Prácticas de la Asignatura de Sistemas, Virtualización y Seguridad

Autores: David Martínez Gómez
Israel Rubio LLarena

Índice de Contenidos

1. Introducción

En la presente memoria se recoge la experiencia realizada en las prácticas de la asignatura de Sistemas, Virtualización y Seguridad del máster de Ingeniería Informática. En ella se pretende realizar una medida del rendimiento real de nuestro ordenador al llevar a cabo tres test de Benchmark: SPEC CPU, SPEC JBB y SPEC Web. Posteriormente se emplearán estos Benchmarks en la máquina virtual con el hipervisor Xen y en su optimización con VMM para poder compararlas. Para terminar, en la última práctica se desplegará un docker sobre la cloud pública de Google y se comparará de nuevo los resultados con los obtenidos en Xen para HVM y PV.

Antes de comenzar, es necesario definir una serie de conceptos que nos facilitarán la comprensión de lo que estamos haciendo:

1.1- Rendimiento de una máquina: el mejor o peor rendimiento de una máquina vendrá dado, principalmente, por dos factores: **latencia** (*latency*: tiempo que tarda en terminar la tarea asignada) y **carga de trabajo asumible** (*throughput*: número de tareas que puede realizar en un determinado tiempo). Estos dos términos son opuestos, por lo que el rendimiento mejora cuanto menor sea la latencia y mayor la carga de trabajo asumible. Para evaluar el rendimiento de un computador podemos utilizar diferentes técnicas:

- Modelos analíticos (matemáticos): útil cuando no se dispone de la máquina, pero tienen limitado su ámbito de utilización por la dificultad de expresar en forma de ecuaciones matemáticas el comportamiento detallado de la máquina y su carga de trabajo. Se utilizan en fases muy tempranas de diseño para realizar estimaciones generales del rendimiento.

- Modelos de simulación (algorítmicos): pueden construirse con mayor precisión, recogiendo especificaciones detalladas de diseño. Sin embargo, estos modelos requieren una gran capacidad computacional cuando se incorporan todos los componentes básicos de la máquina.

- Máquinas reales: se evalúan con el fin de disponer de algún criterio de elección, será necesario disponer de un conjunto de programas representativos de la carga real de trabajo que vaya a tener la máquina, con respecto a los cuales se realicen las medidas (Benchmarks).

1.2- Test Benchmark: el término inglés, que podría ser traducido por "cota", designaba originalmente el punto geográfico de elevación conocida. En informática se usa desde 1975 (fecha de publicación de la obra de Walkowicz[1]), no se traduce y sirve para designar el rendimiento de un proceso, aunque en realidad es el proceso de calcular el tiempo que precisa una máquina para realizar ese proceso y luego compararlo con una referencia documentada. Podemos utilizar diferentes criterios no excluyentes a la hora de clasificar los Benchmarks que se utilizan en la actualidad para evaluar los computadores. Los principales Benchmarks que se utilizan son:

a) *LINPACK*: es una colección de subrutinas Fortran que analizan y resuelven ecuaciones lineales y problemas de mínimos cuadrados. Los sistemas de ecuaciones lineales que contempla utilizan diferentes formas de las matrices: generales, en banda, simétricas, triangulares etc. Lo diseñó Jack Dongarra a finales de los 70 para medir el rendimiento en supercomputadores[2]. Lo componen las siguientes subrutinas:

- matrices generales y en banda.
- matrices definidas positivas.
- matrices en banda definidas positivas.
- matrices indefinidas simétricas.
- matrices triangulares y tridiagonales.
- descomposición de Cholesky.
- descomposición QR.
- actualización de las descomposiciones QR y Cholesky.
- descomposición valores singulares.

b) *TPC* (Transaction Processing Performance Council): es un consorcio de 47 fabricantes de software y hardware entre los que se encuentran Compaq, Digital, IBM, NCR, Sun, Informix, Intel, Microsoft y Oracle dedicado al diseño de Benchmarks para la medida de rendimiento de los sistemas informáticos. Estos Benchmarks analizan la mayoría de los aspectos del sistema en relación con el procesamiento de transacciones, es decir, accesos de consulta y actualización a bases de datos. Tiene el inconveniente de la cantidad de tiempo que requieren las pruebas (meses) haciendo que su coste sea elevado (miles de euros). Los Benchmarks TPC-A, TPC-B, TPC-C y TPC-D no se usan desde 1999, los actuales son:

b.1- TPC-H: es un Benchmark de soporte a la decisión. Consta de un conjunto queries específicas, orientadas a la actividad comercial.

b.2- TPC-R: es similar a TPC-H pero permite optimizaciones adicionales basadas en el conocimiento de las queries.

b.3- TPC-W: Benchmark para medir las transacciones en la WEB. La carga de trabajo se configura en un entorno de comercio que simula las actividades de un servidor de WEB orientado a las transacciones comerciales.

c) *SPEC*[3]: acrónimo de Standard Performance Evaluation Corporation, un consorcio sin ánimo de lucro (aunque entre sus componentes se encuentran diversas empresas privadas de venta de ordenadores) fundado[3] en 1988 en Virginia, Estados Unidos, y que se encarga de crear Benchmark estándar para medir el rendimiento de los ordenadores y publicar los resultados de estos test. Los test Benchmark de SPEC son de pago, están escritos en código C, C# o Fortran y tiene un objetivo real ya que las empresas los emplean para controlar la mejora de los componentes que producen, lo que se conoce con el término "*benchmarketing*". En nuestra práctica emplearemos 3 modelos:

c.1- SPEC CPU 2006: es actualmente la última versión disponible para la medida del rendimiento de la CPU (se han retirado las versiones CPU 92, 95 y 2000), se usa para comparar las cargas de trabajo asumibles por los ordenadores bajo diferentes sistemas. Incluye 29 test:

- CFP2006: aplicaciones intensivas en cálculo numérico con reales. Son 17 test de coma flotante (float point): simulaciones físicas, gráficos 3D, procesadores de imágenes, programas de química computacional, etc.

- CINT2006: aplicaciones en las que domina la aritmética entera, incluyendo procedimientos de búsqueda, operaciones lógicas, etc. Son 12 test de operaciones aritméticas con enteros (integers): compiladores, intérpretes, procesadores de texto, juegos de ajedrez, etc.

c.2- SPEC JBB 2005: la última versión disponible es la 2013 (se han retirado las versiones 2000 y 2005), se evalúa el rendimiento del servidor Java al emular un sistema de tres niveles cliente/servidor. El trabajar con la versión 2005 nos obligará a buscar una versión antigua de Java(1.5). Representa un servidor comercial basado en un sistema de 3 capas: presentación, aplicación y datos (en la cual analizamos la capa intermedia). Emula un Sistema OLTP (OnLine Transaction Processing) que es una base de datos que hace operaciones muy pequeñas de forma continuada.

c.3- SPEC Web 2005: se han retirado las versiones 96, 99, 2005 y 2009. Actualmente no existe ningún SPEC dedicado en exclusiva al estudio web. Consiste en un conjunto de clientes que atacan a un conjunto de servidores web con una base de datos detrás, encapsulada. Son en realidad 3 Benchmarks: una página web de contenido estático, otra de e-commerce (dinámico+estático) y un sistema de banco (el más complejo por precisar SSL).

1.3- Xen[4]: es un hipervisor de código abierto desarrollado por la Universidad de Cambridge a finales de los 90's que permite instancias de uno o varios sistemas operativos diferentes en paralelo dentro de una sola maquina o host. En 2007 fue adquirida por Citrix por 500 millones de dólares.

a) Características:

- Bajo consumo de RAM: Xen utiliza un microkernel diseñado para ser usado con poca cantidad de memoria RAM e interfaz limitada para los Guest, haciéndolo más robusto y seguro que otros Hipervisores.
- Sistema operativo agnóstico: la mayoría de las instalaciones corren con Linux como un conjunto del sistema principal, denominado "dominio cero". Además, soporta sistemas abiertos (como SOLARIS).
- Controlador aislado: Xen tiene la capacidad de alojar el controlador principal de un sistema para ejecutarlo en una máquina virtual. Si el controlador falla o su funcionamiento se ve comprometido, la máquina virtual (VM) contiene un controlador que puede ser reiniciado sin afectar el resto del sistema.
- Paravirtualización: guest completamente paravirtualizados han sido optimizados para correr en una máquina virtual. Esto permite a los invitados ejecutarse mucho más rápido que con una extensión real de hardware (HVM siglas en inglés). Adicionalmente Xen puede correr en hardware que no soporta extensión real de hardware.

b) Tipos:

I. Para Virtualizado (PV): es el modelo más sencillo, todo va paravirtualizado a través de un hiperpost (programa que corre sobre el hardware real para implementar la máquina virtual). Ante toda función que llama al monitor, el hipervisor saca todos los drivers del Dominio 0, no tiene drivers, pero sí gestiona el hardware, aunque no llega a la virtualización híbrida de WM. Se considera "pequeño" porque utiliza unas decenas de hiperllamadas frente a los cientos que debería realizar si estuviera emulando.

II. Virtualización con asistencia de Hardware (HVM): sistema virtualizado, más complejo. Usa las extensiones de virtualización del procesador para ofrecer la capa de virtualización que necesitan las VMs. Los SOs de los dominios no están modificados por lo que requieren de un soporte de emulación hardware, en este caso conocido como QEMU.

III. PVH: híbrido, es el modo más complejo de todos.

1.4- Virtual Machines (VM): una máquina virtual[5] es un duplicado eficiente y aislado de una máquina física que puede ejecutar programas como si fuese una computadora real.

a) Características:

- Duplicado: se comporta de forma idéntica a la máquina real excepto por la existencia de menos recursos disponibles (incluso diferentes entre ejecuciones) y las diferencias de temporización al tratar con dispositivos.
- Aislado: se pueden ejecutar varias VM sin interferencias.
- Eficiente: la VM debería ejecutarse a una velocidad cercana a la real.
- Dependencia: requiere que la mayoría de las instrucciones sean ejecutadas directamente por el hardware.

b) Ventajas:

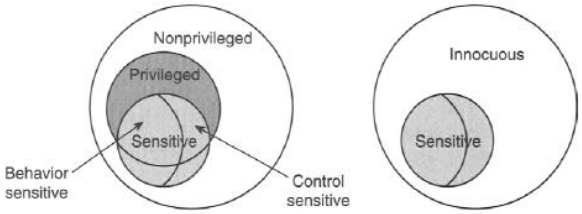
- Rendimiento: consolidación de servidores para usar el hardware más eficientemente.
- Seguridad: al aislar el sistema se permite un acceso muy seguro.
- Multiplataforma: poder ejecutar diferentes sistemas operativos en el mismo ordenador.
- Migración de máquinas virtuales en caliente: con Xen se puede conseguir hacer un balance de cargas sin tiempos muertos, la memoria de la máquina virtual es copiada al destino sin apenas detener su ejecución (una parada muy breve, de 60-300 ms, es necesaria para realizar la sincronización final).

c) Casos de uso práctico:

I. VMWare: proviene de simulador de arquitectura genérico.

II. Desarrollo de contenedores: como alternativa a la portabilidad, uso de Chroot para que el kernel de la VM se comunique más rápidamente con el container.

1.5 - TRAP (trampeo): excepción causada por una condición excepcional que provoca una llamada al kernel antes de devolver el control al proceso original. En las VM se emplea a propósito para impedir que un proceso sea ejecutado por el monitor. Ej.: En las zonas de ejecución fuera de Privileged[5] y que coinciden con Sensitive, monitorizo el código y, al encontrarlo, deben hacer una recompilación binaria; esto es poco frecuente y por eso funciona bien.

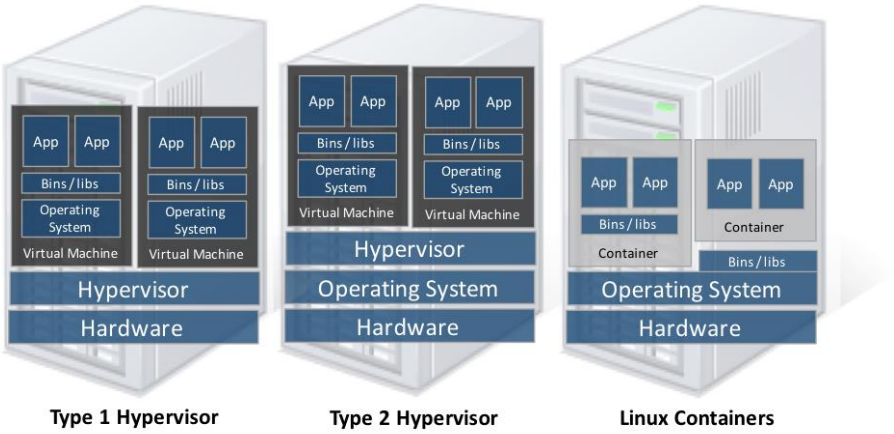


1.6 - Shadow PT (Tabla de páginas en la sombra): las tablas de páginas son una parte integral del Sistema de Memoria Virtual en un sistema operativo y se usa para realizar las traducciones de direcciones de memoria virtual (lógica) a memoria real (física). En general, el sistema operativo mantiene una PT por cada proceso corriendo en el sistema.

En una VM, existe una tabla de páginas en la sombra y, cuando el Guest accede a ella, la Unidad de Manejo de Memoria (MMU) detecta los cambios y la sincroniza con la tabla de páginas real. Para evitarlo, se introduce el Buffer de Traducción Adelantada (Translation Lookaside Buffer o TLB) como mecanismo para evitar el paso por PT.

1.7 - Kernel-based Virtual Machine (KVM) es otro tipo de virtualización completa basado en el uso de hipervisores pero de forma más sencilla que Xen, ya que cada máquina virtual tiene su propio hardware virtualizado: tarjeta de red, disco duro, tarjeta gráfica, etc.

1.8 - Container: tecnología de virtualización en el nivel de sistema operativo que permite que un servidor físico ejecute múltiples instancias de sistemas operativos aislados, conocidos como Servidores Privados Virtuales (SPV o VPS en inglés) o Entornos Virtuales (EV). No provee de una máquina virtual propiamente dicha sino más bien un entorno virtual que tiene su propio espacio de procesos y redes.



1.9 - Docker: es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de Virtualización a nivel de sistema operativo en Linux. Mediante el uso de contenedores, los recursos pueden ser aislados, los servicios restringidos, y se otorga a los procesos la capacidad de tener una visión casi completamente privada del sistema operativo con su propio identificador de espacio de proceso, la estructura del sistema de archivos y las interfaces de red.

2. Realización de las prácticas

2.1 Práctica 1

a) SPEC CPU 2006:

a.1- Pasos previos:

Para esta práctica se usaron los ordenadores del Laboratorio II (puestos 2 y 3) y los ordenadores portátiles personales de los dos alumnos. En un principio se intentó crear una partición de Linux en el portátil, pero, debido a problemas para la configurar la tarjeta de wifi en Debian 5.8 Wheezy en se optó por su instalación en VMWare WorkStation 12 y reducir las prestaciones de la máquina virtual a 2 procesadores con 2 cores cada uno y 8 Gb de RAM.

Para comenzar:

- Se instala en el disco virtual el Debian 5.8 Wheezy descargado de internet.
- Se emplea la configuración inglesa por defecto para que se descarguen los componentes mínimos (configurando la española da problemas por el repositorio).
- Se instalan las VMWare Tools contenidas en la aplicación desde el CD-ROM virtual en el caso del portátil para que no haya problemas con el tamaño de pantalla y red.
- Se instalan C, C# y Fortran: apt-get install gcc g++ gfortran

a.2- Ejecución del Benchmark SPECcpu:

- Descargar archivos SPEC incluidos en la documentación[6] y descomprimir el bz2:
  tar -xf SPECcpu2006.tar.bz2
- Cambiar privilegios de lectura/escritura: chmod -R ug+w p/
- Entrar en config/ y guardar default: cp linux64-amd64-gcc42.cfg default.cfg
- Cambiar ruta de C, C# y Fortran: vi default.cfg
  y buscar la ruta de gcc, g++ y gfortrant y dejar como: usr/bin/g… (elimino lo que sobra)
- Invocar el instalador: ./install.sh
- Sólo aparece una opción: Linux-suse101-AMD64. Aceptamos para continuar (y).
- Elegir la fuente: source shrc
- Compilar: runspec --action=build <**nombre**>
- Correr: runspec --action=run --noreportable --iterations=**x** <**nombre**> --size=<**tipo**>
  - action: tipo de acción: build= compilar y run=correr.
  - noreportable: sinónimo de **-l** , evita error si no está configurado adecuadamente.
  - iteration: nº de veces que se ejecuta, como hubo suficiente tiempo se ejecuta 3 veces y para los resultados se toma la mejor prueba.
  - <**nombre**> : cuál de los benchmarks se ejecuta, con **all** todos a la vez
  - size: selecciono prueba (**test**) o normal (**train**) o referencia (**ref**).

- Se corre una primera vez todos los benchmarks en modo test para ver cuánto duran y elegir uno de la opción FP y otro de INT. En un primer intento no se obtienen datos para 447.dealII ni 481.wrf por un fallo de compilación. Buscando documentación:

447.Deal II: da un fallo del "header", se puede resolver añadiendo un FLAG al fichero de configuración y descargando la librería necesaria: Srcalt = apache_stdcxx_4_2_1

https://www.spec.org/cpu2006/src.alt/README.447.dealII.src.alt.apache_stdcxx_4_2_1.txt

481.wrf: falla en el build, sobra una línea de código: // wrf_data_header_size = 8

https://www.mail-archive.com/gcc-bugs@gcc.gnu.org/msg309327.html

Por falta de tiempo no se pudo realizar la corrección del 447 (el profesor nos aconsejó que no perdiéramos el tiempo en ello). Sí se realizó la corrección del 481.

- Al realizar el primer "run", no se obtienen tampoco datos para *483.xalancbmk* por un fallo de compilación. Se puede solucionar añadiendo el FLAG:

CXXPORTABILITY= -DSPEC_CPU_LINUX -include cstdlib -include cstring

http://cs.mtu.edu/~mmkoksal/blog/?x=entry:entry120124-205001

- La medición se realizó 3 veces por Benchmark y en 2 modos: "test" y "train". Hubiese sido más conveniente hacerlo en modo "ref" pero no pudo realizarse por falta de tiempo (y ya que sólo nos sirve para estimar con qué Benchmark nos quedamos, tampoco era indispensable). Los resultados obtenidos en esta parte fueron:

a) Float Point (FP):

	TEST				TRAIN			
BENCHMARK FP	Secs	Secs	Secs	**T.MEDIO**	Secs	Secs	Secs	**T.MEDIO**
410.bwaves	16,2	13,6	11,3	**13,70**	151	122	123	**132,00**
416.gamess	0,352	0,271	0,27	**0,30**	103	88,3	88	**93,10**
433.milc	9,2	7,62	7,48	**8,10**	14,9	13,1	13,2	**13,73**
434.zeusmp	11,8	9,3	9,08	**10,06**	24,8	20,9	20,9	**22,20**
435.gromacs	1,01	0,835	0,789	**0,88**	87,2	76,4	76,6	**80,07**
436.cactusADM	2,51	2,1	2,01	**2,21**	20,3	17	17,6	**18,30**
437.leslie3d	12	9,09	8,66	**9,92**	61,9	51,7	51,9	**55,17**
444.namd	12,1	9,85	9,84	**10,60**	11,5	9,81	9,82	**10,38**
447.dealII	0	0	0	**0,00**	0	0	0	**0,00**
450.soplex	0,018	0,0145	0,0145	**0,02**	6,08	4,86	4,79	**5,24**
453.povray	**0,533**	**0,387**	**0,383**	**0,43**	**5,53**	**4,72**	**4,51**	**4,92**
454.calculix	0,0415	0,381	0,041	**0,15**	1,31	1,05	1,05	**1,14**
459.GemsFDTD	76,6	1,59	1,47	**26,55**	29,3	25,4	26,1	**26,93**
465.tonto	0,698	0,53	0,533	**0,59**	153	152	153	**152,67**
470.lbm	2,61	2	1,96	**2,19**	26,8	26,9	26,9	**26,87**
481.wrf	0,0988	0,0807	0,0808	**0,09**	0,0767	0,0761	0,0766	**0,08**
482.sphinx3	1,8	1,45	1,44	**1,56**	7,75	7,81	7,86	**7,81**
MEDIA FP	**5,46**				**40,66**			

- Tomamos el valor medio aritmético de las 3 medidas como valor de referencia

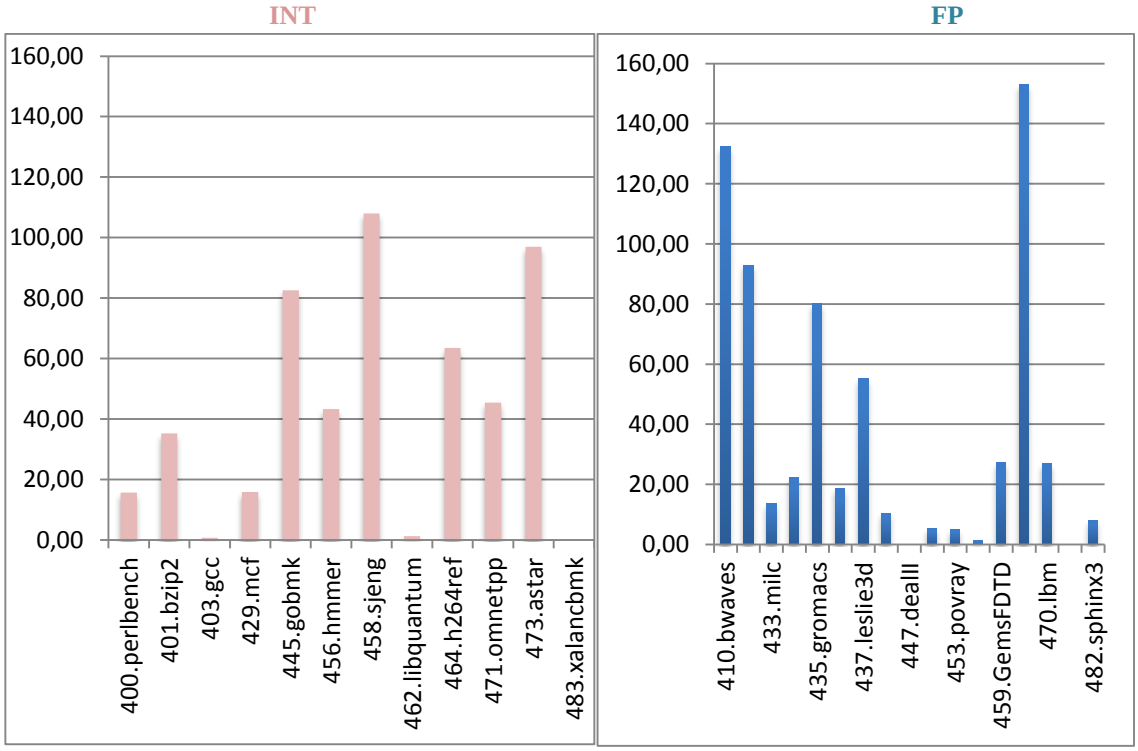
- Hay un valor muy discordante con el resto obtenido en el SPEC 459. Fue debido a que en el momento de realizarse la pantalla del portátil entró en reposo. A raíz de este descubrimiento se desactivó el apagado automático de la pantalla para que no volviera a interferir en las mediciones.

- Por indicación del profesor, de esta parte se eligió el Benchmark: **453.povray.** POV-Ray es un trazador de rayos: es una técnica de renderizado que calcula la imagen de una escena en la que se simula cómo iluminan los rayos de luz las superficies. El usuario puede especificar la localización de la cámara, las fuentes de luz y la posición de los objetos (https://www.spec.org/cpu2006/CFP2006/).

b) Integer (INT):

	TEST				TRAIN			
BENCHMARK INT	Secs	Secs	Secs	**T.MEDIO**	Secs	Secs	Secs	**T.MEDIO**
400.perlbench	3,47	3,59	3,44	**3,50**	16,5	15,3	15,2	**15,67**
401.bzip2	4,88	6,74	4,8	**5,47**	35,3	35,3	35,2	**35,27**
403.gcc	0,971	1,19	0,974	**1,05**	0,702	0,703	0,715	**0,71**
429.mcf	2,9	2,57	2,21	**2,56**	15,8	16	15,7	**15,83**
445.gobmk	15,1	18,9	14,9	**16,30**	82,6	82,4	82,6	**82,53**
456.hmmer	**2,14**	**2,8**	**2,15**	**2,36**	**47,6**	**41,3**	**40,9**	**43,27**
458.sjeng	3,45	4,14	3,16	**3,58**	126	99,1	98,7	**107,93**
462.libquantum	0,0291	0,0439	0,0379	**0,04**	1,53	1,14	1,13	**1,27**
464.h264ref	10,2	14,2	10,3	**11,57**	74,6	58	58	**63,53**
471.omnetpp	0,293	0,377	0,29	**0,32**	53,1	41,5	41,5	**45,37**
473.astar	8,52	9,89	8,19	**8,87**	115	87,8	88	**96,93**
483.xalancbmk	0	0	0	**0,00**	0	0	0	**0,00**
MEDIA INT	**5,06**				**46,21**			

Por indicación del profesor se seleccionó el Benchmark: **456.hmmer.** HMMer es diminutivo de "Profile Hidden Markov Models", consistente en modelos estáticos de multiples secuencias alineadas, empleadas por ejemplo en la secuenciación del ADN (https://www.spec.org/cpu2006/CFP2006/).



Podemos analizar los resultados obtenidos y sacar varias conclusiones:

1. Si nos fijamos en la media de todos los valores, INT es superior a FP con tamaño "train" pero inferior en "test". El comportamiento es justo opuesto en FP. Dado que los valores son de tiempo y en modo "test" algunos son extremadamente pequeños, nos hace sospechar que la opción correcta es la obtenida en modo "train", los cálculos de enteros llevan de media más tiempo que los de coma flotante.

2. Si nos fijamos valor por valor, nuestra máquina da mejores resultados en aplicaciones de enteros (INT) que de coma flotante (FP). El peor tiempo se obtiene para FP en el Benchmark 465 con 152,67s y para INT el peor es el 458 con 107,93s. Los mejores tiempos se obtienen en FP, correspondiente al Benchmark 481 con 0,08s. Entre los Benchmarks INT, el más rápido es el 403 con 0,71s.

a.3- Comparar benchmarks SPECcpu de 32 y 64 bits:

- Cambiamos en el archivo default.cfg el modo de 64 a 32bits:

cp linux64-amd**32**-gcc42.cfg default.cfg

- Reinstalamos las multilibs para recompilar:

apt-get install --reinstall gfortran-4.7-multilib
apt-get install --reinstall gcc-4.7-multilib
apt-get install --reinstall g++-4.7-multilib

- Cambiamos el archivo de configuración por defecto y las rutas:

cp linux64-amd32-gcc42.cfg default32.cfg
Ej. : CC: = /usr/bin/gcc –m32

- Corremos los Benchmarks 453 y 456 con la configuración normal y tamaño "ref":

runspec --action=run --noreportable --iterations=3 --size=ref --tune= base 453 456

BENCHMARK	Secs	Ratio	Secs	Ratio	Secs	Ratio	T.MEDIO	RATIO M.
453.povray	**137**	**38,9**	**136**	**39,1**	**136**	**39,1**	**136,33**	**39,03**

BENCHMARK	Secs	Ratio	Secs	Ratio	Secs	Ratio	T.MEDIO	RATIO M.
456.hmmer	**416**	**22,4**	**413**	**22,6**	**413**	**22,6**	**414,00**	**22,53**

- A continuación, corremos el SPEC con la compilación de 32 bits:

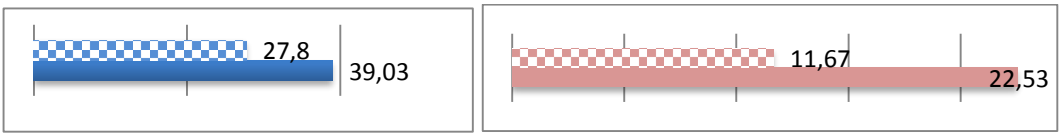
runspec --action=run --noreportable --iterations=3 --size=ref --config=default32.cfg 453 456

BENCHMARK	Secs	Ratio	Secs	Ratio	Secs	Ratio	T.MEDIO	RATIO M.
453.povray	**184**	**28,9**	**208**	**25,6**	**184**	**28,9**	**192,00**	**27,80**

BENCHMARK	Secs	Ratio	Secs	Ratio	Secs	Ratio	T.MEDIO	RATIO M.
456.hmmer	**818**	**11,4**	**860**	**10,9**	**736**	**12,7**	**804,67**	**11,67**

- De este modo podemos comparar los resultados de ratios a 32 y 64 bits para FP e INT: se obtiene mucho mejor ratio en 64-b que en 32-b. Sin embargo, el hecho de que para el INT la mejora sea casi del doble nos hace pensar que los datos no son correctos.

- Si los representamos gráficamente, observamos que en FP la mejora del ratio de 64b a 32b es del 40% pero en INT llega al 93%, lo cual es descabellado. Lo más probable es que, al hacer la práctica en VMWare, se haya producido algún proceso de Windows que ralentizó la máquina al tomar las medidas y los resultados no son fiables.



a.4- Benchmarks mejorando la compilación:

- Por defecto el nivel de mejora de la compilación que hemos usado es el O2. Se podría mejorar usando O3 o utilizando técnicas de GPO: la Compilación Guiada por Profiling es una técnica de optimización que automáticamente realiza una primera ejecución, detectando qué áreas del sistema permiten llevar a cabo una optimización del rendimiento, y pasando esos datos como entrada en una nueva compilación, ya optimizada.

- Para optimizar el SPEC, se modifica de nuevo el fichero de configuración (default2.cfg), añadiendo en el apartado de Optimización los siguiente FLAGS:

PASS1_CFLAGS = -Qprof_gen
PASS1_LDFLAGS = -Qprof_gen
PASS2_CFLAGS = -Qprof_use
PASS2_LDFLAGS = -Qprof_use

- A continuación, corremos el SPEC con la compilación optimizada:

runspec --action=run --noreportable --iterations=3 --size=ref –tune=base --config=default2.cfg 453 456

- Al intentar realizar con tune="base" da error, debemos cambiar a "peak" y, para que dure menos, manteniendo size="ref", modificamos iterations = 1:

FP	ORIGINAL	MEJORADO
Peak Runtime	13	13
Estimated Ratio	39,03	13,3

INT	ORIGINAL	MEJORADO
Peak Runtime	156	155
Estimated Ratio	22,53	5,97

- De nuevo los datos son inconsistentes, la nueva compilación optimizada disminuye el ratio, estaríamos tardando más y teniendo mejor compilación y además, empeora 3 veces en FP y 4 veces en INT. Por tanto, hubiese sido necesario hacer más de una iteración para estar seguro y, además, hacerlo en otro ordenador que no tuviera VMWare porque los datos no son aceptables. Sin embargo, por falta de tiempo ninguna de las dos opciones fue factible.

a.5- Benchmarks dependientes de la memoria y de CPU:

- Se estudian de nuevo el 456 de FP y el 453 de INT.

- Repetimos la ejecución con valores de tune "peak" y empleamos el comando PERF [7]

- Accedemos al path donde se encuentran el conjunto de Benchmarks:

/benchspec/CPU2006/xxx/run/run_base_ref_amd64_m64...xxx

- Para conocer el número de ciclos e instrucciones, abrimos la carpeta de build y ejecuto:

perf stat -e ./xxx rZZZZ > xxx.ref.out

- Los "rZZZZ" se obtienen de la documentación suministrada por el profesor (**64-ia-32-architectures-sw.pdf**), de este modo se obtienen que para las máquinas del laboratorio debo usar: r00C0, r412e, r00C5, r003c

- Como estos valores no funcionan en el portátil personal, no pudimos calcular los IPC ni TPS.

- A través de los resultados de nuestros compañeros, mostrados en las presentaciones, se obtuvo que el trabajo de la CPU es mayor en el Benchmark de FP que en el de INT. Por tanto, **FP es más dependiente de la CPU**. En cuanto a la predicción de saltos el comportamiento es opuesto, por tanto, **INT es más dependiente de la memoria**.

b) SPEC JBB 2005.

Los pasos seguidos en la realización de la práctica fueron:

- Instalar la versión Java JDK 1.5 procedente del repositorio enviado por el profesor (se puede descargar de la página web pero es necesario estar registrado) en /usr/lib/jvm/

- Comprobamos la versión de java con: java –versión

- Nos aparece que la versión activa es la 1.7. Para que se instale la 1.5:

  update-alternatives --install "/usr/bin/java" "java" "/usr/lib/jvm/jdk1.5/bin/java" 1

- Debemos cambiar los privilegios para poder acceder[8]:

  chmod a+x /usr/bin/javaws
  chown -R root:root /usr/lib/jvm/jdk1.5

- Para cambiar entre una versión y otra: update-alternatives --config java

- Descargar el archivo SPEC JBB 2005[9] y descomprimir: tar -xf SPECjbb2005.tar.bz2

- Cambiar privilegios de lectura/escritura: chmod -R ug+w jbb/

- Comprobamos la memoria libre con el comando: free , obteniendo 7 Gb. libres.

- Correr Java: java -Xms<min> -Xmx<max> spec.jbb.JBBmain -propfile SPECjbb.props

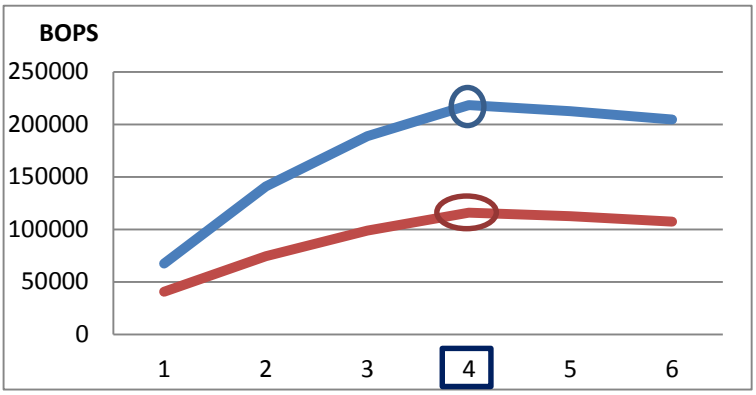
- En nuestro caso empleamos 1000M de memoria (con 256Mb da error en el Warehouse 5 y con 512Mb en el 8), 8 warehouses y los tiempos de ramup y ramdown de 30 segundos y de input.measurement de 240 en SPECjbb.props.

- Correr el benchmark con las dos versiones de java para poder compararlas: source run.sh

- De este modo se obtiene un fichero en el que para cada warehouse se da el número de las transacciones (separadas en New order, Payment, OrderStatus, Delivery, StockLevel y CustReport), que al sumarse y dividirse entre el tiempo de medida (240 segundos) da el throughput en BOPs (bussiness operations per second).

Se obtienen los siguientes resultados:

Warehouses	jdk 1.5	open jdk 1.7
1	40666	67369
2	74441	140749
3	98844	188739
4	115844	218448
5	112397	212553
6	107241	204490
Mejora:		**88,57%**



- Como era de esperar, el punto máximo de BOPs se obtiene en 4 warehouses, coincidiendo con el número de procesadores de la máquina. Si comparamos los BOPs obtenidos en ese punto para la compilación en java 1.5 y con open jdk 1.7, la mejora es de un **88,57%.**

c) SPEC Web 2005.

- Seguimos las indicaciones del profesor para instalar Apache y PHP y configurar los servidores. Comenzamos con "Support".

- Para empezar, ponemos "SIMULTANEOUS_SESSIONS" =100, los RamUp/Down de 30 segundos y RUN de 180 segundos. En la ejecución nos dará un "warning" porque los valores deberían ser otros.

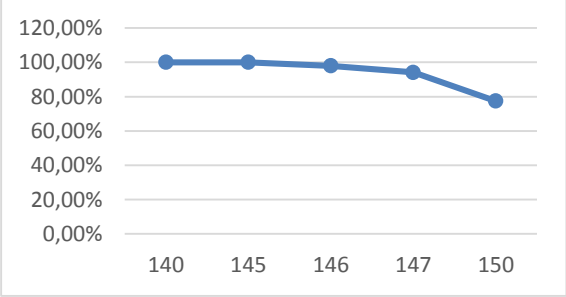
- Comprobamos que el Besim funciona y en una ventana ejecutamos el SpecWebClient.jar, lo dejamos corriendo, y en otra ventana ejecutamos el SpecWeb.jar.

- Cliente: java Xms512m -Xmx512m -jar specwebclient.jar
- Servidor: java -Xms512m -Xmx512m -jar specweb.jar

- De este modo se obtiene que el 100% de peticiones son tolerables, debemos hacer pruebas hasta llegar al 98%. El proceso para obtener el máximo número de sesiones es iterativo: pruebo un valor y si sigue dando 100% lo aumento, sino pruebo con un valor intermedio hasta lograrlo: en nuestro caso lo conseguimos al sexto intento:

100 -> 150 -> 140 -> 147 ->145->146

Sesiones Simultáneas	Tasa soportada Tolerable
100	100,00%
140	100,00%
145	100,00%
146	98,00%
147	94,15%
150	77,46%



- No fuimos capaces de ejecutar la siguiente parte, por falta de tiempo nos quedamos configurando el SSL para que funcionara (daba error de "handshake" porque el certificado lo firmamos nosotros).

2.2 Práctica 2

a) Instalación de Xen.

Los pasos seguidos en la realización de la práctica fueron:

- Aunque en la práctica se dice que se puede necesitar actualizar Kernel del Dom 0 para instalar X, en nuestro caso no fue necesario.

- También se propone como modo de gestión de disco LVM, más eficiente, pero lleva mucho tiempo, por lo que lo en esta parte de la práctica no lo hacemos (aparece más adelante, cuando creemos un disco).

- Comprobamos que la extensión de virtualización está disponible:

  cat /proc/cpuinfo| grep vmx

- Vemos que tiene activada la virtualización con lo que no es necesario hacer nada en la BIOS.

- Buscamos en los repositorios, la instalación de Xen:

  apt-caches earch xen

- Según lo indicado en las prácicas instalamos el hipervisor de xen 4.1:

  apt-get install xen-hypervisor-4.1-amd64

- Configuramos la red en el demonio de Xen, la configuración está en el archivo /etc/xen/xend-config.sxp y basta con activar la linea comentada para Network Bridge

- Reiniciamos y en el arranque de Grub elegimos arrancar, XEN 4.1

- Para cambiar el Grub y hacer que por defecto arranque en Xen, es necesario entrar en el directorio */etc/default/grub* y cambiar la opción por defecto (la primera 0, prueba de fallos 1 y Xen 2):

  GRUB_DEFAULT=2

- Para aplicar el cambio hay que actualizar el Grub:

  update-grub

- Debemos cambiar los parámetros de la CPU y memoria para nuestra Dom0: tamaño de memoria mínima a 1Gb y máxima a 2Gb y 1 CPU:

  GRUB_CMDLINE_XEN_DEFAULT = “dom0_mem=1024M,max:2048Mdom0_max_vcpus=1”

- Reiniciamos y comprobamos que se ejecuta. Además, podemos usar el comando para asegurarnos que no se ha producido ningún error con Xen:

  dmesg | grep xen

- Con esto tenemos instalado Xen, podemos lanzar el siguiente comando para asegurarnos que funciona.

  xm list

b) Creación de la máquina virtual (DomU).

Los pasos seguidos en la realización de la práctica fueron:

- Creamos un fichero monolítico desde la Dom0 que nos servirá como disco de la DomU:

  dd if=/dev/zero of=manual.img bs=1G count=4
  mkfs.ext4 manual.img
  mount manual.img /mnt/
  debootstrap –-arch amd64 wheezy /mnt/ http://ftp.es.debian.org/debian/

- Instalar xen-tools (necesario hacerlo como root, comando: *su* e introduzco la contraseña root):

  apt-get install xen-tools

- Crear el domU con la dirección ip y máscara de sub-red obtenidas y espero a que se instale:

  xen-create-image --hostname=prueba

- Modificamos el archivo CFG , nombre de la MV, configuración del disco para apuntar a el fichero de imágen, etc. Creamos una nueva DomU a partir del .cfg (la opción "-c" muestra la consola):

  xmcreate -c /etc/xen/prueba.cfg

- Una vez arrancada la consola de la máquina virtual, nos damos cuenta que no tenemos la contraseña de root, así salimos con CTR+5.

- Abrimos los log de Xen-tools, concretamente el archivo "prueba.log" (situado en /var/log/xen-tools/) y lo examinamos en busca de los datos de root por defecto:

  - **hostnameprueba**: root
  - **rootpass**: c6jdRkhA

- Buscamos nuestra DomU en la lista:

  xm list

- Como está arrancada abrimos la consola de la DomU:

  xm console my_weezy

- Cambiar la configuración de red para que se obtenga desde DHCP en:

  /etc/network/interfaces

- Vamos a utilizar un volumen lógico LVM como disco, en vez del fichero de imagen, para eso instalamos primero LVM:

  apt-get install lvm2

- Y lo creamos:

  pvcreate /dev/sdb /dev/sdx
  vgcreate vol1 /dev
  lvcreate –L5Gi -n base vol1

- Transferimos la imagen que habíamos creado con el comando:

dd if=/etc/xen/pruba.img of=/dev/vol1/base

- Creamos un duplicado del cfg y hacemos que apunte al nuevo volumen lógico:

cp prueba.cfg pruebalvm.cfg

root = '/dev/xvda1 ro'
disk = ['phy:vol1/base,xvda1,w',]

- Iniciaremos como siempre:

xm create -c pruebalvm.cfg

- Una vez creado el domU se puede crear una imagen del estado actual, denominado "snapshot" [11].
Para crear un snapshot de 5GB (-L5Gi) y basta con darle un nombre "clon1" y el volumen lógico que queremos "clonar"

lvcreate -L5Gi -s -n clon1 /dev/vol1/base

- Para arrancar esta nueva domU desde este snapshot, copiamos el fichero de /etc/xen de "pruebalvm.cfg" a "clon1.cfg" y lo modificamos para que apunte al nuevo volumen lógico y cambiamos el nombre:

cp pruevalvm.cfg clon1.cfg
vi clon1.cfg

disk = ['phy: vol1/clon1,xvda1,w',]
name ='clon1':

- Podemos hacer una copia de seguridad en un fichero comprimido:

dd if=/dev/vol1/clon1 | gzip>clon1.img.gz

- Y recuperar la copia con tan solo descomprimirla:

gunzip clon1.img.gz

- Repetimos el proceso y obtenemos "clon2" con su clon2.cfg que ahora apuntará a:

disk = ['file: clon1.img,xvda1,w',]

- También podríamos volcar la copia a un volumen lógico como se ha visto anteriormente mediante el comando dd.

- Para restaurar desde el snapshot:

lvconvert --merge /dev/vol/clon1 -lv

- Para arrancar el domU desde el snapshot:

xm create -c clon1.cfg

- Si queremos eliminar el snapshot:

lvremove /dev/vol/clon1

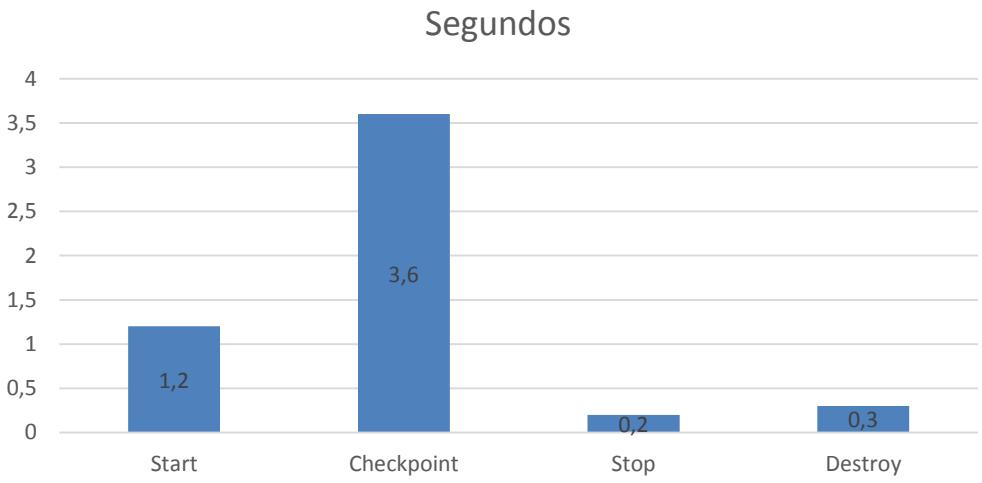
c) Gestión del DomU.

c.1- XenStore:

- Todas las máquinas virtuales han sido creadas y gestionadas de forma manual, para que Xen se encargue de ellas basta con añadirlas al XenStore con el siguiente comando.

xm new manual.cfg

- Empleando el comando "time" seguido de la orden obtengo el tiempo de ejecución:



- Los tiempos de ejecución obtenidos en la mayoría de los casos son de una décima de segundo, sólo en el caso de iniciar el dominio (xm start) el tiempo es de 1.2 segundos, diez veces superior, y de guardar en un punto (xm checkpoint) que el tiempo es de 3.6 segundos, 30 veces superior. Por tanto, es esta última la operación más costosa.

c.2- PVGRUB:

- El PVGRUB permite a las DomU gestionar su propio kernel dentro de la propia domU. Lo vamos a descargar a través de un repositorio backport:

deb http://http.debian.net/debian wheezy-backports main

- Actualizamos e instalamos el kernel 1.36 de Linux:

apt-get -t wheezy-backports install linuximage-3.16.0-0.bpo.4-amd64

- Modificamos los siguientes apartados de la configuración de la domU:

kernel = '/boot/pv-grub-x86_64.gz'
extra = '(hd0)/boot/grub/prueba.cfg'
vcpus = '1'
memory = '256'
disk = ['phy:vol1/base,xvda1,w',]

- Creamos el fichero /boot/grub/grub.cfg y añadimos lo siguiente:

default 0
timeout 5
title "vm-linux 3.16"
root (hd0)
kernel /boot/vmlinuz-3.16.0-0.bpo.4-amd64
root=/dev/xvda1 ro
initrd /boot/initrd.img-3.16.0-0.bpo.4-amd64

c.3- SPEC cpu 2006:

- Para comprobar el rendimiento ejecutaremos a la vez el SPECcpu en los 3 domU creados: prueba, clon1 y clon 2.

Configuración	VM 1	VM 2	VM 3
1vcpu/1024/igual peso	154	154	154
1vcpu/1024/igual peso	423	423	423
4vcpu/1024/igual peso	154	154	154
4vcpu/1024/256-256-32	154	154	154
4vcpu/1024/4096-256-32	423	423	423
4vcpu/2048/4096-256-32	154	154	154
4vcpu/2048/4096-256-32	153	153	309

- Se puede ver que los dos Benchmarks dan consistentemente los mismos resultados a pesar de los cambios de planificación, por lo que parece que corren solo en un core y por eso no compiten.

- En la última prueba se colocó el CAP a 50 (medio core) en la VM3 para probar que la planificación se estaba haciendo bien.

- Cuando realizamos el benchmark 445.gobmk con los 3 domU se obtienen los siguientes tiempos:

  - domUprueba.cfg = 1 minuto 34 segundos
  - domU clon1.cfg = 1 minuto 30 segundos
  - dom2 clon2.cfg = 1 minuto 29 segundos

- Los tiempos son muy parecidos por lo que no podemos hacer distinciones. Por eso, es aconsejable aumentar el peso del dominio “base” 16 veces:

xmsched-credit -d manual -w 4096

- De este modo se obtiene:

  - domUmanual.cfg = 1 minuto 28 segundos
  - domU clon1.cfg = 2 minutos 6 segundos
  - dom2 clon2.cfg = 2 minuto 14 segundos

- Por lo tanto, la que más tarda es la que se obtiene haciendo de la imagen del snapshot (clon2).

2.3 Práctica 3

a) Instalar HVM.

- Primero comprobamos si la virtualización hardware está habilitada o no:

```
xm dmesg | grep -i hvm     ->  (XEN) HVM: ASIDs enabled.
                               (XEN) HVM: VMX enabled
                               (XEN) HVM: Hardware Assisted Paging (HAP) detected
                               (XEN) HVM: HAP page sizes: 4kB, 2MB
```

- Para instalar HVM es necesario el netinst de debian 7.9[13] y crear un fichero de configuración:

```
cp  /etc/xen/manual.cfg    /etc/xen/hvm.cfg
```

- Se configurará la opción bridge por lo que la MV tendrá asignada la ip 192.168.0.5 al bridge xenbr0 Posteriormente, dentro de la máquina se configurará esta misma IP.

- Creo un volumen lógico de 15Gb como hicimos en la práctica anterior (dev/vol3/hvm) y creo la máquina virtual:

```
xm create -c /etc/xen/hvm.cfg
```

- Para poder visualizar la pantalla del domU para instalar debian se accede con el comando:

```
vncviewer 127.0.0.1:0.[idDomU].
```

- Después se realiza una instalación de Debian como en la práctica anterior.

- Configuramos la consola usando grub2, en el fichero de configuración HVM se añade la opción:

```
serial=’pty’
```

- Una vez iniciado el domU, edito el fichero /etc/inittab:

- *comentar* la línea donde aparece: ...:2345
- *descomentar* la línea de la consola serie: sT0:23:respawn:/sbin/getty -L ttyS0 38400 vt100.

- Editamos el fichero de configuración del grub /etc/default/grub:

```
GRUB_CMDLINE_LINUX_DEFAULT=’ console = tty0 console=ttyS0,38400n8’
GRUB_TERMINAL=serial
GRUB_SERIAL_COMMAND="serial -speed=38400 -unit=0 -word=8 -parity=no -stop=1"
```

- Por último, es necesario reconfigurar la red en el domU añadiendo la correspondiente configuración en el fichero /etc/network/interfaces y reiniciando el servicio de networking.

- Este mismo domU nos servirá para PV, sólo tenemos que cambiar el archivo de configuración “pv.cfg” con los cambios que y elegirlo al arrancar:

```
xm create –c pv.cfg
```

- NUNCA debemos arrancar los dos dominios a la vez: debemos cerciorarnos de apagar correctamente uno antes de iniciar otro.

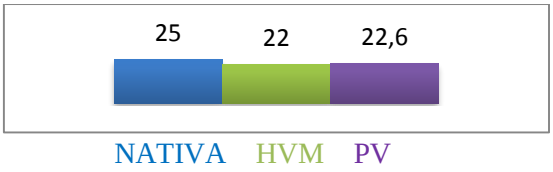
b) Comparación de rendimiento entre máquina virtual y sistema nativo.

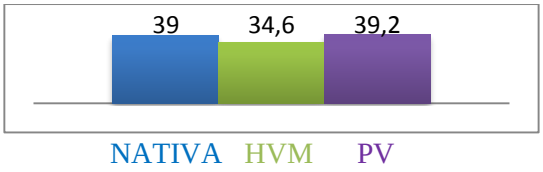
b.1 - SPEC cpu 2006:

- Se ejecutan los benchmarks 453 y 456 de SPEC CPU empleando size="ref" y 1 iteración (no nos dio tiempo a realizarlo 3 veces) para una máquina con PV, otra con HVM y se comparan con lo obtenido en nativo:

456.hmmer	T.MEDIO	RATIO	%
NATIVO	376	25	100,00%
HVM	423	22	88,89%
PV	413	23	91,04%

453.povray	T.MEDIO	RATIO	%
NATIVO	137	39	100,00%
HVM	154	35	89,18%
PV	152	39	90,35%





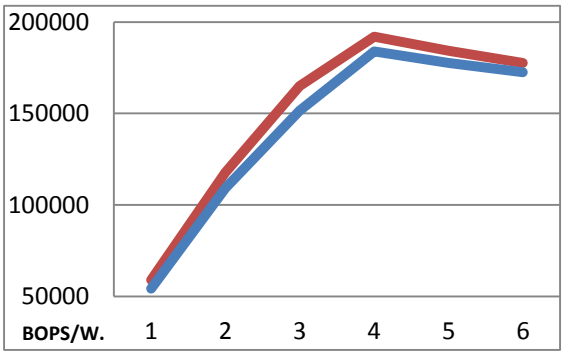
- Los mejores resultados se obtienen para PV frente a HVM pero por muy poca diferencia. En HVM se pierde casi un 11% frente a nativo y en PV alrededor del 9%.

b.2 - SPEC jbb 2005:

- Se instala SPECjbb2005 y se ejecuta primero en HVM y luego en PV. Los resultados obtenidos son:

W. PV	jdk 1.7
1	54211
2	109170
3	151526
4	184005
5	177647
6	172575
NIVEL:	84,23%

W. HVM	jdk 1.7
1	59002
2	117802
3	165047
4	191935
5	184317
6	177593
NIVEL:	87,86%



- Como los mejores resultados se obtienen para HVM, clonaremos este domU y lo ejecutaremos simultáneamente en los clones HVM1 y HVM2 (realizamos la prueba con HVM y HVM1 pero vimos que las modificaciones en el original afectaban a la copia por lo que realizamos un segundo clon).

- La práctica se realiza en el PC de clase (puesto nº2) con openjdk 1.7. Se debería además recompilar con este jdk, aunque en nuestro caso no lo hicimos por falta de tiempo.

- Se deben modificar los "pesos" para alcanzar un 75% respecto a los valores del sistema nativo. Probamos con poner 5 veces mayor uno que otro (768 para HVM1 y 128 para HVM2). Los resultados obtenidos de este modo son:

Warehouses	jdk 1.7
3	145574
4	171015
RESULTADO:	78,29%

- Como la práctica nos pedía que fuera alrededor del 80% de los BOPs obtenidos en nativo, tomamos este dato como bueno y no realizamos más pruebas.

b.3 - SPEC web 2005:

- No fuimos capaces de realizar esta parte a tiempo por lo que no se muestran resultados.

2.4 Práctica 4

a) Containers LXC vs Xen.

- Lo primero es instalar el container

- Descargamos el Kernel de backports
- Configurar cgroups en /etc/fstab
- Crear container con lxc-créate
- Arrancarlo con lxc-start
- Entrar con lxc-attach
- Crear red virtual (virsh)
- Cambiar configuración /var/lib/lxc
- Instalar SPECJBB en el container

- Clonamos el container creado, que se queda como base, en otros dos que se utilizarán para el test

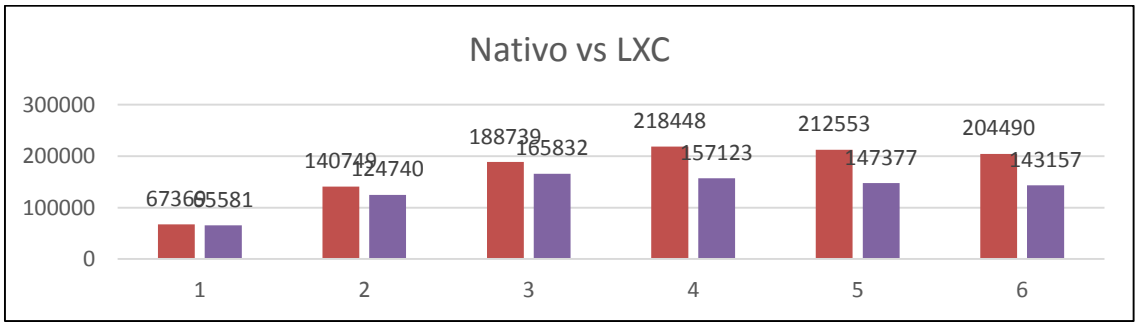
lxc-clone -B aufs -s base my_clon

- Utilizando cgroups podemos planificar un conjunto de procesos, en nuestro caso la configuración para nuestro contenedor estará en:

/sys/fs/cgroup/lxc/my_contenedor/

- Arrancamos el clonado del contenedor y ejecutamos los dos.

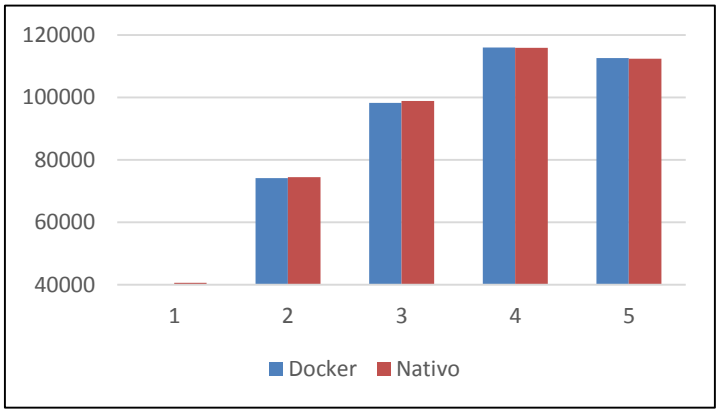
Planificamos Contenedor A: 1-3 Cpus Contenedor B: 1Cpu (74,5%)



b) Dockers.

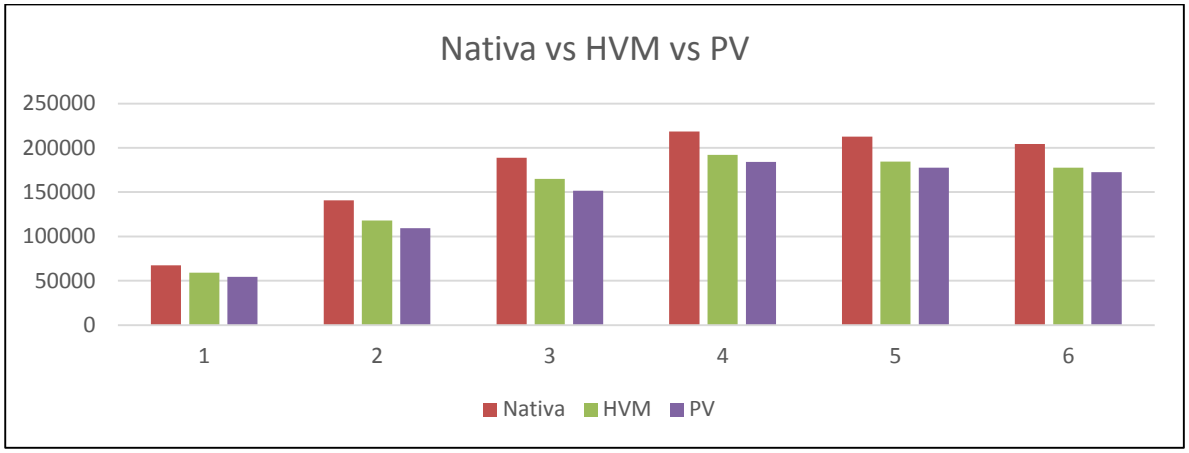
- Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización.

Warehouses	Docker	Nativo
1	40188	40666
2	74171	74441
3	98217	98844
4	116004	115844
5	112595	112397



- Comparando los datos obtenidos en nativo, HVM y PV:

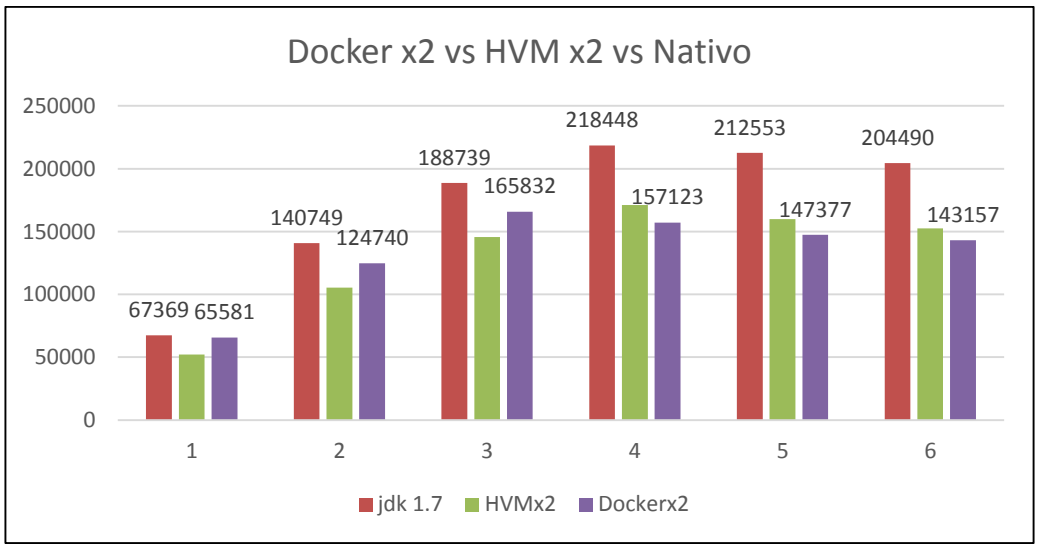
Warehouses	Nativa	HVM	PV
1	67369	59002	54211
2	140749	117802	109170
3	188739	165047	151526
4	218448	191935	184005
5	212553	184317	177647
6	204490	177593	172575



- Creamos un clonado del contenedor y ejecutamos los dos.

Planificamos Contenedor A: 1-3 Cpus Contenedor B: 1Cpu (74,49%)
HVM A 768 HVM B: 128 (78,29%)

Warehouses	jdk 1.7	HVMx2	Dockerx2
1	67369	52270	65581
2	140749	105437	124740
3	188739	145574	165832
4	218448	171015	157123
5	212553	159772	147377
6	204490	152463	143157



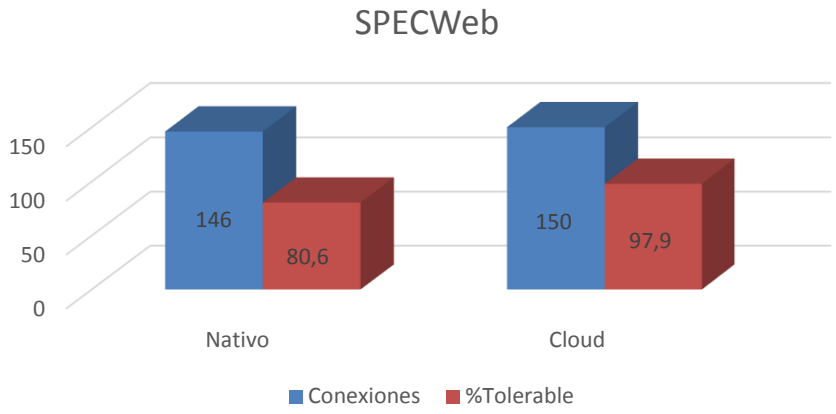
- Como se ve en los resultados, el rendimiento en pico es peor que en HVM, lo que no tendría sentido, pero tiene fácil explicación, la planificación es más favorable a la máquina que se mide en el caso de HVM, si midieramos en conjunto el rendiminto de AyB en el contenedor sería superior.

- Se ha incluido este ejemplo para ver lo fácil que se pierde la ventaja que nos da un contenedor en cuanto a overhead, cuando compartimos servidor con otras máquinas o la configuración de la planificación no es óptima.

c) Google Cloud.

Por falta de tiempo no se han podido finalizar las prácticas con Docker, por eso se ha decidido instalar en el SpecWeb en las instancias de google sin utilizar contenedores.

- Creamos una Instancia de máquina virtual.
- Instalamos y configuramos SPECWeb.
- Probamos en local y obtenemos:

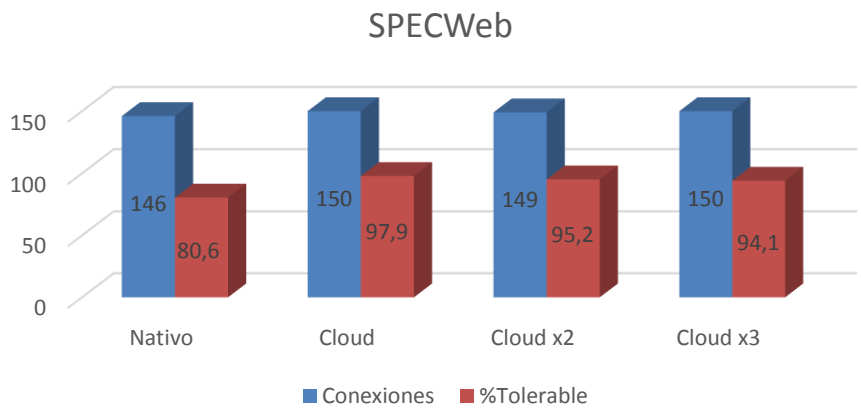


- Ahora que funciona creamos una captura (snapshot) de la VM

- Creamos Otras dos Máquinas desde el SnapShot creado y las reconfiguramos

- Las ponemos IP estática para la LAN y reconfiguramos los ficheros de configuración del specWeb para que apunten a la IP del Webserver y la de Besim a la base de datos.
- Como se ve en esta prueba todas las máquinas se han creado con 1 vCPU y 3,75 GB de RAM, el disco duro es estándar de 20GB

- Las diferentes ejecuciones del SPEC Web nos dan:

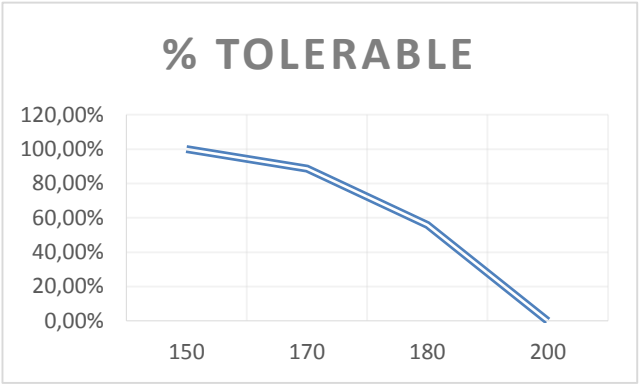


- Los resultados se obtienen para:

  - Cloud lanzando la prueba en una sola máquina, que lanza los clientes, el webserver y la base de datos,
  - Cloud x2 separa la máquina que lanza los clientes del webserver+BBDD
  - Cloud x3, una máquina para los clientes, otra para el Webserver y la tercera para BBDD

- Repetimos con 3 máquinas ahora subiendo a **4 cores y 15GB** de RAM:

Sesiones Simultáneas	% Good	% Tolerable
150	100,00%	100,00%
170	88,70%	99,90%
180	55,80%	77,51%
200	26.2%	28,80%



d) Monitorización de instancias en Google Cloud

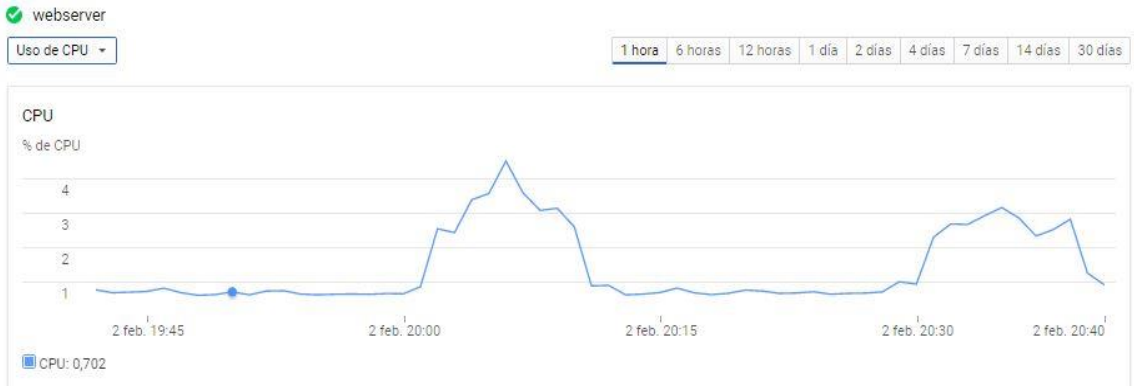
- Se han utilizado las herramientas de monitorización que ofrece google Cloud para ver la reacción de las tres instancias que se han utilizado durante las últimas pruebas.

- Adjuntamos pantallazos donde se pueden ver las reacciones a dos de los test realizados.

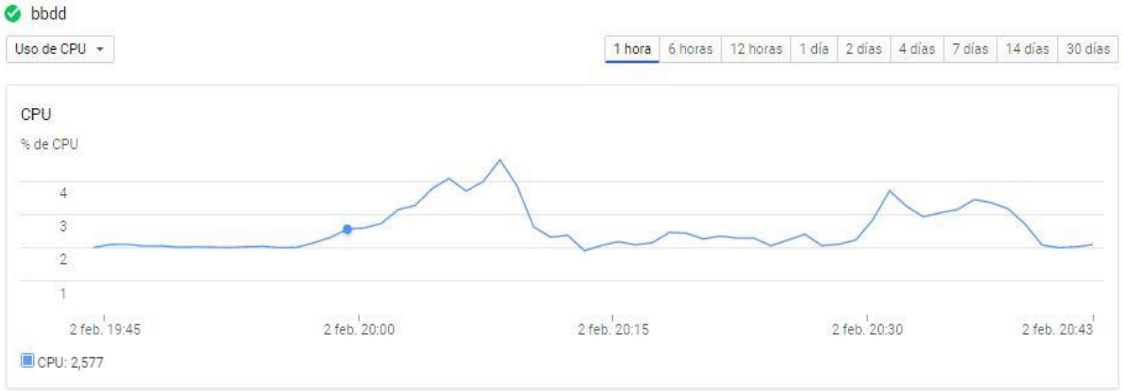
d.1- CPU

- Se puede ver el pico de CPU en la instancia que va a lanzar los clientes durante la preparación de las pruebas, y luego dos pruebas casi consecutivas:

- la primera con 200 conexiones simultáneas,

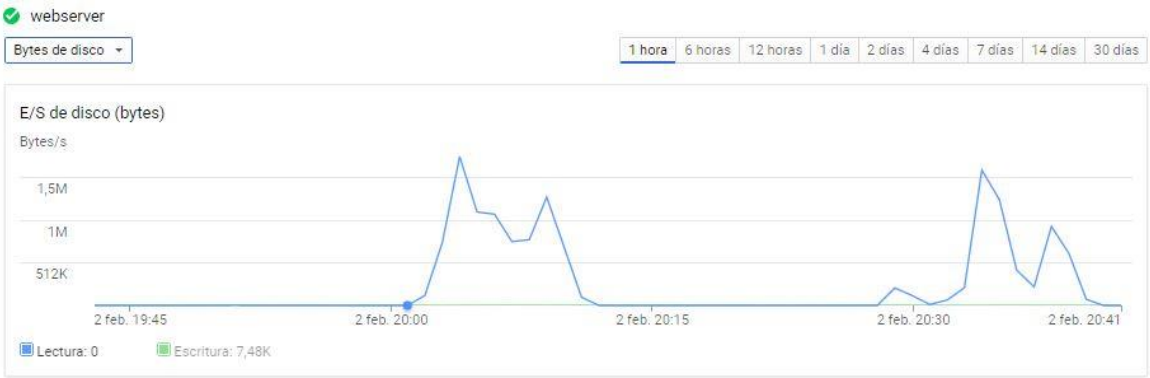


- y la segunda con 150 conexiones simultaneas.



d.2 – DISCO

- Se puede ver el pico de acceso a disco de la instancia de pruebas que se corresponde con la edición de los ficheros de configuración y la preparación para el lanzado de clientes, a partir de ese momento no se accede a disco en esa instancia.

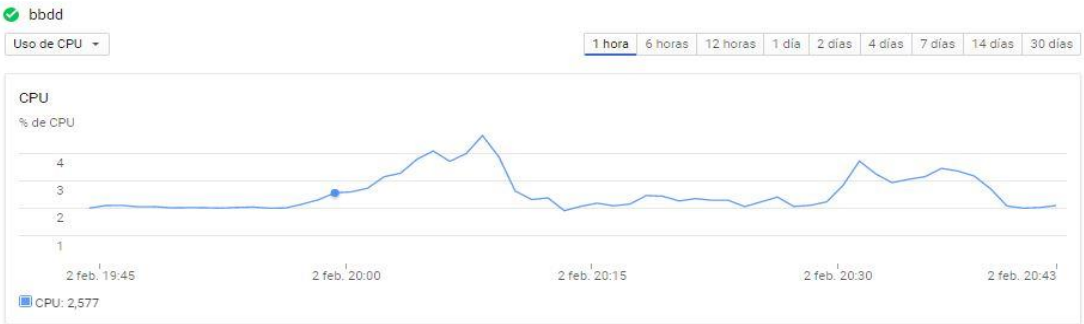


- Se puede ver como en la primera prueba, la de 200 conexiones, el acceso a disco se mantiene casi constante por encima de 1M mientras que en la de 150 apenas supera esa marca en el pico inicial, y luego se mantiene por debajo. Parece una diferencia sustancial en acceso a disco, sobre todo porque parece ser uno de los límites de nuestras instancias, durante todas las pruebas no se han visto valores más altos.

- Así que las siguientes pruebas deberían ir encaminadas a comprobar si efectivamente el factor limitante en este caso es el acceso a disco.

d.3- RED

- En este caso no se aprecian diferencias sustanciales entre el tráfico de red en ambos casos que justifiquen la diferencia de %Tolerable en las pruebas, con lo que en principio suponemos que la red no tiene nada que ver en la caída de rendimiento don 200 conexiones concurrentes.



3. Conclusiones de las prácticas

3.1 Práctica 1

- El uso de Benchmarks nos permite comparar el rendimiento de distintas máquinas al ejecutar en todas ellas la misma carga de trabajo. En la carpeta "Nativos" que se encuentra en GitHub se incluyen copia de todos los ficheros modificados o creados y de los informes obtenidos en la realización de los SPEC CPU de esta parte.

- El SPEC CPU resulta fácil de preparar, pero es largo ya que se deben hacer muchas pruebas, algunas que pueden llegar a durar horas.

- El SPEC JBB resulta más fácil una vez resuelto el problema de instalar Java 1.5, que fue lo que nos dio más problemas.

- El SPEC Web resulta el más difícil de todos, incluso siguiendo las instrucciones daba multitud de errores y sólo para la prueba más sencilla es necesario "tener suerte" a la hora de tomar los valores de sesiones simultáneas porque nunca se sabe por qué valor empezar y cuánto aumentarlo después.

3.2 Práctica 2

- Como esta parte fue prácticamente seguir el guion de la práctica y probar los comandos, no tenemos nada que comentar.

3.3 Práctica 3

- Realizada la práctica anterior, la creación del domU de HVM es sencilla.

- Sí tuvimos ciertos problemas a la hora de arrancar, pero una vez solucionados, el dominio de PV se creó de forma inmediata y las mediciones fueron algo tediosas por el tiempo necesario en prepararlas, en especial varias máquinas a la vez

3.4 Práctica 4

- Hemos podido comprobar en la práctica que la utilización de contenedores es mucho menos laboriosa que la implementación de un sistema de vitalización tipo Xen, produciendo además un overhead mucho menor.

- También hemos visto que al tener varios contenedores ejecutándose de forma concurrente es fácil una mala configuración nos haga perder ese rendimiento que habíamos ganado en una de las máquinas.

- No hemos podido terminar esta parte de las prácticas, básicamente Docker es una abstracción por encima de LXC que nos facilita algunas tareas como backup de máquinas en control de versiones. Una especie de GIT para contenedores. Por eso no es de extrañar que los resultados obtenidos sean casi idénticos a LXC, y que el overhead sea mucho menor que los resultados obtenidos con PV y HVM

4. Bibliografía

[1] "Benchmarking and workload definition" Josephine L. Walkowicz.

[2] Top500: https://www.top500.org/lists/2016/11/

TOP 10 Sites for November 2016

For more information about the sites and systems in the list, click on the links or view the complete list.

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
2	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
3	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
4	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
5	DOE/SC/LBNL/NERSC United States	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939

[3] SPEC:

- Web: http://www.spec.org/
- "The Benchmark Handbook for Database and Transaction Systems" Jim Gray. Capítulo 9: Overview of the SPEC Benchmarks (Kaivalya M. Dixit).

[4] Información sobre Xen: https://wiki.debian.org/es/Xen

[5] "Virtual Machines" J.E.Smith, R.Navir.

- Capítulo 1: historia de las máquinas virtuales (pág. 16)
- Capítulo 8: VM (pág. 381)
- Capítulo 8: instrucciones privilegiadas, figura 8.6 (pág. 385)

[6] Documentación SPEC CPU 2006: https://www.spec.org/cpu2006/Docs/

[7] Tutorial perf: https://perf.wiki.kernel.org/index.php/Tutorial

[8] Cambiar jdk: http://askubuntu.com/questions/522523/how-to-install-java-1-5-jdk-in-ubuntu

[9] Manual de SPEC Jbb: https://www.spec.org/jbb2005/docs/UserGuide.html

[10] Manual de SPEC web: https://www.spec.org/web2005/docs/users_guide.html

[11] Operación de clonado: http://www.tutonics.com/2012/12/lvm-guide-part-2-snapshots.html

[12] Instalación PVGRUB: http://www.atc.unican.es/%7evpuente/SOA/pv-grub-x86_64.gz

[13] Netinst de Debian 7: https://www.debian.org/releases/wheezy/debian-installer/