

# 4. Selección y Configuración de Sistemas Informáticos: *Benchmarking*

## 4.1 Introducción

Un programa de prueba o **benchmark** se define como *un programa o conjunto de programas que evalúan las prestaciones de un sistema informático reproduciendo una carga de trabajo genérica en dicho sistema informático*. Al proceso de comparar dos o más sistemas mediante la obtención de medidas se le denomina *benchmarking*.

En general, para evaluar las prestaciones de un sistema informático es necesario conocer y caracterizar previamente cuál es la carga de trabajo, como se ha visto en temas anteriores. Sin embargo, en muchos casos tal carga no se conoce de antemano, es difícil de caracterizar o es suficientemente amplia como para considerarla una carga genérica.

En general, los benchmarks se agrupan en los denominados *paquetes benchmark* (benchmark suites), que agrupan diferentes programas para medir diferentes aspectos de un sistema informático. Para escoger o diseñar un buen paquete benchmark deben de seguirse los siguientes pasos [Puigjaner94]:

*Determinar los objetivos* de uso del benchmark. Es decir, ver qué sistema se va a medir, y aislar esa parte del sistema del resto para evitar sus efectos. Además, se debe especificar en qué tipo de cargas de trabajo o entorno se va a evaluar: carga científica o entorno de red como servidor, por ejemplo. En la mayor parte de los casos, el uso o diseño de un benchmark se reduce a “demostrar que nuestra máquina es mejor que la de los competidores”, aunque este tipo de objetivos es lo que se debe de evitar.

*Analizar la carga de trabajo de diversos sistemas, o de un mismo sistema con diversas aplicaciones*, para llegar a una carga de trabajo genérica. Por ejemplo, analizar en qué proporción se ejecutan las instrucciones de coma flotante y las de números enteros, o con qué frecuencia y de qué tamaño suelen ser las peticiones en un servidor de ficheros.

*Escoger los programas* según los objetivos determinados. Cada programa mide diferentes subsistemas, y, por tanto, esta será una razón para elegirlos. Diferentes programas, además, ejercitan de forma diferente el sistema. Por ejemplo, si se quiere medir las prestaciones de un ordenador como servidor Web, se escogerán programas tales como SPECWeb o WebBench (de la Ziff-Davis Benchmark operation); pero si el mismo sistema se quiere usar como servidor de ficheros en una red local, se deberá usar SPEC-SFS. Los resultados en un benchmark no implican que se obtengan los mismos resultados en otro que mida un aspecto similar del mismo sistema: por ejemplo, el que un procesador tenga buenas prestaciones en coma flotante con simple precisión no implica que tenga las mismas prestaciones en doble precisión; diferentes procesadores optimizan en su diseño diferentes tipos de cálculo. El análisis de la etapa anterior tendrá que tenerse en cuenta a la hora de evaluar los diferentes programas, o de seleccionar cuales programas de un benchmark interesan o no.

*Escoger las métricas* o mediciones que se van a tomar sobre el sistema.

Dependiendo de la parte del sistema que se quiera medir, así se usarán diferentes métricas. Estas métricas se escogerán entre las examinadas en el capítulo de

Caracterización de la Carga, pero son en general de dos tipos: *velocidad* de una función del sistema informático, y *capacidad* de la misma. Una métrica de tipo velocidad sería el tiempo que tarda en ejecutarse un programa o grupo de programas; una de tipo capacidad el número de usuarios que es capaz de soportar un sistema o el número de instrucciones por ciclo que puede ejecutar un microprocesador. En general, las métricas son función del nivel funcional del sistema o subsistema que se quiere medir.

Se deben de tener en cuenta *todos los factores que influyan en el rendimiento*.

Algunos de esos factores se variarán durante el estudio (por ejemplo, el número de usuarios simulados por el benchmark), y otros permanecerán fijos; en cualquier caso, todos los factores que influyan en el benchmark deberán de ser los mismos para todos los sistemas que se estudien, si se pretende hacer una comparación entre ellos. Por ejemplo, se deberá usar siempre la misma versión del sistema operativo (a no ser que la eficiencia de éste sea una de las cosas que se midan), iguales versiones del compilador, igual cantidad de memoria, de caché e iniciar la medición bajo las mismas condiciones. Habitualmente, en la presentación de los resultados de los benchmark se suele seguir la regla *Full Disclosure*, o *revelación completa*, es decir, una indicación pormenorizada de todas las condiciones bajo las que se lleva a cabo el estudio: fechas, versiones, marcas y modelos y opciones de compilación.

Finalmente, se debe de llevar a cabo algún tipo de *análisis*. La presentación de muchos datos no permite extraer conclusiones: todas las mediciones se deben de resumir y sintetizar en una *figura de mérito* (FDM) que indique, básicamente, cuál es el mejor sistema en el objetivo que se ha marcado, como por ejemplo los índices SPECint95 o BYTEmark usados por SPEC-CPU y Byte, respectivamente.

Los paquetes benchmark, y, en general, las cargas de trabajo de prueba deben de reunir una serie de características para cumplir su función adecuadamente:

*reproductibilidad*: que sea fácil de reproducir para cualquier tipo de sistema, tanto el programa en sí como los resultados a lo largo de diferentes ejecuciones del mismo programa,

*compacidad*: que contenga poco código, y que no sobrecargue el sistema ni tarde demasiado tiempo en ejecutarse,

*compatibilidad*: esta es la característica más importante. La carga de prueba debe de ser compatible con todos los sistemas que sean susceptibles de ser medidos,

*resolución*: el benchmark debe de durar lo suficiente como para que sea fácil diferenciar entre dos sistemas con características similares, y

*escalabilidad*: es decir, que el mismo benchmark sirva para ordenadores con una gama amplia de velocidades, que siga siendo útil a lo largo del tiempo, y que dé resultados significativos, o sea, que sirvan para diferenciar las prestaciones de los sistemas que se evalúan.

Este tema se estructurará en las siguientes secciones: en la sección [2](#) se explicará en qué aplicaciones es necesario, para pasar a comentar en la sección [3](#) cómo se usan. En la sección [4](#) se estudiarán algunos paquetes benchmark habituales, para terminar con los problemas y críticas que reciben los benchmark en general en la sección [5](#).

## 4.2 Aplicaciones de los benchmarks.

Un benchmark, por tanto, se usará en las siguientes ocasiones [Puigjaner94]:

*Adquisición de equipos informáticos.* En la mayoría de los casos, un equipo informático se va a usar para una gama amplia de tareas, desde llevar a cabo operaciones comerciales hasta ejecutar juegos de red. Por tanto, en tales casos una carga genérica reproducirá con más o menos exactitud la carga que va a ejecutar el sistema. Los resultados del benchmark servirán para justificar la compra de uno u otro equipo informático; en algunos casos incluso los resultados de un benchmark servirán como certificación de las prestaciones de un equipo, como en el caso de TPC, benchmark del Transaction Processing Council que evalúa las prestaciones de los ordenadores como sistema de proceso de transacciones.

*Sintonización de un sistema informático.* El ejecutar benchmarks periódicamente sobre un sistema que se está usando, permite ver como se deteriora o, en general, cambia su capacidad a lo largo del tiempo. Además, los benchmarks permiten hallar qué partes del sistema se deben cambiar o mejorar, o cómo ha impactado en el sistema el cambio de alguna de sus partes, o la actualización del sistema operativo o de alguno de sus programas.

*Planificación de la capacidad de un sistema informático:* de la misma forma que se ha indicado en el punto anterior, la evolución de la carga de un sistema y de los resultados de los benchmarks pueden permitir prever qué cambios va a hacer falta llevar a cabo en el futuro, y en qué punto. El ejecutar un benchmark para llevar al límite las capacidades de un sistema puede servir también en este sentido.

*Comparación de diferentes programas que realizan una tarea determinada:* por ejemplo, evaluar cómo diferentes compiladores generan código, cómo se comportan dos sistemas de procesamiento de transacciones, o cómo aprovechan los recursos del sistema dos diferentes sistemas operativos.

*Diseño de sistemas informáticos.* Aunque, evidentemente, no se puede ejecutar un benchmark sobre un sistema que aún no se ha construido, la ejecución de un benchmark puede permitir extraer conclusiones sobre el comportamiento de un sistema para diseños futuros. Por ejemplo, el número de instrucciones por ciclo evaluados con diferentes benchmarks puede permitir extraer conclusiones para el diseño de un pipeline; o a partir de la tasa de aciertos de una caché se puede calcular cuál va a ser su tamaño en versiones sucesivas [Franklin91], o el número de vías, o si se va a separar la caché de datos de la de instrucciones. De la misma forma, los benchmarks se pueden ejecutar a posteriori sobre un sistema informático para evaluar la efectividad de alguna de sus características en un entorno determinado, como hacen Bekerman y otros en [Bekerman95] con el procesador Pentium, descubriendo que el pipeline V de enteros sólo se usa, como máximo, en el 30% de los casos, y mucho menos si se trata de programas comerciales en Windows. Incluso algunos autores, como Hennesy y Patterson [HenPatt96] han usado el comportamiento a bajo diferentes programas benchmark procedentes de SPEC92 como criterio para el diseño de una arquitectura, la DLX.

Como se ve, y en el espíritu de esta asignatura, es necesario ejecutar benchmarks en todos los niveles funcionales de la arquitectura de un sistema informático [Tanenbaum87]: microoperaciones, máquina convencional, sistema operativo, lenguajes orientados a problemas, y programas de aplicación; y durante diferentes etapas de la vida útil del mismo: diseño, prueba, y comercialización. Por supuesto, el tipo de benchmark elegidos, las métricas usadas sobre el mismo, y el análisis de los resultados será diferente en cada caso.

En la siguiente sección se verá cómo se usa un benchmark para alcanzar los objetivos deseados.

## 4.3 Utilización de un benchmark

Los benchmarks son usados por una gran cantidad de personas, sus resultados son también aprovechados por una amplia gama de profesionales, y, por tanto, la mayoría tienen cierto grado de *oficialidad* o *estandarización*. Las especificaciones para los mismos, las reglas para su ejecución y la ejecución física de los mismos se llevan a cabo de forma diferente; este aspecto se examinará en el apartado [3.1](#). De estas ejecuciones se tienen que extraer medidas que sirvan para comparar los sistemas, que se verán en el apartado [3.2](#). No siempre se llevan a cabo los benchmark de forma correcta, es habitual realizar una serie de errores que se contemplarán en el apartado [3.3](#); muchas veces estos errores se realizan intencionadamente, como se verán en los “juegos” examinados en el apartado [3.4](#). Finalmente, a veces son los benchmarks los engañados, como se verá en el apartado [3.5](#).

### 4.3.1 ¿Quién propone un benchmark?

Dado que los benchmarks sirven para comparar sistemas informáticos, deben de tener amplia difusión, y, de hecho, *todos* los sistemas informáticos deben ser evaluados por el mismo benchmark, para que efectivamente puedan ser comparados. Para que este objetivo se lleve a cabo, los benchmarks suelen

Ser ejecutados por una empresa o institución independiente, que habitualmente es pagada por sus servicios. La empresa recibe una copia del sistema a evaluar, realiza el trabajo necesario para que se ejecuten los programas, y entrega los resultados al fabricante. Esto, más o menos, garantiza la independencia de los resultados. En muchos casos, las empresas son revistas o están contratadas por revistas, que presentan los resultados obtenidos a los lectores. Este el caso de NSTL, National Standard Technology Labs, contratados por la revista Byte, o de las revistas del grupo Ziff-Davis, que recurren a la Ziff-Davis Benchmark Operation. En otros casos, son instituciones universitarias, como sucede en el caso del grupo de Investigación de Jack Dongarra, proponente de los benchmarks denominados LINPACK. En este caso, la propia revista o institución suele llevar a cabo los benchmark, aunque también suele dejar en el dominio público los programas para que cualquiera pueda reproducir los resultados.

Ser propuestos por un consorcio de empresas, es decir, son una serie de programas, que todos o la mayoría de los fabricantes conoce y está de acuerdo en respetar los resultados; cada fabricante es responsable de llevar a cabo las pruebas y de publicar los resultados. Esto sucede con SPEC, *System Performance Evaluation Council*, que propone benchmarks para medir diferentes aspectos de un sistema, o GPC, Graphics Performance Council, que hace lo propio para subsistemas gráficos, o TPC, Transaction Processing Council, que propone benchmarks para la medición de sistemas de proceso transaccional.

El propio interesado puede llevar a cabo los benchmarks, siempre que los diferentes sistemas informáticos estén disponibles y cuente con los recursos suficientes para hacerlo. De esta forma, se pueden enfocar claramente los objetivos del estudio, y usar las métricas y resultados que más interesen; además, tiene sentido que los usuarios conozcan como aplicar y qué suelen medir los benchmarks más importantes. Estas métricas se explican en el apartado siguiente.

### 4.3.2 Unidades de medida

Las unidades dependerán totalmente del objetivo del estudio y de qué nivel del sistema informático interese medir. Evidentemente, todas estas medidas reflejan las prestaciones de un sistema completo, puesto que es imposible separar los componentes para medirlos independientemente, aunque tratan de reflejar la eficiencia del sistema a este nivel. A continuación se enumeran las métricas usadas en diferentes niveles, aunque no todas se usan actualmente:

A nivel de *máquina convencional* se pueden usar los MIPS (millones de instrucciones por segundo) o MFLOPS (millones de instrucciones en coma flotante por segundo). Estas medidas han caído en desuso, por su poca representatividad y porque es difícil que se reproduzcan resultados incluso para un mismo sistema, hasta el punto que se les conoce despectivamente como *Meaningless o Marketing index of performance*. Modernamente se usa también los CPI, o ciclos por instrucción, que mide el número medio de ciclos que tarda una instrucción en ejecutarse.

En el nivel de *sistema operativo* se usan medidas tales como velocidad en la ejecución de un determinado programa y número de programas o usuarios concurrentes que tal sistema operativo es capaz de soportar.

En el nivel de *lenguajes orientados a programas*, se mide el tamaño de un programa generado por un compilador, la eficiencia o velocidad de los programas generados por los mismos,

A nivel de *programas de aplicación*, se mide como diferentes programas del mismo segmento son capaces de ejecutar la mismo tipo de carga de trabajo. Generalmente se miden velocidades.

Es habitual, en benchmarks, establecer un sistema informático como base de comparación, de forma que, en vez de dar las cantidades absolutas de las medidas, se den tales cantidades referidas al sistema base. La mayoría de los *índices* presentados en las revistas de Informática y los consorcios de evaluación de prestaciones se hacen de esta forma. En otros casos, el índice de prestaciones incluye el precio del sistema informático, de forma que se puedan comparar no sólo las prestaciones, sino también la relación precio/prestaciones en el caso de que el precio del equipo sea un factor a tener en cuenta.

Independientemente de las métricas que se extraigan del experimento, un benchmark mide prestaciones a todos los niveles funcionales del sistema informático, y por lo tanto todos influyen en los resultados. Desde el sistema operativo, hasta el compilador, pasando por las librerías ya compiladas que se usen para la ejecución de los programas. Por ello hay que tener en cuenta y especificar cada una de ella junto con los resultados de la medición.

### 4.3.3 Errores comunes

En el proceso de medición de las prestaciones de un sistema informático se pueden producir una serie de errores, debidos a la inexperiencia de los realizadores, o bien por no haber tenido en cuenta ciertos factores importantes en las prestaciones de un sistema. Los errores más habituales son los siguientes [Jain91]:

*Representar solamente comportamiento medio* en la carga de trabajo. Cuando se trata de abstraer el comportamiento de un sistema es necesario resumirlo en unos cuantos índices. Sin embargo, no solamente hay que tener en cuenta los índices de tendencia central, o las medias, sino también alguno de los momentos, como la desviación típica. No es lo mismo un sistema que esté funcionando a la mitad de su capacidad todo el tiempo que otro que esté la mitad del tiempo sobrecargado y la mitad del tiempo ocioso. Habrá que tener también en cuenta la distribución que sigue algún elemento de la carga: por ejemplo, el tamaño de ficheros leídos por un programa o solicitados a un sitio Web suele distribuirse según una exponencial: los ficheros pequeños suelen registrar muchas más peticiones que los ficheros grandes.

*Ignorar la distribución desigual de las peticiones de dispositivos.* Cuando un sistema tiene acceso a diversos servidores, todos los cuales proporcionan un servicio, suele suceder que el comportamiento que siguen es encadenado, es decir, se accede muchas veces seguidas a uno de ellos, posteriormente muchas veces seguidas al siguiente, y así sucesivamente. Esto sucede, por ejemplo, cuando un computador tiene acceso a diferentes discos duros. Si se ignoran estas distribuciones, se registrarán tiempos de respuesta muy superiores a los que suceden en la realidad.

*No controlar el nivel de carga de forma apropiada.* En algunos benchmarks en los que se mide el efecto en un sistema de diferente número de usuarios simulados, se puede cambiar una serie de parámetros de la simulación, como el número de usuarios, su “tiempo de reflexión” y las peticiones de recursos de cada uno. Lo más realista es incrementar el número de usuarios, pero dado que esto requiere más emuladores de terminales remotos, o más espacio de almacenamiento, habitualmente se decrementa el “tiempo de reflexión” o se incrementa el número de peticiones. Esto afecta a la carga de forma diferente: los fallos de cache serán mucho menores, por ejemplo, y el incrementar el número de peticiones no sería realista.

*Ignorar los efectos de la cache:* las caches hacen que el orden en el cual se realizan las operaciones tenga importancia, porque puede conducir a que la información que necesita un programa se encuentre o no en la cache. Y las caches se encuentran tanto en la CPU, como en diversos niveles de la jerarquía de memoria; la memoria virtual padece también del mismo problema.

*Ignorar el overhead del monitor.* Tanto el hecho de tomar el tiempo que tarda en realizarse una determinada medida, como el que cada medida tenga que registrarse en el lengo almacenamiento secundario, introduce error en las medidas. Si el overhead es del 10%, los resultados del modelo pueden ser más o menos inexactos, aunque el modelo de la carga implementado en el benchmark lo sea.

*No validar las medidas:* se deben comprobar todas las medidas realizadas por dispositivos hardware, como en cualquier otro experimento; además, si algún resultado no es lo que cabía esperar, se debe de repetir y comprobar por si hay algún error introducido en el dispositivo experimental.

*No asegurarse de las mismas condiciones iniciales,* es decir, de que el estado de la cache sea el mismo, el de los procesos que se están ejecutando también, incluso, si es posible, la fragmentación del disco duro y el espacio que queda libre, pues, como se sabe, estos son dos factores que influyen en la velocidad del mismo. Habitualmente, por ejemplo, los benchmarks para sistemas multiusuario como SPEC se ejecutan en modo monousuario, con la conexión de red eliminada, y como el primer programa después de arrancar el ordenador (en Windows 95, si no se ejecuta el programa justamente después de encenderlo, corre el riesgo de que se quede colgado al mover el ratón. Incluso con esa precaución hay una probabilidad altísima de que se quede colgado). Si las mediciones son muy sensibles a las condiciones iniciales, habrá que incluir otros factores en el modelo de carga.

*No medir las prestaciones del transitorio,* ya que la mayoría de los experimentos están diseñados para predecir las prestaciones bajo condiciones estables. Aunque en algunos casos, cuando el sistema tarda en alcanzar el estado estacionario, y debe de cambiar continuamente de estado, puede ser importante medir las prestaciones de este transitorio.

*Utilizar los porcentajes de uso de los dispositivos para comparar prestaciones.* Los porcentajes de uso son métricas en el sentido de que, para la misma carga de trabajo, se prefiere un uso más bajo. Sin embargo, no sólo hay que tener en cuenta el uso, sino la cantidad de peticiones que se han procesado durante ese tiempo. Por ejemplo, tomar como medida de prestaciones la tasa de fallos de cache puede hacer que un sistema que se quede continuamente ejecutando un bucle de espera tenga unas tasas de acierto de cache bastante altas, mientras que no se hace ningún trabajo útil, mientras que otro que lleve a cabo un trabajo real tenga tasas más bajas de aciertos, pero consiguiendo a la vez un número de peticiones servidas mucho más altas. Estos errores resultan de mezclar las métricas de diferentes niveles funcionales: para medir las prestaciones de una función determinada, se deben de escoger métricas relativas al nivel funcional de la misma. Por ejemplo, los benchmarks SFS de medición de prestaciones de un sistema de ficheros UNIX miden tanto el número medio de operaciones de lectura o escritura, teniendo en cuenta diferentes tamaños de ficheros, como el número de usuarios que teóricamente sería capaz de soportar el sistema. Medir el tiempo que está el disco ocupado efectuando operaciones de lectura o escritura no tendría mucho sentido, pues puede tanto estar mucho tiempo reintentando una lectura fallida con una tasa alta de ocupación, como no haciendo nada con una tasa baja.

*Recoger demasiados datos con muy poco análisis:* el resumir todos los resultados en un índice de la forma adecuada puede ser tan arduo como obtener los resultados en sí. Demasiados resultados pueden convertir la comparación en algo imposible. Sin embargo, este es un error habitual, pues las habilidades que requiere un análisis es diferente a las habilidades requeridas por la medición.

Mientras que estos errores son involuntarios, puede suceder que se cometan errores con el objetivo de probar que el ordenador que se está midiendo es mejor que el de la competencia. Es lo que se llaman *juegos de benchmarking*, como se verá a continuación.

#### **4.3.4 Juegos de benchmarking**

Se puede afirmar de antemano que cuando una compañía dé un índice de prestaciones *desnudo*, sin indicar las condiciones en las que se ha llevado a cabo, está mintiendo. El decir, por ejemplo, que un microprocesador puede ejecutar 22 MIPS, necesita ser cualificado por el programa o conjunto de programas con los que se alcanza tal medida. En cualquier caso, hay otras muchas formas de engañar al público con los resultados de un benchmark, lo cual se ha venido a denominar *benchmarketing* [Jain91]:

*Usar configuraciones diferentes para ejecutar la misma carga de trabajo*, por ejemplo distinta cantidad de memoria, o discos de diferente calidad o tamaño.

*Elegir las especificaciones de forma que favorezcan a una máquina determinada.* Generalmente, los sistemas están optimizados para una aplicación determinada; el probarlos con otra aplicación dará resultados incorrectos; y compararlos con respecto a esos resultados, también.

Usar una *secuencia de trabajos sincronizada*, de forma que el solapamiento entre el trabajo de la CPU y del subsistema de E/S produzcan mejores prestaciones.

*Elegir una carga de trabajo arbitraria*, que puede dar buenas prestaciones para una máquina determinada, pero no tener nada que ver con las prestaciones reales de la misma. Por ejemplo, escoger la “criba de Eratóstenes”, o Whetstone, sin justificarlo en el tipo de trabajo para el cual se va a usar la máquina.

*Usar benchmarks demasiado pequeños:* que pueden hacer que los fallos de página y de cache sean mínimos, de forma que se ignore la ineficiencia de la organización de las mismas. También pueden evitar los efectos de cambio de contexto y el overhead de la entrada/salida. La mayoría de los sistemas usan una amplia variedad de cargas de trabajo; para comparar dos sistemas, se deberían usar tantas cargas de trabajo como fuera posible.

*Proyectar o interpolar resultados de un benchmark* [Bailey91], es decir, medir las prestaciones de un ordenador con un procesador determinado, y proyectar los resultados a otro ordenador con un número de procesadores diferente, o un procesador de la misma familia con más potencia. Aunque se puede predecir el rendimiento ideal teniendo en cuenta la ley de Amdahl y la participación del nuevo procesador en las prestaciones del sistema, habitualmente los componentes de un sistema interaccionan de una forma tan compleja que es imposible usar esas proyecciones. Las prestaciones deben de medirse sobre una máquina real.

	T(A)	T(A)/T(B)	T(B)	T(B)/T(A)
Benchmark 1	5	5	1	0.2
Benchmark 2	5	0.5	10	2
Media Aritmética	<b>5</b>	<b>2.25</b>	<b>5.5</b>	<b>1.1</b>
Media Geométrica	5	1.58	3.16	0.6

*Elegir el sistema base de normalización de forma arbitraria [HenPat96].*

Supóngase, por ejemplo, que al medir dos sistemas usando dos componentes de un benchmark se han obtenido los resultados de la tabla 1. En esta tabla, se presenta una situación bastante habitual, el que los resultados obtenidos en dos mediciones sean bastante dispares para una y otra máquina. Sin embargo, el usar una u otra máquina como índice puede resultar en que la máquina B es un 125% más rápida que la A, si se usa B como base, o que la B sea un 10% más rápida, si se usa la A. La media geométrica, sin embargo, mantiene la relación entre los tiempos de ejecución, sea cual sea la máquina que se toma de base, en este caso,  $\text{tiempo}(A)/\text{tiempo}(B) = 2/3$ . Por esto, la media geométrica se suele usar en la mayoría de los benchmark estándar, como los benchmarks SPEC.

Sin embargo, en la mayoría de los casos, es difícil engañar con los resultados de los benchmarks, ya que ni el código de los mismos ni la presentación de sus resultados están bajo control de los fabricantes. Por eso lo más práctico es engañar directamente al benchmark, como se verá en la siguiente sección.

#### 4.3.5 Engañando a los benchmarks

Dado que las decisiones de compra de muchos consumidores de material informático dependerán de las prestaciones del mismo, tal como han sido medidas por organizaciones, independientes, hay una gran presión para obtener los mejores resultados posibles. Ya que es difícil manipular los resultados, por la verificación a que éstos son sometidos, se intenta manipular los ordenadores de forma que produzcan tiempos de ejecución mejor en los benchmark. Por supuesto, esto no tiene por qué manifestarse como una mejora de prestaciones en las aplicaciones reales que el ordenador va a ejecutar. Las formas más habituales en que se suelen producir estos engaños son los siguientes:

*Detección y optimización del código de un benchmark:* tanto los compiladores como el código del sistema operativo pueden examinar el código que están ejecutando o compilando, ver su parecido con alguno de los benchmarks habituales, y optimizar sólo y exclusivamente ese código, dando resultados mucho mejores que los que se obtendrían con un código similar en una aplicación real o dando resultados optimizados aún en el caso de que el compilador no haya activado esta opción. Esto hizo IBM cuando produjo una versión nueva de su compilador de C para las estaciones de trabajo IBM System/6000, o algunos fabricantes de tarjetas de video que incluyeron código de aceleración de WinBench directamente en un ASIC. Hay incluso una empresa, KAP, que mejora compiladores para que optimicen código de algún benchmark, específicamente Matrix300, esto, a largo plazo no sirve de mucho, porque es. Incluso en algún

caso, había compiladores que eliminaban totalmente el bucle principal del benchmark, como sucedió con Whetstone y un compilador de IBM.

*Análisis del benchmark independiente de la máquina.* En algunos casos, los lenguajes como el Java o los compiladores como el gcc de la GNU, crean un código independiente de la máquina, que posteriormente se ensambla o interpreta por una máquina virtual. En tal caso, se puede analizar ese código, como se hace en [Conte91], para concluir qué características debe de tener un ordenador que ejecute ese código lo más rápido posible. Esto permite, por ejemplo, dimensionar las caches para que se minimicen los fallos de página. Sin embargo, una vez más, las mejoras sólo serán aplicables para los programas incluidos en el benchmark, aunque en este caso los resultados sean de aplicación más genérica.

Precisamente por esta razón, la mayoría de los benchmark estándar cambian cada cierto tiempo. SPEC se revisa cada 3 años, y hasta ahora ha habido tres versiones: SPEC89, 92, 95 y 2000. De esta forma, cuando los fabricantes han *aprendido* a engañar un benchmark, se propone uno nuevo.

## 4.4 Algunos ejemplos de benchmarks

Hoy en día hay una serie de benchmarks, que están aceptados e incluso propuestos por la comunidad de fabricantes de ordenadores y estaciones de trabajo. La mayoría están propuestos por consorcios, como los SPEC, TPC y GPC, que se verán en las tres primeras secciones, aunque hay otros propuestos por editoriales de revistas del ramo, como el ByteMark y el WinBench, que se verán en las siguientes secciones. En cada caso, se examinarán qué tipo de razonamiento ha conducido a elegir la carga de trabajo, las métricas usadas, y las críticas que se han vertido sobre ellos, algunas veces por los mismos fabricantes que los han propuesto.

### 4.4.1 Benchmarks propuestos por SPEC

SPEC, que son las iniciales de *System Performance Evaluation Cooperative*, es un consorcio de fabricantes de microprocesadores, ordenadores y estaciones de trabajo. La misión de SPEC es principalmente desarrollar una serie de programas que se van a utilizar para medir diversos aspectos de las prestaciones de un ordenador, y publicar los resultados de esos tests según han sido proporcionados por los fabricantes.

SPEC consiste en realidad en tres grupos diferentes:

*OSG, o Open Systems Group*, que crea benchmarks para procesadores y sistemas que ejecutan UNIX, Windows NT y VMS.

*HPC, o High Performance Group*, que mide prestaciones de ordenadores dedicados a cálculo intensivo, y

*GPC, o Graphics Performance Characterization Group*, que mide prestaciones de subsistemas gráficos, OpenGL y XWindows.

SPEC solo especifica los programas que se van a utilizar para medir prestaciones y la forma de ejecutarlos, pero no los ejecuta, sino que confía en los fabricantes para ejecutarlos, con la condición de que los resultados serán supervisados por SPEC, y se presentarán en su página Web: <http://www.specbench.org>. En realidad, dado el carácter público de los benchmark, cualquiera puede ejecutarlos (siempre que el ordenador cumpla las características que se especifican en la página Web, como 256 MBs de memoria y 1 GBs de disco).

Los benchmark de SPEC han tenido diversas versiones, la última de las cuales es la SPEC2000; las versiones anteriores son la 89, 92 y 95; siempre hay que tener en cuenta la versión a la hora de comparar prestaciones, y se suele especificar.

La mayoría de los fabricantes de ordenadores incluyen en sus páginas Web las medidas SPEC de sus equipos, eso sí, con las críticas correspondientes y poniendo claramente de relieve donde sobresalen. Incluso se basan en los resultados dados en estos benchmarks para tomar decisiones de diseño con respecto a sus máquinas.

#### 4.4.1.1 Cint/Cfp

En concreto, SPEC95 se compone de dos grupos de programas: CINT95, 8 programas de cálculo de enteros intensivo (como se muestran en el cuadro 2), y CFP95, 10 programas de cálculo intensivo de coma flotante (como se ve en el cuadro 3).

Tras medir lo que tardan en ejecutarse estos programas, se dan una serie de cantidades, algunas relacionadas con la velocidad, y otras relacionadas con el throughput, es decir, el número de programas del mismo tipo que un ordenador es capaz de manejar. Por ejemplo, para SPECint, se dará lo indicado en el cuadro 4.

Todas las tasas son medias geométricas de los resultados obtenidos por cada uno de los diferentes métodos, no aritméticas. Estas medias geométricas se suelen utilizar cuando las prestaciones de los diferentes componentes no se suman, sino que se acumulan, como se verá más adelante. Y además, mantienen la relación entre los tiempos de ejecución de los benchmarks independiente de la máquina que se haya escogido para normalizar.

Tras medir lo que tardan en ejecutarse estos programas, se dan una serie de cantidades, algunas relacionadas con la velocidad, y otras relacionadas con el throughput, es decir, el número de programas del mismo tipo que un ordenador es capaz de manejar. Por ejemplo, para SPECint, se dará lo indicado en el cuadro correspondiente

		Compilación	
		Conservador	Agresivo
Tasa	Velocidad	SPECint_base95	SPECint95
		SPECfp_base95	SPECfp95
	Throughput	SPECint_rate_base95	SPECint_rate95
		SPECfp_rate_base95	SPECfp_rate95

Se miden un total de 72 cantidades. Cada programa se compila de dos formas diferentes, una *conservadora*, con una serie de restricciones (por ejemplo, no más de 4 *flags* de ejecución), y además igual para todos los programas y máquinas, y otra *agresiva*, en la cual se puede usar la máxima optimización permitida por la máquina y el compilador; a la vez, para cada programa se mide el tiempo que tarda en ejecutarse, y el tiempo que tarda en ejecutarse cuando se ejecutan el máximo número de copias simultáneas del programa. Todas las velocidades se normalizan con respecto a un sistema, que depende de la versión del benchmark.

Tanto en el caso de Cint como de Cfp, los índices de “velocidad” suelen medir la eficiencia monoprocesador del sistema, aunque tenga diferentes procesadores; los índices de *throughput* miden la eficiencia del sistema operativo en cambios de contexto, pero también la eficiencia del mismo distribuyendo la carga entre diferentes procesadores, y la eficiencia del acceso a la memoria compartida de los mismos, en el caso de sistemas multiprocesador.

Los resultados son publicados cada trimestre en una publicación periódica, y en el sitio Web. Los del tercer trimestre de 1997 se muestran en las tablas [5](#) y [6](#). Como se puede observar en tales tablas, mientras que las arquitecturas Alpha de Digital y PA-RISC de HP son los más eficientes en SPECint95, vale decir, proceso de enteros y monoprocesador, las arquitecturas MIPS de SGI y las SPARC de Sun son las más eficientes en sus modelos multiprocesador. En estas mismas tablas se puede también ver lo difícil que resulta a veces comparar dos arquitecturas: mientras que las Alpha suelen dar mejores resultados en las compilaciones agresivas, las HP suelen ser mejores en las *base*, así, el sistema nº 8 (Alpha) supera al 7 (HP), y el 6 al 5 en esos aspectos.

Dado que los *rates* afectan más al sistema operativo que a los resultados particulares de una compilación, y éste no cambia en los índices *base* y los *agresivos*, la inversión de prestaciones comentada anteriormente no se da en este caso.

En cuanto a los índices medidos según SPECfp, mostrados en las tablas [7](#) y [8](#), se observa una vez más los Alpha de Digital dominan la escena, acampanados esta vez por los UltraSPARC de Sun; evidentemente, las arquitecturas de HP están optimizadas para una carga de tipo comercial (representada o sintetizada por SPECint), mientras que las Sun lo están para cálculo científico. Y en cuanto a los índices de *throughput*, aparecen los IBM, junto con las Sun y SGI. En este caso, las arquitecturas con mayor número de procesadores son las que consiguen mayor *throughput*.

Algunos fabricantes de ordenadores, a pesar de usar estas medidas, suelen criticar algunos de sus aspectos: la principal crítica es que la media oculta los resultados individuales de cada uno de los tests; uno de los tests puede dar mejor resultado para una máquina A que para otra B y sin embargo, la media ser mejor para B; por ello se presentan, en los informes de *revelación total*, los resultados individuales, para que cada persona pueda mirar los resultados que le interesan.

Estos tests miden también sólo la combinación CPU/memoria/compilador, no el sistema completo, en el cual pueden influir mucho las prestaciones de entrada/salida o las de red; sin embargo, hay otros tests que pueden medir estos subsistemas; esto causa también que algunos fabricantes, como IBM, logren optimizar sus compiladores para que reconozcan en código SPEC y conseguir así mejores resultados. Además, los “\_rate” miden las prestaciones de ordenadores ejecutando muchas copias del mismo programa, no diferentes programas, y por lo tanto son poco representativos de sistemas reales.

En general, hay que hacer un análisis pormenorizado de los resultados de cada uno de los benchmarks. En tal tipo de análisis, llevado a cabo por Sun, encontraron que el test más representativo es `gcc`, por tener código correspondiente a una aplicación real, un compilador usado en toda la gama de estaciones de trabajo, y ser además muy difícil de optimizar. Por otro lado, `compress` es uno de los que más exige al sistema, haciendo énfasis en el ancho de banda entre CPU y memoria, que es esencial en todas las aplicaciones.

En estudios llevados a cabo por Intel [Bekerman95], analizando cómo responden la jerarquía de memoria y el procesador (buffer TLB, cachés, pipelines U y V) a estos benchmarks, halla también que `gcc` y `compress` tienen una baja localidad de las referencias, haciendo que el número de veces que se tiene que acceder a memoria principal es bastante alto. Esto hace, una vez más, que sean bastante buenos indicativos de las prestaciones reales del ordenador. Por otro lado, los CPI (ciclos por instrucción) alcanzados en un Pentium con estos dos benchmarks son 2.62 y 2.15, respectivamente, a pesar del diseño superescalar, de dos pipelines de enteros y uno de números reales, que teóricamente le permitiría varias instrucciones por ciclo. Estudios similares se hacen en cada generación de las diferentes familias de microprocesadores, como por ejemplo en la tercera generación del PowerPC, denominada G3. En este modelo se incluyó un segundo pipeline de enteros tras examinar trazas de ejecución de diversos modelos anteriores.

En algunos casos, las medidas aportadas por los fabricantes son erróneas, como sucedió con los SPEC92 suministrados por Intel para el Pentium Pro a 200Mhz. A causa de un error en los compiladores usados, los resultados dados por uno de los tests eran exagerados. Más adelante, Intel tuvo que reducir el SPECint92 en un 10%. Lo que no consta es hasta qué punto el error era intencionado o no.

TPC (Transaction Processing Performance Council) es un consorcio que propone benchmarks en el área de procesamiento de transacciones. Hay varios benchmarks propuestos; hoy en día los más usados son TPC-C y TPC-D. Ambos tienen como entorno de referencia el siguiente: un sistema cliente, que genera peticiones de bases de datos usando un *front-end*, y un sistema servidor, que ejecuta el intérprete del lenguaje de bases de datos, generalmente SQL, y desde el mismo accede a un sistema de gestión de bases de datos. Cada transacción debe de ser *atómica*, es decir, se lleva a cabo o no completa, y además los sistemas deben de estar disponibles 24/7, 24 horas al día 7 días a la semana.

Estos benchmarks son de muy alto nivel, o más bien, ejercitan todos los niveles del sistema, desde la jerarquía de memoria, incluyendo memoria secundaria, pasando por el sistema operativo, hasta los programas que resuelven un problema concreto.

Las métricas que se proporcionan con estos benchmarks son de dos tipos: peticiones/tiempo, número de usuarios, y precio/prestaciones; hay unas reglas estrictas para evaluar éste último. Se verá a continuación cada uno de los benchmarks por separado.

Herederos de TPC-A y B, en este benchmark se mide el número de transacciones *New Order* se pueden generar por minuto mientras que el sistema está ejecutando peticiones de otros cuatro tipos (Pago, Estado de la Petición, Reparto, Nivel de Stock). Las cinco peticiones tienen que llevarse a cabo en un tiempo máximo determinado; *NewOrder* debe de durar 5 segundos como máximo.

Las métricas que se usan son `tpmC` (transacciones por minuto de tipo C) y `$/tmpC`, dividiendo las mismas por el precio total del sistema.

Similar al anterior, salvo que modela un sistema de *apoyo a la decisión*, lo que se ha dado en llamar *data warehousing* o *data mining*, es decir, peticiones complejas a bases de datos de gran tamaño. En este tipo de entornos suele haber más peticiones de lectura que de escritura, y además se suele acceder a una gran porción de la base de datos en una sola petición.

El entorno de experimentación consiste en responder a 17 peticiones, en diferente orden, mientras que en segundo plano se efectúan operaciones de inserción y borrado. Las órdenes están codificadas en SQL. Las bases de datos a las que se accede tienen diferentes *factores de escala*, indicados con una @; al proporcionar las prestaciones hay que indicar el factor de escala, y no se pueden extrapolar los resultados de un factor de escala a otro superior o inferior. Los resultados obtenidos por TPC tienen valor de certificación; generalmente suelen ser bastante onerosos para una empresa, y a veces incluyen una auditoría por parte de una empresa independiente.

TPC-D tiene tres métricas, dos de prestaciones y una de precio/prestaciones:

*QppD@tamaño*, o métrica de *potencia*, media geométrica del tiempo empleado por las 17 peticiones, la inserción y el borrado.

*QthD@tamaño*, una medida de tipo *throughput*, que caracteriza la habilidad del sistema para soportar una carga de trabajo multiusuario de tipo equilibrado. Se elige un número de usuarios S, cada uno de los cuales ejecuta las 17 peticiones en orden diferente; la inserción y el borrado también se llevan a cabo en segundo plano. El número de usuarios lo elige el que paga la realización del test (*sponsor*). S puede ser 1.

*QphD@tamaño*, media geométrica de las dos métricas. Su principal propósito es dar un solo índice con el cual se pueda calcular la relación precio/prestaciones.

De esta última métrica, se obtiene una métrica precio prestaciones. Por ejemplo, si el coste de un sistema es 5 millones de dólares, se ha tomado las mediciones para una base de datos de 100 GB, y se ha obtenido un índice *QphD@100GB* de 141.42, la métrica precio-prestaciones será  $35.335'34\$/QphD@100\text{ GB}$ . Otros índices se muestran en la tabla 9.

Dada la importancia comercial que tienen estos benchmarks, hay algunas empresas que han vertido críticas sobre ellas, atendiendo sobre todo a que no representan realmente la carga de trabajo generada por un sistema de toma de decisiones. Otra crítica es que son demasiado caros, costándole a una empresa del orden de un millón de dólares. Por tanto, algunas empresas proponen medir también el proceso de carga, la realización de consultas y los procesos de análisis.

BYTEmark es un conjunto de pruebas creada por la revista Byte, y ejecutada por la propia revista sobre cualquier ordenador que se le ponga a mano. Los programas utilizados son de dominio público, y están accesibles desde la página Web de Byte. El documento que describe BYTEmark está en <http://www.byte.com/bmark/bdoc.htm>.

BYTEmark pretende ser un test estándar para todo tipo de sistemas, y por ello consiste en una serie de programas escritos en ANSI C. Esto significa que evalúa tanto la CPU/FPU y memoria como la eficacia del compilador; el razonamiento de Byte es que si el compilador es malo y genera malos programas, también sufrirán los demás programas que se ejecuten en la misma plataforma. A su vez, se suelen dar dos resultados, uno de enteros y otro de coma flotante.

Las ventajas que tiene este test es que se ha diseñado para que sea escalable y multiplataforma, y fácil de usar en ordenadores personales. Se dan tanto los resultados “crudos”, es decir, el número de “iteraciones” del test, y además la relación con respecto a una máquina “base” (inicialmente un Dell con un Pentium a 90 MHz).

<b>Clasificación numérica</b>	Clasifica una matriz de enteros de 32 bits
<b>Clasificación de cadenas</b>	Clasifica cadenas de longitud arbitraria
<b>Campo de bits</b>	Ejecuta funciones de manipulación de bits
<b>Emulación coma flotante</b>	Un paquete de coma flotante
<b>Coefficientes de Fourier</b>	Rutina de análisis numérico para calcular aproximaciones en serie de ondas
<b>Algoritmo de asignación</b>	Asignación de tareas
<b>Compresión de Huffman</b>	Algoritmo de compresión de texto y gráficos
<b>Encriptación IDEA</b>	Algoritmo de cifrado por bloques
<b>Red neuronal</b>	Simulación de un perceptrón multicapa con retropropagación
<b>Descomposición LU</b>	Algoritmo robusto para resolver ecuaciones lineales

Uno de los principales problemas a los que se enfrentó este benchmark fue cuando se informó que el Pentium era más rápido que el Pentium Pro ejecutando código de 16 bits. Esto se debió a un error del compilador, que hacía que el programa fuera más lento cuando se asignaba memoria con la función `malloc()` sin alinear el bloque a un grupo completo de 4 bytes. En versiones posteriores, se corrigió el error.

Este benchmark lo usa la revista como base de todas las comparaciones, aunque habitualmente va corregido por un multiplicador, para que el índice tenga en cuenta otros factores, como usabilidad, y relación precio/prestaciones. Esto es bastante razonable, pues sólo la velocidad no debe ser el único factor a la hora de comparar diferentes ordenadores.

Este benchmark se podría considerar una carga sintética natural, más que una carga artificial ejecutable, como en el caso de los demás benchmarks. Consiste en ejecutar el conocido juego Doom, en modo demo, y medir el tiempo empleado. El propio programa proporciona el número de *gameticks* y el número de *realticks*, cuantos menos de estos últimos se obtengan, más rápido es el sistema. El benchmark especifica cual es la versión de Doom (1.9) que se debe usar, y los demás parámetros: tamaño de pantalla, tarjeta de sonido, teclado y ratón.

Este tipo de benchmarks dan resultados más realistas que los programas sintéticos, tales como 3DBench, ya que ejercita todas las capacidades del sistema, incluyendo el subsistema gráfico. Doom incluye aplicación de texturas en tiempo real, *rendering* de polígonos y sombreado de Gouraud, y por lo tanto es más complejo que el simple dibujo de polígonos de algunos otros benchmarks. Además, evidentemente, gran

número de programas ejecutados en sistemas personales son juegos similares al Doom, y por tanto los resultados son más realistas que en el caso de benchmarks artificiales.

Según los resultados, publicados en el Web, el sistema más rápido para ejecutar Doom es una SGI-Indy, con el microprocesador MIPS R4600 a 133 MHz, seguido por varios sistemas con el microprocesador AMD K6, SPARC, Pentium Pro 233 Mhz, y Pentium MMX.

Hoy en día, la mayoría de los productos de Id Software incluyen demos que se pueden ejecutar desde la consola tecleando `timedemo`. Muchos otros también tienen facilidades similares

En el cuadro [10](#) se resumen los benchmarks comentados, de qué tipo son, quién los ha propuesto, quién los ejecuta y el sistema o subsistema que ejercitan.

Benchmark	Propuest o por	Tipo	Subsistema/función	Ejecutado por
SPEC CPU	SPEC	Artificial Sintético	CPU/Memoria/compilador	Fabricante
SPEC SFS		Artificial Sintético	Sistema de ficheros UNIX	
SPECWeb				
SPEChpc		Artificial Sintético	Sistema como servidor Web	
		Sintético Natural	Sistema completo en cálculo de altas prestaciones	
TPC-C	TPC	Artificial Sintético	Sistema completo para proceso de transacciones	Fabricante supervisado por un auditor
TPC-D		Artificial Sintético	Sistema completo para toma de decisiones	
BYTEmark	Revista BYTE	Artificial Sintético	CPU/Memoria/Compilador	Revista/usuario
DoomBench	Anton Ertl	Sintético Natural	Sistema completo	Usuario

## 4.5 Problemas generales del benchmarking

Los benchmarks, por su propia naturaleza, tienen una serie de problemas. Mientras que los benchmarks se supone que son suficientemente representativos como para comparar diferentes sistemas, esto no siempre es cierto. Los principales problemas que se plantean son los siguientes

*Representatividad de una simple figura de mérito:* los benchmarks tratan de resumir diversas mediciones (72, en el caso de SPECint/fp) en un sólo índice. Evidentemente, las mediciones arrojarán resultados muy diferentes para dos sistemas; algunas de ellas serán mejores en uno de los sistemas y otras en otro. En principio, se podría mejorar algo la representatividad dando varios índices, como sucede con SPEC, que da ocho; además se podría añadir la desviación típica, para dar una idea de la dispersión de tales resultados. Sin embargo, aún así, un sólo índice no arroja ninguna luz sobre los aspectos particulares en los que un sistema supera a otro.

*Representatividad de las cargas artificiales.* Las cargas de trabajo generadas por un benchmark son resultado de un análisis previo, y tratan de resumir las características de una carga de trabajo real. Por ejemplo, mientras que SPECint/fp 92 tenía más programas en coma flotante que de enteros, representando una mayor representatividad de cargas de tipo científico, SPECint/fp 95 tiene más programas de enteros, representando el cambio del mercado hacia la utilización por parte de usuarios comerciales de estaciones de trabajo, que anteriormente se usaban solamente en tareas científicas y de ingeniería. Sin embargo, como tal carga sintética, es evidente que no representan la carga de ningún sistema real. Una vez más, hay que mirar a cada uno de los índices parciales, para concluir cuál es el mejor sistema para el problema o gama de problemas que se van a tratar.

Las prestaciones de un sistema paralelo tienen una variación muy amplia, desde las prestaciones monoprocesador hasta las prestaciones multiprocesador con algoritmos bien paralelizados, por ello hay autores que afirman que *Las prestaciones de un sistema paralelo no existen* (van der Steen, citado en [Hockney96]), es decir, no se pueden definir las prestaciones de un sistema paralelo. Esto indica la imposibilidad de definir una sola métrica que represente las prestaciones de un sistema paralelo en todas las circunstancias; como máximo, se puede indicar una métrica que indica las prestaciones para un algoritmo determinado, con una implementación determinada.

*Cargas artificiales frente a cargas reales:* la crítica más común es que *los usuarios no ejecutan benchmarks, ejecutan programas*. En una estación de trabajo se ejecutan una serie de tareas: edición, depuración, compilación, todo ello desde un entorno gráfico, mientras que en un ordenador personal se trabaja con procesadores de textos, programas gráficos y juegos, también desde un entorno gráfico, e interactivamente. Ello provoca interacciones complejas entre los diferentes programas, el sistema operativo y el usuario que son muy difíciles de

simular en un benchmark, generalmente de pequeño tamaño y consistentes en programas *kernel*, que representan la parte principal de los programas de usuario. Por ello, la diferencia *real* entre las prestaciones de dos sistemas puede ser muy diferente de la predicha por un benchmark.

*Las prestaciones no lo son todo:* también influyen otros factores como el precio, la facilidad de uso, la garantía que ofrece el fabricante y/o el revendedor, y el apoyo técnico que el mismo ofrece. En algunos benchmarks se tiene en cuenta el precio, y en otros se valoran, de forma más o menos subjetiva, los segundos, como sucede en los de la revista Byte. Por ello, independientemente de la velocidad, los factores que conducen a la compra de uno u otro sistema pueden ser otros, y ello no se puede reflejar en un benchmark. Evidentemente, los benchmarks siguen siendo útiles en los demás casos que se han mencionado en la introducción: diseño de sistemas informáticos, por ejemplo.

## 4.6 Bibliografía y enlaces en internet

Los siguientes sitios Web contienen o bien información sobre benchmark, o programas de dominio público que se pueden usar para llevarlos a cabo.

Grupos de news (USENET): `comp.benchmarks`.

FAQs, o ficheros de preguntas frecuentemente preguntadas:

<http://hpwww.epfl.ch/bench/bench.FAQ.html>.

Web de organizaciones que realizan benchmarks:

<http://www.specbench.org>, para la organización SPEC, que controla los benchmark Cint/Cfp, SFS, Web, HPC y otros,

<http://www.netlib.org>,

<http://www.byte.com>, para la revista Byte, que ha publicado ByteMark,

<http://www.tpc.org>, dirección del Transaction Processing Council, que controla los benchmarks de procesamiento de transacciones,

<http://www.gpc.org>, para benchmarks del subsistema gráfico, incluidos aceleradores 3D, y

<http://www.zdbop.com>, de la Ziff-Davis Benchmark Operation, una división de la empresa editorial Ziff-Davis, que publica, entre otras, el PCMagazine.

Web de empresas que comercializan computadores:

<http://www.intel.com>, <http://www.ibm.com>,

<http://www.hp.com>, <http://www.sgi.com>,

<http://www.dec.com>.

Página Web sobre optimización de sistemas:

<http://www.sysopt.com>.

Finalmente, la página Web de la asignatura:  
<http://geneura.ugr.es/~jmerelo/DyEC>, donde se incluyen todos los punteros anteriores, punteros a la asignatura tal como se imparte en otras universidades, y los textos completos de los apuntes de la asignatura.

Realizado por  
Juan Julián Merelo Guervós [jmerelo at geneura.ugr.es](mailto:jmerelo@geneura.ugr.es)  
**Depto de Arquitectura y Tecnología de Computadores**  
**Universidad de Granada**  
URL:<http://geneura.ugr.es/~jmerelo/DyEC/Tema4/DyEC-Tema4.html>