Linux Benchmarking CÓMO

por André D. Balsa, *andrewbalsa@usa.net* <mailto:andrewbalsa@usa.net>
Traducido por: Joaquín Cuenca Abela, *jcuenca@patan.eleinf.uv.es*
v0.12, 15 de Agosto de 1997

El Linux Benchmarking CÓMO trata sobre algunos aspectos asociados con el *benchmarking* en los sistemas Linux y presentas unas herramientas (algo toscas) para realizar medidas del rendimiento de su sistema, que podría proporcionar una cantidad significativa de información en un par de horas. Quizás también ayude a hacer que disminuya el número de artículos sobre el tema que se envían a comp.os.linux.hardware...

Contents

1 Introducción

"Lo que no podemos decir acerca de nosotros mismos debería desaparecer en el silencio."

Ludwig Wittgenstein (1889-1951), filósofo austríaco

Benchmarking significa **medir** la velocidad con la que un ordenador ejecuta una tarea, de forma que se puedan realizar comparaciones entre diferentes combinaciones de programas/componentes. Esta definición **no** tiene en cuenta la sencillez de utilización, estética o ergonomía o cualquier otro tipo de juicio subjetivo.

El *Benchmarking* es una tarea repetitiva, tediosa, y hay que llevar cuidado con los detalles. Es muy normal que los resultados no sean los que uno espera y que estén sujetos a interpretación (puede que hoy en día ésta sea la parte más importante).

Para finalizar, el *benchmarking* trata con hechos y datos, no con opiniones ni aproximaciones.

1.1 ¿Por qué el *benchmarking* es tan importante?

Aparte de las razones apuntadas en el BogoMips Mini-CÓMO (sección 8, párrafo 2), podemos tener que ceñirnos a un presupuesto o satisfacer unas necesidades de rendimiento mientras instalamos un sistema Linux. En otras palabras, cuando tenemos que hacernos las siguientes preguntas:

- ¿Cómo puedo maximizar el rendimiento con un presupuesto dado?
- ¿Cómo puedo minizar costes manteniendo un nivel mínimo en el rendimiento?
- ¿Cómo puedo obtener la mejor relación calidad/precio (con un presupuesto o unas exigencias dadas)?

Tendremos que examinar, comparar o crear *benchmarks*. Minimizar costes sin tener que mantener un rendimiento en particular implica utilizar una máquina con partes desfasadas (aquel viejo 386SX-16 que está tirado en el garaje podría servir) y no necesita *bechmarks*, y maximizar el rendimiento sin que importe el dinero no es una situación muy realista (a menos que quiera poner un Cray en su casa - la unidad de alimentación recubierta con cuero es bonita, ¿verdad?).

El *benchmarking* de por si no tiene sentido, y es una estúpida pérdida de tiempo y dinero; solo tiene sentido como una parte de un proceso, esto es, si tiene que hacer una elección entre dos o más alternativas.

Normalmente otro parámetro a tener en cuenta en el proceso de decisión es el **coste**, pero también la disponibilidad, el servicio, la seguridad, estrategia o cualquier otra característica medible y racional que tenga que ver con un ordenador. Por ejemplo, cuando comparamos el rendimiento de diferentes versiones del núcleo de Linux, la **estabilidad** suele ser más importante que la velocidad.

1.2 Consideraciones no válidas en la medida del rendimiento

Se pueden leer muy amenudo en los grupos de noticias y las listas de correo, pero aun así:

1. Reputación del fabricante (no se puede medir y es insensato).
2. Cuota de mercado del fabricante (insensato e irrelevante).
3. Parámetros irracionales (por ejemplo, supersticiones o prejuicios: ¿Compraría un procesador que se llame 131313ZAP pintado de rosa?)
4. Valor estimado (insensato, irracional y no se puede medir).
5. Cantidad de propaganda: creo que éste es la peor. Personalmente, estoy harto de los logos "XXX inside" o "kkkkkws compatible" (ahora se ha unido a la banda el "aaaaa Powered" - ¿Quién será el próximo?). EMMO[1], los billones de pesetas gastados en campañas de este tipo estarían mejor empleados en equipos de investigación que se ocupen de desarrollar nuevos procesadores libres de fallos, más rápidos y más baratos :-). Ningún tipo de publicidad puede arreglar un fallo en la unidad de coma flotante en la nueva hornada de procesadores que acaba de instalar en su placa base, pero en cambio un procesador rediseñado sí puede hacerlo.
6. La opiniones del tipo "tiene lo que paga" son sólo eso: opiniones. Denme hechos, por favor.

2 Procedimientos de medida e interpretación de resultados

Unas cuantas recomendaciones semiobvias:

1. Primera y principal, **identifique el rendimiento objetivo**. ¿Qué es exactamente lo que trata de medir? ¿De qué forma la medida del rendimiento le ayudará después a tomar una decisión? ¿Cuánto tiempo y cuántos recursos está dispuesto a gastar en el proceso de medida?
2. **Utilice herramientas estándar**. Use una versión del núcleo estable y actualizada, así como un gcc, libc, y una herramienta de medida del rendimiento actualizados y estándares. Abreviando, utilice el LBT (ver más adelante).
3. Dé una **descripción completa** de su configuración (vea el artículo LBT más adelante).
4. Trate de **aislar una variable**. Las medidas comparativas dan más información que las "absolutas". **Ya no puedo insistir más en este punto**.

[1]N.T.: En Mi Modesta Opinión

5. **Verifique sus resultados**. Ejecute sus pruebas unas cuantas veces y compruebe las fluctuaciones en los resultados, de haberlas. Las fluctuaciones inexplicables invalidarán sus resultados.
6. Si cree que su esfuerzo en la medición del rendimiento ha producido información útil, **compártala** con la comunidad Linux de forma **breve** y **concisa**.
7. Por favor, **olvídese de los BogoMips**. Me he prometido que algún día implementaré un rapidísimo ASIC con el bucle de los BogoMips enganchado en él. ¡Entonces veremos lo que tengamos que ver!

2.1 Entendiendo la elección de la herramienta

2.1.1 Las herramientas de medida sintéticas vs. las de aplicaciones

Antes de perder el tiempo escogiendo entre todos los tipos de herramientas de medida, se debe hacer una elección básica entre las herramientas "sintéticas" y las de "aplicación".

Las herramientas sintéticas están especialmente diseñadas para medir el rendimiento de un componente individual de un ordenador, normalmente llevando el componente escogido a su máxima capacidad. Un ejemplo de una herramienta sintética muy conocida es el **Whetstone**, programado originalmente en 1972 por Harold Curnow en FORTRAN (¿o fue en ALGOL?) y todavía ampliamente utilizada. El conjunto de herramientas Whetstone medirá el rendimiento de la unidad de punto flotante de la CPU.

La crítica principal que puede hacérsele a las medidas "sintéticas" es que no representan el rendimiento del sistema en las situaciones de la vida real. Tomemos por ejemplo las herramientas Whetstone: el blucle principal es muy pequeño y podría caber fácilmente en la caché primaria de la CPU, manteniendo el bus de la unidad de punto flotante (FPU) constantemente lleno y ejercitando, por tanto, la FPU a su máxima velocidad. No podemos criticar las herramientas Whetstone por esto, ya que hemos de tener presente que fueron programadas hace 25 años (¡y diseñadas en fechas anteriores!), pero hemos de asegurarnos de que interpretamos los resultados con cuidado cuando medimos microprocesadores modernos.

Otro punto a tener en cuenta sobre los tests sintéticos es que, idealmente, deberían darnos información acerca de un aspecto **específico** del sistema que estamos comprobando, independientemente del resto de componentes: un test sintético sobre la E/S de las tarjetas Ethernet debería devolver el mismo resultado (o al menos similar) independientemente de si se ejecuta en un 386SX-16 con 4 MBytes de RAM o de si se ejecuta en un Pentium 200 MMX con 64 MBytes de RAM. Sin embargo, el test medirá la rendimiento global de la combinación CPU/placa base/Bus/tarjeta Ethernet/Subsistema de memoria/DMA: no es muy útil, ya que la variación en el procesador podría causar un impacto mayor en los resultados que la variación en la tarjeta de red Ethernet (naturalmente, ésto es suponiendo que estamos utilizando la misma combinación de controlador/núcleo, que también pueden producir grandes cambios).

Para terminar, un error muy común es hacer la media de varios tests sintéticos y decir que esta media es una buena representación del rendimiento en la vida real de un sistema.

Aquí tenemos un comentario acerca de los tests FPU, citado con permiso de Cyrix Corp.:

> *"Una Unidad de Coma Flotante (Floating Point Unit, FPU) acelera los programas diseñados para utilizar matemáticas en coma flotante: suelen ser programas de CAD, hojas de cálculo, juegos 3D y diseño de aplicaciones. Sin embargo, hoy en día las aplicaciones más populares para PC utilizan al mismo tiempo instrucciones en enteros y en coma flotante. Como resultado, Cyrix ha decidido poner énfasis en el "paralelismo" a la hora de diseñar el procesador 6x86 para acelerar los programas que entremezclan estos dos tipos de instrucciones.*
>
> *El modelo de exclusión de la unidad de coma flotante de los x86 permite la resolución de instrucciones con enteros mientras se ejecuta una instrucción en coma flotante. Por contra, no se*

puede ejecutar una segunda instrucción en coma flotante si ya se estaba ejecutando anteriormente una. Para eliminar la limitación en el rendimiento creada por el modelo de exclusión de la unidad de coma flotante, el procesador 6x86 puede realizar especulativamente hasta cuatro instrucciones en coma flotante al chip FPU mientras sigue ejecutando instrucciones enteras. Por ejemplo, en una secuencia de código con dos instrucciones en coma flotante (FLTs) seguidas por seis instrucciones enteras (INTs) y seguidas por dos FLTs más, el procesador 6x86 puede mandar las diez instrucciones anteriores a las unidades de ejecución apropiadas antes de que se termine la primera FLT. Si ninguna de las instrucciones falla (el caso típico), la ejecución continua con las unidades de enteros y de coma flotante terminando las instrucciones en paralelo. Si una de las FLTs falla (el caso atípico), la capacidad de ejecución especulativa del 6x86 permite que se restaure el estado del procesador de forma que sea compatible con el modelo de exclusión de la unidad de coma flotante del x86.

Un examen de los test de rendimiento revela que los test sintéticos de la unidad de coma flotante utiliza un código con solo operaciones de coma flotante, que no es lo que nos encontramos en las aplicaciones del mundo real. Este tipo de tests no aprovecha la capacidad de ejecución especulativa presente en el procesador 6x86. Cyrix cree que las pruebas que utilizan herramientas no sintéticas, basadas en aplicaciones del mundo real reflejan mejor el rendimiento real que pueden obtener los usuarios. Las aplicaciones del mundo real contienen instrucciones mezcladas de enteros y de coma flotante y utilizan por tanto, la capacidad de ejecución especulativa del 6x86."

Por lo tanto, la tendencia en los tests de rendimiento es elegir las aplicaciones más comunes y utilizarlas para medir el rendimiento del sistema completo. Por ejemplo, **SPEC**, la organización sin ánimo de lucro que diseñó las herramientas SPECINT y SPECFP, ha lanzado un proyecto para un nuevo conjunto de herramientas basadas en aplicaciones. Pero de nuevo, sería muy raro que alguna herramienta comercial de medida del rendimiento incluya código Linux.

Resumiendo, los tests sintéticos son válidos mientras comprenda sus propósitos y limitaciones. Las herramientas basadas en aplicaciones reflejarán mejor el rendimiento global del sistema, pero no hay ninguna disponible para Linux.

2.1.2 Tests de alto nivel vs. test de bajo nivel

Los tests de bajo nivel miden directamente el rendimiento de los componentes: el reloj de la CPU, los tiempos de la DRAM y de la caché SRAM, tiempo de acceso medio al disco duro, latencia, tiempo de cambio de pista, etc... esto puede ser util en caso de comprar un sistema y no se sabe con que componentes viene, pero una forma mejor de comprobar estos datos es abrir la caja, hacer un listado con los nombres que pueda conseguir y obtener una hoja de características de cada componente encontrado (normalmente disponibles en la red).

Otro uso de los tests de bajo nivel es comprobar que un controlador de núcleo ha sido correctamente instalado para un componente específico: si tiene la hoja de especificaciones del componente, puede comparar los resultados del test de bajo nivel con las especificaciones teóricas (las impresas).

Los tests de alto nivel están más enfocados a medir el rendimiento de la combinación componente/controlador/SO de un aspecto específico del sistema, como por ejemplo el rendimiento de E/S con ficheros, o el rendimiento de una determinada combinación de componentes/controlador/SO/aplicación, p.e. un test Apache en diferentes sistemas.

Por supuesto, todos los tests de bajo nivel son sintéticos. Los tests de alto nivel pueden ser sintéticos o de aplicación.

2.2 Tests estándares disponibles para Linux

EMMO un test sencillo que cualquiera puede hacer cuando actualiza cualquier componentes en su Linux es hacer una compilación del núcleo antes y después de la actualización del componente o del programa y comparar los tiempos de compilación. Si todas las demás combinaciones se mantienen igual, entonces el test es válido como medida del rendimiento en la compilación, y uno ya puede decir que:

> "Cambiar de A a B lleva a una mejora de un x % en el tiempo de compilación del núcleo de Linux bajo estas y estas condiciones".

¡Ni más, ni menos!

Ya que la compilación del núcleo es una tarea muy normal en Linux, y ya que ejercita muchas de las funciones que se realizan normalmente en los tests (excepto el rendimiento con las instrucciones en coma flotante), podemos concluir que es un test **individual** bastante bueno. Sin embargo en muchos casos, los resultados de un test no puede ser reproducido por otros usuarios Linux debido a las variaciones en la configuración de los programas/componentes y por tanto este tipo de pruebas no puede utilizarse como "vara de medida" para comparar distintos sistemas (a menos que nos pongamos todos de acuerdo en compilar un núcleo estándar - ver más adelante).

Desgraciadamente, no hay herramientas de medida del rendimiento específicas para Linux, exceptuando el Byte Linux Benchmarks, que son una versión modificada del The Byte Unix Benchmarks que data de Mayo de 1991 (los módulos de Linux se deben a Jon Tombs, autores originales Ben Smith, Rick Grehan y Tom Yager).

Hay una página central *http://www.silkroad.com/bass/linux/bm.html* para el Byte Linux Benchmarks.

David C. Niemi puso por ahí una versión del Byte Unix Benchmarks mejorada y actualizada. Para evitar confusiones con alguna versión anterior la llamó UnixBench 4.01. Aquí está lo que David escribió sobre sus modificaciones:

> *"La versión original y las versiones ligeramente modificadas de BYTE Unix Benchmarks se diferencian en varias cosas que los hacen ser un indicador inusualmente poco fiable del rendimiento del sistema. He hecho que los valores de mis "índices" parezcan muy diferentes para evitar confusiones con el viejo test."*

David ha creado una lista de correo majordomo para la discusión sobre las pruebas de rendimiento para Linux y para el resto de SOs. Puede unirse a la lista enviando un mensaje a *majordomo@wauug.erols.com* escribiendo en el cuerpo "subscribe bench". El Grupo de Usuarios de Unix del Area de Washington está en proceso de crear una página Web *http://wauug.erols.com/bench* sobre los test de rendimiento en Linux.

También recientemente, Uwe F. Mayer, *mayer@math.vanderbilt.edu* portó el conjunto de pruebas Byte Bytemark a Linux. Éste es un moderno conjunto de herramientas que Rick Grehan envió a BYTE Magazine para comprobar la CPU, FPU y el rendimiento del sistema de memoria de los modernos sistemas de microordenador (hay pruebas estrictamente orientadas al rendimiento del procesador, sin tener en cuenta el rendimiento de la E/S o del sistema).

Uwe también ha creado una página Web *http://math.vanderbilt.edu:80/~mayer/linux/bmark.html* con una base de datos de los resultados de las pruebas de su versión del Linux BYTEmark benchmarks.

Si busca pruebas sintéticas para Linux, en sunsite.unc.edu podrá encontrar unas cuantas. Para comprobar la velocidad relativa de los servidores X y de las tarjetas gráficas, el conjunto de herramientas xbench-0.2 creado por Claus Gittinger está disponible en sunsite.unc.edu, ftp.x.org y otros lugares. Xfree86.org rechaza (prudentemente) el llevar o recomendar ninguna prueba.

El *XFree86-benchmarks Survey* *http://www.goof.com/xbench/* es una página Web con una base de datos de los resultados de x-bench.

Para el intercambio de E/S de disco, el programa hdparm (incluido con muchas distribuciones, si no lo tiene puede encontrarlo en sunsite.unc.edu) puede medir las tasas de transferencia si lo invoca con las opciones -t y -T.

Hay muchas otras herramientas disponibles de forma libre en Internet para comprobar varios aspectos del rendimiento de su Linux.

2.3 Enlaces y referencias

El comp.benchmarks.faq creado por Dave Sill es la referencia estándar en las pruebas de rendimiento. No es específico de Linux, pero es una lectura recomendada para cualquiera que se quiera ver seriamente implicado en las pruebas de rendimiento. Está disponible en muchos FTPs y páginas Web y muestra **56 pruebas diferentes**, con enlaces a direcciones FTP o páginas Web donde se pueden recoger. Algunas de las pruebas que se mencionan son comerciales (SPEC, por ejemplo).

No voy a nombrar todos y cada uno de los tests que se mencionan en comp.benchmarks.faq, pero hay al menos una prueba de bajo nivel que me gustaría comentar: la prueba lmbench *http://reality.sgi.com/lm/lmbench/lmbench.html* de Larry McVoy. Citando a David C. Niemi:

> *"Linus y David Miller la utilizan mucho ya que es capaz de realizar medidas útiles de bajo nivel y también puede medir el trasvase y la latencia de la red si tiene dos ordenadores para hacer los tests. Pero no intenta conseguir algo así como un "rendimiento del sistema" general..."*

Alfred Aburto puso en marcha un lugar FTP bastante completo en cuanto a utilidades **libres** de medida del rendimiento (*ftp://ftp.nosc.mil/pub/aburto*). Las herramientas Whetstone utilizadas en el LBT se encontraron aquí.

También tenemos el **FAQ multiparte de Eugene Miya** que deja regularmente en comp.benchmarks; es una referencia excelente.

3 El Linux Benchmarking Toolkit (LBT)

Quiero proponer un conjunto básico de herramientas de medida para Linux. Es sólo una versión preliminar de un general Linux Benchmarking Toolkit, que será expandido y mejorado. Tómelo como lo que es, esto es, como una propuesta. Si no cree que es un conjunto de herramientas válido, sientase libre de enviarme un correo electrónico con sus críticas y estaré encantado de hacer los cambios y mejoras, si puedo. Sin embargo, antes de tomar una decisión, lea este CÓMO y las referencias mencionadas: las críticas informadas serán bienvenidas, las críticas sin fundamento no.

3.1 Bases lógicas

Ésto es sólo de sentido común:

1. No debe llevar un día el ejecutarlo. Cuando hay que hacer tests comparativos (varias ejecuciones), no hay nadie que esté dispuesto a pasar días tratando de averiguar la mejor configuración de un sistema. Idealmente, el conjunto completo de pruebas debe llevar unos 15 minutos en una máquina media.

2. Todo el código fuente de los programas de estar libremente disponible en la Red, por razones obvias.

3. Los tests deben proporcionar una representación sencilla de los resultados que refleje el rendimiento medido.

4. Debe haber una mezcla de tests sintéticos y de tests de aplicación (con resultados separados, por supuesto).
5. Cada test **sintético** debe ejercitar un subsistema particular hasta su máxima capacidad.
6. Los resultados de los tests **sintéticos NO** deben mezclarse en un sólo resultado general (ésto acaba con la toda la idea que hay detrás de los tests sintéticos, con una considerable pérdida de información).
7. Los tests de aplicación deben consistir en tareas usualmente ejecutadas en los sistemas Linux.

3.2 Selección de herramientas

He seleccionado cinco conjuntos de herramientas, tratando de evitar, en la medida de lo posible, el solapamiento de pruebas. Son éstas:

1. Compilación del Núcleo 2.0.0 (con la configuración por defecto) utilizando gcc.
2. La versión 10/03/97 de Whetstone (la última que ha sacado Roy Longbottom).
3. xbench-0.2 (con los parámetros de ejecución rápida).
4. La versión 4.01 de UnixBench (resultados parciales).
5. La distribución 2 de la versión beta de los test BYTEmark de la revista BYTE Magazine (resultados parciales).

Para las pruebas 4 y 5, "(resultados parciales)" significa que no se tendrán en cuenta todos los resultados producidos por estos tests.

3.3 Duración de las pruebas

1. Compilación del Núcleo 2.0.0: 5 - 30 minutos, dependiendo del rendimiento **real** de su sistema.
2. Whetstone: 100 segundos.
3. Xbench-0.2: $<$ 1 hora.
4. Versión 4.01 de los tests UnixBench: aprox. 15 minutos.
5. Los tests BYTEmark de BYTE Magazine: aprox. 10 minutos.

3.4 Comentarios

3.4.1 Compilación del Núcleo 2.0.0:

- **Qué:** es el único test de aplicación que hay en el LBT.
- El código está ampliamente difundido (finalmente he encontrado alguna utilidad a mis viejos CD-ROMs con Linux).
- Muchos linuxeros recompilan el núcleo a menudo, por lo que es un medida significativa del rendimiento global del sistema.
- El núcleo es grande y gcc utiliza una gran cantidad de memoria: se atenua la importancia de la caché L2.

- Hace un uso frecuente de la E/S al disco.
- Procedimiento para realizar la prueba: conseguir el código de la versión 2.0.0 del núcleo, compilarlo con las opciones por defecto (make config, pulsar Intro repetidamente). El tiempo a informar debería ser el que se inverte en la compilación; esto es, después de que escribe make zImage, **sin** incluir make dep, make clean. Tenga en cuenta que la arquitectura objetivo por defecto del núcleo es i386, de manera que si compila en otras arquitecturas, debería configurar también gcc para hacer una compilación cruzada, teniendo i386 como arquitectura objetivo.
- **Resultados:** tiempo de compilación en minutos y segundos (por favor, no indique las fracciones de segundo).

3.4.2 Whetstone:

- **Qué:** mide el rendimiento de punto flotante puro con un bucle corto. El fuente (en C) es muy legible y es fácil de ver qué operaciones en punto flotante intervienen.
- Es la prueba más corta del LBT :-).
- Es una prueba "Clásica": hay disponibles cifras comparativas, sus defectos y deficiencias son bien conocidos.
- Procedimiento para realizar la prueba: se debería obtener el código en C más reciente del sitio de Aburto. Compile y ejecute en modo de doble precisión. Especifique gcc y -O2 como opciones de precompilador y compilador, y defina POSIX 1 para especificar el tipo de máquina.
- **Resultados:** una cifra del rendimiento de punto flotante en MWIPS.

3.4.3 Xbench-0.2:

- **Qué:** mide el rendimiento del servidor X.
- La medida xStones proporcionada por xbench es una media ponderada de varias pruebas referidas a una vieja estación Sun con una pantalla de un solo bit de profundidad. Hmmm... es cuestionable como test para servidores X modernos, pero sigue siendo la mejor herramienta que he encontrado.
- Procedimiento para realizar la prueba: compilar con -O2. Especificamos unas pocas opciones para una ejecución más rápida:`./xbench -timegoal 3 > results/name_of_your_linux_box.out`. Para obtener la calificación xStones, debemos ejecutar un guión (script) en awk; la manera más rápida es escribir `make summary.ms`. Compruebe el fichero summary.ms: la calificación xStone de su sistema está en la última columna del renglón que tiene el nombre de su máquina que especificó durante la prueba.
- **Resultados:** una figure del rendimiento de X en xStones.
- Nota: esta prueba, tal como está, es obsoleta. Debería ser reescrita.

3.4.4 UnixBench versión 4.01:

- **Qué:** mide el rendimiento global de Unix. Esta prueba ejercitará el rendimiento de E/S de ficheros y multitarea del núcleo.
- He descartado los resultados de todas las pruebas aritméticas, quedándome sólo con los resultados relacionados con el sistema.

- Procedimiento para realizar la prueba: compilar con -O2. Ejecutar con `./Run -1` (ejecutar cada prueba una vez). Encontrará los resultados en el fichero ./results/report. Calcule la media geométrica de los índices EXECL THROUGHPUT, FILECOPY 1, 2, 3, PIPE THROUGHPUT, PIPE-BASED CONTEXT SWITCHING, PROCESS CREATION, SHELL SCRIPTS y SYSTEM CALL OVERHEAD.
- **Resultados:** un índice del sistema.

3.4.5 Banco de pruebas BYTEmark de BYTE Magazine BYTEmark:

- **Qué:** proporciona una buena medida del rendimiento de la CPU. Aquí hay un extracto de la documentación: *"Estas pruebas están pensadas para exponer el límite superior teórico de la arquitectura de CPU, FPU y memoria de un sistema. No pueden medir transferencias de vídeo, disco o red (éstos son dominios de un conjunto de pruebas diferentes). Debería usted, por lo tanto, utilizar los resultados de estas pruebas como parte, no como un todo, en cualquier evaluación de un sistema."*
- He descartado los resultados de la prueba de FPU ya que la prueba Whetstone es representativa del rendimiento de la FPU.
- He dividido las pruebas de enteros en dos grupos: aquellos más representativos del rendimiento memoria-caché-CPU y las pruebas de enteros de la CPU.
- Procedimiento para realizar la prueba: compilar con -O2. Ejecutar la prueba con `./nbench > myresults.dat` o similar. Entonces, de myresults.dat, calcule la media geométrica de los índices de las pruebas STRING SORT, ASSIGNMENT y BITFIELD; éste es el **índice de la memoria**; calcule la media geométrica de los índices de las pruebas NUMERIC SORT, IDEA, HUFFMAN y FP EMULATION; éste es el **índice de enteros**.
- **Resultados:** un índice de memoria y un índice de enteros calculado tal como se explica anteriormente.

3.5 Posibles mejoras

El conjunto ideal de pruebas debería ejecutarse en pocos minutos, con pruebas sintéticas que examinen cada subsistema por separado y pruebas de aplicación que den resultados para diferentes aplicaciones. También debería generar de forma automática un informe completo y quizá enviarlo por correo a la base de datos central en la Web.

No estamos interesados en la portabilidad, pero debería al menos poder ser ejecutado en cualquier versión reciente (> 2.0.0) y 'sabor' (i386, Alpha, Sparc...) de Linux.

Si alguien tiene alguna idea al respecto de probar la red de una manera sencilla, fácil y fiable, con una prueba corta (menos de 30 minutos en configuración y ejecución), por favor, póngase en contacto conmigo.

3.6 El formulario de informe LBT

Aparte de las pruebas, el procedimiento de 'benchmarking' no estaría completo sin un formulario describiendo la configuración, de manera que aquí está (siguiendo la guía de comp.benchmarks.faq):

```
LINUX BENCHMARKING TOOLKIT REPORT FORM
```

```
CPU
==
Vendor:
```

```
Model:
Core clock:
Motherboard vendor:
Mbd. model:
Mbd. chipset:
Bus type:
Bus clock:
Cache total:
Cache type/speed:
SMP (number of processors):
```

```
RAM
====
Total:
Type:
Speed:
```

```
Disk
====
Vendor:
Model:
Size:
Interface:
Driver/Settings:
```

```
Video board
===========
Vendor:
Model:
Bus:
Video RAM type:
Video RAM total:
X server vendor:
X server version:
X server chipset choice:
Resolution/vert. refresh rate:
Color depth:
```

```
Kernel
=====
Version:
Swap size:
```

```
gcc
===
Version:
Options:
libc version:
```

```
Test notes
==========
```

```
RESULTS
========
Linux kernel 2.0.0 Compilation Time: (minutes and seconds)
Whetstones: results are in MWIPS.
Xbench: results are in xstones.
Unixbench Benchmarks 4.01 system INDEX:
BYTEmark integer INDEX:
BYTEmark memory INDEX:
```

```
Comments*
=========
* Este campo se incluye para una posible interpretación de los resultados,
y como tal, es opcional. Podría ser la parte más significativa del
informe, sin embargo, especialmente si está haciendo pruebas comparativas.
```

3.7 Pruebas del rendimiento de la red

Probar el rendimiento de una red es un reto, ya que implica al menos tener dos máquinas, un servidor y un cliente, y por lo tanto el doble de tiempo para configurar, más variables a controlar, etc... En una red ethernet, pienso que su mejor apuesta sería el paquete ttcp. (por expandir)

3.8 Pruebas SMP

Las pruebas SMP son otro reto, y cualquier banco de pruebas diseñado específicamente para probar SMP tendrá dificultades probándose a sí misma en configuraciones de la vida real, ya que los algoritmos que pueden tomar ventaja de SMP son difíciles de realizar. Parece que las últimas versiones del núcleo de Linux (> 2.1.30 o por ahí) harán multiproceso "muy granulado" (*fine-grained*), pero no tengo más información al respecto ahora mismo.

Según David Niemi, *" ... shell8* [parte del Unixbench 4.01]*hace un buen trabajo comparando hardware similare en los modos SMP y UP."*

4 Prueba de ejemplo y resultados

Ejecuté el LBT en la máquina de mi casa, un Linux de clase Pentium que puse a mi lado y usé para escribir este COMO. Aquí tiene el Formulario de Informe LBT de este sistema:

```
LINUX BENCHMARKING TOOLKIT REPORT FORM

CPU

==
```

```
Vendor: Cyrix/IBM

Model: 6x86L P166+

Core clock: 133 MHz

Motherboard vendor: Elite Computer Systems (ECS)

Mbd. model: P5VX-Be

Mbd. chipset: Intel VX

Bus type: PCI

Bus clock: 33 MHz

Cache total: 256 KB

Cache type/speed: Pipeline burst 6 ns

SMP (number of processors): 1

RAM

====

Total: 32 MB

Type: EDO SIMMs

Speed: 60 ns

Disk

====

Vendor: IBM

Model: IBM-DAQA-33240

Size: 3.2 GB

Interface: EIDE

Driver/Settings: Bus Master DMA mode 2

Video board

===========
```

```
Vendor: Generic S3

Model: Trio64-V2

Bus: PCI

Video RAM type: EDO DRAM

Video RAM total: 2 MB

X server vendor: XFree86

X server version: 3.3

X server chipset choice: S3 accelerated

Resolution/vert. refresh rate: 1152x864 @ 70 Hz

Color depth: 16 bits

Kernel

=====

Version: 2.0.29

Swap size: 64 MB

gcc

===

Version: 2.7.2.1

Options: -O2

libc version: 5.4.23

Test notes

==========

Carga muy ligera. Las pruebas anteriores se ejecutaron activando
algunas de las capacidades mejoradas del Cyrix/IBM 6x86, mediante el
programa setx86: fast ADS, fast IORT, Enable DTE, fast LOOP, fast Lin.
VidMem.

RESULTS
```

```
========

Linux kernel 2.0.0 Compilation Time: 7m12s

Whetstones: 38.169 MWIPS.

Xbench: 97243 xStones.

BYTE Unix Benchmarks 4.01 system INDEX: 58.43

BYTEmark integer INDEX: 1.50

BYTEmark memory INDEX: 2.50

Comments

=========

Este es un sistema muy estable con un rendimiento homogéneo, ideal
para el uso en casa o para el desarrollo con Linux. ¡Informaré de los
resultados con un procesador 6x86MX tan pronto como le eche las manos
encima!
```

5 Falsedades y fallos del benchmarking

Después de reunir este COMO empecé a comprender por qué se asocian tan frecuentemente las palabras "falsedad" y "fallo" con el benchmarking...

5.1 Comparar manzanas con naranjas

¿O debería decir Apples[2]con PCs? Es una disputa tan obvia y antigua que no ahondaré en los detalles. Dudo que el tiempo que tarda en cargarse Word en un Mac comparado a la media de un Pentium sea una medida real de nada. Al igual que el tiempo de arranque de un Linux y un Windows NT, etc... Procure lo más posible comprar máquinas idénticas con una sola modificación.

5.2 Información incompleta

Un solo ejemplo ilustrará este fallo común. A menudo uno lee en comp.os.linux.hardware la siguiente frase o similar: "Acabo de poner el procesador XYZ a nnn MHz y ahora compilar el núcleo de linux me lleva sólo i minutos" (ajustar XYZ, nnn e i a lo que se requiera). Es irritante, porque no se da más información, esto es, no sabemos siquiera la cantidad de RAM, tamaño del fichero de intercambio, otras tareas que se ejecutan simultáneamente, versión del núcleo, módulos seleccionados, tipo de disco duro, versión del gcc, etc... Le recomiendo que use el Formulario de Informe LBT, que al menos proporciona un marco de información estándar.

[2] N. del T.: Apple = manzana, pero también una famosa compañía que fabrica ordenadores

5.3 Software/hardware Propietario

Un conocido fabricante de procesadores publicó una vez los resultados de unas pruebas producidas por una versión especial, adaptada, de gcc. Aparte las consideraciones éticas, estos resultados no tenían sentido, ya que el 100% seguiría usando la versión estándar de gcc. Lo mismo va para el hardware propietario. El benchmarking es mucho más útil cuando trata con hardware off-the-shelf y sofware libre.

5.4 Relevancia

Estamos hablando de Linux, ¿no? De manera que deberíamos olvidarnos de pruebas producidas en otros sistemoas operativos (este es un caso especial del "Comparando manzanas y naranjas" que vimos anteriormente). Además, si vamos a hacer pruebas del rendimiento de un servidor Web, **no** debemos resaltar el rendimiento de la FPU ni otra información irrelevante. En tales casos, menos es más. **Tampoco** necesitamos mencionar la edad de nuestro gato, el humor del que estábamos mientras hacíamos la prueba, etc...

6 FAQ (Preguntas Frecuentes)

P1.

¿Hay alguna medida aislada del mérito de los sistemas Linux?

R:

No, gracias al cielo nadie ha salido todavía con una medida Lhinuxstone (tm). Y si hubiera una, no tendría mucho sentido: los sistemas Linux se usan para diversas tareas, desde servidores Web muy cargados hasta estaciones gráficas para uso individual. Ninguna medida de mérito por separado puede describir el rendimiento de un sistema Linux bajo tales situaciones diferentes.

P2.

Entonces, ¿qué tal una docena de cifras resumiendo el rendimiento de diversos sistemas Linux?

R:

Esa sería la situación ideal. Me gustaría ver que se hace realidad. ¿Voluntarios para un **Linux Benchmarking Project**? ¿Con un sitio web y una base de datos en línea, completa y con informes bien diseñados?

P3.

... ¿BogoMips...?

R:

Los BogoMips no tienen nada que ver con el rendimiento de su sistema. Lea el BogoMips Mini-HOWTO.

P4.

¿Cuá es el 'mejor' banco de pruebas para Linux?

R:

Todo depende de qué aspecto del rendimiento de un sistema Linux quiera medir uno. Hay diferentes bancos de pruebas para medir la red (tasa sostenida de transferencia Ethernet), servidores de ficheros (NFS), E/S de disco, FPU, enteros, gráficos, 3D, ancho de banda procesador-memoria, rendimiento CAD, tiempo de transacción, rendimiento SQL, rendimiento de servidor web, rendimiento en tiempo real (*Real-Time*), rendimiento del CD-ROM, rendimiento de Quake (¡!), etc... HDYS[3], no existe ningún conjunto de pruebas para Linux que realice todas estas pruebas.

P5.

¿Cuál es el procesador más rápido sobre el que corre Linux?

R:

¿Más rápido en qué tarea? Si estamos orientados a un gran procesamiento de números, un Alpha de gran velocidad de reloj (600MHz y superior) debería ser más rápido que ninguna otra cosa, ya que los Alpha se han diseñado para ese tipo de rendimiento. Si, por otro lado, uno quiere poner un servidor de news muy rápido, es probable que la elección de un subsistema de disco duro muy rápido y muchísima RAM de un rendimiento mucho más alto que un cambio de procesador, por la misma cantidad de $.

P6.

Permítame rehacer la pregunta anterior, entonces: ¿hay algún procesador que sea más rápido para aplicaciones de propósito general?

R:

Esa es una difícil con truco, pero tiene una respuesta muy sencilla: **NO**. Siempre podremos diseñar un sistema más rápido incluso para aplicaciones de uso general, independientemente del procesador. Normalmente, siendo todos los demás elementos iguales, mayores tasas de reloj darán sistemas de mayor rendimiento (y también más dolores de cabeza). Sacando un viejo Pentium a 100MHz de una (no suele ser así) placa madre actualizable, y enchufando la versión a 200MHz, uno debería sentir el "umppffff" extra. Por supuesto, con sólo 16 MBytes de RAM, se podría haber hecho la misma inversión, más sabiamente, en unos SIMM extra...

P7.

¿De manera que la velocidad de reloj influye en el rendimiento del sistema?

R:

Para la mayoría de las tareas excepto los bucles NOP vacíos (por cierto, son eliminados por los compiladores optimizadores modernos), un aumento en la velocidad del reloj no nos dará un aunmento lineal en rendimiento. Los programas muy pequeños e intensivos que quepan enteros en la caché primaria del procesador (la caché L1, normalmente de 8 o 16K), obtendrán un aumento de rendimiento equivalente al de la velocidad de reloj, pero la mayoría de los programas "reales" son mucho más grandes que eso, tienen bucles que no caben en la caché L1, comparten la caché L2 (externa) con otros procesos, dependen de componentes externos y obtendrán incrementos mucho menores de rendimiento. Esto es así porque la caché L1 funciona a la misma velocidad de reloj que el procesador, mientras que la mayoría de las caché L2 y el resto de los subsistemas (DRAM, por ejemplo, funcionan de forma asíncrona a menores velocidades.

P8.

Bien, entonces, una última pregunta sobre el asunto: ¿cual es el procesador que proporciona una mejor tasa precio/rendimiento para usos de propósito general de Linux?

R:

¡Definir "uso de propósito general de Linux no es fácil! Para cualquier aplicación particular, siempre hay un procesador con EL MEJOR precio/rendimiento en un momento dado, pero cambia tan frecuentemente como los fabricantes sacan al mercado nuevos procesadores, de manera que responder Procesador XYZ ejecutándose a n MHz sería una respuesta válida sólo temporalmente. De todas maneras, el precio

[3]HDYS = Hasta Donde Yo Sé (AFAIK = As Far As I Know)

del procesador es insignificante comparado al precio global del sistema que vamos a poner, De manera que, realmente, la cuestión debería ser ¿cómo podemos maximizar la tasa precio/rendimiento de un sistema dado? Y la respuesta a esa cuestión depende muchísimo de los requerimientos mínimos de rendimiento y en el coste mínimo/máximo establecido para la configuración que estamos considerando. Algunas veces, el hardware que podemos comprar en las tiendas no nos dará el rendimiento mínimo necesario, y la única alternativa serán costosos sistemas RISC. Para algunos usos, recomiendo un sistema equilibrado y homogéneo :-); la elección de un procesador es una decisión importante, pero no más que elegir el tipo y capacidad del disco duro, la cantidad de RAM, la tarjeta de vídeo, etc...

P9.

¿Qué es un incremento "significativo" de rendimiento?

R:

Yo diría que cualquier cosa por debajo de 1% no es significativo (podría ser descrito como "marginal"). Nosotros, los humanos, difícilmente distinguiremos la diferencia entre dos sistemas con una diferencia en tiempo de respuesta del 5%. Por supuesto, algunos de los más duros realizadores de pruebas no son humanos, y le dirán que, comparando dos sistemas con índices de 65'9 y 66'5, este último es "definitivamente más rápido".

P10.

¿Cómo obtengo incrementos "significativos" en renadimiento al menor coste?

R:

Como la mayoría del código fuente para Linux está disponible, un examen atento y un rediseño algorítmico de las subrutinas clave podrían alcanzar incrementos de rendimiento en órdenes de magnitud en algunos casos. Si estamos tratando con un proyecto comercial y no deseamos meternos demasiado en el código C, **podríamos llamar a un consultor de Linux**. Lea el Consultants-HOWTO.

7 Copyright, reconocimientos y miscelánea

7.1 Cómo se produjo este documento

El primer paso fue leer la sección 4 "Escribir y enviar un HOWTO" del HOWTO Index de Greg Hankins.

No sabía absolutamente nada sobre SGML o LaTeX, pero estuve tentado de usar un paquete de generación automática de documentación tras leer varios comentarios sobre las SGML-Tools. Sin embargo, insertar etiquetas manualmente en un documento me recuerda los días en que ensamblé a mano un programa monitor de 512 bytes para un microprocesador de 8 bits ya difunto, de manera que tomé las fuentes de LyX, lo compilé, y usé su modo LinuxDoc. Recomiendo la combinación: **LyX y SGML-Tools**.

7.2 Copyright

derivado de un HOWTO e imponer restricciones adicionales sobre su distribución. Se podrían permitir excepciones a estas restricciones bajo ciertas condiciones; por favor, póngase en contacto con el coordinador del Linux HOWTO en la dirección que damos más adelante.

En resumen, nos gustaría promover la diseminación de esta información a través de cuantos más canales sea posible. Sin embargo, queríamos retener el copyright de los documentos HOWTO, y nos gustaría que nos avisase de cualquier plan para redistribuir los HOWTO.

Si tiene preguntas, diríjase por favor a Greg Hankins, el coordinador de Linux HOWTO en gregh@sunsite.unc.edu mediante correo electrónico o llamando al +1 404 853 9989.

7.3 Nuevas versiones de este documento

Se pondrán las nuevas versiones del Linux Benchmarking-HOWTO en sunsite.unc.edu y en sitios espejo. Allí encontrará otros formatos, como las versiones PostScript y dvi en el directorio other-formats. El Linux Benchmarking-HOWTO también está disponible para clientes WWW como Grail, un navegador Web escrito en Python. También será enviado con regularidad en comp.os.linux.answers.

La versión en castellano de este HOWTO la encontrará en el sitio del Insflug *http://www.insflug.org*

7.4 Realimentación

Se buscan sugerencias y correciones. Se reconocerá a los contribuyentes. No necesito 'flames'.

Siempre me puede localizar en andrewbalsa@usa.net.

7.5 Agradecimientos

David Niemi, el autor del conjunto de aplicaciones Unixbench, ha probado ser una fuente infinita de información y críticas (válidas).

También me gustaría agradecer a Greg Hankins, el coordinador del Linux HOWTO y uno de los mayores contribuyentes al paquete SGML-tools, a Linus Torvalds y a la comunidad Linux al completo. Este HOWTO es mi manera de dar algo a cambio.

7.6 Pliego de descargo

Su experiencia puede variar (y variará). Tenga en cuenta que el benchmarking es un tema delicado, y una actividad que consume grandes cantidades de tiempo y energía.

7.7 Marcas registradas

Pentium y Windows NT son marcas registradas de Intel Corporation y Microsoft Corporation respectivamente.

BYTE y BYTEmark son marcas comerciales de McGraw-Hill, Inc.

Cyrix y 6x86 son marcas comerciales de Cyrix Corporation.

Linux no es una marca comercial, y esperemos que nunca lo sea.