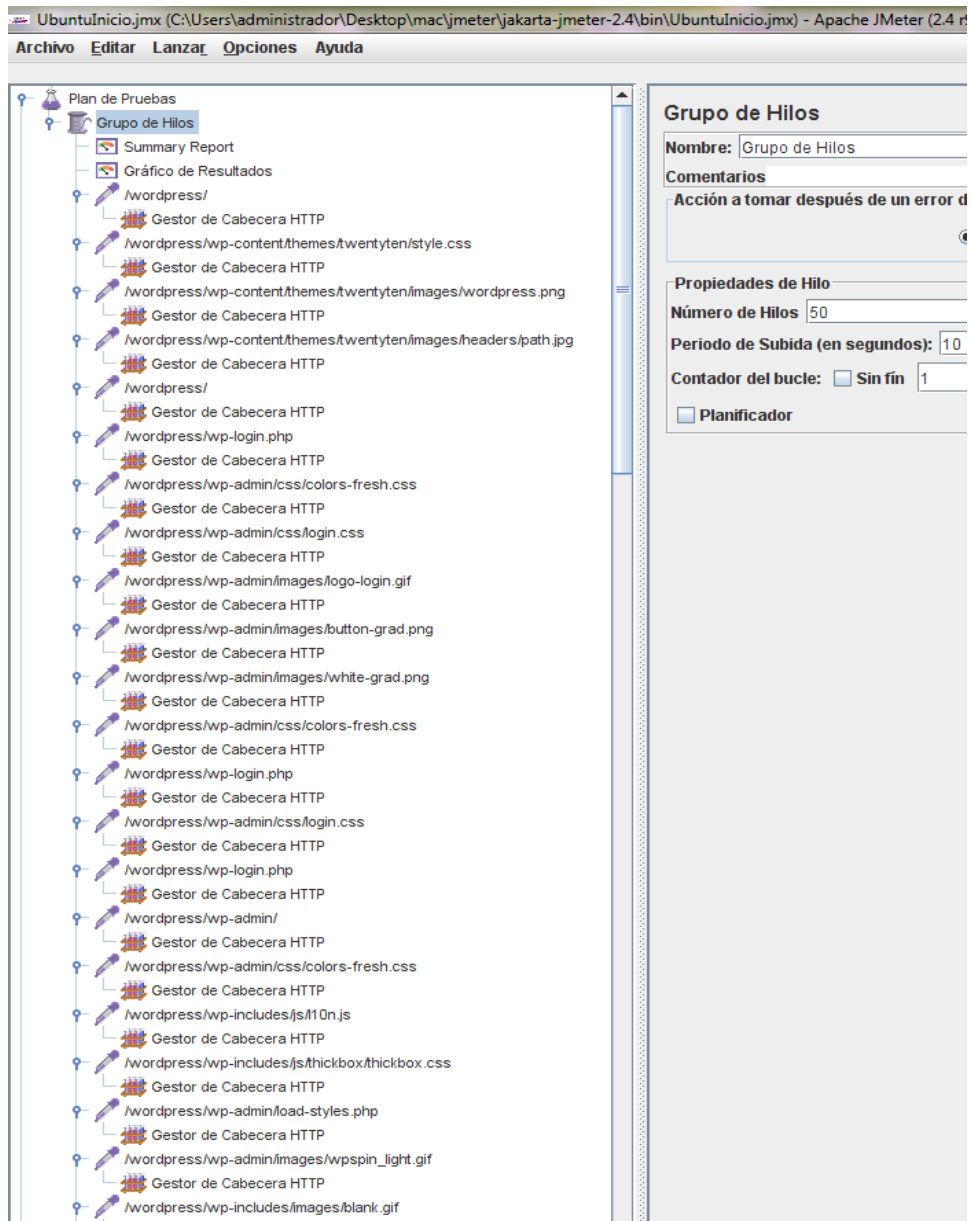


Figura 20. Captura de la secuencia de las páginas visitadas

Con *JMeter* se puede seleccionar una serie de herramientas que analizan y evalúan el test realizado, obteniendo de esta manera gráficas o tablas de resultado que ayudan a la hora de estudiar el rendimiento del sistema.

Hay diferentes “*Listener*”, como se denomina en *JMeter*, con los que obtener los datos del rendimiento del equipo a evaluar. Algunos de ellos son válidos para conocer el rendimiento de la CPU de un equipo u otras medidas que no serán llevadas a cabo en este trabajo. Las dos herramientas o monitores que se utilizarán son: “*Graph Results*” (Gráfico de resultados – Figura 21) y “*Summary Report*” (informe de síntesis – Figura 22). En el primero se obtiene varios gráficos con la evolución de la prueba. Estos gráficos indican

parámetros como el número real de peticiones por minuto que el servidor maneja realmente, lo que nos permite probar con parámetros que aumenten el número de usuarios concurrentes y/o disminuyan los retrasos hasta alcanzar el mayor rendimiento.

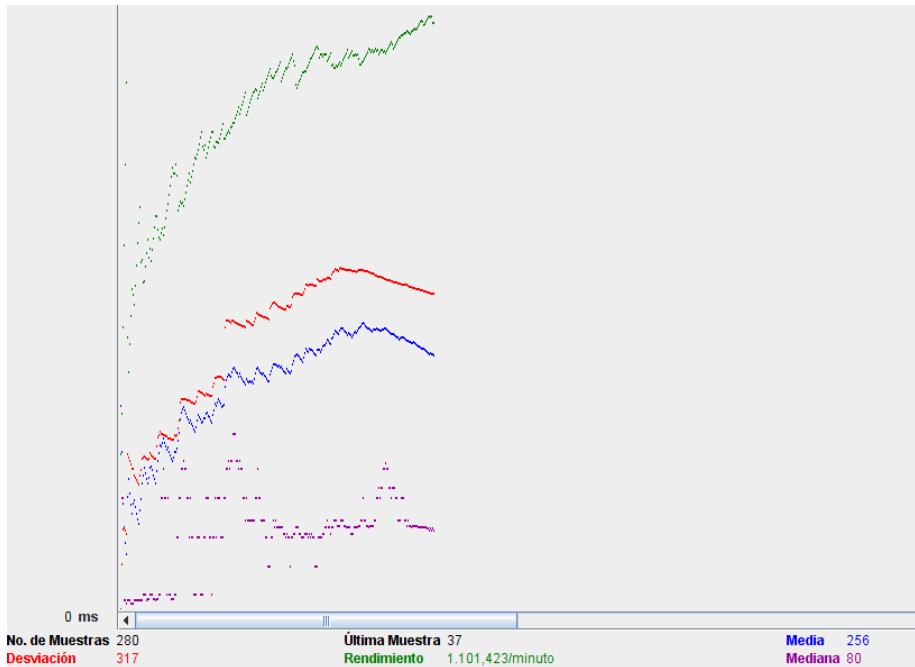


Figura 21. Gráfico de resultado de JMeter

Respecto al informe, se obtiene una tabla con los datos referidos a cada una de las muestras (cada uno de los hilos que forman el test). De cada una de ellas se puede observar el número de veces que se accede a esa muestra, el tiempo medio transcurrido, el rendimiento referido a las páginas por segundo que el servidor ha servido, etc. Además se obtienen los totales de dichos parámetros, así tendremos una idea clara del rendimiento actual del servidor.

Label	# Muestras	Media	Mín	Máx	Std. Dev.	% Error	Rendimiento	Kb/sec	Avg. Bytes
/wp-login.php	30	480	112	1256	335,69	33,33%	2,7/sec	4,93	1868,
/wp-admin/css/...	10	8	4	18	4,83	0,00%	1,0/sec	2,23	2190,
/wp-admin/css/...	20	8	6	13	2,00	0,00%	1,7/sec	51,49	30355,
/wp-admin/ima...	10	5	3	13	2,91	0,00%	1,0/sec	4,90	4816,
/wp-admin/ima...	10	3	2	5	1,04	0,00%	1,0/sec	0,25	243,
/wp-admin/	10	629	254	986	257,19	0,00%	54,6/min	1,89	2133,
/wp-includes/js...	10	7	5	22	5,00	0,00%	55,7/min	3,51	3864,
/wp-includes/js...	20	4	3	11	1,80	0,00%	1,5/sec	0,44	308,
/wp-admin/lnd...	50	605	264	948	177,87	0,00%	3,8/sec	8,07	2172,
/safebrowsing/...	10	171	80	321	85,54	0,00%	52,4/min	0,81	952,
/safebrowsing/...	10	232	75	766	210,30	0,00%	53,5/min	6,50	7460,
/wp-admin/ima...	10	19	3	90	31,34	0,00%	55,1/min	0,25	284,
/wp-admin/pos...	10	585	268	865	213,95	0,00%	54,1/min	1,88	2141,
/	10	341	151	513	135,08	0,00%	56,2/min	8,31	9076,
/wp-includes/js...	10	5	4	6	0,63	0,00%	58,1/min	0,74	786,
/wp-includes/js...	10	5	3	7	1,20	0,00%	58,1/min	1,81	1917,
/wp-includes/c...	10	5	4	7	1,14	0,00%	58,1/min	5,49	5804,
/wp-includes/fi...	10	4	3	6	1,10	0,00%	58,1/min	0,70	737,
/avatar/720e72...	10	567	336	1831	432,35	0,00%	56,2/min	0,26	283,
/avatar/ad5165...	10	106	37	370	104,10	0,00%	1,1/sec	0,31	283,
TOTAL	280	256	2	1831	317,79	3,57%	18,4/sec	77,32	4313,

Figura 22. Tabla del informe de síntesis de JMeter

## 6. Evaluación de los Sistemas Operativos

Tras la instalación y configuración de los sistemas operativos, así como de la herramienta Apache y demás software necesario para hacer funcionar *Wordpress* (como php y mysql), se ha llevado a cabo una serie de pruebas de estrés a cada uno de los sistemas para comprobar su rendimiento.

Para la realización de las pruebas se han utilizado las herramientas de *benchmarking* ya mencionadas (*ab* y *JMeter*). Los resultados de las mismas serán expuestos en los siguientes apartados, realizando un análisis de aquellos aspectos que se consideren más relevantes. Cabe decir que el resultado que se obtenga y sea plasmado en sus respectivas tablas, será la media resultante de la realización de 5 pruebas del mismo tipo y con el mismo software, es decir, en cada una de las pruebas realizadas se ejecutarán 5 pruebas iguales, y la media de las mismas será el resultado que se plasmará en los siguientes apartados.

Las pruebas a realizar en el test inicial consistirán en dos partes. Por un lado realizaremos una prueba con la herramienta *ab* a la página principal de *Wordpress*, en concreto a la página *index.php*. De esta forma se podrá ver el cambio que se produce en cuanto a servir la página de forma dinámica, o servir dicha página estáticamente debido a la caché en disco.

Por otro lado, con la herramienta *JMeter* se podrá realizar un test más completo, ya que se seguirá la secuencia descrita en el apartado 5.3.2. (Figura 20). Como se aprecia en dicha secuencia, se va a proceder a testear tanto páginas estáticas (fundamentalmente *.css*, *.js*) como páginas dinámicas. En este caso, hay algunas páginas dinámicas que no pueden ser cacheadas, por lo que el resultado obtenido será bastante realista en cuanto al rendimiento aportado una vez realizado los cambios de configuración en Apache.

Respecto a los cambios a realizar, se realizarán primero los cambios pertinentes en el servidor web Apache, y una vez obtenido dichos resultados, se realizarán también cambios en la configuración de memoria física y procesador para constatar el rendimiento que los mismos producen en los sistemas.

## 6.1. Test inicial

Con estos test se podrá tener una primera aproximación del rendimiento que ofrecen los sistemas operativos en su tarea de servir páginas web.

Se han llevado a cabo dos test con diferentes valores. En el primero se ha utilizado 10 usuarios concurrentes realizando cada uno de ellos una petición de 100 páginas.

Mientras que en el segundo test se ha realizado una muestra con 50 usuarios concurrentes realizando 500 peticiones. De esta forma se intenta llevar al servidor más al límite.

Tras examinar los resultados de los diferentes test realizados, se ha creado la siguiente tabla (Figura 23) y una gráfica en la que se puede apreciar mejor las diferencias entre cada uno de los sistemas. Como se observa en dicha tabla, el sistema que obtiene mejores resultados es Ubuntu, tanto en el número de páginas que el servidor puede servir por segundo, cómo por la utilización de la memoria física. Esto indica que no por utilizar más memoria, necesariamente se obtiene mayor rendimiento, sino que hay que buscar la configuración y herramientas idóneas para realizar una tarea con el mejor rendimiento.

Los datos referidos en la tabla corresponden a:

- Uso de memoria. Es el uso de memoria Física en el momento en el que se realiza el test.
- Rendimiento (muestras/segundos). Es el número medio de páginas que se han servido a lo largo del test.
- Tiempo de respuesta por petición (segundos). Es el tiempo en segundos de la duración en que el servidor tarda en servir una petición.
- Tiempo de respuesta media de conjunto (segundos). Es el tiempo medio en segundos que ha tardado el servidor en servir todas las páginas solicitadas desde JMeter, donde no sólo se realiza la petición de una página sino de varias.

Test con 10 usuarios y 100 muestras

	Uso de Memoria (%)	Rendimiento (muestras/segundos)	Tiempo de respuesta por petición (segundos)
Ubuntu	35	7,13	0,139
Windows 7	75	3,15	0,364
Mac OS X	50	4,02	0,263

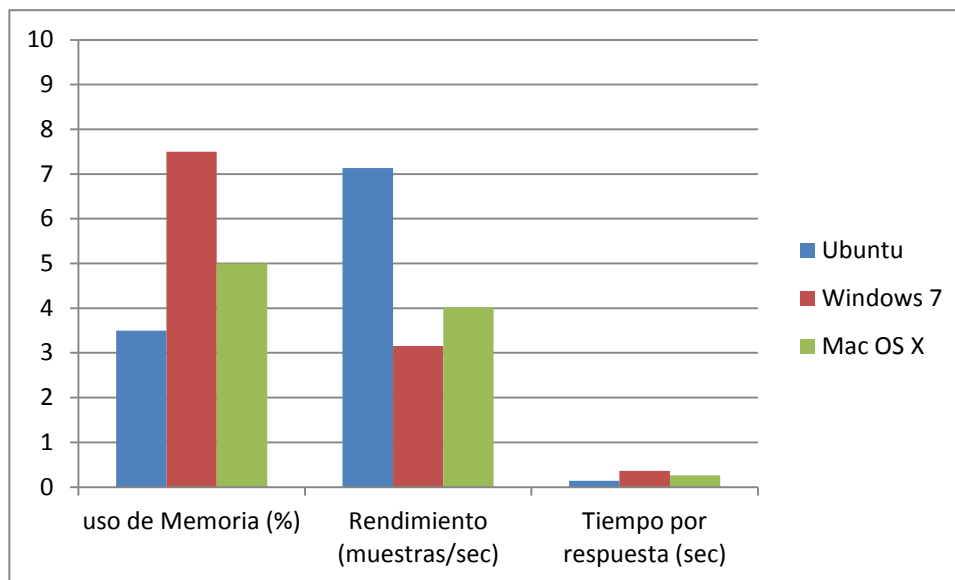


Figura 23. Tabla y gráfico de resultado del test 1.

Se ha realizado un segundo test inicial para observar el comportamiento de los sistemas ante una mayor carga de trabajo. En concreto, se ha configurado los benchmark con 50 conexiones concurrentes cada una de las cuales realiza 500 solicitudes al servidor. Los datos obtenidos se recogen en la siguiente tabla y gráfico (Figura 24).

En esta tabla se recoge el tiempo medio de respuesta de todo el conjunto de páginas analizadas mediante JMeter. Este tiempo nos servirá para compararlo con el test final.

En los datos recogidos se puede apreciar cómo tanto el sistema Ubuntu como el sistema Mac OS X realizan un mayor uso de memoria frente al test anterior, y cómo disminuye en todos los casos el rendimiento. En cualquier caso, sigue siendo Ubuntu el sistema con mejor rendimiento.

En este sentido encontramos un caso curioso, y es que Ubuntu obtiene un mayor rendimiento respecto a Mac OS X, siendo ambos sistemas prácticamente iguales en cuanto a que los dos se basan en la arquitectura de

UNIX. Incluso el uso de memoria es menor. La explicación se puede encontrar en la compilación del kernel. No siendo el estudio del mismo parte de este trabajo, sí se ha podido constatar tras la búsqueda de información que el núcleo de Mac OS X tiene problemas de rendimiento provocados por el Kernel Mach. El estudio [42], realizado por Jasjeet S. Sekhon<sup>6</sup> demuestra que Linux maneja mejor los bloques continuos de memoria de cierto tamaño (en su caso de 35 KB) frente a Mac OS X. Y pone de relevancia que dicho rendimiento se debe a la compilación del kernel de ambos sistemas. De esta forma, Jasjeet S. Sekhon ha constatado que una compilación del kernel cambiando el sistema de gestión de memoria se consigue unos resultados idénticos en ambos sistemas, pero no así con el sistema de gestión estándar de Mac OS X.

Test con 50 usuarios y 500 muestras

	Uso de Memoria (%)	Rendimiento (muestras/segundos)	Tiempo de respuesta (segundos) por petición	Tiempo de respuesta media de conjunto (segundos)
Ubuntu	83	6,89	0,145	2,587
Windows 7	75	2,93	0,417	4,695
Mac OS X	80	3,77	0,265	4,081

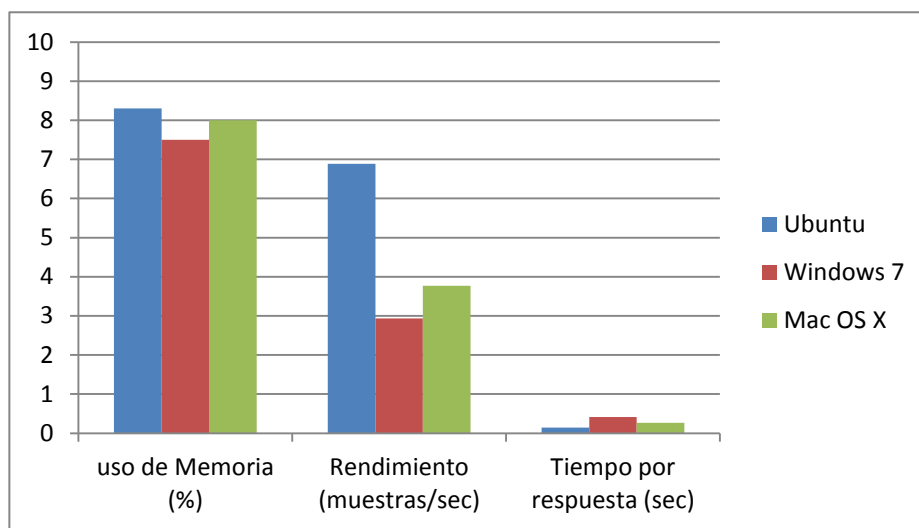


Figura 24. Tabla y gráfico de resultado del test 2.

Aunque no se ha recogido en los resultados anteriores, cabe mencionar que en el segundo test realizado al sistema Ubuntu, en un momento dado el mismo ha tenido que hacer uso de la memoria de intercambio o Swap, lo que ha podido penalizar ligeramente su rendimiento. Esto se aprecia en la siguiente

<sup>6</sup> Profesor de la universidad de Berkeley

captura del monitor de sistema (Figura 25). El uso es sólo del 0,1% por lo que prácticamente es inapreciable el momento en el que el sistema utiliza dicha memoria de intercambio.

También cabe reseñar que el uso de CPU en cada uno de los sistemas ha sido del 100% en todos los casos, es decir, todos los test realizados utilizaban el total de uso de la CPU mientras estaban en ejecución.

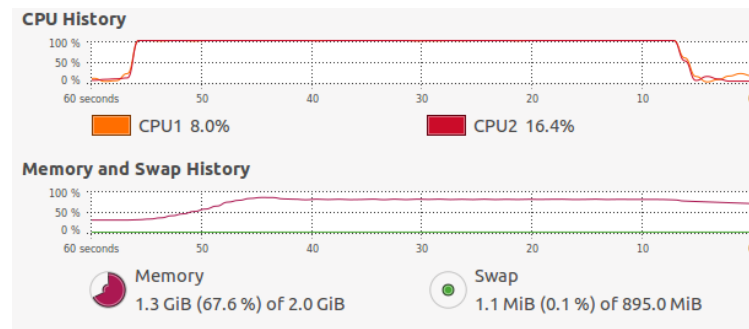


Figura 25. Monitor Sistema Ubuntu (test 2).

## 6.2. Cambio de configuración

Una vez realizado el test inicial se procede a realizar cambios en la configuración de Apache y sus módulos para lograr un rendimiento óptimo.

En concreto, se utilizará como punto de partida la página de apache que indica cómo mejorar el rendimiento [34].

Como indica el recurso mencionado hay que tener cuidado de no utilizar swapping para ejecutar los procesos de apache, ya que aumentaría la latencia de cada petición, lo que llevaría a los usuarios a recargar las páginas incrementando la carga en el servidor.

En este *Apache Performance Tuning* (Mejora del rendimiento de Apache) se indica una serie de recomendaciones dentro de la configuración del software para conseguir aumentar el rendimiento. Estas modificaciones se realizan dentro del fichero de configuración de apache (por defecto “*apache2.conf*”). A parte de estos cambios se realizarán otro tipo de configuraciones que serán comentadas más adelante.

A continuación se irán explicando cada uno de las configuraciones que requiere Apache para lograr un mejor rendimiento, además en cada una de ellas se indicará el rendimiento obtenido con el cambio o no modificación de la

configuración (los valores corresponden a peticiones/segundos que el servidor procesa):

- Respecto a las consideraciones que se plantean, es recomendable deshabilitar la resolución de nombres de los host, ya que en cada petición habría que realizar la resolución de DNS antes de finalizarla, lo que incrementa el tiempo de respuesta. Apache por defecto tiene deshabilitada dicha función, pero cabe reseñar que la activación de la misma comportaría un decremento del rendimiento. El comando a utilizar será “*HostnameLookups off*”.

Test	Ubuntu 10.10	Windows 7	Mac OS X
<i>HostnameLookups off</i>	7.10	2.93	3.85
<i>HostnameLookups on</i>	6.73	2.78	3.66

- Otra media es utilizar es la desactivación de enlaces simbólicos mediante el comando “*Options FollowSymLinks*”, con ello evitamos que Apache busque el contenido fuera de los directorios de la web. De distinta forma ocurre con la directiva de negociación de contenido “*MultiViews*”. Si la tenemos activada, cuando un usuario realiza una petición de una página en un determinado idioma, Apache buscará dentro de los recursos dicha página, por lo que hará que aumente la latencia a la hora de servir las peticiones.
- De un modo similar, Apache cuando recibe una petición mira primero el fichero “*.htaccess*” para ver si hay alguna configuración especial, y busca en todos los subdirectorios hasta la raíz en busca de dicho archivo. Para evitar esto utilizamos el comando “*AllowOverride None*”.

En el siguiente test se han utilizado las variables *FollowSymLinks*, *MultiViews* y *AllowOverride*.

Test	Ubuntu 10.10	Windows 7	Mac OS X
<i>AllowOverride All NO FollowSymLinks MultiViews</i>	6.89	2.93	3.77
<i>AllowOverride None FollowSymLinks NO MultiViews</i>	7.93	3.86	4.62



- Apache permite mantener conexiones permanentes por cada petición, de manera que un usuario con la misma conexión puede obtener páginas html y en la misma conexión obtener a su vez imágenes. Para ello hay que activarlo (“*KeepAlive on*”) que es el valor por defecto de Apache. Además se puede indicar el número de peticiones que se va a permitir con la misma conexión (“*MaxKeepAliveRequest 100*”) y el tiempo de espera entre peticiones antes de cerrar la conexión (“*KeepAliveTimeout 15*”). En el caso de estudio, la aplicación web *Wordpress*, sólo se ha modificado el valor para el tiempo de espera, en concreto se ha bajado dicho valor a “*KeepAliveTimeout 10*”. De esta forma reducimos el tiempo de espera para servir otra página a través de la misma conexión, y no mantenemos conexiones a la espera de recibir una nueva petición.

Test	Ubuntu 10.10	Windows 7	Mac OS X
<i>KeepAlive off</i>	6.33	2.45	3.42
<i>KeepAlive on MaxKeepAliveRequest 100 KeepAliveTimeout 10</i>	6.92	2.92	3.78

- Compresión de las páginas. Se puede configurar Apache para que realice la compresión de las páginas solicitadas. Con ello se reduce el tamaño de las páginas descargadas entre un 30%-50% y por ende el tiempo de conexión. En consecuencia, obtenemos un mejor rendimiento.

Para ello se utiliza el módulo de Apache “*mod\_deflate*”. Lo único que tenemos que hacer es habilitarlo mediante el comando “*a2enmod deflate*”. Dicha compresión puede ser indicada para un solo directorio, un sitio o todos los sitios del servidor. Dependiendo de lo que queramos comprimir utilizaremos el comando “*SetOutputFilter DEFLATE*” dentro de un fichero *.htaccess*, dentro de la directiva “*<directory>*” en el fichero de configuración del sitio o en el fichero de configuración de apache “*apache2.conf*”.

Una vez que tenemos habilitado dicho módulo podemos configurar los tipos de ficheros a comprimir.

Por defecto, dicho módulo comprime todas las páginas html, todos los textos planos, los ficheros de estilo en cascada (“css”) y aplicaciones JavaScript (“js”). Se pueden configurar para cualquier tipo de extensión, pero excepto en las mencionadas, en la mayoría de las extensiones la compresión es mínima por lo que no compensa hacer trabajar a la CPU para dicha compresión.

Test	Ubuntu 10.10	Windows 7	Mac OS X
<i>Mod_deflate</i> desactivado	6.92	2.94	3.77
<i>Mod_deflate</i> activado	7.17	3.12	3.89

- Para evitar que el servidor esté continuamente comprimiendo las páginas solicitadas y con ello aumente la carga de trabajo, se puede utilizar el módulo “*mod\_cache*” junto con el módulo “*mod\_disk\_cache*”. Con la utilización de estos módulos evitamos tener que estar continuamente comprimiendo ciertas páginas estáticas.

Para habilitar los módulos creamos un enlace a los mismos en “*mods\_enabled*” y ejecutamos el comando “*a2enmod cache*” y “*a2enmod disk\_cache*”.

La configuración de la caché se encuentra en el fichero “*disk\_cache.conf*”. En este último, para que funcione la caché en disco, quitamos el comentario a la línea “*CacheEnable disk !*”, y reiniciamos el servidor apache. Con ello estamos cacheando todo el servidor web, pero se puede cachear sólo un directorio.

Los únicos parámetros que aparecen por defecto y que son necesarios son: “*CacheDirLevels*” que indica la creación de un árbol de directorios donde se almacenan los ficheros cacheados ya que si estuvieran en el mismo directorio penalizaría el rendimiento, y “*CacheDirLength*” que es el número de caracteres de cada directorio. Dentro del módulo “*mod\_disk\_cache*” se puede configurar las directivas tamaño máximo y tamaño mínimo de los ficheros.

El módulo “*mod\_cache*” tiene más directivas configurables, entre ellas cabe destacar el tiempo de expiración de la caché. Esto es importante ya

que si no se actualiza la caché se corre el riesgo de que las páginas servidas no estén actualizadas. Si las páginas están en continuo cambio, un tiempo razonable puede ser 30 minutos.

Para limpiar de la caché los archivos que han expirado tenemos la herramienta *htcacheclean*, que se instala junto con el módulo *mod\_disk\_cache*. Se puede configurar dicha herramienta para que se ejecute cada cierto tiempo (lo normal es que sea el mismo tiempo que el de la expiración de las páginas cacheadas) mediante el comando “*htcacheclean -d30 -n -t -p /var/cache/apache2/mod\_disk\_cache -l 100M -i*”. De esta forma le indicamos que se ejecute cada 30 minutos (-d30) en el directorio (-p “/./”) con un límite de almacenamiento de 100 MB (-l 100M). Además se especifica que se borre los directorios vacíos (-t) y que el borrado sea inteligente (-i), de manera que sólo se ejecute cuando haya una modificación en la caché del disco.

Respecto a la expiración de las páginas, también se puede utilizar el módulo de Apache “*mod\_expires*”, donde tenemos una mayor cantidad de directivas para indicar parámetros más específicos.

Test	Ubuntu 10.10	Windows 7	Mac OS X
<i>mod_cache</i> y <i>mod_disk_cache</i> desactivado	7.1	2.92	3.82
<i>mod_cache</i> y <i>mod_disk_cache</i> activado	19.2	13.54	14.95

- Separar las páginas estáticas de las dinámicas. Este es otro aspecto a la hora de mejorar el rendimiento de Apache. Apache al servir una página dinámica utiliza unos 20-30 MB de memoria física. Una vez servida la petición el proceso se queda a la espera de ejecutar otra petición en la misma conexión para evitar realizar continuas conexión. Si la siguiente petición es una imagen, esta petición está utilizando esos 30 Mb de memoria física cuando sólo requiere de unos cuantos Kb. La solución que se puede efectuar es la de separar las páginas dinámicas de las estáticas. Para ello se utilizará los módulos *mod\_proxy* y *mod\_rewrite*.

Además hay que tener instalado un segundo servidor para que uno se encargue de servir las páginas estáticas (servidor ligero) y el otro sirva las páginas dinámicas (servidor pesado).

Como servidor ligero se utilizará *lighttpd*, el mismo simplemente tendrá configurado el puerto 81 y el directorio donde se encuentra *Wordpress*. No tendrá ningún parámetro específico ya que simplemente se le pasará las páginas html, las imágenes, las hojas de estilo (css) y los JavaScript (js).

Como servidor web pesado se utilizará Apache, el cual tendrá toda la configuración descrita a lo largo de este documento.

Respecto a la configuración que tendrá Apache, ya bien sea en el archivo de configuración general (*apache.conf*) o dentro de las directivas de un directorio, para pasar las extensiones de un servidor a otro será la siguiente:

- “*RewriteEngine On*”, activamos el módulo *mod\_rewrite*
- Creamos la regla para indicar todas las imágenes y fichero estáticos que se enviarán al servidor ligero 

```
{RewriteRule ^/(.*)\.(jpg|jpeg/gif/png/swf/pdf/avi/mpeg/mpg/mp3/tgz/gz/css/html/js)$""http://127.0.0.1:81/$1.$2" [P]}
```
- Con el siguiente comando, “*ProxyPassReverse / http://127.0.0.1:81*”, indicamos que se envíe todo lo anterior al servidor ligero que escucha en el puerto 81 y que envíe la cabecera http para seguir utilizando los cookies. De esta forma el usuario no percibe si se está proporcionando las páginas desde un servidor u otro.

Con este tipo de configuración se mejora sobre todo en el uso de memoria física. Respecto al rendimiento depende de la cantidad de ficheros estáticos que se pueden servir desde distintos servidores.

Test	Ubuntu 10.10	Windows 7	Mac OS X
<i>Mod_proxy</i> y <i>mod_rewrite</i> desactivado	7.10	2.93	3.77
<i>Mod_proxy</i> y <i>mod_rewrite</i> activado	8.05	3.87	4.6

- Un aspecto importante a la hora de mejorar el rendimiento es elegir correctamente los parámetros del módulo MPM (*Multi-Processing Modules. Módulos de Procesamiento Múltiple*). Hay módulos de procesamiento específicos para algunas plataformas, por ejemplo para Windows el módulo es “*mpm\_winnt*”, que según la descripción de la página oficial de Apache crea un solo proceso hijo el cual crea tanto *Threads* (hilos) según el número de peticiones.

En cambio para plataformas de tipo UNIX se puede elegir entre varios MPM, lo que puede afectar al rendimiento de Apache. Estos MPM son: *Prefork* y *Worker*. La diferencia entre ambos radica en que mientras en *Prefork* se crean varios procesos hijos, utilizando uno por cada petición que se realiza, en *Worker* se crean varios procesos hijos con varios *Threads* cada uno, utilizando para cada petición un *Thread*. Este último es utilizado principalmente para equipos multiprocesadores y tiene un menor consumo de memoria respecto a *Prefork*.

El MPM a utilizar será *Worker* en el caso de los sistemas UNIX, que para que funcione con php requiere del módulo *mod\_fcgid*. Se ha decidido utilizar este MPM porque una buena forma de lograr un buen rendimiento en los sistemas Windows es utilizar *FastCGI*, ya que tiene más ventajas que ejecutarlo como controlador o como CGI, por eso los resultados obtenidos en dicho test respecto a Windows reflejan el comportamiento de *FastCGI*. Por ende, para que todos los sistemas utilicen los mismos módulos, siempre que se pueda, se ha decidido a utilizar dicho MPM.

Además tras realizar varias pruebas con ambos módulos, el módulo MPM *Worker* obtiene mejores resultados que el módulo MPM *Prefork*.

Para utilizar el MPM *Worker* hay que instalar los módulos *apache2-mpm-worker* y *libapache2-mod-fcgid*, y habilitaremos el módulo fcgi con el siguiente comando “*a2enmod fcgid*”.

Los parámetros de configuración utilizados para los módulos *fcgi* y *Worker*, se han obtenido de los artículos de *2bits*<sup>7</sup>[39], de la web de

---

<sup>7</sup> 2bits es una empresa privada de consultoría especializada en el sistema gestor de contenidos *Drupal*.

RedHat<sup>8</sup> [40], y en el libro de “La Biblia de Servidor Apache2” de Mohammed J. Kabir [41].

Los servidores hijo utilizados al comenzar Apache (*StartServers*) serán 3, con un máximo (*ServerLimit*) de 3000. Esto se produce en los sistemas UNIX, mientras en los sistemas Windows, cómo se ha comentado más arriba, la versión de Apache es multihilo, por lo que un solo proceso maneja todas las solicitudes, de esta forma lo único que podremos configurar será el límite de hilos.

Cada proceso hijo puede generar a su vez un hilo, estableciéndose un límite de 25 hilos por proceso (*ThreadsPerChild*).

Respecto al máximo de solicitudes simultáneas que el servidor puede procesar (*MaxClients*), se establece en 300 peticiones concurrentes.

Se fija un máximo de 10 hilos en espera (*MaxSpareThreads*).

También cabe mencionar que el límite de solicitudes que puede procesar un servidor hijo (*MaxRequestPerChild*) es configurado a 0, por lo que estos procesos nunca mueren.

Test	Ubuntu 10.10	Windows 7	Mac OS X
<i>mpm Prefork</i> activado	7.1	2.93	3.78
<i>mpm_worker</i> activado	18.5	15.4	15.8

Además se han realizado distintas pruebas con los parámetros configurables de Worker. No aparecen los valores para Windows porque no utiliza dicho módulo.

Variables	Valores por Defecto	Valores			
<i>StartServer</i>	2	5	500	100	5
<i>ThreadsPerChild</i>	25	25	10	15	10
<i>MinSpareThreads</i>	25	5	3	3	3
<i>MaxSpareThreads</i>	75	15	10	50	10
<i>MaxClients</i>	150	300	300	150	150
<i>MaxRequestChild</i>	0	0	1000	1000	0
<b>Ubuntu 10.10</b>	18.5	16.8	18.5	19	18.3
<b>Mac OS X</b>	15.8	14.8	15.7	15.8	15.7

<sup>8</sup> Sistema operativo basado en Linux al igual que Ubuntu.

Para aglutinar todas las variables que se han modificado en Apache para obtener un mejor rendimiento, se ha creado el siguiente resumen mediante la siguiente tabla.

VARIABLE	PARÁMETROS	DESCRIPCIÓN
<i>HostnameLookups</i>	<i>No</i>	Deshabilita la resolución de nombres de los equipos que realizan peticiones
<i>FollowSymLinks</i>	Activada	Evitamos que Apache busque un contenido fuera de su directorio.
<i>Multiviews</i>	Desactivada	Si no utilizamos esta directiva evitamos la negociación de contenido en otros idiomas.
<i>AllowOverride</i>	<i>None</i>	Evita que Apache realice una búsqueda por todas las carpetas hasta la raíz del fichero de configuración <i>.htaccess</i>
<i>KeepAlive</i>	On	Habilita la espera en la conexión para realizar varias peticiones con la dicha conexión
<i>MaxKeepAliveRequest</i>	100	Establece el límite de peticiones que puede procesar una conexión
<i>KeepAliveTimeout</i>	10	Establece el tiempo máximo de una conexión antes de cerrarse si no recibe una nueva petición.
<i>mod_deflate</i>	Activado	Si activamos dicho módulo se realizará la compresión de las páginas solicitadas
<i>mod_cache</i> , <i>mod_disk_cache</i>	Activado	Con la activación de estos módulos las páginas solicitadas serán cacheadas en el disco.
<i>mod_proxy</i> , <i>mod_rewrite</i>	Activado	Con estos dos módulos se puede realizar la separación de las páginas estáticas de las dinámicas. Para ello necesitamos un 2º servidor web
<i>mpm_prefork</i> , <i>mpm_worker</i>	El elegido por rendimiento es <i>mpm_worker</i>	Estos módulos están disponibles en la versión de UNIX, y cada uno realiza una gestión de los procesos de manera diferente.
<i>StartServer</i>	3	Número de servidores hijo con los que Apache inicia al recibir una petición
<i>ServerLimit</i>	3000	Límite de servidores hijo que puede utilizar Apache
<i>ThreadsPerChild</i>	25	Hilos que pueden generar cada servidor hijo
<i>MaxClients</i>	300	Número máximo de peticiones concurrentes que Apache procesa
<i>MaxSpareThreads</i>	10	Número máximo de hilos en espera
<i>MaxRequestChild</i>	0	No se establece límite del número de solicitudes que un servidor hijo puede procesar

Figura 26. Resumen cambios realizado en Apache.

Sin querer adelantarnos a los resultados de los test, tras una prueba para comprobar los resultados en cuanto a la diferencia de rendimiento entre los dos distintos módulos MPM en los sistemas UNIX, cabe reseñar que la cantidad de memoria consumida (Figura 27) ante una prueba de 50 usuarios simultáneos con el módulo MPM *Worker* y el módulo *fcgid*, es prácticamente la mitad a la utilizada con el MPM *Prefork*, que fue el utilizado en los test de inicio.

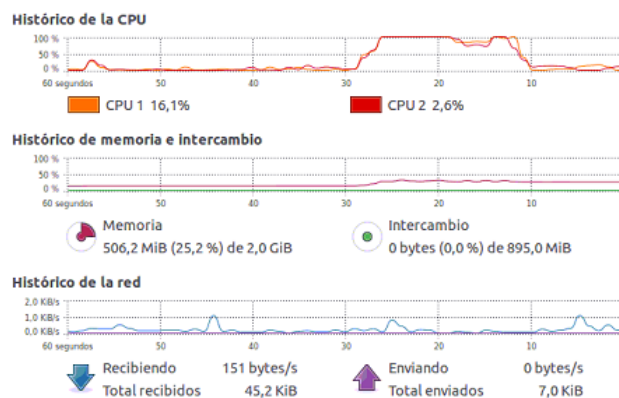


Figura 27. Monitor Sistema Ubuntu con 50 usuarios concurrentes.

Aprovechando que los sistemas están instalados en máquinas virtuales, se pueden utilizar distintas configuraciones en cuanto a memoria y CPU se refiere. Por tanto realizaremos distintas configuraciones para ver el rendimiento con escenarios diferentes. Las configuraciones que podemos utilizar son: la configuración de memoria física, se configurarán los sistemas pertinentes aumentando o disminuyendo su valor, así si actualmente los sistemas disponen de 2 GB, se pueden realizar configuraciones desde 256 MB a 3.5 GB que es el máximo disponible, y la otra variable a cambiar sería el procesador, donde existe la posibilidad de configurar 1, 2, 4, u 8 procesadores, con 1, 2, 4, u 8 Core cada procesador. En este caso debido al hardware disponible sólo se pueden utilizar dos configuraciones: 1 procesador o 2 procesadores, que es lo máximo que puede soportar el equipo utilizado.

### 6.3. Test Final

Antes de realizar todos los cambios de Apache y realizar los respectivos test, es conveniente ver cómo afecta cada cambio en la configuración en cada uno de los sistemas, aunque en algunos de los test se utilizarán varias variables.



Por tanto, la siguiente información (Figura 27) recoge el rendimiento antes de los cambios y después de los cambios.

Una vez implementado dichos cambios en la configuración de Apache y sus módulos, se crearán varias pruebas para conocer el comportamiento de cada uno de los sistemas. De esta forma se podrá tener una idea más clara de cuál de ellos gestiona mejor los procesos y la memoria física, así como quien obtiene mejor tiempo de respuesta ante una petición.

Al igual que se realizó en el test inicial, las herramientas de *benchmark* serán las mismas (*ab* y *JMeter*), y los parámetros a utilizar los ya descritos en los test inicial. Aunque en este caso sólo se realizará los test con 50 usuarios concurrentes para compararlos con los realizados en el test inicial.

Tras examinar los resultados de los diferentes test realizados, se han plasmado en la siguiente tabla y gráfica (Figura 28).

Test con 50 usuarios y 500 muestras

	Uso de Memoria (%)	Rendimiento (muestras/segundos)	Tiempo de respuesta media de conjunto (segundos)
Ubuntu	40	27,6	0,905
Windows 7	65	15,9	1,433
Mac OS X	43	16,3	1,373

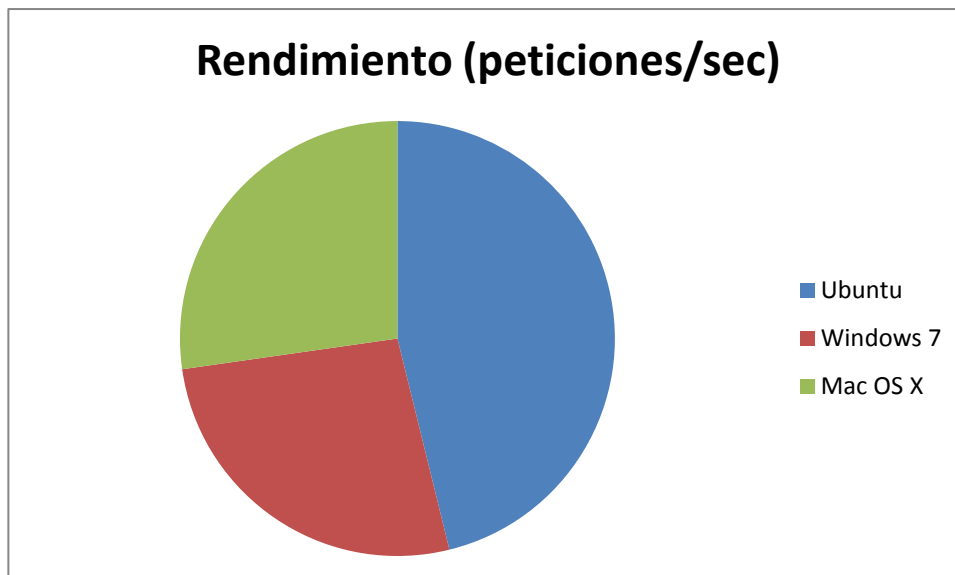


Figura 28. Tabla y gráfico de resultado del test 2.

Como se aprecia en la tabla, sólo se recoge el tiempo medio del conjunto de páginas analizadas. Esto se debe a que en la configuración de los servidores se ha utilizado el módulo caché. Debido a tal causa, los test realizados sobre una única página, nos daban resultados del orden de 100 veces mayor rendimiento, mientras que el test realizado sobre un conjunto de páginas no daba tal valor, por lo que podría llegar a no entenderse bien los resultados.

De todas formas se mostrarán los resultados de los test realizados sobre una sola página para compararlos entre los sistemas.

La lectura principal que se puede hacer de los resultados es la reducción en el consumo de memoria en todos los sistemas. Entre otros, la separación entre páginas dinámicas y estáticas, junto con el módulo de caché de disco, según se ha podido apreciar, son los que más han contribuido a tal reducción.

Otra lectura que podemos realizar es que Ubuntu se posiciona como el mejor sistema para ejecutar un servidor web Apache a la hora de servir páginas web dinámicas e incluso estáticas, prácticamente obtiene el doble de rendimiento frente a los demás sistemas.

La siguiente muestra obtenida con el *apache benchmark*, se observa lo ya comentado del rendimiento, el cual es mucho mayor que los obtenidos hasta ahora. La razón es la de ser una simple página estática y no una página dinámica.

Test con 50 usuarios y 500 muestras

	Rendimiento (muestras/segundos)	Tiempo de respuesta (ms)
Ubuntu	2681.54	3.72
Windows 7	626.67	16.45
Mac OS X	1439.84	6.45

Figura 29. Tabla de resultado del test páginas estáticas.

Para conocer cuál de los tres sistemas tiene mejor respuesta a la escalabilidad de las peticiones de páginas, se han realizado los mismos test pero con una carga de trabajo mayor, es decir, se han realizado test con 100 usuarios concurrentes y 200 usuarios concurrentes. Ambos test han realizado la petición de las páginas de la secuencia descrita en la figura 20.

<b>Peticiones concurrentes</b>	50	100	200
<b>Ubuntu 10.10</b>	27.6	28.0	31.6
<b>Windows 7</b>	15.9	15.9	15.7
<b>Mac OS X</b>	16.3	16.5	16.8

Figura 30. Tabla de resultado de test con distintos usuarios.

Por lo que se aprecia en los resultados, se puede decir que los sistemas basados en UNIX tienen una mejor respuesta al aumento de peticiones con la configuración descrita en estos apartados que el sistema Windows. Probablemente se deba a la gestión de la memoria que realizan dichos sistemas, ya que

## 7. Análisis de resultados

Ante los datos obtenidos de los test, se confirma como el sistema operativo Ubuntu reacciona mejor ante la petición de una solicitud al servidor web Apache. Esto no quiere decir que los otros dos sistemas puedan dar una mayor prestación en otro tipo de escenario.

Aun cuando el sistema Ubuntu ha sido el único que, en un momento dado de los test<sup>9</sup>, ha necesitado usar la memoria Swap para poder servir los procesos que en ese momento requerían de algún tipo de memoria para ejecutarse, en todos los demás test el comportamiento de uso de la misma ha sido totalmente superior a los demás.

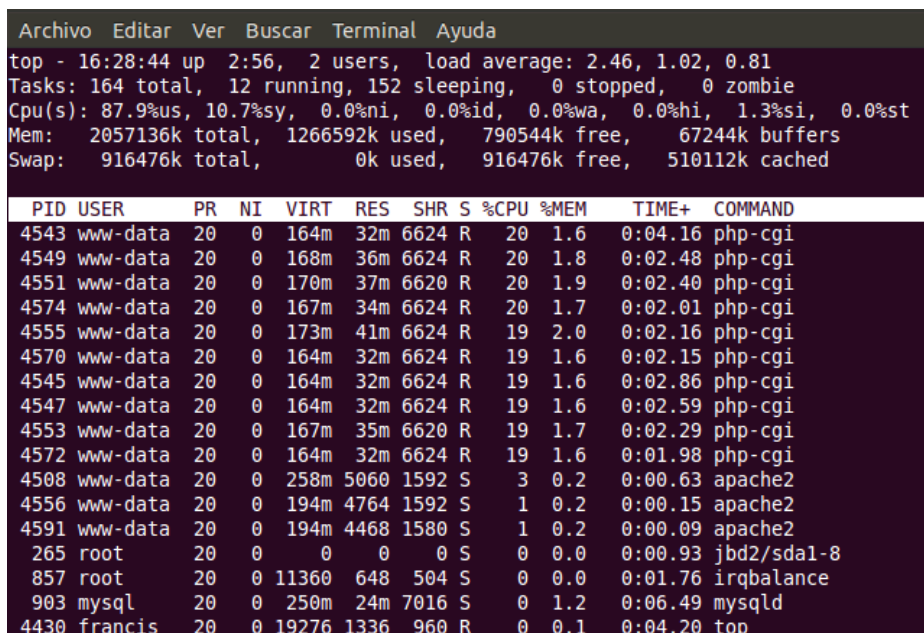
Este cambio, como se ha podido constatar a lo largo de la configuración y modificación de Apache, se debe sobre todo al cambio en el módulo de procesamiento múltiple (MPM), pasando del módulo *Prefork* al módulo *Worker*, y la instalación del módulo *fcgid* (necesario para la ejecución de php). Este último presenta una menor estabilidad frente al módulo *mod\_php*, pero ha sido necesaria su utilización para poder utilizar el módulo *Worker*.

Con este módulo lo que se logra es que el servidor web Apache pase las peticiones a la aplicación FastCGI, que se ejecuta fuera del mismo. El módulo *Worker* no requiere de un proceso por petición, como ocurría en el caso de *Prefork*, sino que un proceso hijo puede crear tantos subprocesos o hilos como peticiones haya.

<sup>9</sup> Página 44 de este trabajo, se puede observar en la figura 24.

En los sistemas UNIX (Figura 31), cuando realizamos varias peticiones, en este caso son 50 usuarios concurrentes, se inician 3 procesos de apache (configurado en la opción del módulo *Worker*), que a su vez crean los respectivos procesos de la aplicación FastCGI.

Además, se puede apreciar como cada proceso *php-cgi* ocupa una media de 32 MB (columna RES) de memoria física, sin embargo los procesos de Apache simplemente ocupan unos cuantos KB. Con el módulo *Prefork* se creaba un proceso de Apache de unos 27 MB por cada una de las peticiones. Al tener 50 peticiones, si las multiplicamos por el valor de memoria de cada proceso es lo que provoca que el consumo de memoria se dispare. Por tanto la mejor configuración respecto a la gestión de memoria es la utilización del módulo *Worker*.



```

Archivo Editar Ver Buscar Terminal Ayuda
top - 16:28:44 up 2:56, 2 users, load average: 2.46, 1.02, 0.81
Tasks: 164 total, 12 running, 152 sleeping, 0 stopped, 0 zombie
Cpu(s): 87.9%us, 10.7%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 1.3%si, 0.0%st
Mem: 2057136k total, 1266592k used, 790544k free, 67244k buffers
Swap: 916476k total, 0k used, 916476k free, 510112k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM   TIME+  COMMAND
 4543 www-data  20   0  164m  32m  6624 R   20  1.6   0:04.16 php-cgi
 4549 www-data  20   0  168m  36m  6624 R   20  1.8   0:02.48 php-cgi
 4551 www-data  20   0  170m  37m  6620 R   20  1.9   0:02.40 php-cgi
 4574 www-data  20   0  167m  34m  6624 R   20  1.7   0:02.01 php-cgi
 4555 www-data  20   0  173m  41m  6624 R   19  2.0   0:02.16 php-cgi
 4570 www-data  20   0  164m  32m  6624 R   19  1.6   0:02.15 php-cgi
 4545 www-data  20   0  164m  32m  6624 R   19  1.6   0:02.86 php-cgi
 4547 www-data  20   0  164m  32m  6624 R   19  1.6   0:02.59 php-cgi
 4553 www-data  20   0  167m  35m  6620 R   19  1.7   0:02.29 php-cgi
 4572 www-data  20   0  164m  32m  6624 R   19  1.6   0:01.98 php-cgi
 4508 www-data  20   0  258m 5060 1592 S    3  0.2   0:00.63 apache2
 4556 www-data  20   0  194m 4764 1592 S    1  0.2   0:00.15 apache2
 4591 www-data  20   0  194m 4468 1580 S    1  0.2   0:00.09 apache2
   265 root      20   0     0     0     0 S    0  0.0   0:00.93 jbd2/sda1-8
   857 root      20   0 11360  648  504 S    0  0.0   0:01.76 irqbalance
   903 mysql     20   0  250m  24m  7016 S    0  1.2   0:06.49 mysqld
  4430 francis  20   0 19276 1336  960 R    0  0.1   0:04.20 top
    
```

Figura 31. Procesos obtenidos con el comando *Top* en Ubuntu.

En el sistema Windows no podemos configurar estos parámetros. El módulo utilizado es *mpm\_winnt*, donde sólo se puede configurar el máximo de hilos que un proceso puede crear, por lo que se crearán tantos hilos como peticiones se realicen. El consumo de memoria es mayor en los sistemas Windows que en los sistemas UNIX. Además como se ha podido constatar, los sistemas UNIX consumen un 10% menos de memoria que los sistemas Windows en cuanto a la memoria necesaria para la ejecución del propio sistema.

Respecto a la mejora de consumo de memoria, se puede decir que todos los sistemas han mejorado mediante la configuración de un segundo servidor que sirva

las páginas estáticas. De este modo todas las peticiones que lleven consigo una imagen, un fichero html, etc. serán servidas por un segundo servidor, en este caso por el servidor *lighttpd*. De esta forma todas las peticiones a imágenes, páginas estáticas, páginas de estilo no utilizarán un proceso de apache, sino que serán servidas por el servidor ligero *lighttpd*, cuyos procesos ocupan una mínima parte de la memoria física. Aunque como ya se ha comentado, la utilización del *mpm\_Worker* o *FastCGI* ha contribuido más a la mejora de dicha utilización.

También cabe decir que ante una escalabilidad de los usuarios concurrentes, el sistema Windows tiene peor respuesta frente a los demás, debido probablemente a la utilización de la memoria que realiza dicho sistema frente a los demás.

En conclusión, en este análisis se puede decir que los sistemas UNIX tienen un mayor control de los procesos, en cuanto que Windows concentra todas las peticiones en un solo proceso. Pero hay que decir, que el uso de *FastCGI* en los sistemas UNIX puede ocasionar problemas a la hora de ejecutar las páginas PHP siempre que éste no disponga de su propio espacio de memoria [43]. Algo que no le ocurre a Windows por ser un sistema multi-hilo.

Se puede concluir por tanto que el sistema operativo Ubuntu es el más idóneo de los tres sistemas evaluados para ser utilizado como servidor de páginas web, utilizando para ello el servidor Apache.

## 8. Conclusiones

El trabajo final de carrera ha consistido en la evaluación del rendimiento de tres sistemas operativos. Los sistemas escogidos son de uso común: Windows 7 Profesional, Ubuntu 10.10 y Mac OS X 10.6, por lo que no se ha buscado qué sistema de todos los existentes en el mercado proporciona mejor rendimiento.

Dicho trabajo ha estado comprendido en varias fases, las cuales, estando relacionadas unas con otras, comportaban distintos retos a conseguir. En una primera fase teórica, he podido profundizar en cómo gestionan los procesos y la memoria los sistemas que utilizamos a diario, y comprender por qué unos sistemas realizan la misma labor de forma diferente. Por ejemplo, algunos sistemas Windows no reconocen 4 GB de memoria RAM, y es que al ser sistemas de 32 bits, su espacio de direcciones está limitado a 4 GB. Otro ejemplo es porqué en Windows aparece sólo

un proceso en ejecución de Apache cuando tiene varias solicitudes simultáneas, mientras que en UNIX aparecen varios procesos siendo la misma situación.

El resto del trabajo ha sido mayoritariamente práctico, lo que contribuye a que el mismo haya sido más agradable de realizar, pero no exento de problemas y búsqueda de información.

En esta parte práctica se ha tenido que ir salvando distintos obstáculos, como la instalación de los sistemas en máquinas virtuales y la configuración del servidor Apache en los mismos. Ambas tareas no eran nuevas para mí, excepto el caso del sistema Mac OS X, el cual nunca lo había instalado en una máquina virtual, y mucho menos utilizado como servidor web.

Unos de los primeros problemas encontrados ha sido la de intentar utilizar como Hypervisor la versión VMWare ESXi [15]. Este Hypervisor, tanto por ser gratuito como por las prestaciones que suele proporcionar, era la primera elección, pero debido a la incompatibilidad con el hardware disponible para el trabajo no ha sido posible su instalación, por lo que se ha tenido que utilizar la versión VMWare Workstation [16]. Cabe decir que podríamos haber utilizado cualquier otro Hypervisor existente en el mercado.

Dentro de los retos a los que he tenido que enfrentarme en este trabajo he de destacar dos de ellos por el tiempo que me ha supuesto a la hora de conseguirlos, y por ende por los problemas que me han ocasionado. Se trata de lograr una configuración de Apache que obtuviera el máximo rendimiento del mismo mediante la separación de las peticiones estáticas de las dinámicas y la elección del módulo MPM como módulo de multiprocesamiento. Han sido las dos metas logradas con mayor esfuerzo.

Considero que el objetivo del trabajo ha sido cumplido, en tanto que se ha podido comprobar cuál de los tres sistemas obtiene mejor rendimiento ante un mismo problema.

A nivel personal, me llevo la satisfacción de haber realizado un trabajo que está relacionado con mi actividad profesional actual. Me refiero sobre todo a la configuración de Apache como servidor web y a las configuraciones que se pueden realizar en el mismo para obtener el máximo rendimiento. Ya que hasta la fecha los conocimientos sobre el tema estaban lejos de lo aprendido con el trabajo.

## 9. Bibliografía y enlaces

Sistemas Operativos:

- [1] **Tanenbaum Andrew S.** (2003): *Sistemas Operativos Modernos*. (2ª Edición) México: Prentice Hall
- [2] **Héctor Jairo Ortiz Pabón** (2005): *Sistemas Operativos Modernos*. (1ª Edición) Medellín-Colombia: Sello
- [3] **Tanenbaum Andrew S.** (1996): *Sistemas Operativos Distribuidos*. (1ª Edición) México: Prentice Hall
- [4] **Irene Rodil Jiménez y Camino Pardo de Vega** (2010): *Operaciones auxiliares con tecnologías de la información y la comunicación*. (1ª Edición) Madrid: Paraninfo
- [5] API Windows: [http://msdn.microsoft.com/en-us/library/aa383749\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/aa383749(v=vs.85).aspx)
- [6] Sistema Operativo Ubuntu: <http://www.ubuntu.com/>
- [7] Sistema Operativo Mac OS X: <http://developer.apple.com/library/mac/navigation/>
- [8] Sistema Operativo Microsoft: <http://www.microsoft.com/es/es/default.aspx>
- [9] Sistema UNIX: <http://www.unix.org/>
- [10] Tienda de Microsoft: <http://emea.microsoftstore.com/es/es-ES>
- [11] Servidor Web Apache: <http://www.apache.org/>

Máquinas Virtuales:

- [12] [http://es.wikipedia.org/wiki/M%C3%A1quina\\_virtual](http://es.wikipedia.org/wiki/M%C3%A1quina_virtual)
- [13] VMWare: <http://www.vmware.com/es/>
- [14] VMWare Server: <http://www.vmware.com/products/server/overview.html>
- [15] VMWare ESXi: <http://www.vmware.com/products/vsphere-hypervisor/overview.html>
- [16] VMWare WorkStation: <http://www.vmware.com/products/workstation/>
- [17] Máquina Virtual XEN: <http://www.xen.org/>
- [18] Máquina Virtual, Virtual Box: <http://www.virtualbox.org/>
- [19] Máquina Virtual Virtual-PC: <http://www.microsoft.com/windows/virtual-pc/>
- [20] Servidor Hyper-V: <http://www.microsoft.com/hyper-v-server/en/us/default.aspx>
- [21] Sun: <http://www.oracle.com/us/sun/index.html>
- [22] Máquina Virtual Java (JVM): <http://www.java.com/es/about/>

[23] Framework de Microsoft .NET: <http://www.microsoft.com/net/>

### Benchmarking:

[24] ¿Qué es el benchmarking? <http://www.elblogsalmon.com/conceptos-de-economia/que-es-el-benchmarking>

[25] Software de test *JMeter*: <http://jakarta.apache.org/jmeter/>

[26] Software de test Apache Benchmark (ab): <http://httpd.apache.org>

[27] Descarga de Mysql : <http://dev.mysql.com/downloads/mysql/>

### Sistemas Virtuales:

[28] Virtuozzo: <http://www.parallels.com/es/products/pvc46/>

[29] Linux-VServer: [http://linux-vserver.org/Welcome\\_to\\_Linux-VServer.org](http://linux-vserver.org/Welcome_to_Linux-VServer.org)

[30] Solaris Containers: <http://www.oracle.com/technetwork/systems/containers/index.html>

[31] OpenVZ: [http://wiki.openvz.org/Main\\_Page](http://wiki.openvz.org/Main_Page)

[32] FreeBSD Jails: <http://www.freebsd.org/>

[33] Ranking servidores web <http://news.netcraft.com/>

### Mejora del rendimiento de Apache:

[34] *Performance Tuning Apache*: <http://httpd.apache.org/docs/2.0/misc/perf-tuning.html>

[35] Máximo rendimiento de Apache:

[http://www.howtoforge.com/configuring\\_apache\\_for\\_maximum\\_performance](http://www.howtoforge.com/configuring_apache_for_maximum_performance)

[36] **Rob Flickenger** (2003): *Linux Server Hacks*. (1ª Edición) EE.UU.: O'Reilly Media

[37] Módulo *Prefork* de Apache: <http://httpd.apache.org/docs/2.0/mod/prefork.html>

[38] Módulo *Worker* de Apache: <http://httpd.apache.org/docs/2.0/mod/worker.html>

[39] Apache con fcgi: <http://2bits.com/>

[40] Directrices MPM específicas: [http://docs.redhat.com/docs/es-ES/Red\\_Hat\\_Enterprise\\_Linux/5/html/Deployment\\_Guide/s2-apache-mpm-containers.html](http://docs.redhat.com/docs/es-ES/Red_Hat_Enterprise_Linux/5/html/Deployment_Guide/s2-apache-mpm-containers.html)

[41] **Mohammed J. Kabir** (2002): *La Biblia de Servidor Apache 2*. Madrid: Anaya

[42] Linux versus Mac OS X en Intel Core <http://sekhon.berkeley.edu/macosx/>

[43] Instalación de PHP <http://php.net/manual/es/faq.installation.php>