

1. Sistemas Informáticos y su Evaluación

1.1 Introducción: necesidad de la evaluación de un sistema informático

Durante el ciclo de vida de un sistema informático, resulta muchas veces necesario evaluar sus prestaciones, habitualmente con el objetivo de mejorarlas o bien de comparar diversos sistemas informáticos entre sí. Esa evaluación de prestaciones se debe hacer de forma objetiva, para que puedan compararse diversos valores a lo largo del tiempo o bien los valores para diversos sistemas informáticos. Tales mediciones pueden servir también para identificar los problemas que tiene un sistema informático, con el objetivo de solucionarlos.

En concreto, se necesitará evaluar objetivamente las prestaciones de un sistema informático a lo largo de las siguientes fases de su ciclo de vida:

- Diseño de un ordenador o de un sistema informático, durante el cual es necesario saber, a priori, cuales van a ser las prestaciones del mismo.
- Adquisición de un sistema, durante la cual será necesario evaluar cuál es la configuración necesaria, y elegir entre varias posibles.
- Explotación y ampliación de un sistema, durante la cual hay que examinar cuales son los problemas que se presentan, y solucionarlos sobre la marcha, o cuáles de los componentes del sistema es necesario cambiar para maximizar el aumento de prestaciones. A veces es necesario conocer las prestaciones o la utilización de un ordenador por la política de su uso: por ejemplo, para cobrar por su uso, por usuarios o departamentos, o para garantizar que se cumplen reglas como que no se pueden usar programas interactivos por más de un tiempo determinado.

Los objetivos de una evaluación suelen ser alguno de los siguiente

- Comparar alternativas
- Determinar el impacto de una nueva característica, por ejemplo, añadir un disco duro nuevo
- Sintonizar el sistema, es decir, hacer que funcione mejor según algún punto de vista (nunca se puede hacer que vaya mejor según todos los puntos de vista).
- Identificar prestaciones relativas entre diferentes sistemas.
- Depuración de prestaciones, es decir, identificar los fallos del sistema que hacen que vaya más lento
- Poner unas expectativas sobre el uso del sistema, por ejemplo, cuántas conexiones es capaz de soportar una bases de datos simultáneamente, o cuántas peticiones un sitio web

Hay otros sistemas operativos OS X, el sistema operativo de los Apple iMac, los sistemas operativos de los grandes mainframes: OS/390, por ejemplo. Sin embargo, Unix/Linux y WindowsNT/2K/XP copan el 90%, o quizá más, de los servidores actuales. También están los diferentes miembros de la familia BSD: [NetBSD](#), [FreeBSD](#), y [OpenBSD](#): otros Unix gratuitos, no tan extendidos, ni tan fáciles de instalar como Linux, pero que, como suele suceder, tienen un grupo de adeptos bastante fuerte. En el caso del OpenBSD, se pone énfasis en la seguridad; mientras que en el caso del NetBSD, se trata de potenciar sobre todo la portabilidad.

Los sistemas de referencia que se van a usar serán habitualmente ordenadores con el sistema operativo UNIX. Aunque en general todos los UNIX son bastante similares, es decir, funcionan en todos las mismas órdenes (porque todos usan el mismo intérprete de comandos) y librerías, hay sutiles diferencias entre los dos tipos principales de UNIX, los System V, y los BSD; sin embargo, hoy en día, el UNIX más extendido, con mucho, es Linux (en sus diferentes distribuciones), seguido de lejos por Solaris, la versión de Sun.

En esta asignatura se tratará también con otro de los sistemas operativos más usados en la actualidad, [Windows NT/2K/XP](#). En este caso, se trata de un sistema operativo multiproceso, multihebras, y, en el caso de XP, multiusuario.

Un ordenador se compone de muchos subsistemas diferentes, tanto software, como hardware, y todos interaccionan entre sí para dar el resultado que observa un usuario. El procesador, los diferentes elementos de la jerarquía de memoria, el sistema operativo, compiladores, la cantidad de usuarios, todos tienen un impacto en las prestaciones del sistema. En concreto, los sistemas con los que trabaja el ordenador se pueden dividir de la forma siguiente:

- Hardware: la parte física del ordenador, compuesta a su vez del subsistema CPU-memoria, entrada/salida, y conectada a este subsistema, la red, el sistema de almacenamiento rápido, y los gráficos, principalmente, aparte de otros subsistemas, que no suelen influir tanto en las prestaciones (por ejemplo, sistemas de copias de respaldo, impresoras, CDs y ese tipo de cosas).
- Software: también el sistema operativo y los programas de usuario hacen uso de diferentes capas, que van desde la más baja llamada capa de abstracción de hardware, para pasar al kernel o microkernel. Más arriba están los diferentes subsistemas del sistema operativo: entrada salida, seguridad, gestión de objetos, gestor de procesos, memoria virtual y demás (para saber más sobre el tema, consultar Byte, Jul 1996).

Aproximación pachanguera al análisis de prestaciones A pesar de la sabiduría y la ciencia que se encuentra en el análisis de prestaciones, lo más habitual es que poca gente le haga caso, salvo que se estén jugando muchos cuartos. Lo más normal es usar alguna de las opciones siguientes

- Comprar el segundo más barato. No te vas a comprar el más barato, porque serías un cutre, así que te compras el segundo más barato. Eso sí te llega el presupuesto.
- Opción burro grande: comprar el procesador más rápido para la pasta que tengamos. Esta opción la suelen usar, sobre todo, los gamers.
- Reinstalar Windows. Cuando el ordenador va lento, lo mejor es reinstalar Windows; el peso de los bits es lo que no le deja andar
- Recompilar el kernel: este para los linuxeros. Para los menos fanáticos, bajarse las últimas versiones de todo con el `apt-get` o el RedCarpet

En el caso de un sistema monousuario, que es lo más habitual, también es necesaria la evaluación de prestaciones con el objeto de aprovechar al máximo las posibilidades del hardware; la mayoría de los sistemas operativos serios ofrecen una serie de posibilidades de medición y sintonización, que se deberán de aprovechar, como se tratará en el [tema 3](#); en el caso de un sistema informático compuesto por una red con varias decenas de recursos, periféricos, usuarios, y que se use las 24 horas del día, y en el cual la adquisición de un nuevo componente sea un proceso largo y costoso, será imprescindible evaluar los problemas del sistema y sus posibles soluciones.

Una primera aproximación a la medición de prestaciones son las ofrecidas por el fabricante de cada uno de los componentes del ordenador. Un ordenador con una tarjeta de vídeo rápida será habitualmente más rápido que otro con la tarjeta de vídeo más lenta, o con más memoria (al tener más memoria, hay que transferir más información en cada instante, y por tanto hace más lento el sistema), si todos los demás componentes son iguales. Pero ¿habría mucha diferencia de velocidad, si la velocidad es lo que importa, entre una tarjeta rápida con mucha memoria y otra lenta con menos memoria? ¿Como influiría en eso la velocidad del microprocesador? ¿Y el tipo de bus al que se conecta la tarjeta? Todas esas preguntas y muchas más, tratarán de responderse a lo largo de la asignatura.

Todos los servicios que ofrece un sistema como UNIX están administrados a través de *daemons*. Los *daemons* son programas, cuyo nombre normalmente acaba en *d*, que administran un servicio determinado, como *ftp* o *telnet*. En un momento determinado, hay muchos *daemons* funcionando; se suelen iniciar al arrancar el sistema.

Dado que UNIX es un sistema operativo de red, existe un método de hacer disponibles los sistemas de ficheros (*filesystems*) locales a toda la red, que se denomina NFS, o network filesystem. Este NFS permite a cualquier ordenador en la red acceder a los demás discos duros puestos a su disposición como si se tratase de directorios locales, es decir, que existe un solo sistema de ficheros en toda la red.

Normalmente hay una persona o personas dedicadas a la administración de un sistema UNIX, los denominados superusuarios o administradores del sistema. Estas personas tienen privilegios para hacer cosas que habitualmente no tienen los demás usuarios. Todas las órdenes de UNIX tienen una página de manual; estas páginas están distribuidas en diferentes secciones, que se suelen indicar entre paréntesis después del nombre del comando. Las secciones relevantes a la evaluación de prestaciones de sistemas son la 4 o 7 (formatos de fichero) y 1m u 8 (administración del sistema).

En este capítulo, veremos primero qué fases se siguen en la evaluación de un sistema informático en la sección [1.2](#). Posteriormente se verán qué técnicas ([1.3](#)) y herramientas ([sección 1.4](#)) se usan para la medición de prestaciones de un sistema informático, comenzando con la carga del sistema; finalmente se verá (en la [sección 1.5](#)) qué es lo que se debe medir y qué tipos de magnitudes hay.

1 Ejercicios de autoevaluación

1. Indicar qué tipo de medidas sueles tomar para medir las prestaciones de un ordenador.
 2. Indicar en qué casos de los que te encuentras en tu trabajo diario necesitarás medir las prestaciones del ordenador.
 3. Indicar en qué casos percibes una falta de prestaciones de los ordenadores que sueles manejar.
 4. Mirar qué servicios hay activos en nuestro ordenador personal y en algún otro ordenador al que tengamos acceso.
- ¿Qué usas para saber los servicios que hay activos? ¿Sabes lo que hacen? ¿Si suprimes alguno de ellos, qué pasa?

1.2 Fases en la evaluación de un sistema informático

Para que se vea lo que tienen que sufrir los informáticos, y especialmente los administradores de sistemas que se tienen que dedicar, entre otras cosas, a solucionar los problemas de los demás, es interesante consultar la [Pringao-HOWTO](#) y la tabla de los [30 sentimientos de un informático](#).

Lo que hace que a veces se comporten como el [operador bastardo del infierno](#).

Durante la evaluación de cualquier sistema informático, hay que seguir las siguientes fases:

1. Especificar los objetivos y definir el sistema: una medición de prestaciones no tiene sentido sin objetivos. Se debe de definir claramente cuál es el sistema además, para medir exclusivamente eso. Es decir, si se quieren medir las prestaciones de la memoria de un sistema, hay que aislar lo que pertenece a ella, y eliminar en lo posible de la medición la influencia de todos los demás factores.
2. Hacer una lista de los servicios que ofrece el sistema y sus posibles resultados: es decir, un sistema puede dar un resultado válido, inválido o simplemente no dar ningún resultado, en cualquier caso, habrá que medir la tasa de sucesos de uno u otro tipo.
3. Seleccionar las métricas, es decir, los criterios para comparar prestaciones.
4. Listar los parámetros que pueden afectar a las prestaciones, que se dividen entre las características del sistema, y la carga de trabajo a la cual está sometido; los primeros no varían para todos los sistemas que tengan el mismo hardware; pero el segundo varía entre diversas instalaciones.
5. Factores a estudiar, de los parámetros anteriores, algunos se variarán durante el estudio, los denominados factores. Los diferentes valores que tomarán durante el estudio se denominan niveles.
6. Seleccionar las técnicas de evaluación: entre la modelización, simulación y medición de un sistema real. La selección de la técnica dependerá del tiempo y el dinero disponibles, aunque lo más habitual es que se lleven a cabo benchmarks.
7. Seleccionar la carga de trabajo, es decir, la carga a la que se va a someter el sistema para medirlo.
8. Diseñar los experimentos, dividiéndolos en niveles o valores que tomarán los factores. Inicialmente, se suele diseñar un experimento con muchos factores, pero pocos niveles, para, una vez vistos cuáles son los factores que influyen más en el experimento, concentrarse en esos.
9. Analizar e interpretar los datos; no basta con medir, sino que hay que sintetizar los datos de las medidas, y extraer conclusiones de ellos.
10. Presentar los resultados: lo cual es muy importante, tanto si se presenta a una clase como si se presenta a un gerente que debe de tomar una decisión sobre qué comprar. Llegados a este punto, puede ser necesario comenzar otra vez el estudio desde el principio.

1.3 Técnicas de evaluación de un sistema informático

Este apartado correspondería al punto 3 de la sección anterior; hablaremos de las técnicas más habituales usadas para evaluar

un sistema, que son las siguientes: medición, modelado y simulación.

La medición consiste en tomar medidas directamente sobre el sistema en el que uno está interesado, usando también la carga adecuada, o bien una parte de la misma, que es lo que se suele denominar, en general, carga sintética.

Cuando se trata de evaluar un sistema incompleto, o que no se ha construido aún, hace falta construir un modelo analítico del mismo, es decir, usando fórmulas y ecuaciones diferenciales, tratar de hallar a partir de los valores conocidos o estimados de ciertos parámetros, los valores de los que nos van a interesar.

Por último, se puede simular el sistema, usando algún lenguaje de simulación, como el **SIMULA**, o cualquier otro lenguaje orientado a objetos con las herramientas gráficas adecuadas. Generalmente se usa simulación antes de construir un sistema, especialmente cuando se construyen nuevos microprocesadores, y se basa su estudio en las versiones anteriores de los microprocesadores. Para ello se suelen usar máquinas masivamente paralelas o superordenadores. Por ejemplo, la empresa **VirtuTech** creó un simulador llamado **VirtuHammer** para simular uno de los últimos procesadores de AMD, los **Claw/SledgeHammer**; también ofrece una licencia de su producto **Simics** gratuita para uso académico.

El clásico **99 bottles of beers on a wall** se escribiría en simula de la siguiente forma:

```
BEGIN
COMMENT
  Simula version of 99 beers
  Maciej Macowicz (mmacep@pl.fr)
  Status: UNTESTED ;
;
INTEGER bottles;
FOR bottles:= 99 STEP -1 UNTIL 1 DO
BEGIN
  OutInt(bottles,1);
  OutText("bottle(s) of beer on the wall, ");
  OutInt(bottles,1);
  OutText("bottle(s) of beer");
  OutImage;
  OutText("Take one down, pass it around, ");
  OutInt(bottles,1);
  OutText("bottle(s) of beer on the wall, ");
END;
OutText("1 bottle of beer on the wall, one bottle of beer.");
OutImage;
OutText("Take one down, pass it around, no more bottles of beer on the wall");
OutImage
END
```

Hay implementaciones gratuitas de Simula tales como [esta](#). Sin embargo, no se puede decir que sea precisamente un lenguaje popular.

El problema en cada caso es qué técnica usar. La mayoría de las veces se recurre a la medición: las herramientas existen ya, y sólo hay que aplicarlas a nuestro sistema; sin embargo, si el sistema no existe, la única forma de medir sus prestaciones es mediante simulación y modelación.

En cuanto al tiempo que se tarda en obtener resultados, lo más rápido es usar un modelo analítico: simplemente se aplican ecuaciones; las mediciones tardan un poco más (sobre todo, teniendo en cuenta la variabilidad de las cargas de trabajo durante el tiempo); por último, la simulación es lo más lento, pues hay que hacer un programa y evaluar los resultados.

En cuanto a la exactitud, por supuesto mediciones sobre el propio sistema es lo más exacto, seguido por la simulación, ya que en ella se ponen casi todos los elementos del sistema real, y por último, el modelo analítico, porque requiere gran cantidad de suposiciones. También habría que tener en cuenta el coste (normalmente la medición es bastante cara), y, por supuesto, lo vendibles que son los resultados (en este caso, lo mejor son mediciones).

2 Ejercicios de autoevaluación

1. Especificar en qué consistirían los 10 pasos anteriores en el caso de la evaluación de los siguientes sistemas: un compilador, un proveedor de servicio ADSL, una tarjeta gráfica, una impresora. .
2. Buscar sistemas gratuitos de simulación, especialmente para hardware.

1.4 Medición de la carga de un sistema

Dado que la medición es el sistema más habitual de análisis de prestaciones, suele haber herramientas genéricas para hacerlos. Son los denominados monitores, que "son herramientas de medición que permiten seguir el comportamiento de los principales elementos de un sistema informático cuando éste se halla sometido a una carga de trabajo determinada". En el caso de ordenadores personales o estaciones de trabajo son habitualmente programas, que o bien están incluidos en la distribución del sistema operativo (UNIX, Windows NT/2000/XP), o son utilidades externas (Windows 95/98); también pueden ser trozos de código unidos a un programa, como en el caso de los profilers de aplicaciones, o aparatos de medición, como en el caso de los monitores de prestaciones de la red.

Aunque durante la historia de la informática se han utilizado muchos tipos de monitores, los más habituales hoy en día son programas destinados a evaluar un sistema informático completo o una parte del mismo, y programas o aparatos de medición para medir las prestaciones de una red de ordenadores.

Según la forma como miden las prestaciones, los monitores se denominan de eventos o acontecimientos, que se activan cada vez que sucede un proceso determinado en el ordenador (como leer de disco, por ejemplo), o de muestreo. Los segundos son los más habituales, se activan a intervalos de tiempo fijos, o aleatorios. Lo importante de los monitores es que introduzcan poco overhead, es decir, que interfieran poco en el funcionamiento del sistema; los primeros, en general, interfieren mucho más que los segundos. Un monitor, además, puede ser en tiempo real o en modo batch, según presente los resultados durante su ejecución o al final. Según la presentación de los resultados, también pueden ser monitores gráficos o basados en texto.

La mayoría de las veces, los monitores incluyen dos partes: un cliente y un servidor, sobre todo en el caso de los monitores gráficos. La parte cliente interroga a los servidores, que pueden ejecutarse en la máquina local o en otra máquina, y presenta los datos en forma gráfica, bien mediante diagramas de barras que indican porcentajes de utilización o mediante strip charts, es decir, diagramas temporales que representan en abscisas el tiempo y en ordenadas la utilización del recurso.

En cuanto a los monitores software, hay también principalmente dos tipos: profilers, o programas pegados a otros que miden sus prestaciones, y programas que miden el estado de un sistema. Se verán a continuación.

1.4.1 Profilers

Los profilers son trozos de código linkados a un programa, y que son llamados cada cierto tiempo. Habitualmente, hay que instruir al compilador para que incluya esta opción de profiling. Estos trozos de programa generan un fichero, que es luego analizado por otros programas; el análisis muestra el tiempo empleado en cada una de los procedimientos de un programa y el número de veces que se ha llamado, de forma que el hábil programador pueda optimizar esos procedimientos. Esto puede tener un gran impacto en las prestaciones del programa; si un programa que tarda normalmente 10 segundos en ejecutarse se tira el 50% del tiempo en un procedimiento determinado, y, mediante alguna hábil técnica de optimización, conseguimos reducir el tiempo empleado en ese procedimiento a la mitad, habremos logrado que el programa tarde en ejecutarse solo 7.5 segundos, un 25% menos. Para hacer eso hay diversas técnicas, algunas de las cuales se muestran en la [referencia 2](#).

Flat profile:

Each sample counts as 0.01 seconds.

%	cumulative	self	self	total		
time	seconds	seconds	calls	ms/call	ms/call	name
12.26	0.19	0.19	837698	0.00	0.00	eoRng::uniform(double)
7.10	0.30	0.11	1095925	0.00	0.00	vector<bool, __default_alloc_template<true, 0> >::begin
7.10	0.41	0.11	907355	0.00	0.00	vector<bool, __default_alloc_template<true, 0> >::size
7.10	0.52	0.11	41897	0.00	0.01	__copy_d_H3Z20_bit_const_iteratorZ14_bit_iteratorZi
5.16	0.60	0.08	778944	0.00	0.00	__ml_C20_bit_const_iterator
4.52	0.67	0.07	931658	0.00	0.00	__ml_C14_bit_iterator
4.52	0.74	0.07	701600	0.00	0.00	eoRng::flip(float)
3.87	0.80	0.06	910452	0.00	0.00	__pp_14_bit_iterator
3.87	0.86	0.06	837698	0.00	0.00	eoRng::rand(void)
3.23	0.91	0.05	1710602	0.00	0.00	__bit_reference::__bit_reference(unsigned int *, unsig
3.23	0.96	0.05	990533	0.00	0.00	__mi_C20_bit_const_iteratorG20_bit_const_iterator
3.23	1.01	0.05	912052	0.00	0.00	__bit_iterator::bump_up(void)
3.23	1.06	0.05	673552	0.00	0.00	__bit_const_iterator::bump_up(void)
3.23	1.11	0.05	40000	0.00	0.01	eoBitMutation<eoBit<double> >::operator()(eoBit<double
2.58	1.15	0.04	1027694	0.00	0.00	__opb_C15_bit_reference
2.58	1.19	0.04	81096	0.00	0.00	EO<double>::operator<(EO<double> const &) const
1.94	1.22	0.03	2045477	0.00	0.00	__bit_const_iterator::__bit_const_iterator(__bit_iterat

Ejemplo de listado generado por un profiler en Linux para un programa que implementa un ejemplo de algoritmo genético. En este caso, la rutina que más tiempo consume es la de generación de números aleatorios, que además aparece varias veces dentro del listado. Esto es bastante habitual en muchos programas. En todo caso, el tiempo empleado no es muy alto, alrededor del 12%, por lo tanto no merece mucho la pena mejorar la velocidad de esas rutinas, pues tendrán una incidencia mínima en las prestaciones totales del programa.

Evidentemente, para sacar partido de estas técnicas hace falta tener el código fuente para que los nombres de los procedimientos puedan ser leídos por el profiler. Algunos sistemas operativos (tales como el SysV) incluyen también depuradores de kernel. SGI ha desarrollado Linux tiene un sistema para Linux llamado `kernelprof`, que se aplica como un parche al kernel; las medidas se pueden examinar luego con una utilidad de línea de comandos con el mismo nombre.

Para usar un profiler en diferentes sistemas operativos y con diferentes compiladores, hace falta lo siguiente:

- En Linux o Unix, basta con compilar usando la opción `-p` (para `prof`) o `-pg` para `gprof`:

```
bash$ gcc -pg mipogama.cpp -o mipogama
```

Posteriormente, se ejecuta el programa, que genera automáticamente un fichero `gmon.out`. Este fichero se analiza usando el profiler:

```
bash$ prof mipogama gmon.out
```

- En WinXX, tanto si se usan los compiladores de Borland como los de Microsoft, se tiene que especificar al compilador que se va a hacer un profiling del programa en las opciones de compilación, y posteriormente se llama al profiler.

3 Ejercicios de autoevaluación

1. Buscar y usar un profiler sobre un programa propio, en el lenguaje de programación que sea. Indicar qué es necesario para usarlo, y, una vez aplicado sobre un programa, decir qué mejoras se pueden hacer sobre el mismo.

1.4.2 Métricas de la carga de trabajo más comunes

La medición de diversas magnitudes relacionadas con su funcionamiento, como las siguientes, es necesaria tanto para detectar los problemas que afectan al sistema como para evaluar los resultados de un benchmark:

- Throughput o número de peticiones procesadas en la unidad de tiempo; se puede aplicar tanto a procesadores como a unidades de entrada salida, como, por ejemplo, a tarjetas gráficas (triángulos dibujados en la unidad de tiempo).
- Tiempo de respuesta: opuesto del throughput (para un solo elemento), es el tiempo que se tarda en procesar una petición.
- Eficiencia es la tasa del throughput máximo al throughput que se consigue de forma efectiva.
- Ancho de banda: bits por segundo que es capaz de procesar el sistema.
- Porcentaje de utilización de diversos componentes y solapamiento de los mismos, es decir, la proporción de tiempo que cada uno de los dispositivos está funcionando y la proporción de tiempo durante la cual están funcionando simultáneamente. Cuanto mayor sea el solapamiento, mayor será la eficiencia del sistema, pues ningún dispositivo estará esperando a otro para funcionar.
- Overhead: es decir, tiempo usado en tareas que no son directamente del usuario.
- Factores relacionados con la multiprogramación: tales como el tiempo usado en cambiar de contexto.
- Factores relacionados con la memoria virtual: fallos de página, número de veces que se ha hecho swapping.
- Factores relacionados con la memoria caché de CPU: similares a lo dicho con la memoria virtual, y aparte veces que se vacían los buffers TLB, por ejemplo.
- Otros subsistemas: red, gráficos (que tienen mucha importancia últimamente).

1.4.3 A qué se dedica el tiempo

Antes de intentar mejorar las prestaciones de cualquier sistema, es decir, básicamente hacerlo más rápido, hay que saber en qué invierte el tiempo el sistema. Para medir el tiempo que tarda en ejecutarse un programa, una de las medidas básicas de las prestaciones de cualquier sistema, se puede ejecutar una utilidad tal como `/bin/time`, `/usr/bin/time` o `time` (para diferenciarlos del `time` que es parte del intérprete de comandos), que son el equivalente en las dos versiones del sistema operativo; y que son diferentes a una función del shell, también llamada `time`. Esta utilidad, usada así, da los siguientes resultados (para el mismo programa al que le hicimos el profiling anteriormente):

```
3.63user 0.02system 0:03.75elapsed 97%CPU (0avgtext+0avgdata 0maxresident)k
0inputs+0outputs (171major+33minor)pagefaults 0swaps
```

Lo cual significa lo siguiente: el programa ha tardado 3.75 segundos reales, desde que se ha tecleado Enter hasta que ha salido; pero de esos 3.75 segundos, en realidad sólo 3.63 + 0.02 ha estado en la CPU; el resto del tiempo ésta no le ha hecho caso. Y además, 0.02 segundos son tiempo de sistema, es decir, tiempo invertido en ejecutar llamadas del programa al kernel, y código de UNIX en general. Sólo 3.63 segundos ha estado realmente ejecutando código del usuario; y este es el que está bajo su control completo. El resto es información sobre lo que el programa ha leído y escrito, los fallos de página, y las veces que ha tenido que ser enviado al espacio de swap.

Desde el punto de vista de la sintonización del sistema y la mejora de prestaciones, la diferencia entre tiempo usuario+sistema y tiempo total está totalmente fuera de control del usuario, ya que depende de lo que todo los demás estén haciendo (aunque siempre puede chivarse al superusuario y que los echen a todos), así que veremos en qué consiste el tiempo de usuario+sistema:

- Tiempo de usuario, o tiempo de CPU en estado usuario: tiempo que está la CPU ejecutando código del usuario; tiempo en que está ejecutando las librerías del usuario.
- Tiempo del sistema, que en gran parte está bajo control del superusuario: tiempo que está ejecutando llamadas al sistema y tareas

- administrativas relacionadas con el programa: paginación y cosas así.
- El resto del tiempo se divide en tiempo en la red, tiempo de E/S, tiempo que se tarda en ejecutar otros programas, y prestaciones de la memoria virtual.

Ejemplo: ¿A qué dedica su tiempo la internet? Pues, como sistema complejo que es, el tiempo que tarda en presentarse una página Web, es decir, desde que metemos el URL en el cuadro y aparece "Done" en la parte baja del navegador, se invierte a muchas cosas diferentes:

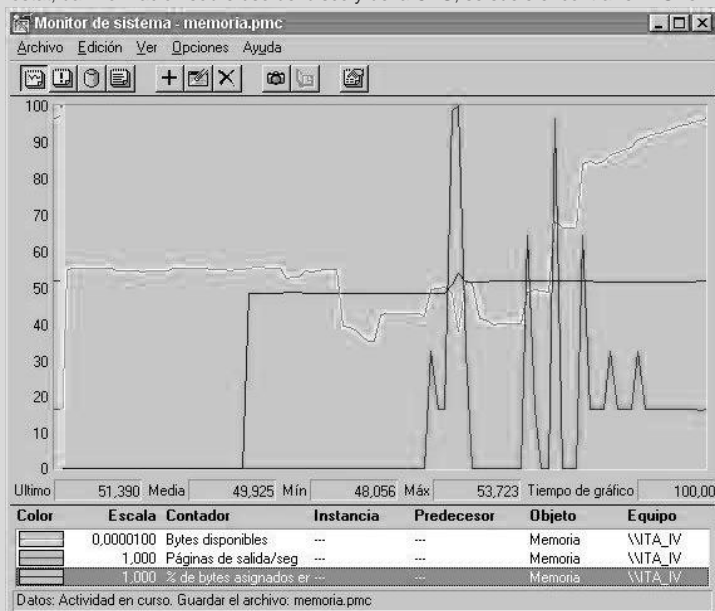
- La petición tiene que ir por alguna red interna de la compañía de telecomunicaciones (como Infovia Plus para Telefónica), en algunos casos, al servidor, desde el punto de presencia local (que permite que se hagan solamente llamadas locales). Ese canal es compartido por cientos, e incluso miles, de usuarios, y el ancho de banda que el proveedor compra a la compañía telefónica está limitado; por lo tanto, en todos lo que suceda de ahora en adelante, habrá que tener en cuenta que peticiones y respuestas pasarán por tal canal .
- El nombre del servidor que se quiere consultar se tiene que buscar en el servidor de nombres, que pertenece a la subred del proveedor; este servidor de nombres normalmente no tendrá la dirección, y pasará la petición a su servidor jerárquicamente superior, y así sucesivamente, hasta que encuentre un servidor de nombres, o DNS (Domain Name System), que tenga el nombre y devuelva la dirección internet correspondiente.
- En ese momento, el cliente o navegador se conecta con el puerto 80 del servidor Web del cual queremos obtener la página, y le envía una petición del estilo `GET estapaginar1.html`. Para ello, por supuesto, tiene que pasar por toda la internet; aunque son pocos bytes, y no tiene demasiada importancia. Si se está usando un proxy, y se detecta que la página que se busca está en el proxy, ahí acaba el tema.
- Esta petición es procesada por el programa `httpd`, sea cual sea (*Apache, por ejemplo*); y por supuesto, tiene uno que compartirlo con todos los que estén accediendo al sitio Web en ese momento. Generalmente se envía un servidor diferente para cada petición, hasta un máximo que se define en el servidor. Cada petición ocupa un proceso o una hebra, ocupa memoria, y, por supuesto, el ancho de banda del servidor Web.
- Procesada la petición, hay que enviar tanto la página como sus contenidos por Internet, y ahí viene lo bueno. Hay que tener en cuenta la distancia "internet", que no es la distancia en kilómetros, sino en hops, o saltos; la información tiene que dividirse en paquetes TCP/IP, enviarse por diversas pasarelas, en cada una de las cuales tiene un retraso, hasta el cliente. La orden `traceroute` permite saber la distancia internet de un punto a otro.
- Por último, cuando ya ha llegado la información, el ordenador tiene que presentarla; es decir, interpretar HTML, componer la página, y decodificar los gráficos, para todo lo cual usa la memoria del ordenador local.

Como se ve, en todo esto el usuario tiene poco control, salvo en el primer paso, es decir, elección del proveedor y cómo se accede a él, y en el último, presentación, en el resto, lo único que se puede hacer es comprarse una internet nueva.

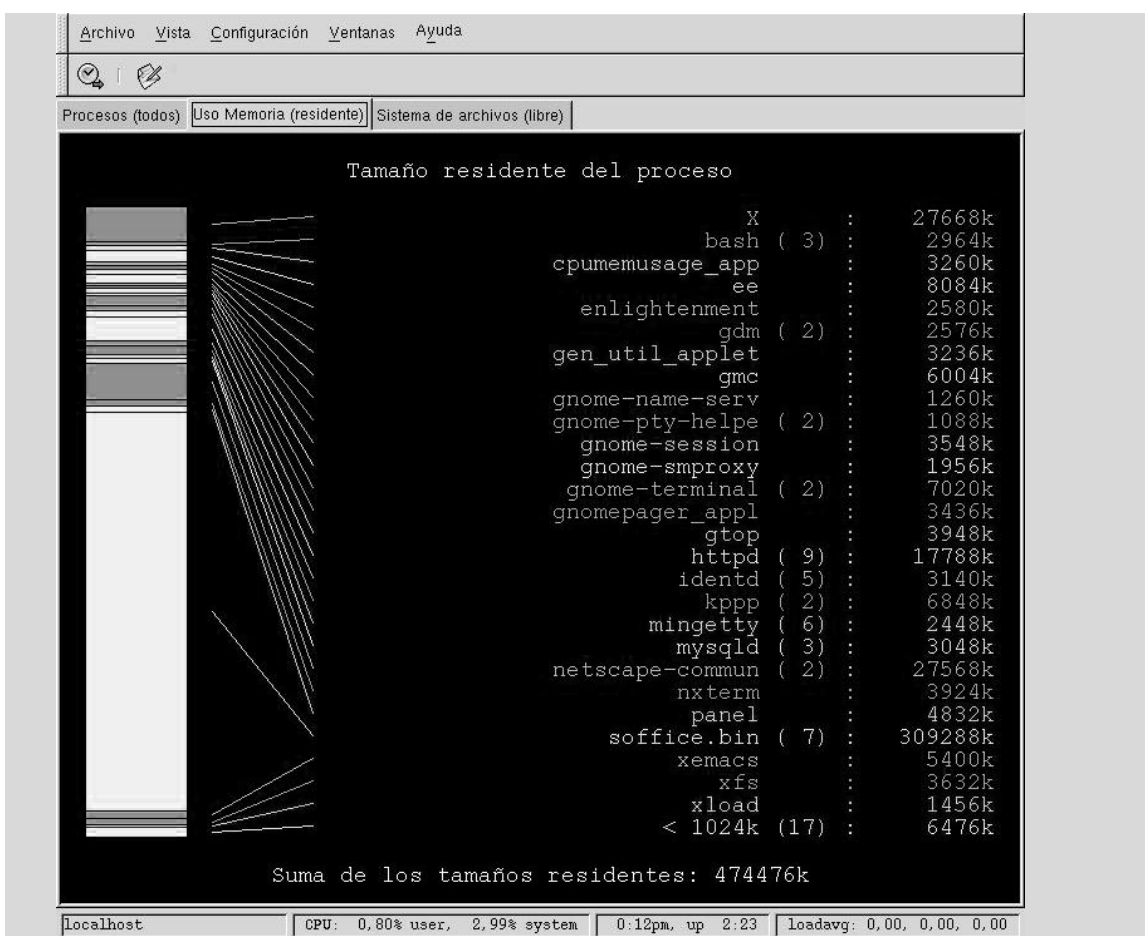
1.4.4 Programas de monitorización de la actividad del sistema

Los siguientes programas son útiles para monitorizar la actividad del sistema, o ayudan a preparar un sistema para ser monitorizado, aunque no sean monitores en sí. En *mi carpeta BackFlip de monitores* se colocan periódicamente nueva información sobre programas de monitorización.

- Cron y crontab, que no sirve para monitorizar, pero sí para ejecutar programas de monitorización a intervalos regulares.
- Uptime y ruptime, que dice la media de carga del sistema.
- Ps dice que procesos se están ejecutando en cada momento.
- Top es un programa un tanto más complejo, que trabaja a terminal complito, y que da información sobre los procesos del sistema y su carga, y además permite interactivamente cargarse algún proceso, si da mucho por saco.
- Iostat, da información sobre uso del disco y de la CPU, se suele encontrar en BSDs

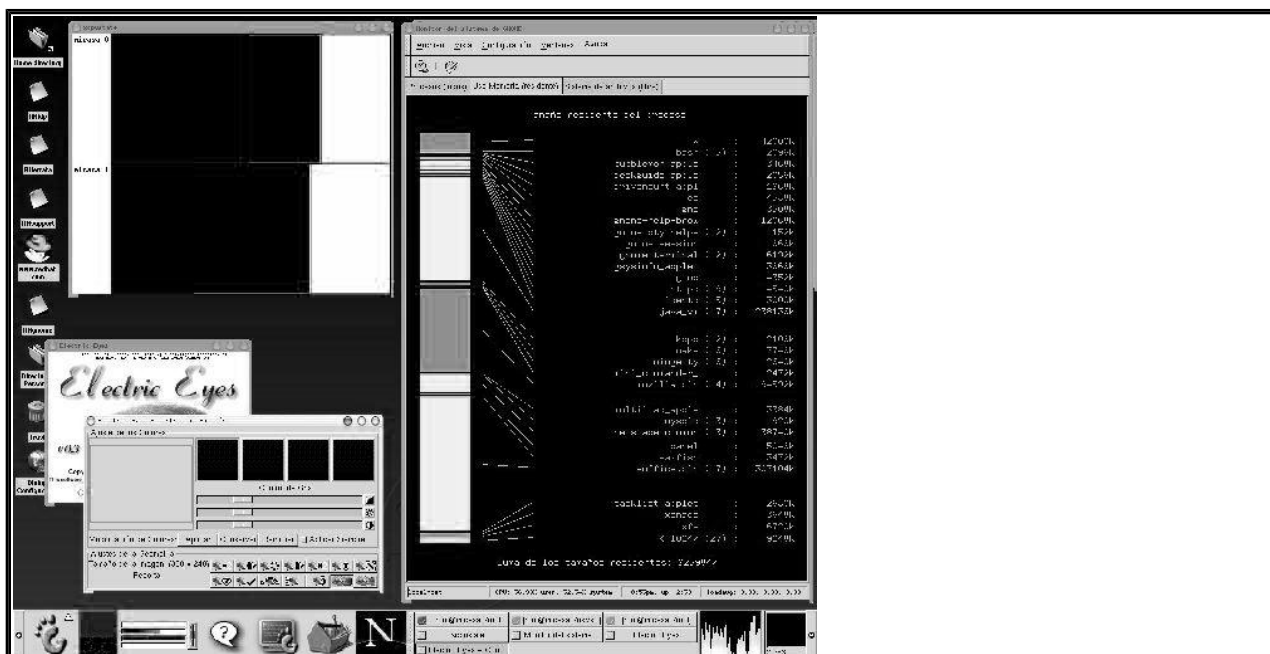


- El administrador de tareas de Windows NT da el uso de CPU y de memoria; también el monitor del sistema da una información equivalente al sar. Está en programas -> herramientas administrativas (común) -> monitor del sistema.
- Sar, o system activity report; en system 5 da información sobre uso de la CPU
- Sa da información sobre que órdenes se han ejecutado y que recursos de CPU han necesitado (sólo BSD), prdaily hace lo mismo en SysV.
- Perfmeter se usa en sistemas OpenView, xload y otros programas hacen algo similar en Linux.
- En otros sistemas operativos, como en el Irix (que está a punto de fenecer, si es que no ha fenecido ya), hay programas como `gr_osview`, que muestran de forma gráfica todos los aspectos del sistema.
- El entorno de usuario `Gnome` viene con el `gtop`, un monitor gráfico equivalente a `top` y `vmstat`, que presenta los procesos que se están ejecutando en cada momento y la memoria que usan. También tiene una serie de monitores que se pueden usar como "applets" en la barra de `Gnome`: de disco, de CPU...



Otros programas sirven para evaluar las prestaciones de una red; o al menos los retrasos que la red introduce en las prestaciones de un ordenador.

- De forma equivalente a los vmstat e iostat, existe netstat, con múltiples opciones; algunas de ellas permiten medir el número de colisiones y errores que hay en la red.
- Ping es la utilidad más simple: simplemente mide lo que tarda un paquete, del tamaño que uno quiera, en ir y volver a un sistema determinado; da una idea de si ese sistema está conectado (independientemente de los protocolos que esté ejecutando), y del retraso que hay. No es en realidad un monitor, salvo que uno quiera medir la velocidad de la red, y lo analice luego con alguna otra herramienta.
- Traceroute no solamente dice cuánto se tarde de un punto de Internet a otro, sino por dónde pasa y cuál es el retraso en cada uno de los nodos. Por alguna razón, en algunos ordenadores hace falta ser superusuario para usarlo.



En esta imagen se pueden ver, empezando arriba a la izquierda y en sentido de las agujas del reloj:

- xcpustate, monitor interactivo que mide el estado de las 2 CPUs del sistema (2 Pentium 450 MHz), indicando el porcentaje dedicado a usuario, sistema, y ocioso
- gtop, indicando ocupación de memoria por cada uno de los procesos; a destacar los 200 megas y pico de la máquina virtual Java y los 300 del StarOffice.
- En el panel de CPU, a la derecha, monitor de carga de la CPU, del propio GNOME; hacia la izquierda, gsysinfo, que indica en un solo panel la media de carga, uso de CPU, uso de

En general, cada sistema operativo viene con multitud de programas de monitorización de la carga del sistema, algunos de ellos en forma de pequeñas aplicaciones (applets) que pueden tenerse siempre ejecutándose (por ejemplo, en la barra de Gnome o en la barra de estado del XEmacs). Es conveniente echar un vistazo a las descripciones de los programas del sistema operativo, a ver cuál es la más conveniente; también conviene mirar en sitios web de distribución de aplicaciones gratuitas, tales como [Freshmeat](#) o [TuCows](#).

Por ejemplo, se puede usar el programa `vmstat` para analizar la actividad de un sistema, de la forma siguiente:

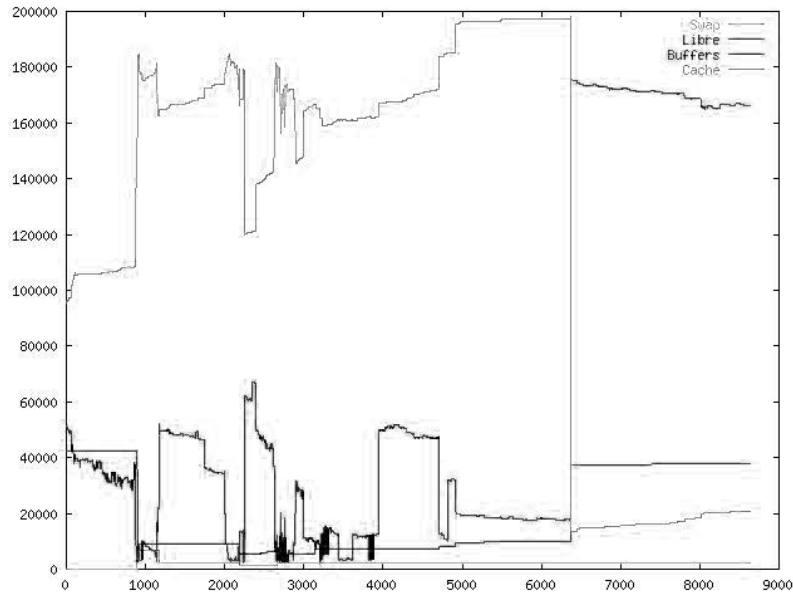
- Se ejecuta durante un tiempo determinado, de la forma siguiente:

```
bash$ vmstat 10 8640
```

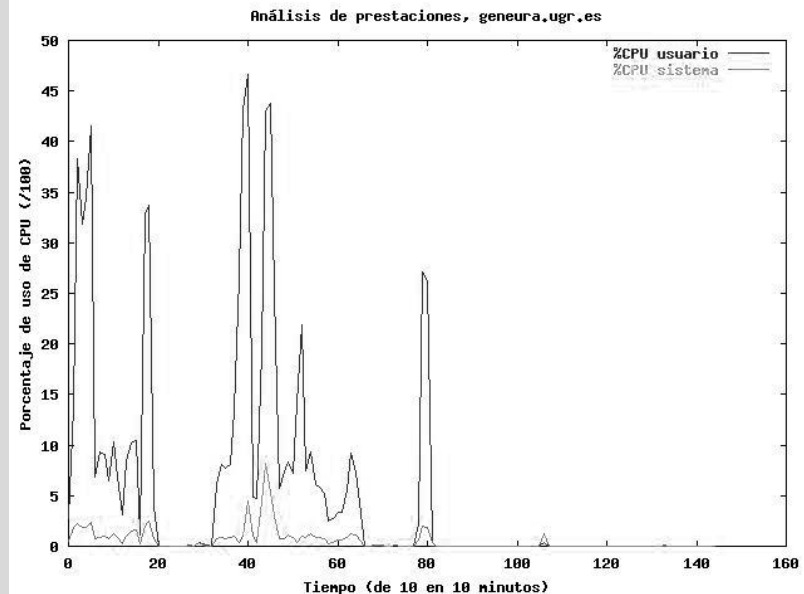
que ejecuta durante un día entero `vmstat`, a intervalos de 10 segundos. Con eso, en Linux, se obtiene un fichero de la forma siguiente:

```
procs
r  b  w  swpd  free  buff  cache  si  so  bi  bo  in  cs  us  sy  id
0  0  0      0 50840 42504 95704 0  0   1   1 124  32  0  0 100
0  0  0      0 50840 42504 95704 0  0   0   6  79  25  0  1  99
0  0  0      0 51436 42504 95708 0  0   0   8 225 267  3  1  96
0  0  0      0 51436 42504 95708 0  0   0   1 121  78  0  0 100
```

- Tal cual, el código anterior nos dice poco. Habrá que hacer algún programa para extraer filas o columnas. Si trazamos usando `gnuplot` las columnas relativas a la memoria, obtenemos algo así



- donde se ve claramente que no hay grandes problemas de memoria, ya que la utilización del swap es bastante bajo. Puede haber algún procesamiento adicional, por ejemplo, en las variables que cambian muy rápido, tales como los porcentajes de tiempo de CPU dedicados a usuario, sistema y el tiempo que la CPU está ociosa. En ese caso, para ver más claras las tendencias, se pueden hacer medias cada número determinado de filas, por ejemplo, 60 (que correspondería a 10 minutos). Eso es lo que hace el programa en `PERL_avg.pl`. Después de este procesamiento, se puede trazar con mucha más claridad las cantidades anteriores usando `gnuplot`



1.5 Selección de las métricas de prestaciones

Para cada estudio, hay que decidir qué conjunto de criterios de prestaciones o métricas van a usarse. Se puede empezar listando los servicios dados por el sistema; para cada petición, hay tres respuestas posibles:

- La petición se ha realizado correctamente.
- La petición se ha realizado incorrectamente.

- La petición no se ha podido realizar.

En caso de que se haya llevado a cabo correctamente, las prestaciones se miden por el tiempo que se ha tardado en realizar la petición, la tasa a la cual el servicio ha sido realizado, y los recursos consumidos mientras se lleva a cabo el servicio, es decir, tiempo/tasa/recurso; estas tres métricas se denominan también responsividad, productividad y utilización.

Por ejemplo, para una pasarela de red (gateway), la responsividad se mide por su tiempo de respuesta, el tiempo entre la llegada y la salida de un paquete; su productividad por el número de paquetes que envía por unidad de tiempo, y su utilización el porcentaje de tiempo que los recursos se usan en una unidad de tiempo determinada.

Todas estas métricas miden, en resumen, la velocidad. Los dos segundos casos se resumen en la fiabilidad y la disponibilidad del sistema. Cada servicio que ofrece un sistema debe de tener una serie de métricas de velocidad, fiabilidad y disponibilidad. Por ejemplo, la fiabilidad se puede medir en tiempo medio entre fallos (MTBF, mean time between failures), y la disponibilidad en el número de horas al año que no está disponible debido a un fallo. Generalmente, estos factores son difíciles de evaluar, y habitualmente se considera en vez de ello las garantías del fabricante.

Los tres criterios que se suelen seguir para elegir un subconjunto de todas las métricas suelen ser: variabilidad baja (para que no haya que repetir las mediciones muchas veces), que no haya redundancia (que no haya métricas que dependan unas de otras), y complejidad (que definan de forma completa las prestaciones de un sistema).

Ejemplo: Medir las prestaciones de diferentes tarjetas gráficas. Básicamente, el servicio que ofrecen es dibujar: textos, gráficos, polígonos, texturas, movimiento de bloques de bits. En este caso es poco probable que se den métricas del tipo fiabilidad y disponibilidad (si no funciona, se devuelve), pero en cuanto a la velocidad, se pueden considerar: velocidad de dibujo y relleno de polígonos, velocidad de dibujo de tipos de letras diferentes, velocidad de descompresión MPEG (si ofrece ese servicio), velocidad de polígonos OpenGL, velocidad en Direct3D, velocidad de dibujo de texturas. ¿Qué métricas se podrían elegir? Byte, en febrero de 1997, usó una serie de tests para 2D, usando programas comerciales, y los programas Viewperf para medir prestaciones en OpenGL y Tunnel para Direct3D. Los tiempos se usaron para generar un índice. Dado que hay tarjetas optimizadas para uno de los dos entornos, lo mejor es compararlas en ambos, y además en el caso más normal, es decir, 2D.

Las métricas de prestaciones se suelen clasificar de la forma siguiente:

- Más alto es mejor, HB, higher is better; es decir que es mejor cuanto más alta, como la velocidad, o el throughput de un sistema.
- Menor es mejor, LB, Lower is better; es decir, que los valores inferiores son los mejores, como el tiempo de respuesta o el número de fallos de página.
- Nominal es mejor, NB, Nominal is best, no son buenos los valores altos ni los bajos; por ejemplo, la utilización es un valor de este tipo. Utilización baja significa infrautilización, y utilización alta hace que los tiempos de respuesta sean altos.

4 Ejercicios de autoevaluación

1. Indicar las métricas que se usarían, y de qué tipo son (más-es-mejor, menos-es-mejor, nominal-es-mejor), en los siguientes sistemas: tarjeta gráfica, impresora, programa servidor web, ordenador servidor web .

1.6 Bibliografía y enlaces en internet

Evaluación y explotación de sistemas informáticos . R. Puigjaner, J. J. Serrano, A. Rubio.

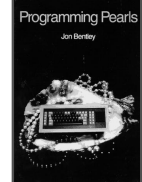
System Performance Tuning, de Mike Loukides, O'Reilly, un libro un tanto obsoleto, pero que contiene todas las técnicas básicas para analizar las prestaciones de los diversos subsistemas de un ordenador: memoria, disco, red. Desde la red de la Universidad de Granada (y cualquier otra red que tenga acceso) puedes acceder a [la versión online del libro](#)



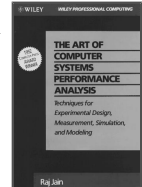
Específicamente para el lenguaje Java, se puede consultar *Java Performance Tuning, second edition*, de Jack Shirazi, un libro concentrado, como es natural, en Java, en cómo incrementar las prestaciones de la máquina virtual y en cómo mejorar los programas escritos en Java . Desde la red de la Universidad de Granada (y cualquier otra red que tenga acceso) puedes acceder a [la versión online del libro](#) (de la primera edición); el capítulo 2 es el más relevante para la asignatura.



Programming Pearls, de Jon Bentley, Addison Wesley, 1989. Colección de las columnas publicadas por el autor en Communications of the ACM, son una colección de técnicas y trucos utilizables para programar en cualquier lenguaje y ordenador.



The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation and modelling, Raj Jain, Wiley, 1992. Por la fecha de publicación, se ve que tiene el mismo problema que el primero; sin embargo, tiene un enfoque ameno y completo al problema del análisis de prestaciones.



Web Performance Tuning: Speeding Up the Web (O'Reilly Nutshell) by Patrick Killelea, Linda Mui; un libro un tanto básico; se concentra demasiado en cosas como aumento de las prestaciones del cliente, pero no está mal del todo. También puedes acceder a él desde [Safari](#) (pero a la primera edición, y no completa).



