ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

"COMPARATIVA DE PRESTACIONES DE SERVIDORES VIRTUALIZADOS"

Adrián Barbáchano Cirión

Daniel Morató Osés

Pamplona, 16 de Septiembre de 2010

ÍNDICE DE CONTENIDO:

RESUMEN....4
1. INTRODUCCIÓN....5
1.1 ¿Qué es la virtualización?....5
1.2 ¿Por qué virtualizar?....5
1.3 Formas de virtualización....7
1.4 Tipos de virtualización....8
1.5 Soluciones de virtualización....8
1.5.1 VMware....8
1.5.1.1 ¿Qué es una infraestructura virtual?....10
1.5.1.2 Sistemas operativos soportados....11
1.5.2 QEMU....11
1.5.3 Parallels....12
1.5.4 Virtual PC....14
1.5.5 VirtualBox....15
1.5.5.1 Funcionamiento de la máquina virtual de VirtualBox....16
1.5.5.2 Técnicas avanzadas: escaneo de código, análisis y parcheo....18
1.5.5.3 Características importantes....18
1.5.5.4 Sistemas operativos soportados....20
1.5.6 User Mode Linux....21
1.5.7 KVM....22
1.5.8 XEN....23
1.6 Comparativa....24

2. ESCENARIO....25
2.1 Qué Vm se usarán....25
2.2 Software y herramientas....25
2.2.1 Apache....25
2.2.2 JMeter....28
2.2.3 PHP....29

2.2.4 BASH....29
2.2.5 AWK....30
2.2.6 Sed....30
2.2.7 Gnuplot....30
2.2.8 Tac....30

2.3 Problemática....30
2.3.1 Conexiones persistentes....30
2.3.2 Caché....31
2.3.3 Aleatoriedad....33

3. PRUEBAS, RESULTADOS Y CONCLUSIONES....35
3.1 Prueba 1....35
3.2 Prueba 2....38

4. CONCLUSIONES....43

5. BIBLIOGRAFÍA....44

6. ANEXO I - Intel VT-x ("Vanderpool")....45

7. ANEXO II – Scripts....46
7.1 sacanumerosazar.php y limpianumerosazar.bsh....46
7.2 graficatamaños.bsh y graficatamaños.dem....46
7.3 sacamedia.bsh....47
7.4 escribe_nombres_ficheros.bsh....47
7.5 crea_archivos_numeros_azar.bsh....47
7.6 borramemoriacompartida.php....47
7.7 contador_semaforo.php....47
7.8 jmeter.php, borraprincipioyfinal.bsh, ordena.bsh, basegnuplot....47
7.9 50megasv1.php y 50megasv2.php....48

ÍNDICE DE FIGURAS:

- Figura 1: Hipervisor nativo....8
- Figura 2: Hipervisor hosted....8
- Figura 3: Hipervisor (hosted) de VMware Workstation....9
- Figura 4: Infraestructura virtual....10
- Figura 5: Niveles de protección....17
- Figura 6: VirtualBox con SO host Linux y SO invitado Windows....19
- Figura 7: Estructura de los procesos en Linux....22
- Figura 8: Estructura de los procesos con UML....22
- Figura 9: Prueba de caché....32
- Figura 10: Fórmula de la distribución de Pareto....34
- Figura 11: Incógnita x despejada....34
- Figura 12: 30 hilos y ListenBacklog por defecto....36
- Figura 13: Peticiones/min en función de la cola y de las peticiones simultáneas....37
- Figura 14: Peticiones/min en función del tiempo medio en VMware....38
- Figura 15: Peticiones/min en función del tiempo medio en VirtualBox....39
- Figura 16: VMware y VirtualBox con 20 posiciones de cola....39
- Figura 17: Vmware y VirtualBox con 40 posiciones de cola....40
- Figura 18: Errores obtenidos con VMware....40
- Figura 19: Errores obtenidos con VirtualBox....41
- Figura 20: Porcentaje de errores con ListenBacklog 1....41
- Figura 21: Porcentaje de errores con ListenBacklog 10....42
- Figura 22: Tamaño de los 15.000 primeros números de la semilla....46

RESUMEN

El término virtualización ya existe desde hace años pero su necesidad y aplicación es cada día mayor. Existen muchas diferentes soluciones de virtualización y muchas son las diferencias existentes entre todas ellas: el precio, forma de virtualizar, tipo de virtualización, compatibilidades, etc... Todo esta información se puede obtener visitando la respectiva página web oficial de cada producto, lo que no se puede conocer es el rendimiento que ofrece cada uno de ellos y eso sería muy importante para saber cuál elegir. En este proyecto se pretende comparar varias máquinas virtuales. Para ello se realizarán peticiones web a través de un software llamado JMeter con el que guardaremos todos los datos de cada petición. Las peticiones web se realizarán a un servidor Web Apache, el cual se irá modificando la configuración respecto al número máximo de peticiones que puede atender a la vez y el número de peticiones máximo que puede tener en cola.

Al realizar dichas peticiones web, se contará el tiempo medio que tarda cada petición, el número de peticiones por minuto, cuántos errores ha habido, etc.. Para ello habrá un script PHP que se encargue de leer los datos y calcular los datos.

El estudio que se va a realizar, es el de una gran población pidiendo una página web, prácticamente en el mismo momento de manera que para el servidor sea una carga no muy ligera y poder comprobar que solución de virtualización de las que se prueban es más eficiente.

Hay ciertos puntos que tratar para que la simulación sea lo más realista posible:

- Para suponer una gran población, en el JMeter se especifican el número de hilos que se van a usar y el número de peticiones que hace cada hilo.
- No se permitirán conexiones persistentes pues cada petición será considerada que proviene de un usuario diferente y debe establecerse otra conexión.
- Los ordenadores con los que se hacen las pruebas deben estar con el mínimo de programas abiertos y siempre en la misma situación.
- No pedir siempre el mismo fichero ni que sean todos los ficheros de un tamaño en concreto. Debe haber una cierta aleatoriedad para que se asemeje lo más posible a un escenario real.
- Es imprescindible evitar la memoria caché ya que el tiempo de algunas peticiones se reduciría.
- Las máquinas virtuales deben estar configuradas exactamente igual y deben contener los mismos ficheros.

Tras realizar diferentes pruebas se comparan los resultados de rendimiento y errores para sacar conclusiones acerca de que software se comporta mejor.

1.INTRODUCCIÓN

1.1 ¿Qué es la virtualización?

Tal y como indica la palabra, se va a tratar algo virtual, es decir, no físico. Lo que se va a virtualizar es un PC, a lo cual se le llamará máquina virtual. Una máquina virtual es un sistema operativo (SO) invitado que funciona sobre un sistema operativo anfitrión. Por lo tanto una máquina virtual permite tener varios ordenadores virtuales ejecutándose sobre el mismo ordenador físico.

La virtualización se ha convertido en un tema de actualidad dentro del mundo IT debido a su potencial para provocar un cambio fundamental en la forma en que se consideran y administran los recursos informáticos.

Un sistema virtual por software es una aplicación que simula un sistema físico (un PC, un Server) con unas características de hardware determinadas. Un software de virtualización proporciona un ambiente de ejecución similar a todos los efectos a un ordenador físico (excepto en el puro acceso físico al hardware simulado), con CPU, BIOS, tarjeta de vídeo, memoria RAM, tarjeta de red, conexión USB, disco duro, etc.

Las máquinas virtuales no funcionan tan rápido como si el SO fuera instalado de forma tradicional. El rendimiento del sistema virtual varía dependiendo de las características del sistema físico en el que se ejecute, y de los recursos virtuales (CPU, RAM, etc.) asignados al sistema virtual.

El término de virtualización empezó a usarse en los '60 cuando se creó el sistema experimental IBM M44/44X. La creación y administración de máquinas virtuales ha sido llamado virtualización de plataforma o virtualización de servidor más recientemente. Se empezó a virtualizar principalmente porque si un equipo estaba ocupado, otro usuario no podía usarlo. Para finales de los '80, las computadoras eran considerablemente más potentes, numerosas y baratas que en los '60, y la necesidad de multiplexar un equipo para varios usuarios se había eliminado mediante el uso de OS multitarea y multiusuario que son los que se conocen a día de hoy. En otras palabras: la razón original para virtualizar equipos completos había desaparecido. Sin embargo eso no significa que ahí terminará la virtualización, hubo un parón en la investigación de máquinas virtuales pero en los '90 se volvió a retomar la idea y año tras año se ha ido poniendo más en práctica. En estos años han aparecido nuevos software de virtualización importantes tales como VMware (1998), VirtualBox (2007), etc. Debido a la demanda existente por todo tipo de usuarios, han ido sacando nuevas versiones y actualmente siguen trabajando en ello para mejorar el rendimiento, corregir fallos y ser compatible con los nuevos SO.

1.2 ¿Por qué virtualizar?

Son muchas las razones por las que es beneficioso virtualizar. Tanto para un usuario final como para empresas, para todos hay ventajas:

1. Se puede hacer funcionar varios sistemas operativos a la vez en un mismo ordenador. De esta manera no hace falta reiniciar para entrar a un SO distinto si se desea usar una aplicación que no está disponible en el SO en el que se está trabajando.

2. En un mismo ordenador es posible usar SO que sin virtualizar no se podría. Es posible generar una máquina virtual con el hardware que se necesite aunque no se disponga de él físicamente, por lo tanto, se puede virtualizar un SO que requiera un hardware específico mientras que sería imposible instalarlo. Por ejemplo se podría virtualizar un SO de 64 bits teniendo en la máquina física un procesador de 32 bits.
3. Reducción de costes: virtualizando se reduce el hardware y por lo tanto se reducen los costes de compras de equipos y/o piezas informáticas, además el ahorro energético que supone tener menos ordenadores encendidos. Según datos extraídos de la página www.vmware.com, en total se calcula que en una empresa mediana o grande se puede llegar a ahorrar un 50% en hardware y un 80% en energía eléctrica y por cada servidor virtualizado se ahorran unos 3000$ anuales (unos 2500€). El ahorro de tener más ordenadores también tiene otra ventaja que empresas y particulares buscan, que es el ahorro de espacio físico. Se ahorra un espacio de entre un 50% y un 80%.

4. Reducir el tiempo de inactividad (desaprovechamiento de los equipos) y mejorar la fiabilidad con la continuidad del negocio y recuperación de datos ante desastres.

5. Por lo general, los servidores de las empresas, aún siendo empresas grandes, están muy inutilizados ya que tienen varios servidores potentes preparados para horas puntas pero el resto del tiempo tienen poca carga de trabajo a no ser que estén virtualizados y así ser más útiles.

   Actualmente, en muchos centros de datos sin virtualización, las cargas de trabajo de un servidor están lejos de consumir sus recursos resultando un desperdicio de infraestructura e inversión. La virtualización permite combinar varios sistemas operativos y sus cargas de trabajo en un único servidor compartido con suficientes recursos para cubrir la demanda de recursos. El resultado es un incremento en la utilización. Un falso mito es que los servidores no deberían ser forzados a trabajar a su máxima capacidad. Lo que sí es cierto es que los servidores tienen que poder cubrir las cargas de trabajo, es decir, si se dispone de un servidor cubriendo perfectamente todas las cargas de trabajo y rindiendo a su máxima capacidad (mejor 3 servidores funcionando a la máxima capacidad que 4 ó 5 trabajando suavemente) se estará amortizando la inversión en infraestructura. Con la correcta planificación y conocimiento de las cargas de trabajo, la virtualización permite incrementar la utilización de servidores mientras se decrementa el número de servidores físicos necesarios. Lo difícil es saber qué cargas de trabajo se van a tener aunque solo sean puntualmente y no quedarse cortos a la hora de adquirir servidores ni que sobren.

6. Servidores rápidos de provisión/reorientación

7. Fácil creación y restauración de una copia de seguridad. El disco duro de la máquina virtual es un archivo por lo que la copia de seguridad es simple de realizar, copiar dicho archivo a una unidad externa para mantener la copia segura. La restauración al contrario que un PC en el que hay que formatear el disco duro e instalar el SO, los drivers y los programas, con la máquina virtual solo se necesita el archivo y el programa manejador de la maquina virtual. El tiempo de restauración es bastante más rápido que con un sistema tradicional.

8. Sirven para usarlas como entornos separados para realizar pruebas con programas que se intuyen que pueden ir mal o no se está seguro de su funcionamiento, etc. Para empresas que se dedican al desarrollo software, la virtualización es indispensable.

9. Se reducen el número de equipos y así se reduce la complejidad global de la infraestructura de organizaciones y empresas por lo que resulta más fácil entender la red. De esta manera en caso de existir algún fallo puntual en la red es más sencillo detectar dónde puede estar el error.
10. Se puede modificar el hardware del host sin problema, no hace falta instalar drivers ni modificar nada a las máquinas virtuales para que puedan funcionar.

Sin duda las ventajas se pueden resumir en ahorro y fiabilidad.

Desventajas:

1. Dado que la virtualización es una capa intermedia entre el nivel físico y el sistema operativo que funciona en el hardware emulado, la velocidad de ejecución de este último es menor que sin virtualización, pero en la mayoría de los casos suficiente para usarse en entornos de producción.

2. Adquirir el software adecuado para virtualizar, que casi siempre, tiene un coste, y algunas empresas en la que los empleados no tienen amplios conocimientos de informática, habría que contratar profesionales para mantener el software. En definitiva, se ahorra en máquinas, pero se obtendrá un coste adicional en licencias y posiblemente en mantenimiento. Pero en todo caso, seguiría siendo un coste menor que comprar un nuevo ordenador o servidor y mantenerlo.

3. Pueden existir picos de carga, es decir, que en determinados momentos, el servidor no pueda con sus entornos virtuales por causa de algún proceso. Si bien es posible configurar qué pueden usar cada uno de los entornos virtuales, al final serán menos recursos que si estuviera en una máquina para cada uno de ellos. En definitiva, hay que estar atento al dimensionamiento si no se quieren problemas en este sentido.

4. Algunos fabricantes de software consideran cada entorno virtual como una máquina diferente, y por tanto, exigen que se compre una licencia para cada servidor, aunque sea virtual.

1.3 Formas de virtualización

Un VMM (Virtual Machine Monitor) o hipervisor es un software que permite que múltiples OS corran de manera simultánea en un mismo equipo.

Hay dos tipos de hipervisor:

1. Nativo: el hipervisor funciona directamente sobre el hardware físico para ofrecer todas las funcionalidades del mismo a las máquinas virtuales que corren sobre él.
2. Hosted: el hipervisor se ejecuta sobre un SO que corre sobre el hardware físico del ordenador. El hipervisor captura las llamadas a hardware de la máquina virtual y se las pasa al SO sobre el que corre y éste a su vez las pasa al hardware físico subyacente.

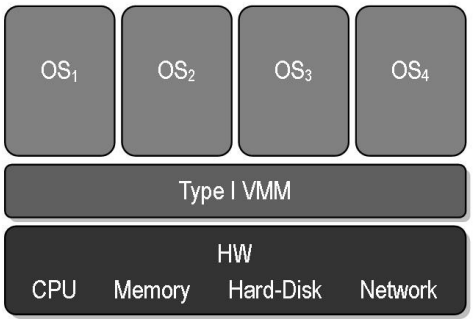

Figura 1: Hipervisor nativo
Fuente: wikipedia.es

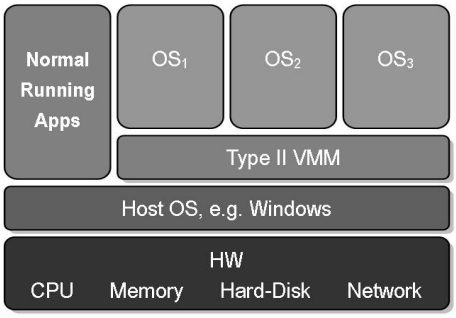

Figura 2:Hipervisor hosted
Fuente: wikipedia.es

En la figura 1 aparece el esquema de cómo sería un hipervisor nativo y en la figura 2 el esquema del hosted.

1.4 Tipos de virtualización

1. Virtualización por Hardware: es un enfoque de virtualización de plataforma que permite virtualización completa usando la ayuda de las capacidades del hardware, principalmente de los procesadores del host. La virtualización asistida por hardware también se conoce como virtualización acelerada. Por ejemplo VirtualBox en ocasiones especiales usar este tipo de virtualización.
2. Virtualización completa: la máquina virtual simula un hardware suficiente para permitir un sistema operativo invitado sin modificar (uno diseñado para la misma CPU, es decir, usa el mismo conjunto de instrucciones que la máquina anfitrión) para funcionar de forma aislada. Típicamente, muchas instancias pueden correr al mismo tiempo. Ejemplo: VMware Workstation y VirtualBox.
3. Virtualización parcial: donde algunos, pero no todos de los recursos son simulados. Actualmente no se usa, se utilizaba en el IBM M44/44X y en otros sistemas antiguos.
4. Paravirtualización: entorno de virtualización que requiere modificaciones de los SO invitados a cambio de mayor eficiencia. Un ejemplo sería Xen.

1.5 Soluciones de virtualización

Obviamente el mundo de la virtualización no es sólo cosa de uno, existen varios productos software, algunos de pago y otros no, entre los más conocidos están los siguientes:

1.5.1 VMware

VMware es una empresa dedicada a la virtualización por software. Tiene varios productos diferentes de forma que satisfacen las necesidades de los distintos tipos de clientes/usuarios. Algunos conocidos son VMware Server, VMware Workstation y VMware Player.

VMware Player y Server son gratuitos y Workstation es de pago. El VMware Player reproduce máquinas virtuales realizadas con el VMware Workstation.

El producto VMware Server es parecido al Workstation pero orientado a más demanda de trabajo, puede tener más máquinas funcionando a la vez.

VMware virtualiza una plataforma x86, de forma que la mayor parte de las instrucciones en VMware se ejecutan directamente sobre el hardware físico.

Los productos de VMware usan la CPU para ejecutar código directamente siempre que sea posible. Cuando no se pueda operar la ejecución directa como con el kernel-level y el código en modo real, VMware reescribe el código dinámicamente lo que se llama "traducción binaria" o BT. El código traducido se almacena en la memoria, por lo general al final del espacio de direcciones, cuyos mecanismos de segmentación pueden proteger y hacer invisible. Por estas razones VMware opera más rápido que los emuladores, funcionando, según la página oficial de VMware (www.vmware.com), a más del 80% de velocidad que si el sistema virtualizado funcionara directamente sobre el mismo hardware físico en el cual funciona el sistema operativo anfitrión.

El enfoque de VMware evita algunas de las dificultades de la virtualización en las plataformas basadas en x86. Las máquinas virtuales de VMware pueden hacer frente a las instrucciones mediante la sustitución de ellas, o simplemente ejecutando el código del kernel en modo de usuario. Sustituyendo las instrucciones se corre el riesgo de que el código pueda fallar al intentar encontrar el contenido esperado si se lee a sí mismo; uno no puede proteger el código contra la lectura mientras que permite la ejecución normal, y esto hace que sea más complicado el reemplazar.

VMware inserta directamente una capa de software en el hardware del ordenador o en el sistema operativo host. Esta capa de software crea máquinas virtuales y contiene un monitor de máquina virtual o "hipervisor" que asigna recursos de hardware de forma dinámica y transparente, para poder ejecutar varios sistemas operativos de forma simultánea en un único ordenador físico sin ni siquiera darse cuenta de ello. VMware Workstation y VMware Server contienen hipervisor hosted, al contrario que VMware ESX que contiene hipervisor de tipo nativo.

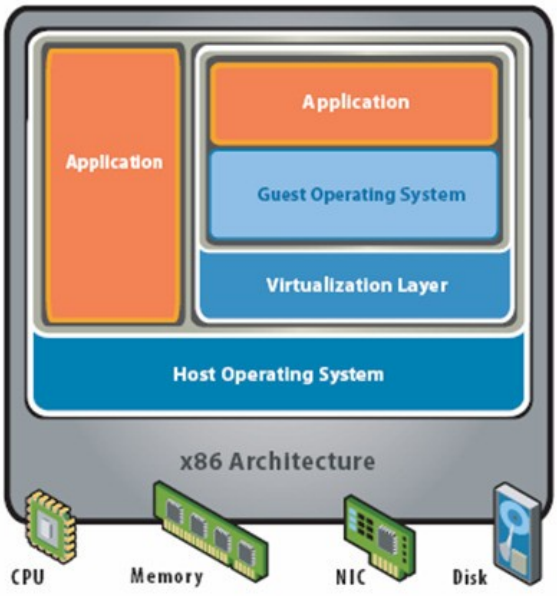

Figura 3: Hipervisor (hosted) de VMware Workstation
Fuente: www.vmware.com

En esta imagen (Figura 3) se puede ver que la capa de software de la que se hablaba se llama capa de virtualización (Virtualization Layer). Se observa que sobre un hardware de arquitectura x86 se ejecuta un sistema operativo que será el sistema operativo anfitrión. Sobre este sistema operativo se ejecutan aplicaciones y una aplicación puede ser una máquina virtual que a su vez está compuesta por la capa de virtualización (la misma para todas las máquinas virtuales del mismo host), el SO invitado y las aplicaciones que corren en ese host invitado. Es un ejemplo claro de hipervisor de tipo hosted.

Empresas de todo el mundo y de todos los tamaños usan soluciones de VMware ya que tiene productos que se adaptan a la medida de todo tipo de empresas, ya sea para crear servidores, infraestructuras virtuales, etc..

No obstante, la virtualización de un ordenador físico único es sólo un concepto básico. VMware ofrece una sólida plataforma de virtualización que puede ampliarse por cientos de dispositivos de almacenamiento y ordenadores físicos interconectados para formar una *infraestructura virtual* completa.

1.5.1.1 ¿Qué es una infraestructura virtual?

Básicamente, una infraestructura virtual consiste en el mapping dinámico de recursos físicos en función de las necesidades de la empresa. Una máquina virtual representa los recursos físicos de un único ordenador, mientras que una infraestructura virtual representa los recursos físicos de la totalidad del entorno de IT, aglutinando ordenadores x86, así como su red y almacenamiento asociados, en un pool unificado de recursos de IT.

Estructuralmente, una infraestructura virtual consta de los siguientes componentes:

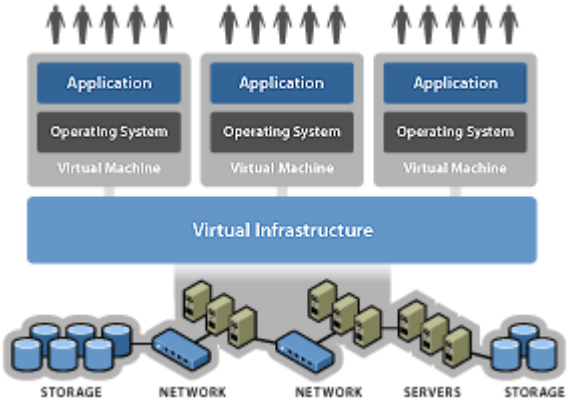

Figura 4: Infraestructura virtual
Fuente:www.vmware.com

- Hipervisor de un solo nodo para hacer posible la virtualización de todos los ordenadores x86.
- Un conjunto de servicios de infraestructura de sistemas distribuida basada en la virtualización, como gestión de recursos, para optimizar los recursos disponibles entre las máquinas virtuales.

Mediante la separación de la totalidad del entorno de software de su infraestructura de hardware subyacente, la virtualización hace posible la reunión de varios servidores, estructuras de almacenamiento y redes en pools de recursos compartidos que se pueden asignar de forma dinámica, segura y fiable a las aplicaciones según sea necesario.

Este enfoque permite a las organizaciones crear una infraestructura informática con altos niveles de utilización, disponibilidad, automatización y flexibilidad utilizando componentes básicos de servidores económicos y estándar del sector.

Es posible eliminar prácticamente las paradas planificadas, usando la tecnología VMotion, que permite trasladar las máquinas virtuales mientras funcionan de una máquina física a otra. De esta manera se puede liberar a una máquina física para trabajos de mantenimiento.

1.5.1.2 Sistemas operativos soportados

Soporta los SO como invitados:
Windows, Linux, Solaris, FreeBSD, Virtual appliances, Netware, OS/2, SCO, BeOS y Darwin en todas sus versiones. No virtualiza Mac OS oficialmente pero sí que circulan por Internet discos de instalación de Mac OS parcheados para instalar natívamente o en VMware. Lo único que hace falta es un procesador compatible con las instrucciones de tipo SSE2 y SSE3.

1.5.2 QEMU

QEMU fue escrito por Fabrice Bellard y es software libre. Es un emulador de procesadores basado en la traducción dinámica de binarios (conversión del código binario de la arquitectura fuente en código entendible por la arquitectura huésped). QEMU también tiene capacidades de virtualización dentro de un sistema operativo, ya sea Linux, Windows, o cualquiera de los sistemas operativos admitidos, (de hecho es la forma más común de uso). Esta máquina virtual puede ejecutarse en cualquier tipo de microprocesador o arquitectura (x86, x86-64, PowerPC, etc.). Puede arrancar varios sistemas operativos, incluyendo entre otros Linux, Microsoft Windows, DOS, y BSD. La mayoría del programa está bajo licencia LGPL y el modo de emulación tiene licencia GPL de GNU.

El objetivo principal es emular un sistema operativo dentro de otro sin tener que reparticionar el disco duro, empleando para su ubicación cualquier directorio dentro de éste.

El programa no dispone de GUI, pero existe otro programa llamado QEMU manager que permite tener interfaz gráfica si se utiliza QEMU desde Windows. También existe una versión para Linux llamado qemu-launcher.

La extensión del archivo de imagen de disco duro es .dsk.

QEMU posee dos modos de operación:

1. Emulación del modo usuario:

Puede ejecutar procesos compilados para un tipo de CPU en otro tipo de CPU. Wine y DOSEMU son los principales objetivos de QEMU.

2. Modo de emulación completo de sistema de ordenador :

Características de QEMU:

- Soporta emulación de x86, AMD64 y PowerPC principalmente.
- Alta velocidad - algunas aplicaciones pueden correr a una velocidad cercana al tiempo real.
- Implementa el formato de imagen de disco Copy-On-Write. Se puede declarar una unidad virtual multi-gigabyte, la imagen de disco ocupará solamente el espacio actualmente utilizado. Se almacena en un formato especial, qcow2.

- Implementa la superposición de imágenes. Se puede mantener una instantánea del sistema invitado, y escribir cambios en un archivo de imagen separado. Si el sistema invitado se colapsa, es sencillo volver a la instantánea del sistema huésped.
- Soporte para ejecutar binarios de Linux en otras arquitecturas.
- Es posible salvar y restaurar el estado de la máquina (por ejemplo programas en ejecución.)
- Emulación de tarjetas de red virtuales.
- El Sistema Operativo invitado no necesita ser modificado o parcheado.
- Mejoras en el rendimiento cuando se usa el módulo del kernel KQEMU
- Las utilidades de línea de comandos permiten un control total de QEMU sin tener que ejecutar X11 (Entorno gráfico).
- Control remoto de la máquina emulada a través del servidor VNC integrado.

Inconvenientes:

- Soporte incompleto para las arquitecturas utilizadas menos frecuentemente.
- Soporte incompleto de controladores (tarjetas de vídeo, sonido, E/S) para los SO invtados, por lo tanto se tiene una sobrecarga importante en aplicaciones multimedia.
- No compila con versiones recientes de GCC (no soporta las versiones 4.x)

QEMU está integrado en varias soluciones de virtualización como Xen-HVM, KVM, etc..sin embargo la más importante es VirtualBox: En enero de 2007 se lanzó la primera versión de VirtualBox. Usaba algunos de los dispositivos hardware virtuales de QEMU y tiene una recompilación dinámica que está basada en QEMU. Como VirtualBox, QEMU ejecuta casi todo el código del SO invitado nativamente en el host a través del VMM (Virtual Machine Manager).

User Mode Linux (UML) fue la primera solución de virtualización antes que QEMU en lanzar un kernel de Linux como un proceso sin necesitar parches del kernel del anfitrión.Sin embargo, UML requiere pesados parches de kernel mientras QEMU acepta kernels de Linux sin parchear. El precio a pagar es que QEMU es más lento.

1.5.3 Parallels

Parallels es otro software de virtualización de pago al igual que VMware. No obstante a través de la web se puede obtener una versión de prueba con un tiempo limitado de uso.

Como otros productos software de virtualización, Parallels usa la tecnología hipervisor, la cual es una pequeña capa entre el SO anfitrión y el ordenador. El hipervisor controla directamente algunos de los recursos de el ordenador y proporciona un interfaz a los monitores de máquinas virtuales y al SO anfitrión. El hipervisor de Parallels además soporta tecnología de virtualización hardware como Intel VT-x y AMD-V.

Se puede instalar tanto en Windows como Linux y Mac OS.

Como SO huésped permite a los siguientes:

- Microsoft Windows
- Linux
- Mac OS
- FreeBSD
- OS/2 and eComStation™
- Sun Solaris
- MS-DOS

El motor de virtualización de Parallels habilita cada máquina virtual para operar de forma similar a un ordenador físico. Cada máquina virtual tiene su propio conjunto de hardware: procesador, RAM, disquetera, lectores de CD y DVD, dispositivos de Entrada/Salida y disco duro.

Se le pueden añadir funcionalidades como la herramienta de sincronización del portapapeles para copiar un texto en una máquina virtual y pegarlo en otra o en el SO anfitrión. Otra funcionalidad es la sincronización del ratón. Cuando el usuario pasa el ratón por la pantalla lo captura y cuando lo saca lo libera.

La configuración del hardware para cada máquina virtual está definida en un archivo de configuración especial con la extensión .pvs. Éste contiene toda la información sobre los dispositivos virtuales usados por la máquina virtual y los archivos conectados a esos dispositivos. Se genera cuando se crea la máquina virtual con el asistente de nueva máquina virtual.

Una máquina virtual de Parallels tiene como mínimo 2 archivos:

1. Un archivo de configuración
2. Una archivo de imagen de disco duro

Generalmente suele haber más: un fichero para cada disco duro virtual adicional y achivos de salida para puertos virtuales. Como un caso excepcional, una máquina virtual puede tener solo un archivo, el de configuración, es decir, sin imagen de disco duro. Esto se usa para arrancar la máquina virtual de un Live CD. Un Live CD es un sistema operativo ya instalado en un CD que se carga en RAM y no hace falta grabar nada en disco duro.

El disco duro virtual puede ser uno de los dos siguientes formatos:

- Plano: Un disco plano tiene el mismo tamaño desde el momento en el que es creado. EL SO invitado opera un poco más rápido en un disco plano. Un disco plano puede ser convertido a uno expandible con la ayuda de herramienta de imagen de Parallels (Parallels Image Tool).

- Expandible: Un disco expandible es pequeño al principio y crece según vas añadiendo aplicaciones y datos a la máquina virtual. El tamaño del disco especificado cuando el disco es creado es el tamaño máximo al que el disco puede crecer. Usando discos expandibles ahorra espacio en el disco duro del ordenador. Al igual que los discos planos, los expandibles también se puede cambiar a planos con la misma herramienta (Parallels Image Tool).

1.5.4 Virtual PC

Es un programa desarrollado por Connectix y comprado por Microsoft para crear ordenadores virtuales. Es decir, su función es emular un hardware sobre el que funcionen varios sistemas operativos. Con esto se puede conseguir ejecutar varios sistemas operativos en la misma máquina a la vez y hacer que se comuniquen entre ellos. La versión 2007 se encuentra disponible de forma gratuita.

Virtual PC virtualiza un PC estándar y su hardware asociado. Puede correr SO Windows y algunos SO como Linux pueden funcionar pero no están oficialmente confirmados.

Virtual PC tiene las siguientes características:

- Soporte y redirección de USB:conectar periféricos tales como dispositivos flash y cámaras digitales, e imprimir desde el SO invitado a las impresoras del SO anfitrión.
- Aplicación integrada de publicación y puesta en marcha: por ejemplo se ejecutan aplicaciones de Windows XP desde el escritorio de Windows 7, similar a la imagen de la figura 6 que se muestra más adelante (ver pág. 19), en la cual se ejecutan aplicaciones de Windows desde Linux.
- Soporte multihilo:ejecutar múltiples máquinas virtuales al mismo tiempo, cada una en su propio hilo para mejorar la estabilidad y el rendimiento.
- Integración con Windows Explorer: controla todas las máquinas virtuales desde una sola carpeta Explorer (%usuario%\Virtual Machines).

Implementación:

- La versión de Virtual PC de Macintosh usa recompilación dinámica para traducir código x86 usados por los PCs al equivalente código PowerPC para Macs.
- La versión de Virtual PC de Windows usa recompilación dinámica pero solo para traducir código de modo kernel modo real x86 a código modo usuario x86.
- Virtual PC y Virtual Server encapsulan los discos duros virtuales a archivos con formato VHD (Virtual Hard Disk).

Virtual PC, en el caso de la versión para Windows, no emula el procesador sino que deja que él mismo ejecute las instrucciones en el entorno emulado.

Posee los siguientes archivos:

- VHD: VHD (Virtual Hard Disk) es un archivo en el ordenador físico que usa una máquina virtual como disco duro y realiza todas las lecturas y escrituras.
- VMC: Es el archivo de configuración de la máquina virtual donde son almacenadas todas las configuraciones para una máquina virtual.

En algunos procesadores es seguro realizar ejecución directa de las instrucciones pero otros requieren emular:

Modo real	Emulación
Modo Virtual 8086 (v86)	Ejecución directa
Modo protegido ring 3	Ejecución directa (con algunas excepciones)
Modo protegido ring 0	Emulación a menos que se sepa que es seguro.

El código de ring 0, 1 y 2 del SO invitado son ejecutados en el ring 1. El código de ring 3 es ejecutado en el ring 3.

Tipos de discos virtuales:
1.Disco virtual expandible dinámicamente.

2.Disco virtual fijo.

3.Diferenciación**:** es un disco que se basa en otro existente y lo único que se almacena en el nuevo disco son las cosas que añadas, pero lo que ya hubiera sigue siendo lo mismo.

4.Vinculado a un disco duro: se crea una imagen de un disco duro existente.

1.5.5 VirtualBox

La virtualización es por naturaleza extraordinariamente compleja, especialmente en hardware x86. Por eso aunque VirtualBox sea open-source y por ello se puede modificar el código fuente del mismo, entenderlo requiere, una gran comprensión sobre los detalles de la arquitectura x86, así como un gran conocimiento acerca de la implementación de los sistemas operativos anfitrión e invitados. A continuación se detalla cómo funciona VirtualBox.

Los procesos de Virtualbox:

Cuando se arranca el interfaz gráfico de usuario de Virtualbox (GUI), por lo menos un proceso extra se inicia en el camino - el proceso VboxSVC. Una vez que se arranca la máquina virtual desde la GUI, se tienen dos ventanas: una es la ventana principal y la otra es la de la máquina virtual; pero 3 procesos corriendo. Si se analiza el administrador de tareas del sistema (en Windows) o algún monitor del sistema (en Linux) se puede observar:

- VirtualBox: la GUI para la ventana principal.
- Otro proceso VirtualBox que fue empezado con el parámetro -startvm el cual significa que su proceso GUI actúa como una shell para la máquina virtual.
- VboxSVC, este servicio el cual se ejecuta en segundo plano para hacer un seguimiento de todos los procesos involucrados. Esto es automáticamente ejecutado por el primer proceso GUI.
- (En Linux, hay otro proceso demonio , llamado VboxXPCOMIPCD).

Para el sistema operativo anfitrión, la máquina virtual que corre “dentro” de la segunda ventana le parece como un programa normal. VirtualBox tiene una muy buena conducta en ese aspecto: toma el control sobre una larga parte del ordenador ejecutando un sistema operativo completo con sus propios conjuntos de procesos, drivers y dispositivos dentro de este proceso de máquina virtual, pero el sistema operativo anfitrión no se da mucha cuenta de todo esto. Cualquiera que sea la máquina virtual, será simplemente otro proceso más en el ordenador físico.

Hay dos tipos de encapsulado con los archivos y procesos de VirtualBox:

1. Arquitectura cliente/servidor. Todos los aspectos de VirtualBox y las máquinas virtuales que se están ejecutando se pueden controlar con el API COM / XPCOM. Por ejemplo, hay una utilidad de línea de comandos llamada VBoxManage que permite controlar máquinas virtuales al igual que la interfaz gráfica de usuario (de hecho, muchas de las operaciones más complejas todavía no están soportados por la interfaz gráfica de usuario como por ejemplo el redireccionamiento de puertos que se verá más adelante). Se puede, por ejemplo, iniciar una máquina virtual desde la interfaz gráfica de usuario (haciendo clic en el botón "Inicio") y pararla desde VBoxManage. Por ello, el proceso de servicio (VboxSVC) es necesario: realiza un seguimiento de las máquinas virtuales que están funcionando y en que estado están.
2. Arquitectura frontend/backend: lo más importante de VirtualBox está escondido en una librería compartida, VBOXVMM.dll en Windows ó VBOXVMM.so en Linux. Esto puede ser considerado un backend o caja negra que es estática y es relativamente fácil escribir otro frontend sin tener que alterar los detalles de la virtualización x86.

De hecho, VirtualBox ya viene con unos cuantos frontends:

- El Qt GUI (interfaz gráfico), el más conocido y usado.
- VBoxManage, una utilidad de línea de comandos que permite control todas las potentes características de VirtualBox. Muy importante para hacer configuraciones avanzadas de la máquina virtual.
- Un GUI plano basado en SDL, con menos cosas que el Qt GUI. Es muy útil para uso en empresas como pruebas durante desarrollo de software. Para controlar las máquinas virtuales, se usa entonces VboxManage (la utilidad para manejar máquinas virtuales desde la línea de comandos.
- Un protocolo servidor de escritorio remoto (RDP), el cual es solo consola y muestra gráficos por pantalla, pero permite que ordenadores remotos se conecten a él. Esto es especialmente útil para empresas que quieren consolidar sus PCs clientes en unos pocos servidores. Los PCs clientes están entonces meramente visualizando datos RDP producidos por los varios procesos de servidores RDP en unos grandes servidores, los cuales virtualizan PCs clientes “reales”.

1.5.5.1 Funcionamiento de la máquina virtual de VirtualBox

Como se ha dicho anteriormente, una máquina virtual no es más que un proceso desde la perspectiva del sistema operativo anfitrión. El sistema operativo anfitrión no necesita ajustarse mucho para afianzar la virtualización. A pesar de que hay un controlador ring-0 (modo kernel) que debe ser cargado en el sistema operativo anfitrión para que VirtualBox funcione, este controlador ring-0 no hace gran cosa.

Sólo es necesario para algunas tareas específicas tales como:

- Alojamiento de memoria física para la máquina virtual.
- Guardado y restaurado de los registros de la CPU y tablas de los descriptores cuando ocurre una interrupción del ordenador mientras se está ejecutando código ring-3 del sistema operativo invitado.
- Cuando se cambia de ring-3 del SO anfitrión (tareas con poca prioridad) al contexto del SO invitado.

- Activar o desactivar VT-x (VT-x representa la tecnología de Intel para la virtualización hardware en la plataforma x86).

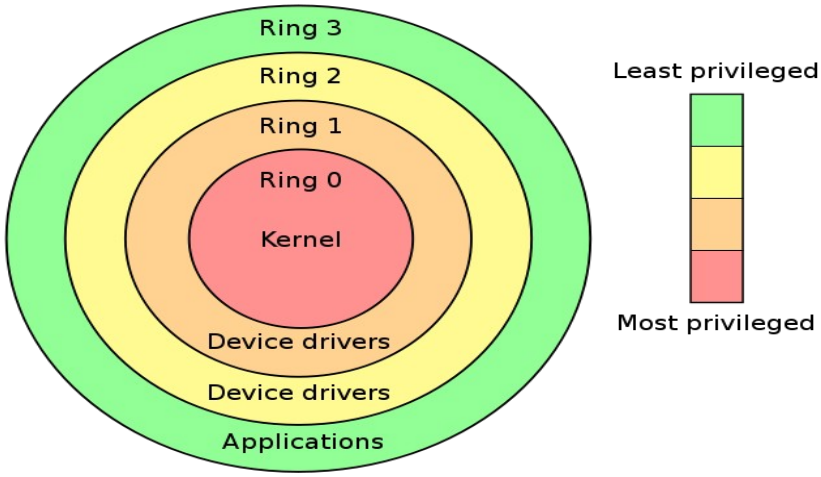

Figura 5: Niveles de protección
Fuente: http://en.wikipedia.org

El controlador ring-0 no se mete con la programación del SO o de gestión de procesos. El sistema operativo invitado completo, incluyendo sus cientos de procesos, es solo programado cuando el sistema operativo anfitrión le da a la máquina virtual una porción de tiempo.

Después de que una máquina virtual haya arrancado, desde el punto de vista del procesador, el ordenador puede estar en uno de varios estados:

1. La CPU puede estar ejecutando código ring-3 del SO anfitrión o código ring-0 del SO anfitrión, lo cual sucedería si no está funcionando VirtualBox.
2. La CPU puede estar emulando código del SO invitado (para el SO anfitrión es considerado una aplicación). Básicamente VirtualBox intenta ejecutar código invitado nativamente, tanto como sea posible. Pero puede lentamente emular código del SO invitado como último recurso cuando no está seguro de qué está haciendo el sistema invitado o cuando la penalización de rendimiento de emulación no es muy alta. El emulador de VirtualBox alojado en /src/emulator/ y se basa en QEMU.
3. La CPU puede estar corriendo código nativo de ring-3 del SO invitado. En VirtualBox, se llama a esto raw ring 3. Esto es, por supuesto, la forma más eficiente de hacer funcionar al SO invitado y afortunadamente no se abandona este modo muy a menudo. Cuanto más se abandone más lenta será la máquina virtual si se compara con un sistema operativo nativo, ya que todos los cambios de contexto hacen perder mucho tiempo y al final es un tiempo en el que no avanza la ejecución de la máquina virtual.
4. La CPU puede estar ejecutando código nativo de ring-0 del SO invitado. El SO invitado piensa que está ejecutando código de ring-0, pero VirtualBox ha engañado al SO invitado para en cambio entrar en ring-1 (el cual no es habitualmente usado por los sistemas operativos x86).

Entonces en el contexto del SO invitado hay:

- ring 3 (afortunadamente ejecutada en "raw mode" todo el tiempo)
- ring 1 (del cual el SO invitado piensa que es el ring 0 como se ha dicho anteriormente)
- ring 0 (el cual es el código de VirtualBox). Este código ring-0 es el hipervisor.

1.5.5.2 Técnicas avanzadas: escaneo de código, análisis y parcheo

Es necesario manipular el SO huésped para ejecutar el código ring-0 en el ring-1. En el ring-1 no se permiten instrucciones privilegiadas y eso causa errores. Por cada fallo el hypervisor tiene que emular el código para solucionar el problema. Además debido a fallos de diseño en la arquitectura AMD64, algunas instrucciones que deberían fallar en ring-1 no lo hacen. Es necesario que estas instrucciones se encuentren y sustituyan. Para estos problemas, VirtualBox hace uso de unas técnicas únicas que llama "Patch Manager" (PATM) y "Code Scanning and Analysis Manager" (CSAM) que significan controlador de paches y administración de escaneo y análisis de código. Antes de ejecutar código de ring 0, VirtualBox escanea el código recursívamente para descubrir instrucciones que puedan llegar a ser problemáticas. Entonces realiza el parcheo in-situ, es decir, cambia la instrucción con un salto a la memoria de hipervisor donde un generador de código integrado coloca una implementación más adecuada y que viene a tener la misma función que lo que se quería ejecutar directamente. En realidad, esta es una tarea muy compleja, ya que hay muchas situaciones extrañas para descubrir y manejar correctamente.

Además, cada vez que se produce un fallo, analiza la causa del fallo para determinar si es posible reparar el código infractor para prevenir fallos peores en el futuro. Esto es un buen trabajo y se pueden reducir fallos causados por la virtualización a una tasa que rinde mucho mejor que un recompilador típico o incluso de tecnología VT-x.

1.5.5.3 Características importantes

He aquí las características más importantes de VirtualBox:

- Portabilidad: VirtualBox funciona en un largo número de sistemas operativos de 32 y 64 bits. VirtualBox funciona idénticamente en todas las plataformas y se usa el mismo formato de imagen y de archivo. Esto permite que puedas arrancar una máquina virtual creada en un ordenador, en otro ordenador diferente aunque tenga diferente sistema operativo; por ejemplo puedes crear una máquina virtual en Windows y arrancarla en Linux.

- Además se pueden exportar e importar fácilmente las máquinas virtuales usando Open Virtualization Format que está disponible en VirtualBox desde la versión 2.20. Se crea el formato ovf, que es un estandard en máquinas virtuales. Con este tipo de archivo se pueden importar OVFs que fueron creados con otro software de virtualización como VMware.

- No requiere virtualización de hardware salvo para los siguientes escenarios: ciertos sistemas operativos raros como OS/2 hacen uso de instrucciones extrañas de procesador y para ello VirtualBox requiere activar virtualización por hardware y también para soportar sistemas operativos de 64 bits.

- Adiciones del invitado: carpetas compartidas, virtualización 3D: Los paquetes llamados "Guest Additions" pueden ser instalados para mejorar el rendimiento y proveer integración adicional y comunicación con el SO anfitrión. Después de instalar estos paquetes una máquina virtual será capaz de soportar automáticamente el ajuste de la resolución de pantalla, aceleración de gráficos 3D, etc. También se puede compartir carpetas entre en el sistema operativo y el anfitrión. Otro gran detalle es que se pueden llegar a integrar las ventanas de la máquina virtual en nuestro escritorio y parecer que están en el sistema operativo host. Desaparece el escritorio del SO invitado pero se mantiene la barra de inicio y las ventanas. Para entenderlo mejor observar la figura 6 .

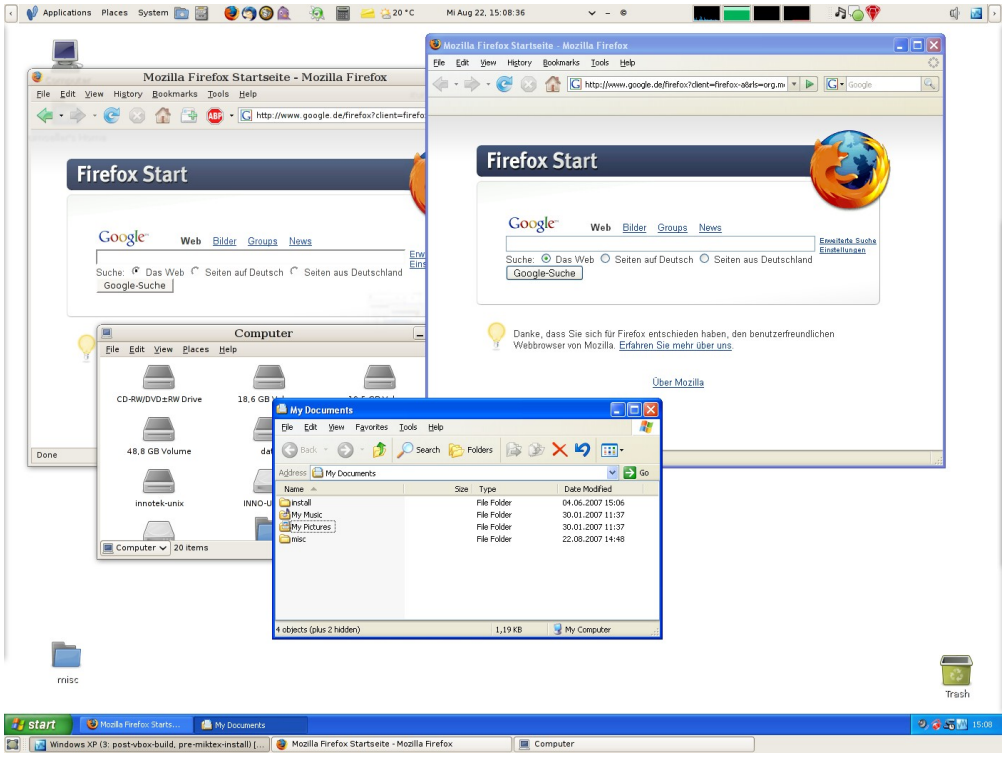

Figura 6: VirtualBox con SO host Linux y SO invitado Windows

Aquí, (figura 6) el SO anfitrión es Linux y el invitado es Windows XP, se mantienen las barras de tareas de los dos sistemas y las ventanas de los sistemas comparten el mismo entorno de pantalla, se manejan a la vez los dos SO sin aparentemente cambiar al programa de virtualización. Este modo de visualización es opcional tras instalar los Guest Additions, se debe pulsar Tecla Anfitrión (Host Key)+L.

- Gran soporte hardware**:** entre otros, VirtualBox soporta:
  - Multiprocesamiento del invitado (SMP)**.** VirtualBox puede tener hasta 32 CPUs virtuales en una máquina virtual, independientemente de el número de CPUs que se disponga en el ordenador.
  - Soporta dispositivos USB 2.0**:** VirtualBox implementa un controlador virtual USB and y te permite conectar dispositivos USB arbitrarios a la máquina virtual sin tener que instalar drivers específicos en el SO anfitrión.
  - Compatibilidad hardware: VirtualBox virtualiza una amplia gama de dispositivos virtuales, entre ellos , discos IDE, SATA y SCSI, bastantes tarjetas de red virtuales y tarjetas de sonido, puerto serie virtual y paralelo.

- Soporte total de ACPI: El ACPI (The Advanced Configuration and Power Interface) está totalmente soportado por VirtualBox. VirtualBox puede informar a un sistema operativo del estado de la energía. Para sistemas móviles funcionando en batería, el invitado puede activar el modo ahorro de energía y notificarle al usuario la energía restante.
- Arranque por red PXE : Las tarjetas de red virtuales integradas de VirtualBox soportan completamente un arranque remoto por Preboot Execution Environment (PXE).

- Copias arbitrarias: VirtualBox puede guardar instantáneas arbitrarias del estado de la máquina virtual conocidas como snapshots. Se puede ir hacia atrás en el tiempo y volver a cualquier instantánea para empezar a crear una configuración de la máquina virtual alternativa .
- Arquitectura limpia: VirtualBox tiene un diseño extremadamente modular con interfaces bien definidas de programación interna y una clara separación de código de cliente y servidor. Esto hace que sea fácil de controlar desde varias interfaces a la vez: por ejemplo, se puede iniciar una máquina virtual simplemente haciendo clic en un botón en la interfaz gráfica de usuario de VirtualBox, y luego controlar la máquina desde la línea de comandos, o incluso de forma remota.
- Visualización de máquina remota: Puedes arrancar cualquier máquina virtual en un programa especial de VirtualBox que actúa como servidor para VRDP (VirtualBox Remote Desktop Protocol). Con esta característica VirtualBox proporciona un alto rendimiento en acceso remoto a una máquina virtual. Como VirtualBox no se basa en el servidor RDP que está construido en Microsoft Windows, un servidor VRDP ha sido construido directamente en la capa de virtualización. Además funciona en cualquier SO (incluido en modo texto). Dentro de esta característica se encuentran dos cosas destacables:
  - Autentificación expansible RDP: VirtualBox soporta Winlogon en Windows y PAM en Linux para la autentificación RDP. Además incluye un SDK (software development kit= kit de desarrolo de software) fácil de usar que permite crear una interfaz distinta para otros métodos de autentificación.
  - USB por RDP: Cuando un usuario está conectado remotamente a una máquina virtual remota a través de RDP puede conectar un dispositivo USB en su ordenador local y puede acceder al USB desde la máquina virtual remota.

1.5.5.4 Sistemas operativos soportados

Se identifican por un lado el SO en el que se puede instalar y por otro lado el SO que se puede instalar como máquina virtual.

SO anfitriones:

- Windows
- Mac OS X
- Linux
- Solaris

SO invitados:

- Windows: Todos
- LINUX 2.4 y 2.6
- Solaris 10, OpenSolaris
- FreeBSD
- OpenBSD
- OS/2 Warp 4.5

Disco duro virtual
El disco duro virtual puede ser uno de los dos siguientes tipos:

- Tamaño fijo: Un disco de tamaño fijo tiene el mismo tamaño desde el momento en el que es creado. EL SO invitado opera un poco más rápido en un disco plano. Se puede ampliar el tamaño usando herramientas externas a VirtualBox como pmagic.
- Expandible: Un disco expandible es pequeño al principio y crece según vas añadiendo aplicaciones y datos a la máquina virtual. El tamaño del disco especificado cuando el disco es creado es el tamaño máximo al que el disco puede crecer. Usando discos expansibles ahorras espacio en el disco duro de tu ordenador.
  El fichero de disco duro virtual tiene el formato .vdi.

Una opción interesante que tiene VirtualBox es que se puede crear un disco duro virtual a partir de una partición de un disco duro físico. Por ejemplo si el usuario se encuentra en Linux y tiene instalado Windows XP en otra partición puede crear una imagen de disco duro virtual a partir de esa partición y al arrancarla el resultado sería el de haber arrancado el Windows que ya tenía.

1.5.6 User Mode Linux

User Mode Linux es el Kernel de Linux portado a su propia interfaz de llamadas al sistema. Provee una especie de máquina virtual, que corre Linux como un proceso dentro de otro kernel de Linux. La diferencia entre un kernel UML y un kernel ordinario es que el kernel UML no se comunica directamente con el hardware. En su lugar, los comandos pasan al kernel real del sistema que lo alberga, el cual controla la comunicación con el hardware. Las instrucciones pasan eficientemente desde el kernel de UML hasta el kernel del host. UML puede ejecutar código nativo y alcanza un retardo de un 20% como máximo respecto de la ejecución del mismo código en el host.

Es una forma segura de correr versiones de Linux y programas. Se pueden correr programas experimentales, probar nuevos kernels o distribuciones y estudiar el interior del kernel, sin arriesgar la instalación principal.

Linux en modo de usuario provee una máquina virtual que puede tener más recursos virtuales de hardware y software que el computador en el que corre. El almacenamiento en disco de la máquina virtual se aloja en un archivo. Se puede configurar la máquina virtual sólo con el hardware que se necesite. Si se configura correctamente el acceso limitado de la máquina virtual, nada que se haga en ella puede cambiar el computador real ni el software. Es importante aclarar que los procesos dentro de UML corren natívamente, en el procesador de la máquina huésped.

Numerosas cosas se vuelven posibles con el uso de UML. Se pueden arrancar servicios de red desde un entorno UML y siguen siendo totalmente apartados del sistema Linux principal en el cual se ejecuta el entorno UML. Los administradores pueden usar UML para establecer honeypots, los cuales permiten probar la seguridad de los ordenadores y redes que poseen.

También se puede usar para ofrecer web hosting, así cada cliente tiene acceso de root pero no del sistema físico sino del virtual y así evitar problemas debidos al cliente o personas ajenas.
La siguiente imagen (figura 7) es la estructura habitual de los procesos en Linux:

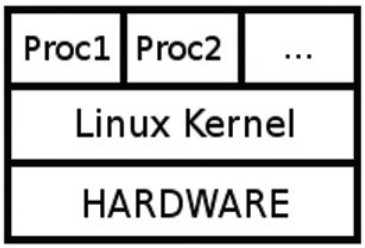

Figura 7: Estructura de los procesos en Linux

Y ésta es la estructura de Linux con UML, como se puede ver, el proceso 1 (proc1) se ejecuta sobre el sistema host Linux y el proceso 2 (el SO invitado) lo hace sobre un sistema virtual UML.

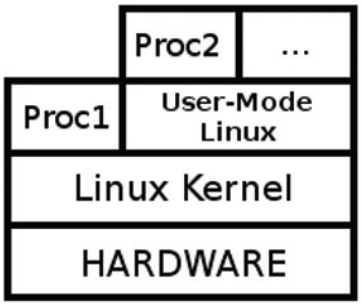

Figura 8: Estructura de los procesos con UML

VNUML (Virtual Network User Mode Linux) es un software de código abierto basado en UML. Su finalidad es definir y probar simulaciones de escenarios de red compleja. Su objetivo es ayudar en las pruebas de las aplicaciones de red y servicios a través de pruebas complejas y las redes dentro de una máquina Linux, sin tener que crearlo todo físicamente.

1.5.7 KVM

KVM es creado y mantenido por Qumranetes. Es un programa de código abierto (open source). Las siglas KVM significan Kernel-Based Virtual Machine, con esta aplicación se puede realizar una virtualización completa en Linux sobre hardware x86. Se compone de dos módulos: uno es un módulo del kernel (kvm.ko) que proporciona la base de la infraestructura de virtualización, y el otro es un módulo específico de procesador kvm-intel.ko para los procesadores de marca Intel y kvm-amd.ko para los procesadores de marca AMD.

Para hacer funcionar KVM es necesario una versión de QEMU particular diseñada para KVM. Permite tener en funcionamiento varias máquinas virtuales simultáneamente y cada máquina tiene su propio hardware virtualizado.

KVM soporta como SO huéspedes Linux (32 y 64 bits) y Windows (sólo 32 bits).

1.5.8 XEN

Xen es otro software de virtualización, se trata de un software libre bajo la licencia GNU GPL.

El hipervisor de Xen es la capa más baja y privilegiada. Sobre esta capa están uno o más SO huéspedes. El primer SO huésped arrancado se le nombra en Xen dominio 0 (dom0), éste se inicia automáticamente cuando se arranca el hipervisor y recibe privilegios especiales y acceso directo al hardware físico (ring 0).

Xen usa un modo de virtualización llamado paravirtualización para obtener un alto rendimiento.

El rendimiento que se obtiene con esta técnica de virtualización son mejores que los rendimientos que se obtienen con las técnicas tradicionales de virtualización. Para hacer uso de esta técnica el SO anfitrión es modificado para poderse ejecutar en el ring 1 y dejando el ring 0 (más privilegiado) para el Xen. De esta manera Xen controla los recursos a los que el SO anfitrión puede acceder.

Como SO anfitrión solo permite distribuciones de NetBSD, Linux y Solaris, pero permite más SO huéspedes: los SO nombrados anteriormente y FreeBSD, Windows XP.

1.6 Comparativa

Nombre	Técnica de Virtualización	Velocidad relativa al mismo SO host.	SO Anfitrión	SO Huésped
KVM	Virtualización de hardware (QEMU)	Rendimiento casi nativo.	Linux	Linux, Windows, FreeBSD, Solaris
Parallels	Virtualización de hardware	Casi nativo	Windows, Linux	Windows, Linux, FreeBSD, Solaris
QEMU	Virtualización de hardware	Rendimiento casi nativo.	Windows, Linux, Mac OS X, Solaris, FreeBSD, OpenBSD, BeOS	Linux, Windows
UML	"Porting" (Adaptar un SO para hacerlo usable en un entorno)	Casi nativo	Linux	Linux
VirtualBox	Virtualización	Casi nativo	Windows,Linux , MAC OS	Windows y Linux
VirtualPC	Virtualización de hardware	Casi nativo en Windows despacio en MAC OS	Windows o MAC OS X	Windows y Linux
VMware	Paravirtualización y virtualización	Casi nativo.	Windows, Linux,MAC	Windows, Linux, Solaris, FreeBSD, etc
Xen	Paravirtualización	Nativo.	NetBSD, Linux, Solaris	FreeBSD, NetBSD, Linux, Solaris, Windows XP

2.ESCENARIO

2.1 Qué Vm se usarán

Para este proyecto se usarán 2 VM: VMware y VirtualBox.

La razón por la que se ha elegido VMware y VirtualBox es porque son hoy en día los más usados y soportan múltiples versiones de SO tanto host como huéspedes. Además uno es de pago y el otro no y es interesante ver si merece la pena pagar por VMware o usar VirtualBox. También está la ventaja de que es posible convertir un disco duro virtual de VMware a VirtualBox por lo que se tendría todo igual y la prueba sería más real.

Cuando uno necesita usar una máquina virtual nunca sabe que software usar, cuál le merece más la pena, por compatibilidades, facilidad de uso, precio, etc. Toda la gente duda entre usar VMware y VirtualBox ya que ambos tienen compatibilidad con gran número de sistemas operativos. A nivel de usuario entre VMware y VirtualBox, quizás sea mejor VirtualBox por el coste y rapidez de instalación (es gratuito, se descarga de internet y ocupa menos que VMware) que es lo que un usuario normal busca. Sin embargo como servidor web lo más importante es el rendimiento que ofrece cada uno de ellos y el porcentaje de errores con cargas altas y eso es lo que hay que comparar.

VMware al igual que VirtualBox inserta una capa de software en el SO host o en el hardware del ordenador. Esta capa de software crea máquinas virtuales y contiene un hipervisor que asigna los recursos de hardware de manera dinámica y transparente. Gracias a esto se pueden ejecutar varios sistemas operativos de manera simultánea en el mismo ordenador físico compartiendo recursos de hardware a los que cada uno de ellos tendrá acceso cuando los necesite. Las máquinas virtuales de VMware o VirtualBox encapsulan un conjunto completo de recursos hardware así como el SO y sus aplicaciones dentro de un paquete software. Esto hace que las máquinas virtuales sean portables y se puedan guardar en USB o NAS.

2.2 Software y herramientas

Durante la realización de este PFC, se ha usado el siguiente software.

2.2.1 Apache

Es necesario la aplicación de un servidor HTTP Apache versión 2 en el lado del servidor (máquina virtual) para servir las peticiones web que realice el cliente. El servidor web Apache es uno de los proyectos que maneja la Apache Software Foundation. Existen varias razones por las que se ha usado este servidor web y no otro de los que existen en el mercado:

1. El servidor web de Apache es el más usado debido en parte a que es multiplataforma, permite usar varios lenguajes en el lado del servidor y es de código abierto. A continuación se incluyen datos obtenidos de la página http://news.netcraft.com/ acerca de la cuota de mercado de los servidores web más usados.

Desarrollador	Dominios en Agosto 2010	%
Apache	119.664.128	56,06
Microsoft	53.434.586	25,03
Google	15.526.781	7,27
nginx	11.713.607	5,49
lighttpd	1.821.824	0,85

Sin duda Apache domina el mercado de los servidores web y como se desea que la simulación que se haga sea lo más realista posible, lo más conveniente es usar lo típico.

2. Como se ha dicho en el punto anterior, Apache es de código abierto y eso facilita mucho a la hora de hacer configuraciones del servidor.
3. Es gratuito lo que conlleva un ahorro de licencia.
4. Al ser el más usado en muchos años, es del que se dispone más información y documentación en Internet, y en el caso de tener algún problema es muy probable que a través de foros y otros medios se dé con la solución mientras que con otros servidores web sería más difícil de conseguir.

Hay que saber que Apache tiene varios módulos de multiprocesamiento, los más usados son mpm_prefork y mpm_worker. Para conocer cual se dispone en el ordenador se teclea en el terminal apache2 -V. El módulo a usar se define en el momento de compilación de Apache.

El módulo mpm_prefork es la configuración predeterminada en casi todas las instalaciones de Apache. Este módulo no usa hilos sino procesos. Es el mejor método de multiprocesamiento de Apache ya que aisla cada petición, por lo que un problema en un proceso no afecta al resto.

El módulo mpm_worker implementa procesos e hilos a la vez. Apache arranca varios subprocesos y estos mantienen varios hilos que son los que se encargaran de procesar las peticiones. Un subproceso puede atender a varios clientes a la vez. El número de clientes que puede atender a la vez se define en ThreadsPerChild. Un fallo crítico afectaría a varias peticiones, al contrario que mpm_prefork que no afectaría al resto. Como un subproceso puede atender a varias peticiones a la vez, se requieren menos recursos para atender a cierto número de peticiones simultáneas.

Para la realización de este proyecto se va a usar el módulo mpm_prefork ya que no se desea que un error afecte a varias peticiones.

Se comprueba que Apache está compilado con dicho módulo.

```
tlm@ubuntu:~$ apache2 -V
Server version:...
…
Server MPM:  Prefork
forked:          yes
…
Server compiled with....
-D APACHE_MPM_DIR=”server/mpm/prefork”
```

Accediendo al archivo de configuración se pueden cambiar parámetros según lo que se necesite. Dado que el estudio requiere probar varias configuraciones del servidor web se hará uso de este archivo.

Dentro de este archivo de configuración se puede encontrar una sección dedicada al módulo mpm_prefork y convendría conocer mejor sus parámetros y la forma en que afecta al servidor de Apache. Las directivas que existen en esta sección son las siguientes:

StartServers: Indica cuántos procesos creará Apache al arrancar.

MaxSpareServers: Número máximo de procesos inactivos esperando peticiones.

MinSpareServers: Número mínimo de procesos inactivos esperando peticiones.

MaxClients: Número máximo de peticiones que se pueden ejecutar a la vez.

MaxRequestsPerChild: Número máximo de peticiones que cada proceso puede atender. Al ponerlo a 0 el proceso nunca terminará.

Además se añade otra directiva: ListenBacklog la cual permite definir cuál será el tamaño de la cola de peticiones pendientes de procesar de Apache. Por defecto ListenBacklog está definido a 511. Un ejemplo de configuración del módulo mpm_prefork sería el siguiente:

```
<IfModule mpm_prefork_module>
    StartServers          5
    MinSpareServers       5
    MaxSpareServers       10
    MaxClients            150
    MaxRequestsPerChild   0
    ListenBacklog         200
</IfModule>
```

Los 3 primeros parámetros no son muy importantes ya que solo sirven para indicar a Apache cúantos procesos deben crearse al arrancar Apache o cuántos deben estar inactivos esperando. Modificar estos parámetros solo modificaría los tiempos del comienzo que es cuando se generan los procesos, pero cuando haya tantos procesos de Apache como el número fijado en MaxClients no hay que generar más procesos Apache y no afecta (salvo que los termine por agotar el MaxRequestPerChild y los vuelva a lanzar. Si el valor es 0, no hay límite)

Cada vez que se haga algún cambio en el archivo de configuración es necesario teclear /etc/init.d/apache2 restart en el terminal para que los cambios surtan efecto.

Cuando se instala Apache en la máquina virtual, antes de continuar es necesario saber si se dispone ya de acceso desde el exterior a la máquina virtual. Se comprueba si desde un ordenador externo se consigue acceder a la máquina virtual. El resultado es negativo. Para cada máquina virtual este paso es diferente:

1-VMware: mirando la documentación de VMware Workstation 7, para hacer port forwarding es necesario seguir estos pasos:

- Tener todas las máquinas virtuales de VMware paradas y el programa cerrado.
- Parar VMware con /etc/init.d/vmware stop

Editar el archivo /etc/vmware/vmnet8/nat/nat.conf, donde esté la sección [incomingtcp] (si no está se añade) se escribe " 8080 = 192.168.182.128:80 " que significa que los paquetes TCP que lleguen por el puerto 8080 al SO anfitrión, se los redirija al 192.168.182.128 , el SO huésped, por el puerto 80.

- Volver a arrancar VMware con /etc/init.d/vmware start

2-VirtualBox:

- Primer requisito, al igual que VMware, tener la máquina virtual parada.

- $ VBoxManage setextradata "*[nombre VM]*"

  "VBoxInternal/Devices/pcnet/0/LUN#0/Config/*[servicio]*/GuestPort" *[puerto VM]*

- $ VBoxManage setextradata "*[nombre VM]*"

  "VBoxInternal/Devices/pcnet/0/LUN#0/Config/*[servicio]*/HostPort" *[puerto Host]*

- $ VBoxManage setextradata "*[nombre VM]*"

  "VBoxInternal/Devices/pcnet/0/LUN#0/Config/*[servicio]*/Protocol" *[protocolo]*

Al *puerto de VM* se le debe poner *80* y al del host se le pone por ejemplo *9090* así todo lo que reciba por ese puerto lo mandará al puerto 80 de la VM. El servicio se le puede poner *web* y el *protocolo TCP*. Si se hubiera puesto el puerto 8080 no funcionaría porque entraría en conflicto con el redireccionamiento del puerto que se ha configurado en VMware en el punto anterior.

Se comprueba de nuevo en un navegador si poniendo la ip de la máquina que alberga la máquina virtual y el puerto correspondiente, se consigue ver el index.

2.2.2 JMeter

El programa que se usará para realizar las peticiones desde el lado del cliente se llama JMeter.

Se empieza por crear un Plan de pruebas al que se le añade un grupo de hilos. En esta sección se define cuántos hilos habrá, en cuánto tiempo tienen que arrancarse todos y la duración de la prueba o el número de peticiones que realiza cada hilo. Se añade un muestreador "Petición HTTP" en el que hay que definir la ruta del archivo que se solicitará, la dirección IP de la máquina que tiene la máquina virtual y el puerto por el cual redirige el trafico web a la máquina virtual, en este caso el 8080.

Se añaden dos listeners para guardar los datos obtenidos de las pruebas a ficheros. Se usan dos listeners y no uno ya que uno será para guardar los resultados de las peticiones correctas y el otro de las peticiones erróneas o fallidas. Se puede configurar qué datos se desea que aparezcan y cuáles no y en qué formato (xml o csv). Se elige el formato csv (comma separated values) ya que para la realización de un script que lea los campos deseados es más fácil y el mismo archivo en formato xml ocupa más del doble puesto que es necesario para cada campo el nombre, por ejemplo: timestamp=10972132. Mientras que con csv simplemente aparecen los valores separados por comas pero no aparecen los nombres que se repetirían cada línea.

Los campos que se necesita que aparezcan en el fichero de las peticiones respondidas son el timestamp (tiempo en que se realiza la petición, cuenta el momento del syn), time elapsed (tiempo total de la petición desde el SYN hasta el FIN+ACK), time latency (tiempo desde que se realiza la petición hasta que se recibe la primera respuesta), bytes de la petición, número de hilo que realiza la petición,...

Del fichero de errores no hace falta entrar en detalles, no es de vital importancia ver cual es la razón del error, solo interesa contar los errores, en otras palabras, cuantas líneas hay en el fichero de errores ya que JMeter escribe una línea por cada error.

Se añade un temporizador: hay temporizador gaussiano y uniforme. El gaussiano pausa cada hilo entre petición y petición por una cantidad de tiempo variable (se le indica un tiempo máximo), con una particularidad, la mayoría de los intervalos de tiempo obtenidos se acercan a un valor específico. El temporizador uniforme pausa cada hilo por una cantidad de tiempo aleatoria en la que cada intervalo de tiempo tiene la misma probabilidad de suceder. Se opta por el temporizador uniforme. A este temporizador se le indican dos parámetros:

1. Máximo número aleatorio de pausa: es el tiempo máximo que puede pausarse un hilo. Se define en milisegundos.
2. Retraso constante: Tiempo mínimo que se retrasa un hilo.

El tiempo que está pausado un hilo después de haber terminado una petición hasta lanzar otra es la suma del retraso constante y del máximo número aleatorio.

2.2.3 PHP

PHP es un lenguaje de programación interpretado, es decir, se ejecuta con un intérprete al contrario de los lenguajes compilados. Fue diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero también puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica. Hoy en día está instalado en un gran número de servidores.

Se usa este lenguaje debido a que se necesita hacer una página web dinámica y por su sencillez, además permite ejecutar comandos de la shell con la función shell_exec, lo cual sirve para llamar a otros scripts realizados en bash.

Aunque PHP no es un lenguaje muy seguro, en este escenario no se va a tener en cuenta la seguridad ya que no es de gran importancia.

2.2.4 BASH

Bash es un programa informático cuya función consiste en interpretar órdenes. Está basado en la shell de Unix y es compatible con POSIX. Es el intérprete de comandos por defecto en la mayoría de las distribuciones de Linux. Su nombre es un acrónimo de Bourne-Again Shell, haciendo un juego de palabras (born-again significa renacimiento) sobre el Bourne shell (sh), que fue uno de los primeros intérpretes importantes de Unix.

Con este intérprete se pueden usar comandos como sed para el preprocesado de ficheros, awk, gnuplot...

2.2.5 AWK

AWK es un lenguaje de programación diseñado para procesar datos, ya sean ficheros o flujos de datos. El nombre AWK deriva de los apellidos de los autores: Alfred Aho, Peter Weinberger, y Brian Kernighan. awk, cuando está escrito todo en minúsculas, hace referencia al programa de Unix.
AWK es ejemplo de un lenguaje de programación que usa ámpliamente el tipo de datos de listas asociativas (es decir, listas indexadas por cadenas clave), y expresiones regulares. Este lenguaje de programación se utilizará debido a su facilidad de uso, se accede muy fácilmente a las palabras de una línea en concreto y así se puede trabajar con esos datos.

2.2.6 Sed

Sed es un editor de flujos y se utiliza para el preprocesado de ficheros, como por ejemplo borrar lineas vacías, quitar la parte decimal a los tamaños de los ficheros, etc.. Así se consigue tener los ficheros con la estructura que se necesita.

2.2.7 Gnuplot

Es un programa que permite generar gráficas a partir de ficheros. Se usa gnuplot para ver cómo evoluciona el tiempo medio de las peticiones, el tamaño de los ficheros para ver como están de dispersos,...

2.2.8 Tac

Herramienta de Linux que sirve para mostrar un fichero de texto empezando por la última línea y terminando por la primera. Su nombre se basa en el comando cat, pues ambos hacen lo mismo pero al revés.

2.3 Problemática

Hay que simular una gran población que realiza peticiones web al servidor, que es una máquina virtual de Ubuntu Server. Se generará un número determinado de hilos y cada uno de ellos realizará varias peticiones con un tiempo aleatorio de pausa entre cada petición del mismo hilo, cuyo máximo y mínimo se establecerá.

La idea es que el escenario sea lo más realista posible para que tenga sentido, por lo tanto hace falta preparar todo para que sea adecuado a lo que se necesita.

2.3.1 Conexiones persistentes

Probando con la configuración de Apache sin modificar, se observa que si un hilo realiza una petición y al poco tiempo realiza otra petición, no vuelve a mandar un paquete con el flag SYN para establecer la conexión con el servidor porque ya la tiene establecida (conexiones persistentes). Este caso no es realista ya que se desea considerar que cada petición proviene de un usuario diferente y para ello es necesario que haya una conexión por petición. Por lo tanto, cada vez que un hilo hace una petición establecerá una nueva conexión y la cerrará al recibir la página solicitada.

Existen varias maneras de eliminar las conexiones persistentes:

- Usando HTTP/1.0 en vez de HTTP/1.1 que es el que siempre se usa y por defecto realiza conexiones persistentes a no ser que se especifique lo contrario. HTTP/1.0 es una versión más antigua (año 1996) y no crea conexiones persistentes. Sí que existe alguna implementación de HTTP/1.0 que incluyen una cabecera para indicar conexión persistente pero este diseño no lo reconocen los proxys intermedios.
- Apache también dispone de una directiva para elegir entre conexiones persistentes o no en el archivo de configuración apache2.conf o httpd.conf localizado en el directorio /etc/apache/. Esta directiva se la conoce como KeepAlive. Por defecto Apache tiene esta directiva en On, estableciendo esta directiva a Off evitaría las conexiones persistentes.

  La manera más fácil de eliminar las conexiones persistentes es modificando el archivo de configuración de Apache (la 2º opción de las expuestas). No obstante, haciendo este cambio hay que ser conscientes de que se elimina la posibilidad de que un usuario pida varias páginas en poco tiempo y que deberían entrar en la misma conexión pero se acerca más a la realidad la forma en la que ha sido enfocado este problema anteriormente.

Tras modificar el archivo de configuración se comprueba que se realiza una conexión por petición. Esta comprobación se puede realizar con un solo hilo que realice varias peticiones, una tras otra, y mientras tanto observar con un programa como wireshark o tcpdump (programas cuyo uso es capturar los paquetes que circulan por los interfaces de red) si solo existe un paquete con flag SYN y otro con flag FIN, o hay tantos paquetes con flag SYN y FIN como peticiones realiza el hilo. Se filtran los paquetes que son de las direcciones IP de los ordenadores que intervienen y que tengan el flag de SYN o FIN activado. Dado que las peticiones son casi inmediatas, si no fuera por el cambio que se ha hecho entrarían todas las peticiones dentro de la misma conexión, por lo que se puede confirmar que se ha solucionado el problema de las conexiones persistentes y para cada petición se establecerá una conexión.

2.3.2 Caché

Otro tema a tratar es la caché: es obvio que acceder a datos de la caché es mucho más rápido que acceder a datos de disco, por lo que si el servidor sirve una página que la tiene en caché lo hará en menos tiempo que si no la tiene en caché. Es necesario comprobar si la caché afecta mucho o es inapreciable.

Se realiza una prueba con un fichero de 50 megabytes, y otra prueba con 100 ficheros de 50MB los cuales se pedirán por orden (51200kb_1.html, 51200kb_2.html...), que aunque se pidan muchas veces no entran en la caché ya que solo se dispone de 2GB de memoria RAM. Para la realización de la prueba se preparan dos ficheros php que se encargan de redireccionar al archivo correcto. Ambos tienen exáctamente las mismas instrucciones, aunque para el archivo PHP que sirve siempre el mismo archivo es algo inútil, pero así la penalización de tiempo que sufre el archivo que tiene que redireccionar cada vez a un archivo diferente este también la sufra y la diferencia de tiempos sea solo la caché y no la caché más algunas instrucciones extras. Solo se diferencian en que uno redirecciona siempre al mismo fichero y el otro a una variable que es un fichero del 1 al 100.

Es necesario pensar los parámetros necesarios para la prueba. Para esta prueba será suficiente un solo cliente así que no importan los parámetros de Maxclients y ListenBacklog.

Tras establecer solamente un hilo en Jmeter, se indica el tiempo que se desea correr la prueba:con 1000segundos (algo más de 16minutos) será suficiente para que se establezcan los tiempos medios por petición. En esta prueba no hay que usar temporizador aleatorio o en su caso ponerle 0ms ya que se desea que cada vez que finalice una petición, el hilo vuelva a solicitar otra página.

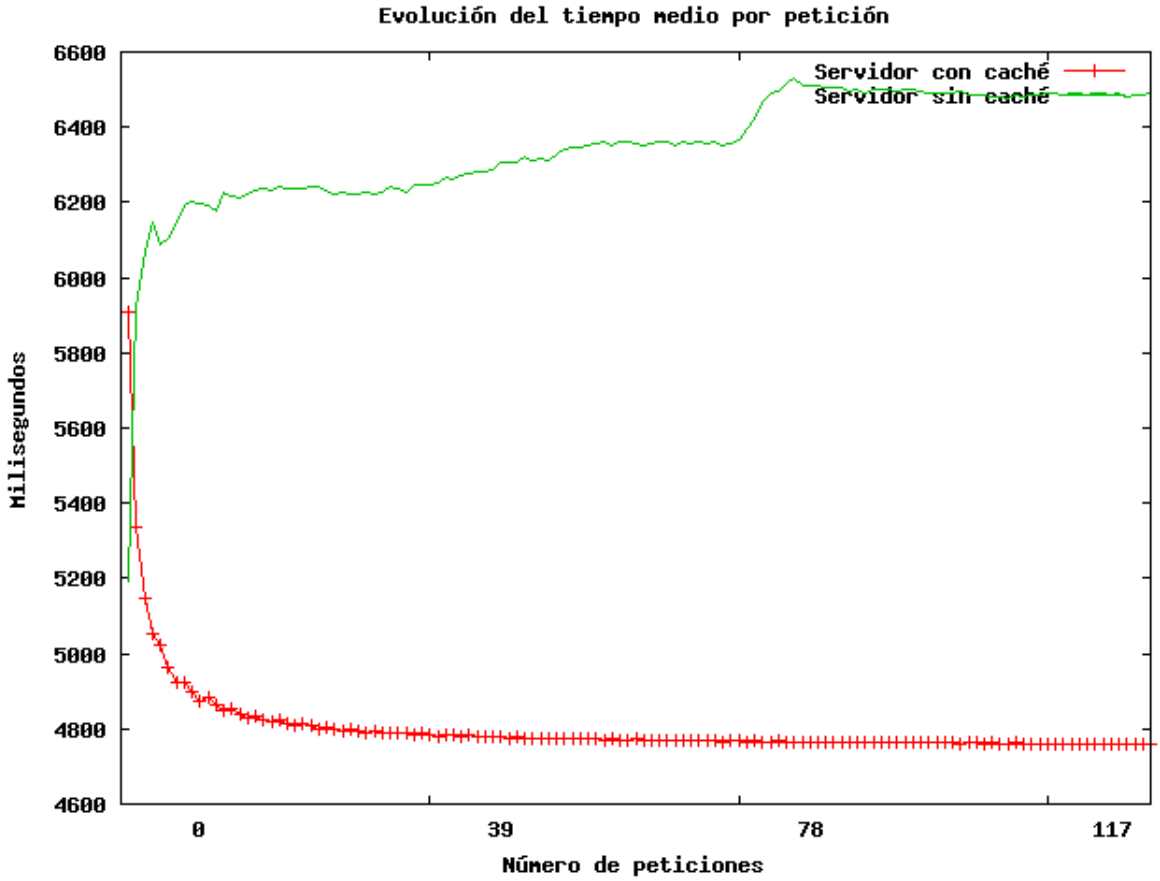

Figura 9: Prueba de caché

Para asegurarse de que los resultados son ciertos se repiten las 2 pruebas de nuevo y se comprueba que el resultado es el mismo. Los resultados son muy claros,indican que hay una considerable diferencia ya que las curvas de la gráfica no comparten ninguna similitud. La prueba en la que el servidor no ha usado la memoria caché, la primera petición ha tenido un tiempo aproximado de 5000ms y ha ido creciendo un poco estabilizándose en 6502ms de tiempo medio por petición. Con esta prueba se ha conseguido resolver 9 peticiones por minuto. La otra prueba en la que el servidor sí ha usado la memoria caché, la primera petición también comienza con un tiempo de 5000ms aproximadamente y petición tras petición va bajando el tiempo medio hasta que se estabiliza por completo a los 4755ms. Este es el tiempo medio por petición al acabar la prueba. Si bien la otra prueba resolvía 9 peticiones/min, esta llega a las 13 peticiones/min. Teniendo en cuenta que son 50MB por cada fichero la diferencia es notable por lo que hay que tenerla en cuenta y se decide que los archivos que se pidan no estén ya en caché. Para asegurarse de ello todos los archivos que se soliciten serán diferentes aunque algunos tengan el mismo tamaño.

2.3.3 Aleatoriedad

Una de las necesidades es que no se pidan los archivos en un orden en concreto como por ejemplo de menor a mayor tamaño porque eso no suele suceder nunca. Lo que se busca es que haya una cierta aleatoriedad y parezca un escenario real. Obviamente si se trabaja con números aleatorios surge otro problema: para cada número aleatorio se tiene que tener un fichero del tamaño que le corresponde, y sería un gran coste de tiempo generarlo cuando se solicita, cosa que no sucede en la realidad, por lo que tiene que estar ya preparado en el directorio del que cuelgan los ficheros web /var/www/. No se puede tener ficheros de todos los tamaños , tienen que ser números aleatorios pero a la vez se tienen que saber cuales van a ser.

La primera idea es generar muchos números de una semilla y guardarlos en un fichero para así cada vez que se hace una petición a la página php que se encarga de redirigir, leería el fichero línea a línea hasta leer una línea que estuviera sin marcar (cada vez que se redirige, la línea que se lee debería ser marcada) y redirigir a esa página web. Esto no se puede hacer porque la petición número 20.000 leerá 20.000 líneas para saber a qué fichero tiene que acceder, con lo que supondría una gran pérdida de rendimiento. Es necesario hacerlo de otra forma, esta es inviable.

Otra idea es que cada petición genere un número aleatorio y con ese número aleatorio redirigirse a un fichero del mismo tamaño. Antes de nada, se genera para una cierta semilla muchos números aleatorios que cumplen esa semilla. Se crea un script para ello llamado sacanumerosazar.php al cual se le pasa por GET el valor de la semilla. Al ejecutarlo da un error por estar demasiado tiempo en ejecución. Resolver este problema es muy sencillo:en /etc/php5/apache2/php.ini se modifica max_execution_time 30 por max_execution_time 3000 para que al generar los números de la semilla no de error y se pare a los 30 segundos por timeout. Tras solucionar este problema se hacen 2 pruebas para ver cómo funciona la aleatoriedad de php y ver si se puede saber qué ficheros van a pedir.

1. Una de ellas es hacer srand( ) para colocar una semilla y rand( ) y en las siguientes peticiones solamente hacer rand( ) para ver si los números continúan la secuencia de la semilla o no.
2. En la primera petición srand( ) y rand( ), guardar ese número y en la siguiente petición srand(número_guardado) y rand( ) y también ver si los números generados continúan la secuencia de números de la semilla.

La prueba 2 no da los resultados que se buscan, en cambio la prueba 1 sí. Al intentar aplicar esto en el servidor surge otro inconveniente: cada proceso de Apache tiene su propia semilla srand, por lo que cada petición no es el siguiente número de la secuencia de la semilla. Con solo un proceso de Apache funcionaría pero eso significa fijar MaxClients a 1 y el escenario no sería realista.

Por lo tanto la solución es volver a la idea de leer un fichero pero hacerlo de manera más eficiente. En vez de leer línea a línea sería ideal saber qué línea hay que leer, es decir, tener un contador. Se barajan varias opciones como la de escribir el número de línea en un fichero o la de usar un espacio de memoria compartida. Se elige la opción de la memoria compartida y así se consigue ahorrar más tiempo pues es más rápido acceder a memoria que a disco. En la memoria compartida se guarda pues un contador, y ese numero será el número de línea que tiene que leer para saber a que fichero acceder. Este fichero deberá estar perfectamente estructurado ya que se accederá directamente a la línea con fseek y fgets por lo que hay que saber perfectamente en que bytes empieza la línea y en qué bytes acaba.

Para preparar este fichero se realiza lo siguiente:

1. Con un script se generan los 15.000 primeros números de la semilla que no sean exageradamente grandes. Dado que se ha realizado el script sacanumerosazar.php anteriormente y ya no se va a volver a usar, se reutiliza modificándolo para esta ocasión.
2. A través del mismo script se convierten esos 15.000 números que se sitúan entre 0 y 1, que significan la probabilidad acumulada de un fichero de tamaño *x*. La distribución que se va a usar es la distribución de Pareto cuya fórmula es:

$$F(x) = 1 - \left(\frac{b}{x}\right)^a$$

Figura 10: Fórmula de la distribución de Pareto.

Esta fórmula (fig.10) tiene varias incógnitas: x,a,b y F(x).

Se sabe que *F(x)* que es el número aleatorio, *b* en este caso es el tamaño mínimo de fichero (0,1MB) y la *a* se fija. Si *a* es mayor que 2 se trata de una ley de potencia, si es menor o igual a 1 la media no está definida y si *a* es menor o igual a 2 la varianza no está definida. Solamente falta por conocer la incógnita *x* que es el tamaño. Despejando la fórmula quedaría así.

$$x = \sqrt[a]{b^a/(1-F(X))}$$

Figura 11: Incógnita x despejada

3. Los números obtenidos en *x*, se consideran los tamaños de los ficheros. Como salen números decimales se ha optado por eliminar los decimales aunque se podría redondear.
4. Como se repite varias veces el mismo tamaño de fichero y no se desea que cuando se pida uno esté cacheado, por ejemplo se llamarán los archivos de la forma 537_00001 y 537_00002 para dos archivos diferentes de 537KB. Estos nombres de archivos serán almacenados en un fichero para que cuando haya una petición, obteniendo el número de contador (llámese *n*), vaya a la línea *n* y leyendo la línea conoce el nombre del fichero añadiendo .html .

Ahora hay un problema más simple: que si hay varias peticiones a la vez el contador de la memoria compartida puede ser modificado de manera no deseada. Para garantizar que no sucede nada extraño se usarán semáforos. Hay que intentar que el semáforo esté bloqueado el mínimo tiempo posible para no afectar al resto de peticiones que están a la espera, ya que el semáforo tendrá un max_acquire de 1 y así solo poder ser modificado por un solo proceso. Así que cuando se tiene el permiso del semáforo, se lee el número de la memoria compartida y se escribe en la memoria compartida ese número +1 y se libera el semáforo.

Teniendo en $sem_id el identificador del semáforo y en $shm_id la memoria compartida quedaría así:

```
sem_acquire($sem_id);
$numero = ereg_replace("[^ A-Za-z0-9_]", "", (shmop_read($shm_id,0,100)));
        shmop_write($shm_id,$numero+1,0);
        sem_release($sem_id);
        shmop_close($shm_id);
```

Cuando se escribe en la memoria compartida, se pone el offset (3º parámetro de la función de escritura) a 0 para que empiece a escribir desde el principio y sobreescriba al número anterior.

Nota: para poder manejar memoria compartida y semáforos desde PHP es necesario tener el php compilado con estas opciones:

--enable-shmop

--enable-sysvsem

Hay que conseguir que la máquina virtual tenga mucha carga, hay que llevarla al límite. Para saber la carga de trabajo que tiene se puede hacer uso del comando *top* y fijarse en los datos que ofrece en la cabecera.

Las abreviaciones significan:

wa= waiting for IO

us=userspace

sy=system/kernel

id=idle

hi=hardware interrupts

si=software interrupts

También se puede mirar el estado de CPU, memoria y swap en el "Monitor del sistema" que incluye la distribución de Ubuntu que está instalada en los ordenadores de los que se hacen uso. Siempre que se haga alguna prueba de rendimiento hay que estar atentos a que ninguna computadora esté usando en ningún momento memoria swap ya que en algunas pruebas realizadas ha habido un poco de memoria swap en uso y éstas han dejado entrever una pérdida de rendimiento y han tenido que ser repetidas tras un reinicio del ordenador.

3. PRUEBAS, RESULTADOS Y CONCLUSIONES

Una vez solucionados todos los problemas que evitaban conseguir una simulación realista, se procede a realizar pruebas.

3.1 Prueba 1

Para la primera prueba se había pensado en poner 30 hilos. La prueba irá determinada por el número de veces que pide cada petición y no por duración ya que podría darse el caso de que se pidan más archivos que los que disponemos. Como son 15.000 ficheros diferentes corresponden a 500 peticiones por hilo. Hay que modificar el archivo de configuración y adaptarlo a la prueba. La directiva ListenBacklog se deja por defecto y se cambia el valor de MaxClients por 1, 2, 4, 8, 16 y 30. Además para cada valor que se introduzca en MaxClients se cambia en JMeter el apartado del temporizador aleatorio uniforme poniendo diferentes tiempos de retraso constante y de pausa aleatoria.

En concreto se prueba con los siguientes valores:

Tiempo aleatorio de pausa (ms)	Tiempo de retardo constante (ms)
125	65
250	125
500	250
1000	500

Se guardan todos los datos de resultados y errores y se pasa a ejecutar scripts para leer esos datos y mostrar un resumen de cada prueba. Se anotan los datos que ofrece el resumen y se realiza una gráficas para observar cómo aumentan o disminuyen el número de peticiones/minuto en función del tiempo de pausa medio entre cada petición del mismo hilo.

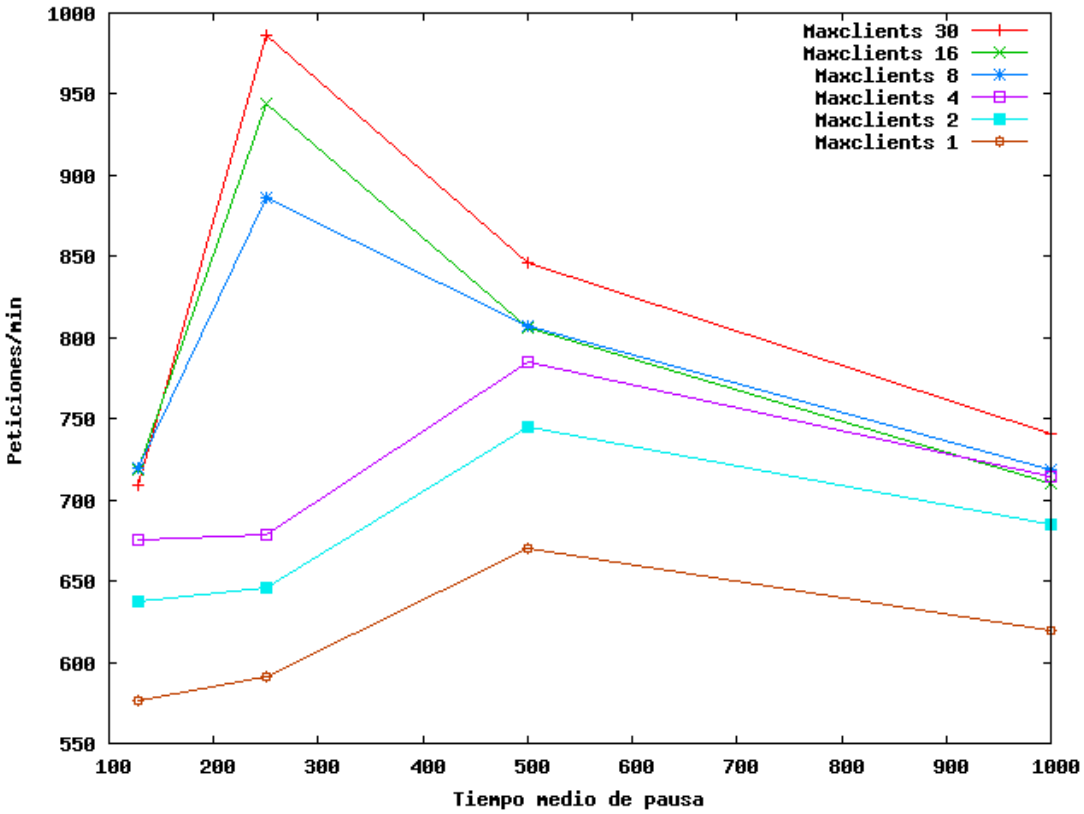

Figura 12: 30 hilos y ListenBacklog por defecto

Se coge la mejor parte que es el tiempo de pausa medio 250 y se prueba a ir bajando el número de posiciones en cola: 16, 8, 4, 2 y 1. A posteriori se vuelve a realizar otra gráfica poniendo en el eje Y las peticiones por minuto y en el eje X el número de peticiones procesadas de manera simultánea (MaxClients).

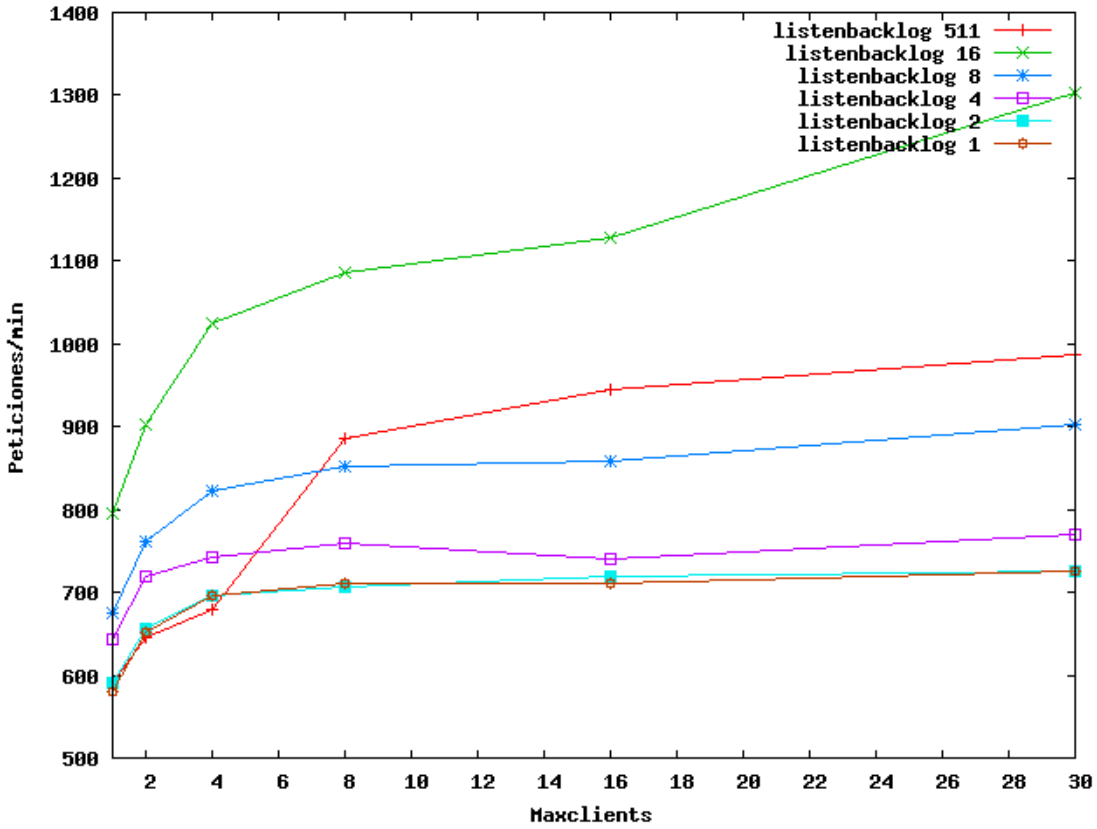

Figura 13: Peticiones/min en función de la cola y de las peticiones simultáneas

En realidad, el número de peticiones por minuto que resuelve el servidor debería poder ser un valor de entrada, es decir, a menor tiempo de pausa entre petición, más peticiones recibe el servidor y más peticiones podría resolver por minuto. Sin embargo en la práctica no ha sucedido lo mismo. El problema reside en que un hilo no lanza una petición hasta que se ha resuelto otra, por lo que el número de peticiones por minuto que recibe el servidor están reguladas por el número de peticiones que resuelve el servidor. En otras palabras, si el servidor resuelve 60 peticiones cada segundo, recibirá 60 peticiones cada segundo. La conclusión es que la carga de trabajo se autorregula en función del trabajo que gestiona.

Respecto a la figura 13 se puede observar que a medida que aumenta el número de MaxClients aumenta el número de peticiones por minuto. Al principio se nota más pero luego llega a un momento en que por falta de recursos no puede ascender e incluso puede llegar a disminuir el número de peticiones que puede resolver el servidor por unidad de tiempo.

Además también se observa que a mayor cola, más peticiones es capaz de resolver, salvo con la cola por defecto en la cual según la documentación de Apache (http://httpd.apache.org/docs/), normalmente el propio SO toma un valor más bajo y en este sentido cada SO puede actuar de forma diferente, por lo que se puede pensar que la cola no es de ese tamaño, o incluso que en función de la carga o del número de MaxClients la incrementa ya que empieza la curva con pocas peticiones/min y más tarde tiene una pendiente ascendente que el resto de líneas no hacen. Aunque se disponga de solo 1 posición de cola y MaxClients 1 y se lancen los 30 hilos, el SO siempre responde a los paquetes con el flag SYN activado para evitar perder peticiones.

3.2 Prueba 2

Se considera oportuno realizar otra prueba pero incrementando el número de hilos de 30 a 100 para intentar evitar que el servidor se autorregule. Se propone usar 50 procesos simultáneos y como máximo 40 peticiones en cola para que Apache no pueda mantener a todas las peticiones en todo momento. En concreto se usarán colas de 1, 10, 20 y 40 posiciones. En el JMeter se introducen valores de pausa similares a los de la prueba anterior e incluso alguno más alto. Tras obtener los datos se representan.

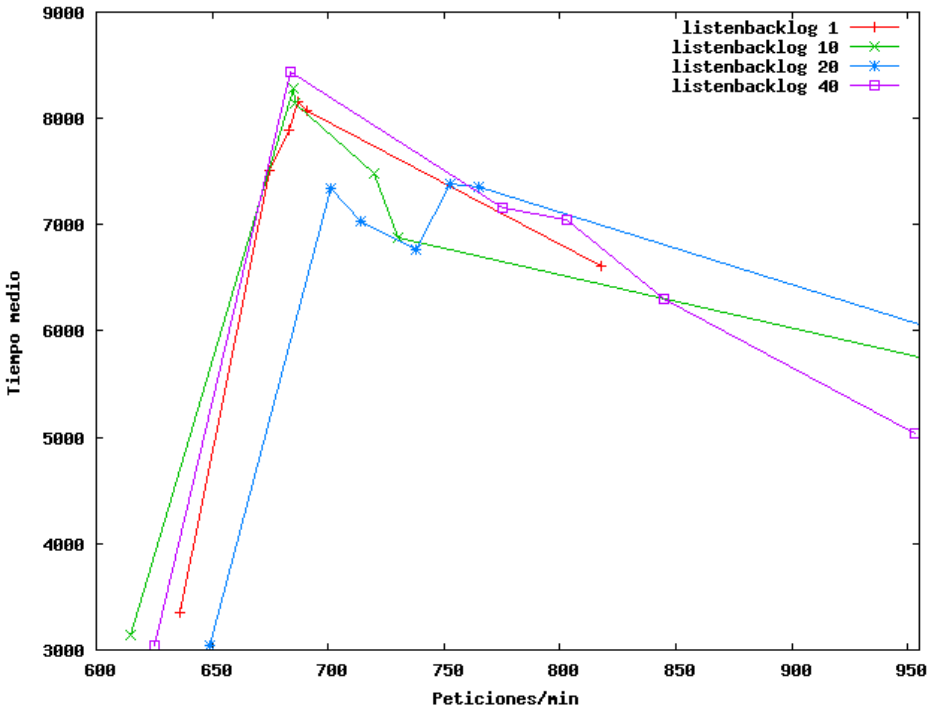

Figura 14: Peticiones/min en función del tiempo medio en VMware

La figura 14 muestra que entre 600 y 650 peticiones por minuto VMware resuelve las peticiones con un tiempo medio de 3000ms. Cuando más tardan es con 700 peticiones por minuto con un tiempo medio de 8300ms aproximadamente. Los tiempos con diferentes tipos de cola deberían variar más. El problema está en no saber qué hace el OS con la cola ya que el comando ListenBacklog que se introduce en el fichero de configuración le pasa el valor al comando listen de UNIX y es el SO el que decide poner otra cola o no. Si el número de la cola es pequeño es posible que lo ignore y elija uno.

Se realiza el mismo experimento con VirtualBox y se consiguen los siguientes resultados:

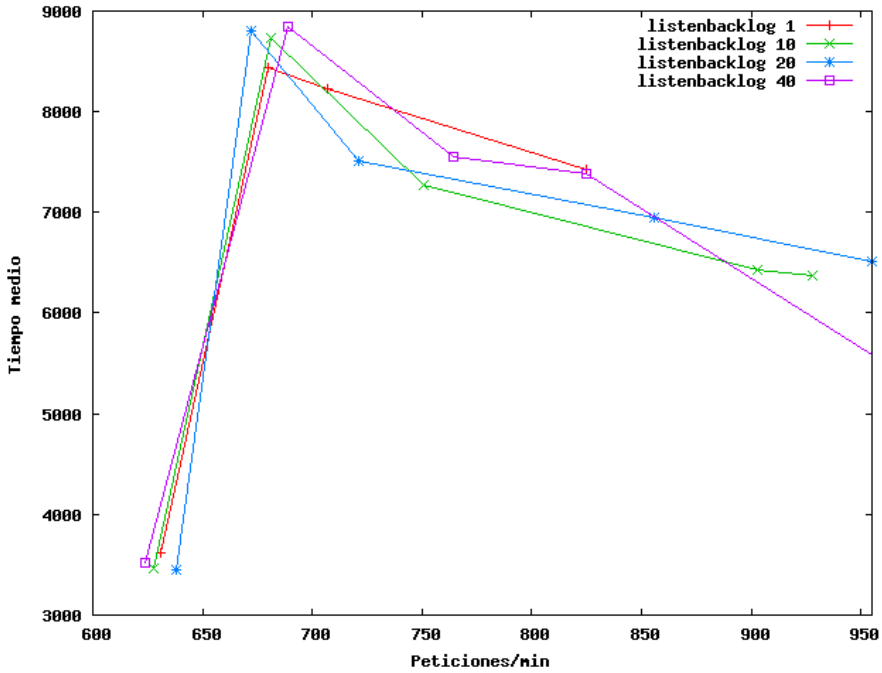

Figura 15: Peticiones/min en función del tiempo medio en VirtualBox

Como se ha visto antes, los tiempos de VMware empiezan sobre 3000ms cuando hay 600-650 peticiones por minuto y llega hasta 8300ms aproximadamente con unas 700 peticiones/minuto. Sin embargo con VirtualBox estos tiempos son algo más elevados pues con la misma carga que VMware, tanto en el mínimo tiempo como en el máximo son unos 300-400ms más elevado el tiempo medio por petición. Por lo tanto VMware ha obtenido mejor rendimiento que su rival VirtualBox.

Para observar esta diferencia de tiempos, es más perceptible representando por pares las líneas de las gráficas más significativas. Así pues, la gráfica correspondiente a las líneas con 20 posiciones de cola (la más cambiante) quedaría de la siguiente manera:

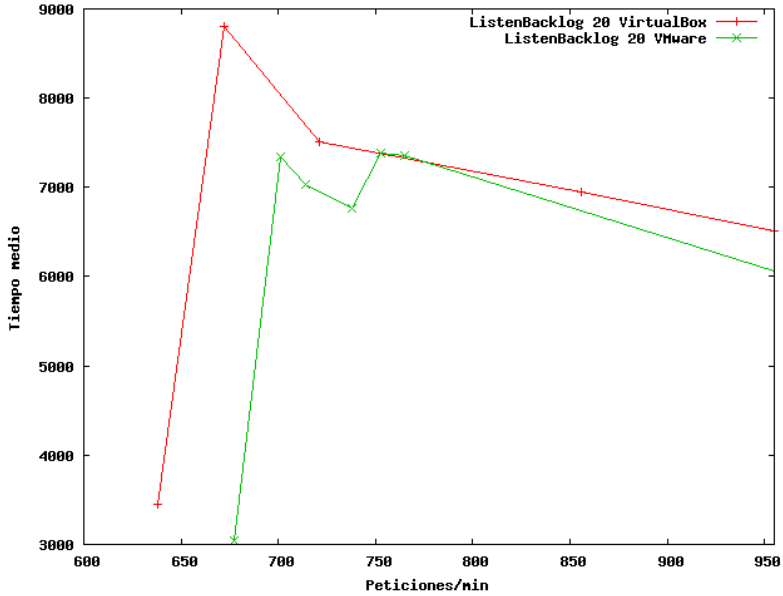

Figura 16: VMware y VirtualBox con 20 posiciones de cola

Es evidente que VirtualBox ha dado algo peores resultados que VMware. El resto de líneas no cambian mucho de unas a otras. Se genera una gráfica con cualquiera de ellas de los dos programas, por ejemplo con la de 40 posiciones de cola.

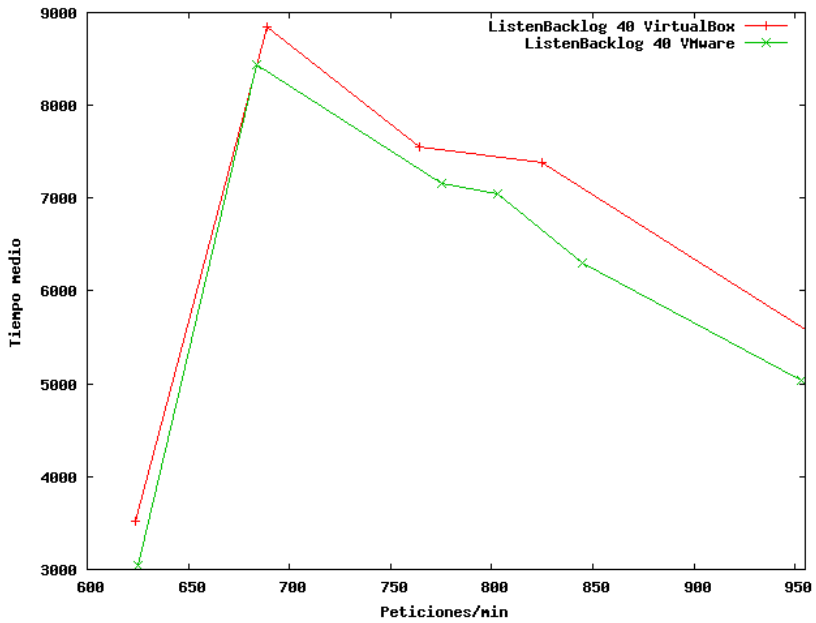

Figura 17: VMware y VirtualBox con 40 posiciones de cola

En esta gráfica (figura 17) también se puede apreciar una leve diferencia de tiempo por peticiones entre ambos programas siendo VMware otra vez más eficiente que VirtualBox.

Respecto a los errores, se consiguen las 2 siguientes gráficas:

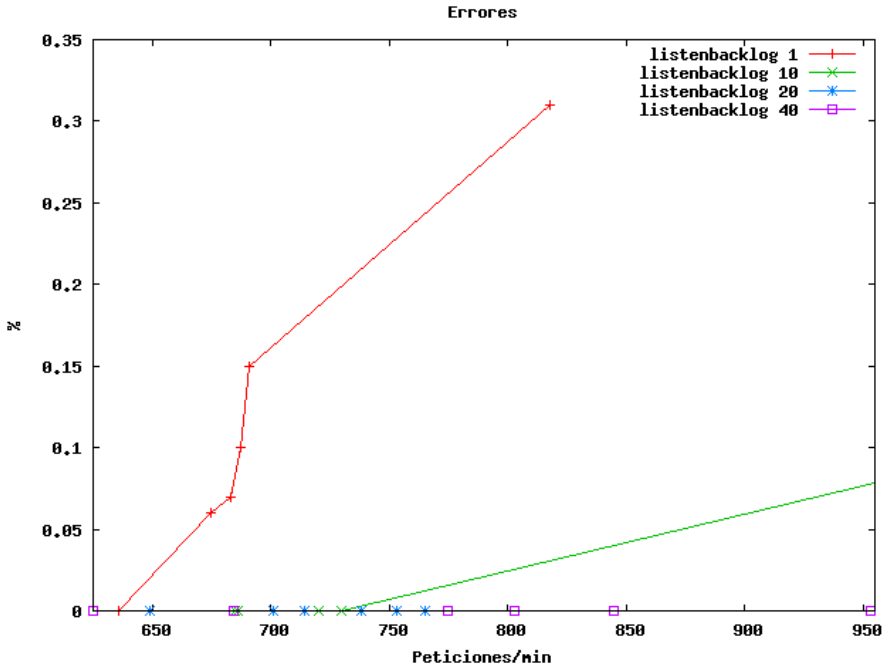

Figura 18: Errores obtenidos con VMware

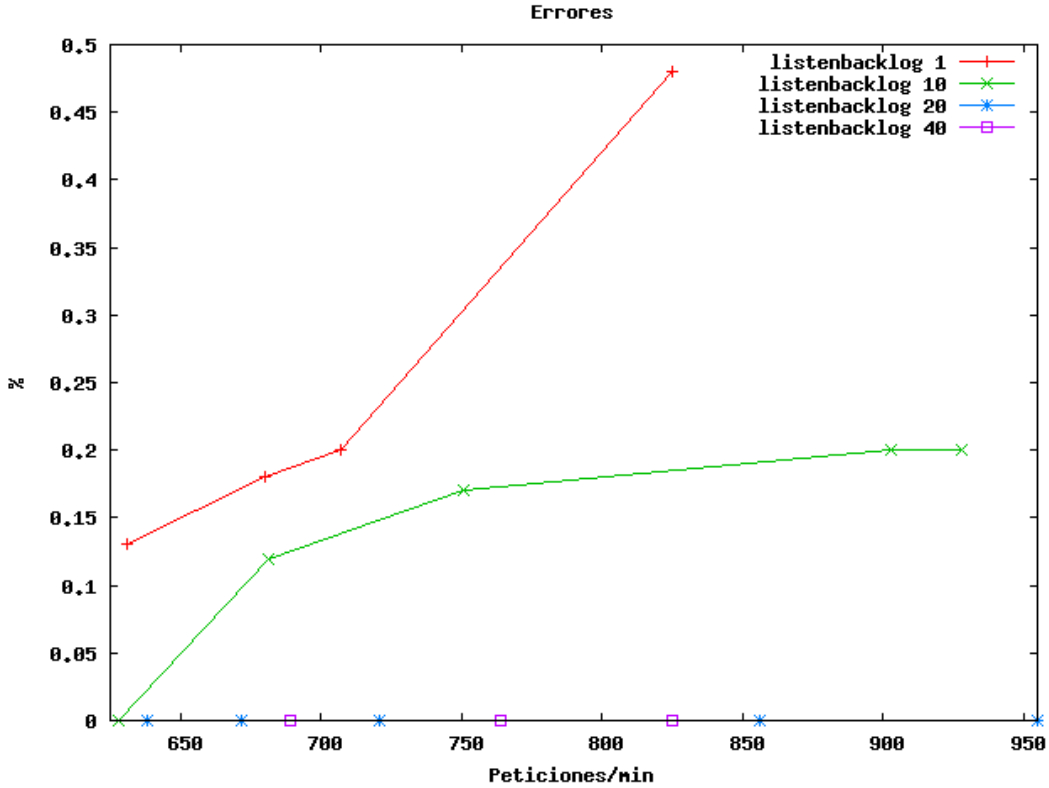

Figura 19: Errores obtenidos con VirtualBox

Claramente se observa que sólo obtenemos errores con 1 y 10 posiciones de cola. Para comprobar cúal es el que ha tenido más errores se grafican por una parte los errores obtenidos con las 2 máquinas virtuales con ListenBacklog 1 y por otra parte con ListenBacklog 10.

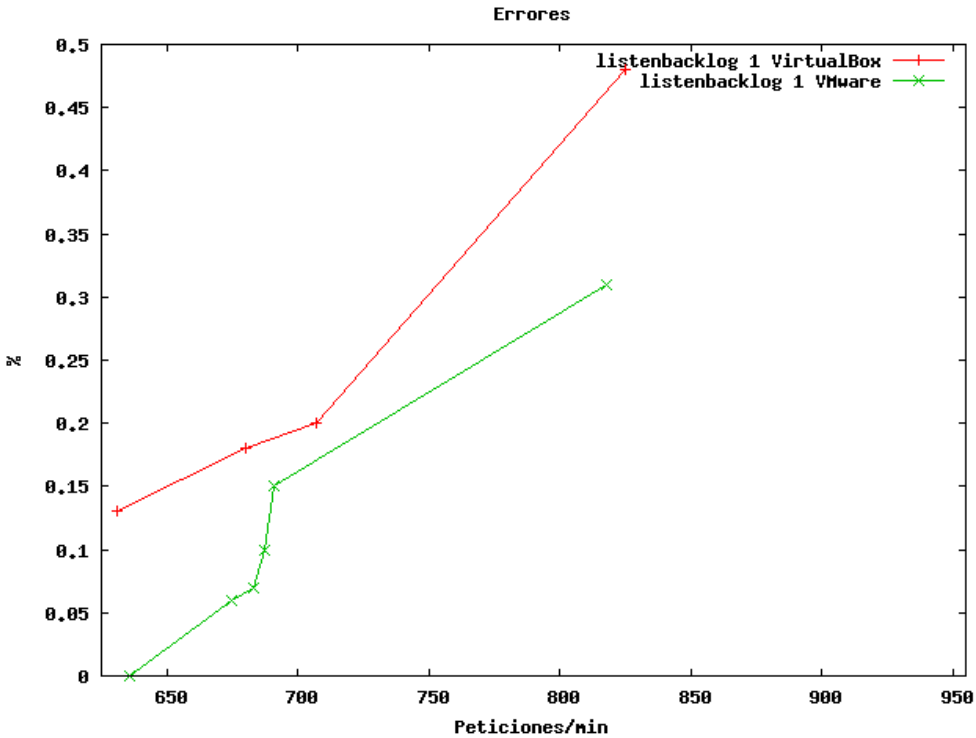

Figura 20: Porcentaje de errores con ListenBacklog 1

En esta gráfica (figura 20) aparece el porcentaje de errores obtenidos con VMware en verde y el de VirtualBox en rojo. Con todas las cargas que se han probado VirtualBox ha dado más errores que VMware con esta cola.

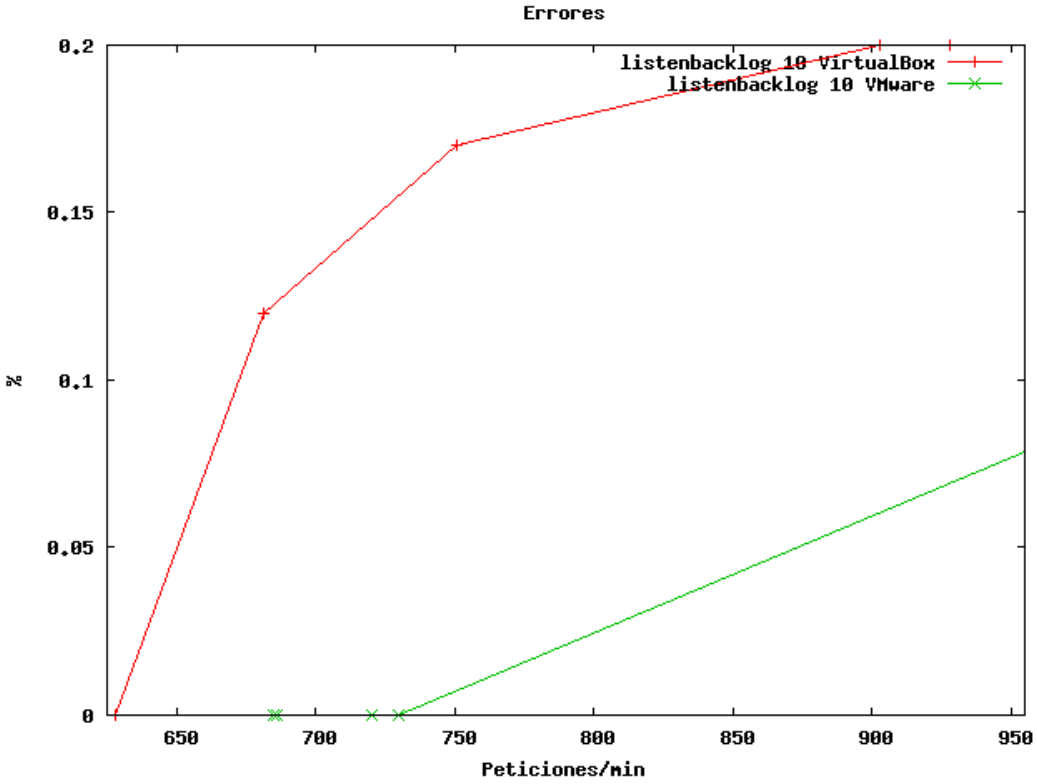

Figura 21: Porcentaje de errores con ListenBacklog 10

En este caso, con 10 posiciones de cola también ha sido la máquina virtual de VirtualBox la que ha dado más fallos, por lo cual es un punto en contra que altas cargas tenga más errores. Sin duda en este aspecto se ha comportado mejor VMware.

Resumiendo, VMware tiene un mejor rendimiento que VirtualBox y con menor porcentaje de errores. Aunque no solo eso cuenta a la hora de elegir entre un software o otro. Como se ha dicho anteriormente VMware es de pago y VirtualBox no, por lo que dependiendo del tipo de cliente, necesidades y presupuesto optará por uno o otro. Por ejemplo para un usuario final no es tan importante un poco menos de rendimiento y hasta un 0.48% de errores ya que a lo mejor nunca va a tener tanta carga pero para una empresa importante que necesite reducir al mínimo los errores y maximizar el rendimiento es muy probable que optara por VMware a pesar de tener que pagar licencias.

4. CONCLUSIONES

En este Proyecto de Fin de Carrera, se ha podido aprender:

- La importancia de la memoria caché. Se ha visto que haciendo peticiones de archivos de 50MB con memoria caché tardaban 2 segundos menos que con caché.

- Así como se ha visto la importancia de la memoria caché, también se ha visto la importancia de la memoria swap, la cual hacía que las pruebas realizadas con JMeter dieran resultados de rendimiento por debajo de lo esperado.

- El método de funcionamiento de la aleatoriedad de PHP y que cada proceso de Apache tiene una semilla para generar números aleatorios. Así si solo hubiera un proceso, dos peticiones a una página que hace rand( ); serán números consecutivos de la secuencia de números de la misma semilla.

- El SO siempre responde a los paquetes que lleguen con el flag SYN activado independientemente de la cola de la que se disponga (incluso con una posición de cola).

- El tamaño de la cola que se indique en la configuración de Apache se pasa al comando listen de UNIX y es el SO el que se encarga de decidir poner ese valor o uno basado en ese valor, o de ignorarlo y poner el que crea conveniente.

- El rendimiento de VMware es algo mayor que el de VirtualBox.

- VMware minimiza más los errores que VirtualBox.

- Con menos posiciones de cola hay más errores, algo evidente.

- De la forma en que se han hecho las peticiones, es necesario muchos hilos para que el servidor no autorregule la carga de trabajo.

5. BIBLIOGRAFÍA

[1] Página oficial de Vmware www.vmware.com

[2] Página oficial de VirtualBox www.virtualbox.com

[3] Biblioteca virtual www.wikipedia.es

[4] Aspectos básicos de virtualización http://itechthoughts.wordpress.com/2009/11

[5] Información sobre la distribución del dominio del mercado de los servidores web http://news.netcraft.com

[6] Página oficial de User Mode Linux http://user-mode-linux.sourceforge.net

[7] Página oficial de Apache http://httpd.apache.org

[8] Página oficial de QEMU http://wiki.qemu.org

[9] Página oficial de VirtualPC http://www.microsoft.com/windows/virtual-pc

[10] Página oficial de Parallels www.parallels.com/

[11] Página oficial de Xen www.xen.org/

[12] Página oficial de KVM www.linux-kvm.org/

[13] Página oficial de PHP www.php.net

[14] Información de comandos de Linux linuxmanpages.com

[15] Página oficial de Gnuplot www.gnuplot.com

[16] Página oficial de Jmeter http://jakarta.apache.org/jmeter/

6. ANEXO I: Intel VT-x ("Vanderpool")

Con los procesadores "VT-x", "Vanderpool", "IVT" o "VMX", Intel introdujo el soporte de virtualización hardware. Con la extensión VMX un procesador siempre opera en uno de los siguientes dos modos:

- Modo root, este comportamiento es muy similar al modo estandard de operación (sin VMX), y éste es el contexto en el que un monitor de maquina virtual (VMM) o hipervisor funciona.
- Modo no root, (o contexto de invitado) está diseñado para funcionar una máquina virtual.

Una característica notable es que los 4 niveles de privilegios (rings) son compatibles en cualquier modo por lo que el software del SO invitado puede teóricamente funcionar en cualquiera de ellos. VT-x define entonces las transiciones de modo root a modo no-root (y viceversa) y los llama "VM entry" (entrada de máquina virtual) y "VM exit" (salida de máquina virtual).

En modo no root, el procesador causará automáticamente salidas de la máquina virtual para ciertas instrucciones privilegiadas y eventos. Para algunas de estas instrucciones, es configurable incluso qué salidas de máquina virtual deberían ocurrir.
El hipervisor en maquinas que no son VT-x, está en el ring 0 del contexto del invitado, por debajo del código de ring-0 del invitado que de hecho se ejecuta en el ring 1. Cuando VT-x está activado, el hipervisor puede estar de forma segura en el ring 0 del contexto del anfitrión y se consigue activar de formar automáticamente por el uso de nuevas salidas de máquinas virtuales.

7. ANEXO II: Scripts

7.1 sacanumerosazar.php y limpianumerosazar.bsh

El script sacanumerosazar.php se encarga de generar los números aleatorios y trabajar con ellos. Se le pasa como parámetro por GET el valor de la semilla. Se ha usado como semilla el 0. Recoge la semilla y como por problemas de espacio solo se van generar 15.000 ficheros, se generan los 15.002 primeros números de la semilla ya que tras probar con 15.000 se observan dos ficheros que superan los 400MB y 500MB (ver figura 22), por lo que esos dos ficheros se borran más tarde de la lista para no generarlos. Tras conseguir un número aleatorio entre 0 y 1, usando la función de distribución de probabilidad de Pareto, se logra el tamaño del fichero correspondiente a dicho número aleatorio. Este script guarda en el fichero *'numerosazar.txt'* por cada línea el tamaño correspondiente a cada número aleatorio. Tras terminar con este fichero procede a llamar al script limpianumerosazar.bsh con parámetro de entrada el fichero generado *'numerosazar.txt'*.

El script limpianumerosazar.bsh se encarga de borrar las líneas vacías que puedan aparecer al principio y final del fichero, y dado que los ficheros aparecen con decimales, eliminar la parte decimal. Se podría optar por redondear a la unidad, las 2 opciones serían correctas.

7.2 graficatamaños.bsh y graficatamaños.dem

Con el fichero de 'numerosazar.txt' ya generado y limpio, se puede ver una gráfica de la distribución de los tamaños. Con el fichero graficatamaños.bsh se pone en el fichero 'numerosazar1.txt' el número de fichero (número de línea) y su tamaño. El archivo graficatamaños.dem está preparado para ser usado con la herramienta gnuplot. Se especifican los datos necesarios para generar la gráfica a través del archivo 'numerosazar1.txt'. La llamada a gnuplot se realiza desde el script graficatamaños.bsh.

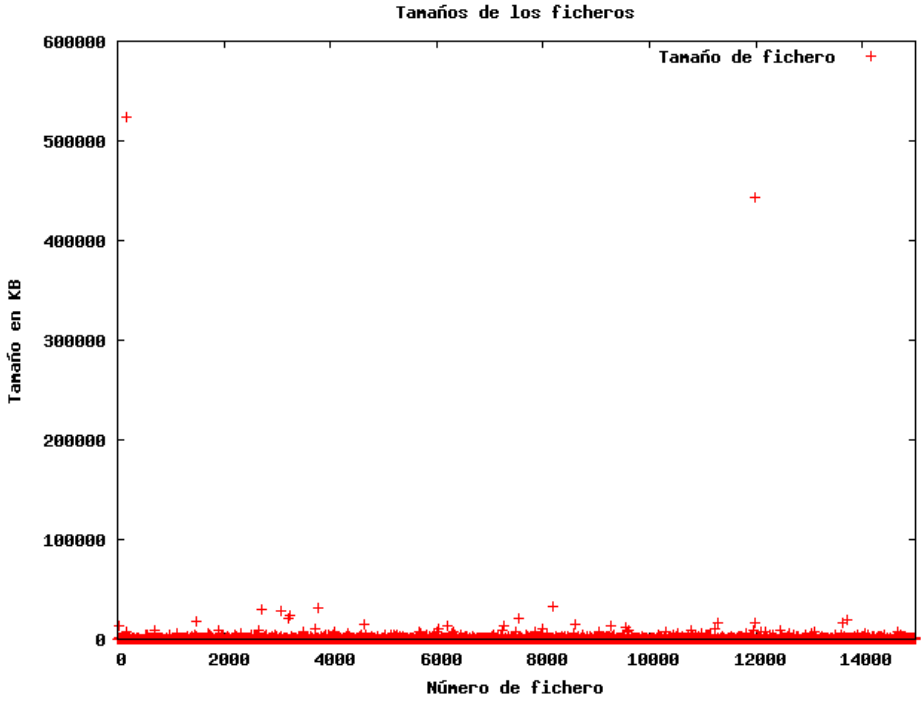

Figura 22: Tamaño de los 15.000 primeros números de la semilla

7.3 sacamedia.bsh

Este script solamente sirve para saber la media de los tamaños de los ficheros que se han generado/se van a generar del fichero 'numerosazar.txt'.

7.4 escribe_nombres_ficheros.bsh

Pasamos como entrada el archivo 'numerosazar.txt'. Este script se dedica a contar las apariciones de cada tamaño y sobre el mismo archivo escribir en cada tamaño el caracter '_' seguido de una numeración que va desde 1 hasta el número de veces que aparece ese tamaño en la lista. Además se le añaden ceros a la izquierda del número para que todas las líneas tengan el mismo número de bytes .He considerado suficiente 13 caracteres para el tamaño,el número que será único por tamaño y el caracter '_'. El número de ceros que añadamos será la diferencia entre 12 y la suma de caracteres del tamaño y del número que se le va a asignar a ese tamaño para identificarlo.
Así un ejemplo de fichero sería: 181_000000057 .

7.5 crea_archivos_numeros_azar.bsh

Dedicado a generar los archivos que se van a pedir. Es necesario introducirle como parámetro de entrada el fichero 'numerosazar.txt' habiendo ejecutado anteriormente escribe_nombres_ficheros.bsh. Automáticamente por cada línea genera un fichero con el tamaño que ha leído (el del ejemplo anterior seria 181 KB) y el nombre de fichero que le asigna es lo que lee de la línea añadiéndole el formato .html, quedando el del ejemplo anterior 181_000000057.html

7.6 borramemoriacompartida.php

Borra la memoria compartida en la dirección 0xff3 y la vuelve a crear escribiendo un 1 para inicializar el contador.

7.7contador_semaforo.php

Realmente será la página que se pida y ella se encarga de redirigirnos a la página que nos "toca". Como el fichero 'numerosazar.txt' está perfectamente estructurado y sabemos que cada 13 bytes hay una línea, entonces la primera línea será desde el byte 0 hasta el 14, la segunda línea desde el 14 hasta el 28,etc.. Así sabiendo el número de línea accederíamos al byte que le corresponde haciendo num_linea*14-14. De ahí leemos 14 bytes y redireccionamos a lo que ha leído añadiendo el formato .html . El semáforo solamente se utiliza para leer y escribir en la zona de memoria compartida y después se libera cuanto antes para que el resto de procesos puedan acceder a la memoria.

7.8 jmeter.php, borraprincipioyfinal.bsh, ordena.bsh, basegnuplot

El archivo jmeter.php es de postprocesado. Una vez se tengan los resultados de una prueba en ficheros (fichero de resultados y fichero de errores), se abre este archivo y en el formulario se introducen los datos de la ubicación de estos ficheros. Llama al script borraprincipioyfinal.bsh para eliminar las primeras y últimas peticiones ya que en ese estado el servidor no tiene suficiente carga.

Tras ello, el script jmeter.php se encarga de calcular el número medio de peticiones por minuto, el tiempo medio de respuesta, tiempo medio por paquete, tamaño medio de ficheros pedidos, porcentaje de errores, etc..

Además genera una gráfica con la evolución del tiempo medio que cuesta llegar a cada paquete. Llama al archivo ordena.bsh ya que se desea ordenar los paquetes por el momento en que se lanzaron en la máquina del Jmeter para así poder hacer la gráfica de cada paquete. Para hacer la gráfica se hace uso de un fichero base , 'basegnuplot' al que en función del número de ficheros que se hayan pedido se le cambian unos datos con sed y así se dejan las xtics de la gráfica correctamente (xtics se llama a cada "tic", separación, en el eje x). En este script php surge un problema en el formulario. Por defecto PHP solo deja subir ficheros de no mas de 2MB. Sin embargo a veces hay que subir ficheros algo más pesados. Este problema se soluciona fácilmente modificando el archivo de configuración de PHP /etc/php5/apache2/php.ini y donde se encuentra la línea con el texto upload_max_filesize 2M se sustituye por 100M por ejemplo y con ello se garantiza no volver a tener dicho error. Si este servidor fuera a ser usado por clientes de Internet no sería conveniente poner un valor tan alto ya que si existiera una sección de subir cualquier fichero y suben ficheros grandes sería muy perjudicial para el espacio en disco duro del servidor.

7.9 50megasv1.php y 50megasv2.php

Estos archivos son los que se han usado para realizar las pruebas de la memoria caché. Con el archivo 50megasv1.php siempre redirige al mismo fichero y con 50megasv2.php cada vez redirige a otro fichero distinto de los 100 diferentes que hay.