

1. Introducción

1.1. Evaluación del rendimiento de un sistema informático

Se podría definir la evaluación del rendimiento de un sistema informático como la medida de cómo un software determinado está utilizando el hardware con una determinada combinación de programas, que constituyen lo que se denomina carga del sistema (figura 1.1).

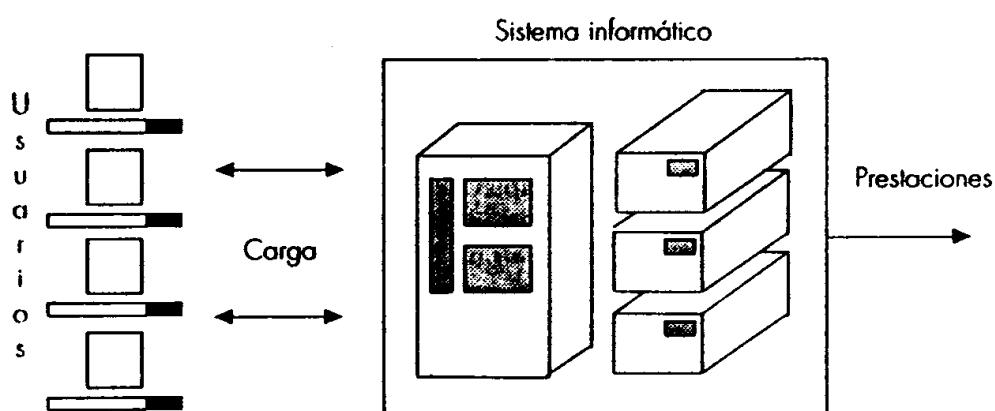


FIGURA 1.1.

Tal vez la mayor dificultad de la evaluación de prestaciones proviene del hecho que la carga real de un sistema informático cambia continuamente y las mediciones no pueden repetirse a no ser que se trabaje en un entorno controlado de carga, técnica que más adelante se describirá. Por otro lado, las variables o índices significativos de la medición de las prestaciones pueden variar de una instalación a otra, en función de la configuración y uso del sistema. Asimismo, los valores de esas variables o índices que permiten considerar el buen o mal funcionamiento del sistema también varían de una instalación a otra.

Los usuarios o las necesidades de servicio exigen que un sistema informático deba cumplir una serie de requisitos o especificaciones de ejecución para considerar que su funcionamiento es aceptable.

Actualmente, todos los usuarios de los sistemas informáticos, diseñadores, administradores, etc. están interesados en los estudios de evaluación del comportamiento de su sistema, con el objetivo de obtener las mejores prestaciones al mínimo costo.

Las actividades que forman parte del estudio del comportamiento de un sistema se denominan de *evaluación de sus prestaciones*.

Las prestaciones de un sistema son el criterio clave en el diseño, selección, compra y uso de un sistema informático.

1.2. Necesidad de la evaluación de prestaciones

Esta necesidad ha aparecido como una consecuencia natural del aumento de la potencia y de la complejidad de los sistemas informáticos. Los primeros computadores estaban concebidos para que los utilizara el propio programador que, personalmente, controlaba lo que sucedía en el sistema mientras se ejecutaba el programa. Eran los tiempos en que prácticamente no existía el software y en que los elementos fundamentales para la evaluación del comportamiento consistían en la longitud de la palabra, el conjunto de instrucciones y su implantación, el ciclo de base de la CPU, el tiempo de ejecución de una instrucción característica (normalmente la instrucción de sumar), etc.

La aparición de las ayudas software, la de los periféricos cada vez más sofisticados, y la de las unidades centrales más complejas (multiprocesadores, *pipelines*, memorias cache, etc.) con sistemas de interrupciones muy sofisticados, el aumento de la dimensión de las memorias, etc., han hecho que la evaluación del comportamiento se haya convertido en un cuerpo de doctrina en el que no sólo se ha de considerar el hardware, sino también las facilidades proporcionadas por el software al acercar la máquina a los usuarios, provocando entonces la aparición del *overhead* (es decir, de los gastos generales de la máquina para repartir los recursos entre los distintos usuarios) que lleva asociado el software. Además, la evaluación del comportamiento de un sistema hay que hacerla teniendo presente que ha de satisfacer a unos usuarios locales o remotos que, en general, se ven a sí mismos como únicos usuarios del sistema en la ejecución de sus programas. Estas consideraciones hacen comprender que la evaluación del comportamiento no es una tarea sencilla, ya que ha de tener en cuenta muchos y variados aspectos del hardware, del software y de las aplicaciones que se han de llevar a cabo en el sistema informático.

En consecuencia, es necesario evaluar un sistema, para comprobar que su funcionamiento es correcto, es decir, el esperado.

También va a influir esta evaluación en las decisiones de diseño, implantación, compra y modificación de los sistemas informáticos.

Así pues, la evaluación de un sistema es necesaria en todas las etapas de su ciclo de vida; por tanto, los evaluadores de un sistema deben ser sus diseñadores, fabricantes, vendedores, administradores y usuarios según la etapa que se considere.

La evaluación va a ser necesaria cuando un diseñador de sistemas quiere comparar un cierto número de diseños alternativos y encontrar el mejor; lo mismo le ocurre al administrador de un sistema, que quiere saber qué sistema es el más adecuado para ejecutar un determinado conjunto de aplicaciones.

Otro de los objetivos que se pretende al evaluar un sistema informático es encontrar los factores que impiden un funcionamiento adecuado, como pueden ser los cuellos de botella. También la predicción del comportamiento del sistema con nuevas cargas (planificación de la capacidad) constituye otro objetivo de la evaluación de prestaciones.

Para realizar la evaluación de un sistema se debe de partir de unos datos y se deben fijar unos objetivos. Estos deben de estar muy claros, es decir, hay que saber con antelación el problema que se va a resolver.

Por lo tanto es necesario evaluar un sistema informático cuando se quiere:

- Diseñar una máquina.
- Diseñar un sistema informático.
- Seleccionar y configurar un sistema informático.
- Planificar la capacidad del sistema informático.
- Sintonizar o ajustar un sistema informático.

En todos los casos expuestos se ha visto que el comportamiento del sistema no es algo independiente del entorno de trabajo sino que está íntimamente relacionado con el uso a que se destinará. Por ello, dentro del ámbito de la evaluación de prestaciones se incluye un apartado destinado a:

- Caracterizar y predecir la carga.

A continuación se describen estas distintas tareas.

1.2.1. Diseñar una máquina

Durante las fases de diseño de un computador hay que tomar una serie de decisiones que pueden tener una gran influencia en el comportamiento del mismo. Por lo tanto, es necesario estudiar el comportamiento del sistema antes de su implantación para ver cuál de las posibles opciones de diseño disponibles es la mejor teniendo en cuenta el entorno en el que se implantará.

1.2.2. Diseñar un sistema informático

Durante las fases de diseño de una aplicación informática, especialmente si se trata de un sistema de tiempo real o acoplado, hay que tomar una serie de decisiones que pueden tener una gran influencia en el comportamiento del mismo. Por lo tanto, es necesario estudiar el comportamiento del sistema antes

de su implantación para ver cuál de las posibles opciones de diseño disponibles es la mejor teniendo en cuenta el entorno de trabajo establecido.

1.2.3. Seleccionar y configurar un sistema informático

Cuando se desea seleccionar una nueva máquina, hay que ver qué configuración es la más adecuada para atender un determinado conjunto de trabajos y qué fabricante ofrece mejor relación precio/prestaciones para esa situación.

1.2.4. Planificar la capacidad del sistema informático

A lo largo de la vida de un sistema, por diversas razones, como el aumento vegetativo de la carga o la incorporación de nuevas aplicaciones, es necesario aumentar su capacidad y, en consecuencia, es necesario planificar la capacidad del sistema con posibles nuevas configuraciones.

1.2.5. Sintonizar o ajustar un sistema informático

Debido al crecimiento vegetativo de la carga de un sistema informático, cuando las prestaciones del mismo decrecen, es necesario ajustar o cambiar alguno de los parámetros del sistema operativo, normalmente para mejorar el comportamiento del sistema. En algunos casos si el sistema (hardware más software básico) no se puede variar, hay que intentar mejorar el comportamiento del sistema modificando la carga (programas).

1.2.6. Caracterizar y predecir la carga

El comportamiento de un sistema es muy dependiente de la carga aplicada al mismo; por lo tanto conocer o caracterizar la carga que tiene un sistema y predecir su carga futura es fundamental para poder a estudiar el comportamiento del mismo.

1.2.7. Conclusión

Siempre que se observe un problema en el rendimiento de un sistema informático o queramos seleccionar una configuración o un sistema o diseñar una máquina es necesario proceder a evaluar sus prestaciones.

1.3. Magnitudes que se deben medir

Para evaluar el comportamiento de un sistema hay que hacer referencia a unas medidas cuantitativas o unos parámetros que van a caracterizar el comportamiento del hardware y software del ordenador o que van a hacer referencia a cómo el usuario (visión externa) y el responsable del sistema (visión interna) ven su comportamiento.

Estas magnitudes y parámetros están relacionados con tres tipos de medidas correspondientes a:

- Consumo de tiempos.

- Utilización de recursos o dispositivos.
- Trabajo realizado por el sistema o componentes del mismo.

La relación que se expone a continuación no pretende ser una lista exhaustiva de las magnitudes que siempre deben medirse en todas las instalaciones sino una lista de variables que normalmente se acostumbran a medir, pero sin que ello sea óbice en cada instalación concreta para añadir variables no consideradas aquí o para eliminar algunas de ellas por carecer de significado en un contexto dado.

1.3.1. Variables externas o perceptibles por el usuario

a) *Productividad o throughput*

Es la cantidad de trabajo útil ejecutado por unidad de tiempo en un entorno de carga determinado (normalmente se mide en trabajos por hora o en transacciones por segundo).

b) *Capacidad*

Es la máxima cantidad de trabajo útil que se puede realizar por unidad de tiempo en un entorno de carga determinado.

c) *Tiempo de respuesta*

Es el tiempo transcurrido entre la entrega de un trabajo o una transacción al sistema y la recepción del resultado o la respuesta.

1.3.2. Variables internas o del sistema

a) *Factor de utilización de un componente*

Es el porcentaje de tiempo durante el cual un componente del sistema informático (CPU, canal, dispositivo de E/S, etc.) ha sido realmente utilizado.

b) *Solapamiento de componentes*

Es el porcentaje de tiempo durante el cual dos o más componentes del sistema han sido utilizados simultáneamente.

c) *Overhead*

Es el porcentaje de tiempo que los distintos dispositivos del sistema (CPU, discos, memoria, etc.) han sido utilizados en tareas del sistema no directamente imputables a ninguno de los trabajos en curso.

d) *Factor de carga de multiprogramación*

Es la relación entre el tiempo de respuesta de un trabajo en un determinado entorno de multiprogramación y su tiempo de respuesta en monoprogramación.

e) *Factor de ganancia de multiprogramación*

Es la relación entre el tiempo total necesario para ejecutar un conjunto de programas secuencialmente en monoprogramación y en multiprogramación.

f) *Frecuencia de fallo de página*

Es el número de fallos de página que se producen por unidad de tiempo en un sistema de memoria virtual paginada.

g) *Frecuencia de swapping*

Es el número de programas expulsados de memoria por unidad de tiempo a causa de falta de espacio o con el fin de permitir su reorganización para recuperar espacio en ella o para disminuir la paginación.

1.3.3. Otras magnitudes relativas al comportamiento

Hay otras medidas relacionadas con el comportamiento del sistema pero no directamente con las prestaciones que en muchos casos también es importante tenerlas en cuenta, como:

- *Fiabilidad*: Es la función del tiempo definida como la probabilidad condicional de que el sistema trabajará correctamente a lo largo del intervalo de tiempo $[t_0, t]$, si trabajaba correctamente en el instante t_0 . En otras palabras es la probabilidad que el sistema trabaje correctamente a lo largo de un intervalo de tiempo dado. Se mide por la probabilidad de fallos por unidad de tiempo, o por el tiempo medio entre fallos.
- *Disponibilidad*: Es la función del tiempo definida como la probabilidad de que el sistema esté trabajando correctamente y que esté disponible para realizar sus funciones en el instante considerado t . Incluye la posibilidad de que el sistema pueda haber estado averiado y posteriormente reparado.
- *Seguridad*: Es la probabilidad de que el sistema esté realizando correctamente sus funciones o parado de forma tal que no perturbe el funcionamiento de otros sistemas ni comprometa la seguridad de las personas relacionadas con él.
- *Performabilidad*: En muchos casos, es posible diseñar un sistema que continúe trabajando correctamente después de la aparición de fallos de hardware o software, pero con un nivel de prestaciones de alguna forma disminuido. Es pues una función del tiempo definida como la probabilidad de que las prestaciones del sistema estarán por encima de un cierto nivel en un instante determinado.
- *Mantenibilidad*: Es la medida de la facilidad con que un sistema puede ser reparado después de un fallo. En términos cuantitativos es la probabilidad de que un sistema averiado pueda ser reparado y devuelto al estado operacional dentro de un período de tiempo determinado.

1.4. Magnitudes que caracterizan la carga

Tanto en el caso de tomar medidas de un sistema existente como cuando se recaban datos para predecir el comportamiento de uno inexistente, se plantea el problema de representar la carga de manera que se pueda considerar que el sistema se comporta de forma característica para proceder entonces a efectuar las mediciones. En consecuencia, para evaluar correctamente las prestaciones de un sistema informático, la carga debe de ser seleccionada cuidadosamente.

El término *carga de prueba* se utiliza para denominar la carga usada en el estudio de prestaciones. Esta carga puede ser *real* o *sintética*.

La carga *real* se observa en un sistema durante su funcionamiento normal. Su inconveniente es que no permite repeticiones para eliminar los errores de medición, y por ello es difícilmente utilizable como *carga de prueba*.

La carga *sintética* está constituida por un conjunto de programas extraídos o no de la carga real del sistema informático que la reproduce de forma compacta. Se utiliza más porque puede utilizarse repetidamente y también porque puede modificarse sin afectar a la operatividad del sistema.

En muchos sistemas la evaluación se suele realizar en un sistema aparte del real, es decir, creándose dos sistemas paralelos. Uno será el sistema real y el otro el de test. En el primero trabajarán, por así decirlo, aquellos usuarios que explotan el sistema. En el segundo, trabajarán aquellos técnicos que desarrollan el sistema o que están encargados de su buen funcionamiento.

El común denominador de todos estos problemas reside en la determinación de las magnitudes que caracterizan la carga del sistema. Dependerán del tipo y modo de trabajo del sistema. Algunas de estas magnitudes características se exponen a continuación.

1.4.1. Para cada componente de la carga

a) Tiempo de CPU por trabajo

Es el tiempo total de CPU necesario para ejecutar un trabajo (programa, transacción, etc.) en un sistema determinado. Evidentemente es función directa del número de instrucciones que se ejecutan para realizar ese trabajo, del volumen de datos procesados y de la velocidad del procesador.

b) Número de operaciones de E/S por trabajo

Es el número total de operaciones de entrada/salida que requiere la ejecución de un trabajo. Evidentemente dicho número total conviene desglosarlo según el dispositivo, el archivo, etc. sobre el que se realizan.

c) Características de las operaciones de E/S por trabajo

Hacen referencia al soporte (cinta, disco, etc.) y, en el caso de discos, a la posición que ocupa el archivo sobre el que se efectúan. Todo ello tiene una influencia directa en el tiempo necesario para realizar una operación de E/S.

d) Prioridad

Es la que el usuario asigna a cada uno de los trabajos que somete al sistema.

e) Memoria

Es la que requiere ocupar, para su ejecución, un trabajo determinado. Puede ser constante –memoria real– o variable –memoria virtual paginada (conjunto de trabajo) o segmentada–, según la gestión que el sistema operativo haga de la memoria.

f) Localidad de las referencias

Es el tiempo en el que todas las referencias a memoria hechas por un trabajo permanecen dentro de una página (segmento) o conjunto de páginas (segmentos).

Si se considera la ejecución de un programa como una sucesión de referencias a memoria, la localidad del programa será tanto mayor cuanto más tiempo esté dentro de la página (segmento) o conjunto de páginas (segmentos) considerados. Es la aparente contradicción de medir una magnitud ligada al espacio (localidad) mediante un tiempo.

1.4.2. Para el conjunto de la carga

a) Tiempo entre llegadas

Es el tiempo entre dos requerimientos sucesivos para un servicio (ejecución de un trabajo o transacción) del sistema.

b) Frecuencia de llegada

Es el número medio de llegadas de nuevas peticiones de ejecución que se producen por unidad de tiempo. Evidentemente es la inversa del tiempo medio entre llegadas.

c) Distribución de trabajos

Define la proporción existente entre las ejecuciones de los distintos trabajos que constituyen la carga.

1.4.3. Para cargas conversacionales

a) Tiempo de reflexión del usuario

Es el tiempo que el usuario de un terminal de un sistema interactivo necesita para generar una nueva petición al sistema (es decir, es el tiempo de leer la respuesta previa, de pensar en la nueva acción que se vaya a tomar, función de la respuesta recibida y de teclearla).

b) Número de usuarios simultáneos

Es el número de usuarios interactivos que trabajan simultáneamente sobre el mismo sistema en un instante dado.

c) Intensidad del usuario

Es la relación entre el tiempo de respuesta de una petición y el tiempo de reflexión del usuario.

1.5. Magnitudes para controlar el comportamiento

Hasta ahora se ha estudiado lo que hay que medir y cómo se escoge cuándo hay que medir; pero, ¿qué hay que hacer si el comportamiento del sistema no es satisfactorio? ¿Cuáles son los resortes que hay que mover para mejorar el comportamiento del sistema?

Las modificaciones que se pueden introducir en un sistema para mejorar su comportamiento pueden hacerse en todos los niveles que influyen en el comportamiento del mismo mediante:

- Ajuste de los parámetros del sistema operativo.
- Modificación de las políticas de gestión del sistema operativo.
- Equilibrado de la distribución de cargas.
- Modificación o sustitución de componentes hardware del sistema.
- Modificación de los programas.

A continuación se analizan con algún detalle el significado de estas distintas opciones.

1.5.1. Ajuste de los parámetros del sistema operativo

La relación que sigue no pretende ser exhaustiva, ni pretende que en todos los sistemas operativos puedan modificarse con la misma facilidad los parámetros expuestos, pero sí es una lista de parámetros de un sistema operativo que pueden modificarse con facilidad y cuya variación puede influir en el comportamiento del sistema.

a) Tamaño del quantum

Es la cantidad de tiempo de uso ininterrumpido de la CPU que un sistema de tiempo compartido asigna a los diferentes trabajos. En ciertos sistemas, no existe uno sino varios quanta para las distintas prioridades internas de los distintos trabajos por lo que no es un solo quantum el que hay que cambiar sino buscar un equilibrio adecuado entre todos ellos.

Es evidente que si el quantum es demasiado grande se favorece a los trabajos con mucho uso de CPU y si es demasiado pequeño se puede introducir un over-

head importante con los continuos cambios de contexto de un programa a otro cada vez que se agota el quantum.

b) Prioridad interna

Es el nivel inicial de prioridad interna que recibe un programa en función de la prioridad externa asignada.

c) Factor de multiprogramación

Es el número máximo de trabajos que están simultáneamente en memoria principal y, por lo tanto, que tienen opción a utilizar la CPU y los demás recursos activos del sistema.

Cuanto mayor sea este valor tanto mejor aprovechamiento se puede tener de todos los recursos del sistema, aunque también aumentará el *overhead*.

d) Tamaño de la partición de memoria

Es la cantidad fija de memoria principal asignada a una cola de trabajos.

Hay que adecuar esas particiones a los tamaños de los programas de la instalación y a su frecuencia de ejecución.

e) Tamaño de la ventana

Es el intervalo de tiempo durante el cual el sistema toma medidas para determinar el conjunto de trabajo de un programa en un entorno de memoria virtual paginada que use esa política.

Evidentemente, según el período de tiempo durante el que se tomen medidas para determinar el conjunto de trabajo, éste varía y, por lo tanto, el valor medio del conjunto de trabajo estará afectado por los valores que intervengan en su cálculo.

f) Máxima frecuencia de fallo de página

Es la frecuencia máxima de fallo de página permitida. A partir del instante en que se alcanza, se efectúa la suspensión o *swapping* de alguno de los trabajos en curso para evitar el excesivo *overhead* que se estaba generando.

g) Índice de supervivencia de las páginas

Es el número de ráfagas de CPU recibidas por un programa antes de que saque de la memoria principal una página que no haya sido referenciada durante ese período.

h) Número de usuarios simultáneos

Es el máximo número de usuarios de terminal permitidos por un sistema de tiempo compartido.

1.5.2. Modificación de las políticas de gestión del sistema operativo

Teniendo en cuenta que las políticas propuestas por los fabricantes en sus sistemas operativos son para atender eficientemente a usuarios con una carga promedio, es posible que, en determinados sistemas, aquéllas no sean las más adecuadas en un caso concreto, por lo que puede ser conveniente y recomendable la sustitución de la rutina encargada de la gestión de un determinado recurso por otra que realice una política más idónea a unas necesidades concretas.

Este cambio lleva asociado evidentemente, el riesgo de problemas en el sistema operativo durante el período de depuración y puesta a punto de la nueva rutina. Además hay que tener en cuenta que habrá que adaptar el cambio a las sucesivas versiones del sistema operativo.

1.5.3. Equilibrado de la distribución de cargas

El ideal, inalcanzable, de utilizar por igual todos los dispositivos del sistema informático debe sustituirse por el de utilizarlos de la forma más uniforme posible. No obstante, con frecuencia, el uso de recursos es notablemente desequilibrado y se deben disponer los cambios (de asignación de dispositivos periféricos a los canales, de ubicación de archivos en disco, de ejecución simultánea de programas que requieren los mismos recursos, de distribución de componentes software en la jerarquía de memoria, de políticas de explotación, etc.) necesarios para tender a lograr el equilibrio deseado.

Este tipo de corrección acostumbra, en muchos casos, a proporcionar mejoras espectaculares en el comportamiento del sistema.

1.5.4. Sustitución o ampliación de los componentes del sistema

Cuando el recurso a los métodos anteriores resulta ineficaz o inaplicable, se está abocado a la modificación de la configuración del sistema, bien sea sustituyendo determinados elementos por otros de mayor capacidad o rapidez, bien sea por aumento del número de dispositivos que constituyen la configuración del sistema. No obstante, la ampliación de la configuración debe hacerse de forma tal que se despeje el cuello de botella (elemento cuya saturación impide la mejora del comportamiento) que se pueda haber detectado, pues, de lo contrario, el comportamiento conjunto del sistema ampliado no varía de forma significativa.

1.5.5. Modificación de los programas

Para mejorar el comportamiento de un sistema informático puede recurrirse a modificar los programas de forma que su ejecución promedio requiera menos recursos (tiempo de CPU, número de E/S, etc.), bien sea por recodificación de los caminos del programa recorridos con mayor actividad, bien sea por un montaje que agrupe en la misma página o segmento aquellos módulos del programa que deben coexistir en memoria para la ejecución del programa, etc. Además es

conveniente verificar periódicamente que las hipótesis que justificaron una determinada concepción de un programa siguen siendo válidas.

Este procedimiento de mejora provoca la modificación de la carga, por lo que, aun cuando es una forma eficaz y frecuente de mejorar el comportamiento del sistema, normalmente se considerará la carga como un dato del problema que no se puede modificar.

1.6. Sistemas de referencia

A lo largo del texto se hará constante referencia a tres tipos de sistemas o tipos de funcionamiento de un sistema informático:

- Sistema por lotes o *batch*.
- Sistema interactivo o por demanda.
- Sistema transaccional.

Estos sistemas servirán de referencia para el estudio de las prestaciones y su evaluación.

1.6.1. Trabajo en lotes o *batch*

En muchas instalaciones de empresas, bancos, etc., el ordenador se queda por la noche en marcha ejecutando una serie de programas que se dejan almacenados en memoria. En estos sistemas (figura 1.2), el responsable de la explotación decide los trabajos que deben estar en ejecución en cada instante; así pues, la planificación interna del sistema operativo está ayudada por la externa humana.

Estos trabajos realizan ciclos de uso de la CPU y de los discos de forma continua hasta que finalizan.

Índices característicos de las prestaciones de estos sistemas son los siguientes:

- *Turnaround time* (equivalente al tiempo de respuesta en estos sistemas) es el tiempo que transcurre desde que se lanza la ejecución de un trabajo hasta que termina y en el que la generación de cada petición depende de la recepción de la respuesta en la petición.
- Productividad medida en trabajos por unidad de tiempo (segundo u hora).

1.6.2. Sistema transaccional

Un sistema transaccional es aquél en que un conjunto de terminales remotos conectados al sistema interactúan con un conjunto determinado de programas; cada una de las interacciones constituye lo que se denomina una transacción (figura 1.3). Como ejemplos de este tipo de sistemas se pueden citar el sistema informático de un banco o caja de ahorros, o el de un sistema de reserva de billetes, o el que recibe medidas de un satélite. En sistemas de este tipo el sis-

tema de planificación interna del sistema operativo ha de ser capaz de gestionar las peticiones que llegan al sistema y no está ayudado en forma alguna por la planificación humana externa ya que desde cada terminal se tiene la sensación de ser el dueño absoluto del sistema y se tiene la posibilidad de lanzar cualquier transacción en cualquier instante.

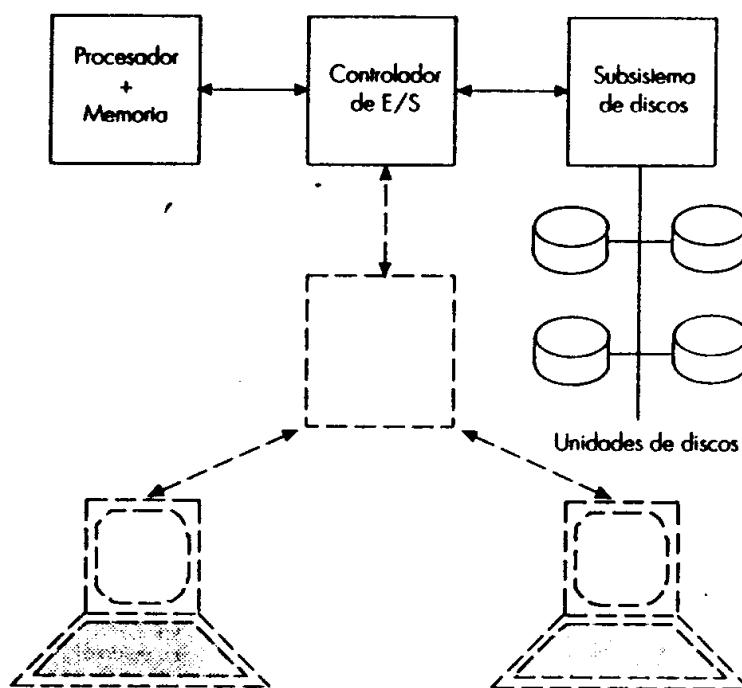


FIGURA 1.2.

El sistema de este tipo queda definido por el flujo de transacciones que le llega y su índice de prestaciones característico es:

- Tiempo de respuesta, que es la suma de los tiempos siguientes:
 - *Tiempo de reacción*, que es el tiempo que transcurre desde que la transacción llega al sistema hasta que comienza su ejecución.
 - *Tiempo de ejecución*, que es el que transcurre desde que el sistema comienza la ejecución de la transacción hasta que termina.
 - *Tiempo de retorno*, que es el que transcurre desde que finaliza la ejecución hasta que, eventualmente, se completa la respuesta hacia el usuario.

1.6.3. Sistema interactivo o por demanda

Un sistema interactivo (figura 1.4) es aquél en que los usuarios acceden a él desde terminales remotos teniendo acceso a la totalidad del sistema operativo. En

estos sistemas, un usuario desde un terminal, después de un tiempo de pensar o de reflexión, da una orden al terminal que pasa a procesarse por el conjunto CPU y discos y que después de un tiempo dado produce una respuesta en el terminal. Se trabajará de modo interactivo de acuerdo con el siguiente ciclo:

ORDEN → RESPUESTA → ORDEN → RESPUESTA → ...

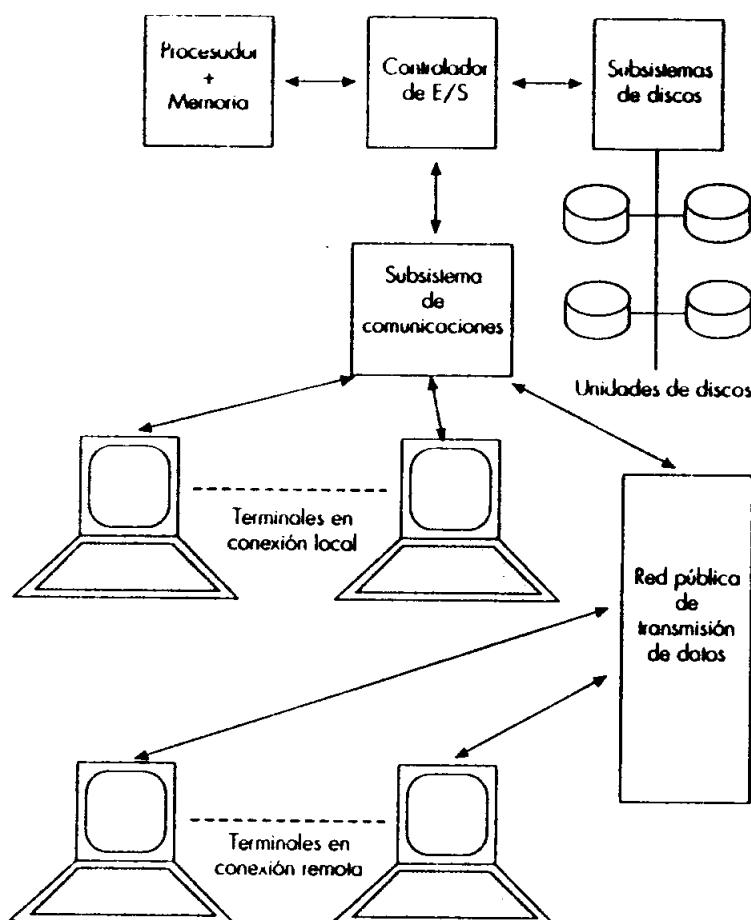


FIGURA 1.3.

En estos sistemas, pues, la generación de una nueva petición depende de la recepción de la respuesta a la petición anterior.

En estos sistemas tampoco la planificación humana ayuda a la planificación del sistema operativo. Un sistema de este tipo queda definido por:

- Número de usuarios que tiene conectados.
- Tiempo de reflexión de los usuarios, que es el tiempo que transcurre desde que el usuario recibe la respuesta hasta que éste finaliza la preparación de la siguiente petición. Incluye, por lo tanto, el tiempo necesario para leer la respuesta del sistema, reflexionar sobre la siguiente acción y teclearla.

Y los índices de prestaciones característicos son:

- Tiempo de respuesta, que será la suma de los tiempos siguientes:
 - *Tiempo de reacción*, que es el tiempo que transcurre desde que el usuario finaliza la petición hasta que el sistema comienza su ejecución.
 - *Tiempo de ejecución*, que es el que transcurre desde que el sistema comienza la ejecución de la petición hasta que termina.
 - *Tiempo de retorno*, que es el que transcurre desde que finaliza la ejecución hasta que se completa la recepción de la respuesta en el usuario.
- Productividad, medida en peticiones por unidad de tiempo, aunque, como se verá más adelante, estas cuatro variables están relacionadas y el conocimiento de tres de ellas permite determinar la cuarta.

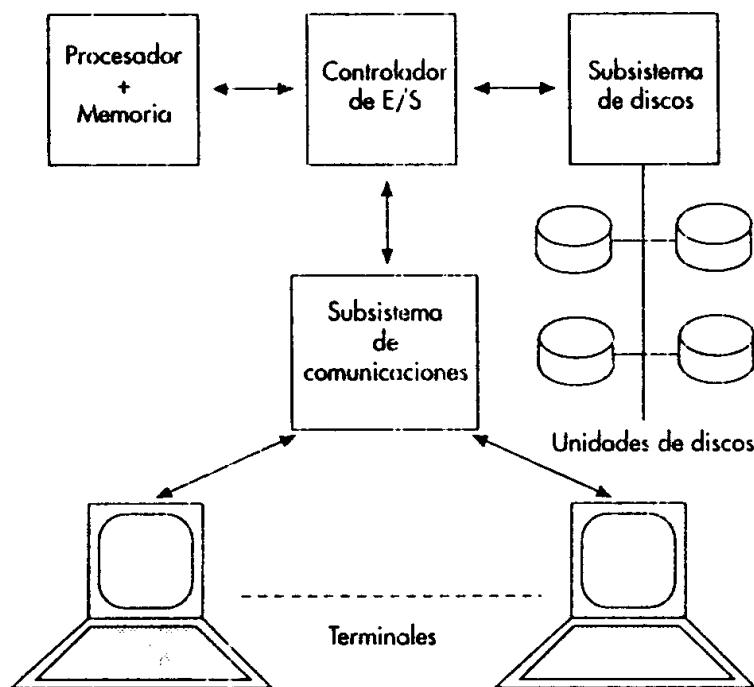


FIGURA 1.4.

1.7. Técnicas más comunes a la hora de evaluar un sistema

Se denominan *técnicas de evaluación* a los métodos y herramientas que permiten obtener los índices de prestaciones de un sistema que está ejecutando una carga dada y con unos valores determinados de parámetros del sistema.

Las técnicas pueden ser de tres tipos:

- Monitorización
- Modelado
- *Benchmarking*

1.7.1. Monitorización

Los monitores son las 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. Pueden existir realizaciones hardware, software o mixtas. Aunque su objetivo es la medición de las prestaciones, se les denomina monitores ya que, debido a la imposibilidad de reproducir situaciones con la carga real, estos instrumentos hacen un seguimiento de lo que sucede en el sistema, es decir, lo monitorizan.

Estas herramientas son imprescindibles para evaluar el comportamiento de un sistema existente, a pesar de las perturbaciones que pueden introducir en el sistema cuyo comportamiento se va a evaluar.

Aparte de su utilización directa para tomar medidas de un sistema existente, permiten determinar la aproximación de una carga de *benchmark* a la carga real, obtener datos para la construcción de modelos y su validación posterior, etc.

1.7.2. Modelado

Ésta es la herramienta que hay que utilizar cuando se trata de evaluar el comportamiento de un sistema en el que hay algún elemento (hardware o software) que no está instalado.

En general, se fundamentan en la teoría de colas, pudiéndose considerar las colas o bien de forma individual o bien unidas formando redes abiertas o cerradas. Su tratamiento se puede realizar mediante los métodos analíticos que proporcionan las teorías de colas y de redes de colas, o por medio de la simulación.

Las técnicas de *simulación* consisten en la construcción de un programa que reproduce el comportamiento temporal del sistema, basándose en sus estados y sus transiciones. Los resultados se obtienen por extracción de estadísticas del comportamiento simulado del sistema.

Las técnicas analíticas se basan en la resolución mediante fórmulas cerradas o algoritmos aproximados de las ecuaciones matemáticas que representan el equilibrio que existe entre los eventos o transiciones de estado discontinuas que se producen en el sistema.

La limitación de los métodos analíticos es su incapacidad para tratar determinadas estructuras y comportamientos de las colas que existen en los sistemas informáticos. Los métodos de simulación no tienen estas limitaciones, pero, en general, son mucho más caros que los analíticos tanto en tiempo de cálculo como en esfuerzo de puesta a punto.

No obstante, la principal dificultad de esta herramienta reside en la obtención de datos lo suficientemente precisos para ejecutar el modelo y obtener resultados con el grado de aproximación que se exige.

1.7.3. *Benchmarking*

Es un método bastante frecuente de comparar sistemas informáticos frente a una carga característica de una instalación concreta, efectuándose la comparación, básicamente, a partir del tiempo necesario para su ejecución. Generalizando se puede considerar como la medición del comportamiento sobre un prototípico. Variantes de este método se usan para evaluar la potencia relativa de un sistema a lo largo de su ciclo de vida, para contrastar monitores y para validar modelos.

Las principales dificultades que se plantean son:

- cómo determinar esa carga característica, de forma que sea suficientemente reducida para ser manejable y suficientemente extensa para ser representativa;
- cómo valorar el aprovechamiento que hacen los programas de las peculiaridades de los distintos sistemas.

2. Técnicas de medida y de presentación de los resultados

2.1. Herramientas de medida: Monitores

2.1.1. Introducción

A la hora de realizar cualquier tipo de estudio sobre un sistema informático (ampliación, utilización, sintonización, etc.), es necesario tener información real acerca de lo que está ocurriendo en ese sistema. Para ello, es de vital importancia disponer de herramientas que permitan obtener esa información. Estas herramientas son conocidas con el nombre de monitores.

Un *monitor* es una herramienta utilizada para observar la actividad de un sistema informático mientras es utilizado por sus usuarios. En general, observan el comportamiento del sistema, recogen datos estadísticos de la ejecución de los programas, analizan los datos recogidos y presentan los resultados.

Una característica de una medida debe ser su repetibilidad, esto es, la posibilidad de realizar una medición diferentes veces, y que el resultado de este proceso (la medida) sea siempre el mismo. En caso de que no sea así, las sucesivas medidas se usan para reducir los errores intrínsecos del proceso de medición.

Ahora bien, en informática, el resultado de una medición será distinto unas veces de otras, ya que, normalmente, no es posible repetir las mismas condiciones de carga y en los mismos instantes. Por ello, se habla de monitorización y no de medición ya que lo que estrictamente se efectúa es un seguimiento de la actividad realizada.

El desarrollo de tales instrumentos para el seguimiento del comportamiento de sistemas informáticos utiliza las dos tecnologías que conviven en un sistema informático: el hardware y el software.

Otro objetivo de una herramienta de medida es cuantificar los resultados de una observación.

Los monitores pueden ser utilizados por todos los usuarios del sistema, desde los analistas de sistemas hasta los gestores y programadores del mismo.

La información aportada por el monitor puede ser útil:

- Para el usuario y administrador, pues puede ser necesario conocer una serie de características del sistema (capacidad, posibilidad de ampliación, planificación, etc.).
- Un administrador de sistema puede conocer la utilización de los recursos del sistema o el cuello de botella del mismo.
- Un programador puede encontrar los segmentos de software más utilizados y, en consecuencia, puede mejorar el comportamiento de los programas.
- Un administrador puede ajustar los parámetros del sistema y mejorar las prestaciones del mismo.
- Un analista de sistemas puede caracterizar la carga y crear cargas de prueba, cuyos resultados se pueden utilizar para planificar la carga del sistema.
- Un analista de sistemas puede encontrar los parámetros de un modelo y tener datos más reales para los modelos que representan el comportamiento temporal del sistema.
- Para el propio sistema, en aquellos que permiten su adaptación dinámica con la carga.

2.1.2. Características

Se ha dicho que el objetivo de un instrumento de monitorización consiste en cuantificar los resultados de la observación de un sistema informático. El desarrollo de herramientas para observar el comportamiento de un sistema informático sometido a una carga ha seguido dos direcciones separadas, caracterizadas por las tecnologías que se usan en un sistema informático. Los monitores software pueden ser programas, que detectan estados del sistema, o conjuntos de instrucciones, que se podrían denominar sondas software, capaces ambos de detectar acontecimientos. Los monitores hardware son dispositivos electrónicos que deben conectarse a puntos específicos del sistema (mediante sondas electrónicas) para detectar señales (niveles o impulsos de tensión) que caracterizan los fenómenos que deben observarse.

Aunque los monitores no se utilizan estrictamente como instrumentos de medida, su calidad viene definida por las mismas características, es decir:

- *Sobrecarga o interferencia.* Como todas las herramientas que miden fenómenos físicos, los instrumentos que observan el comportamiento de los sistemas informáticos, sustraen energía del sistema que están midiendo. La energía consumida por el instrumento de medida debe ser tan poca como sea posible de forma que la perturbación introducida por el instrumento no altere los resultados de la observación. En lo concerniente a los

monitores hardware este peligro existe a través de los puntos de conexión; no obstante, el uso de sondas de muy alta impedancia puede hacer despreciable esta perturbación. Por otro lado, los monitores software aumentan la carga del sistema y alteran, por consiguiente, su comportamiento. Sin embargo, permiten la obtención de medidas inalcanzables por un monitor hardware (aunque lo contrario también es cierto) y presentan ciertas ventajas de tipo operativo, por lo que se intentará reducir al mínimo posible los efectos de su presencia.

- *Precisión.* Otra característica de las herramientas de medida es su exactitud, la cual puede expresarse por el error que puede afectar al valor de los datos recogidos. Estos errores son debidos a diferentes causas: la interferencia del propio monitor, a una incorrecta instalación o utilización, al número de dígitos para representar la medición, etc. Cuando se utiliza un conjunto de herramientas cada una de ellas aporta su inexactitud al error total. Normalmente, este coeficiente de error deberá ser adjuntado por el fabricante, pero no lo suele dar, siendo el usuario quien debe evaluarlo, siempre que sea posible, por contraste de la herramienta.
- *Resolución.* Es la máxima frecuencia a la que se pueden detectar y registrar correctamente los datos. Es decir, mide la capacidad de la herramienta de separar dos acontecimientos consecutivos en el tiempo.
- *Ámbito o dominio de medida.* Hace referencia al tipo de acontecimientos que puede detectar y, en consecuencia, a aquellas características que es capaz de observar y medir.
- *Anchura de entrada.* Es el máximo número de bits de información de entrada que el monitor puede extraer en paralelo y procesar cuando se produce un acontecimiento.
- *Capacidad de reducción de datos.* Es la capacidad que puede tener el monitor de analizar, procesar y empaquetar datos durante la monitorización para un mejor tratamiento y comprensión de los mismos y para reducir el espacio necesario para almacenar los resultados.
- *Compatibilidad.* El hardware y el software de monitorización deben ser fácilmente adaptables a cualquier entorno y a cualquier requerimiento de la aplicación. Las herramientas no deben ser específicas para una determinada aplicación o sistema. Durante la operación de monitorización, la herramienta debe poder adaptarse de forma interactiva a las necesidades que puedan ir presentándose.
- *Coste.* Como en cada equipo industrial, el coste es un factor muy importante a tener en cuenta. Hay que considerar no sólo el coste de adquisición sino también los de instalación, mantenimiento, formación y operación.
- *Facilidad de instalación.* Otro factor a tener en cuenta es la facilidad de instalación del monitor, y también la facilidad de retirarlo del sistema, en caso de no ser necesario.

- *Facilidad de utilización.* Las herramientas de monitorización y medida deben ofrecer al usuario una interfaz que pueda ser utilizada por cualquier programador y no sólo por personal experto. El usuario debe poder centrar fácilmente su interés de forma dinámica desde cuestiones de alto nivel, como paralelismo, mensajes, etc., hasta otras de más bajo nivel o más puntuales, como procesos en espera, tiempos de retardo, etc.

Otros requerimientos típicos de estas herramientas, que es necesario tener en cuenta a la hora de realizar o seleccionar un monitor, son los siguientes:

- *Monitorización continua durante la operación.* Considerando un sistema distribuido que cambia continuamente por la creación y extinción de procesos y por la modificación de la estructura de interconexión entre las unidades distribuidas, es necesario tener un monitor que refleje la secuencia de modificaciones y la utilización de las unidades.
- *Presentación orientada a la aplicación.* La gran cantidad de datos de monitorización y prestaciones debe presentarse de forma orientada a la aplicación, reflejando la organización y la semántica del programa, dando los resultados preferiblemente en forma de tablas y gráficos. El almacenaje eficiente de la información puede aportar detalles acerca del pasado en caso de errores o comportamiento anormal del programa, a la vez que presentar resultados más detallados para analizar los problemas de prestaciones y diseño.
- *Integración.* La instrumentación debe ser transparente a los usuarios, debiendo permitir la realización de una monitorización continua del sistema sin una especial preparación. En general, la información acerca del sistema debe estar integrada en éste, debiendo concebirse esta integración en fase de diseño del monitor.
- *Realimentación.* En algunos casos, para conseguir un comportamiento adaptativo del sistema, los resultados de la monitorización deben ser realimentados al sistema para mejorar su comportamiento, como, por ejemplo, para algoritmos de equilibrado de la carga.

2.2. Conceptos de medida

Cuando se está frente a fenómenos de naturaleza compleja, que resultan de la coexistencia e interacción de muchos factores, se necesita describir con tanta precisión como sea posible lo que se está observando, puesto que se deben explorar las conexiones entre los acontecimientos para que sea posible explicarlos y predecirlos. Un método vital para alcanzar este objetivo es el de la medida, que permite tratar cuantitativamente los fenómenos; es decir, permite sugerir, verificar y establecer las relaciones entre las cantidades que caracterizan un fenómeno.

El estudio de un sistema informático, tanto cuando se está diseñando, como cuando está trabajando, requiere el uso de técnicas de medida, que son particu-

larmente útiles en la determinación del comportamiento del sistema. ¿Qué significa medir (o, mejor, monitorizar) un sistema? En general, significa recoger información de la actividad del sistema mientras atiende a los usuarios.

Las medidas (u observaciones) pueden clasificarse en dos grandes categorías, atendiendo a quien las va a usar: las requeridas por los usuarios del sistema y las requeridas por el propio sistema. Todas las medidas concernientes a la utilización de los recursos del sistema llevadas a cabo para evaluar su comportamiento, controlar su uso y planificar la adición de nuevos recursos pertenecen a la primera clase. Las efectuadas para el propio control del sistema, que le permiten adaptarse dinámicamente a los factores que condicionan su actividad (principalmente la carga de trabajo) y mantener un nivel adecuado de comportamiento externo, pertenecen a la segunda clase.

En general, hay que tener en cuenta que ni el hardware ni el software de un sistema informático se han diseñado con el propósito de ser medidos (u observados) al menos en sus primeros tiempos; actualmente esta concepción ha cambiado desde que se ha asumido la necesidad de conocer el comportamiento de un sistema informático. Este defecto de concepción, a menudo, restringe las medidas (u observaciones) que pueden realizarse de un sistema y requiere que se modifiquen algunas de sus funciones o que se añadan instrumentos caros para superar dichas restricciones.

La mesurabilidad de un sistema informático se puede definir como una función de la información que se puede obtener con un monitor y del coste de las mediciones. Es decir, la mesurabilidad de un sistema puede variar entre dos extremos, correspondiendo a la posibilidad de medir (observar) cada componente individual del sistema con el nivel de detalle deseado y a la total inaccesibilidad al sistema, respectivamente.

Hay diversas técnicas para medir (observar) un sistema informático; su elección depende del tipo de análisis que deba efectuarse y del nivel al que deba realizarse. Puesto que un acontecimiento es un cambio del estado del sistema, una forma de recoger datos de determinadas actividades del sistema es capturar todos los acontecimientos asociados a los cambios de estado y registrarlos en el mismo orden en que se producen. En este caso, la medición (observación) se efectúa por detección de acontecimientos.

Una técnica que, con frecuencia, se prefiere a la que se acaba de proponer, ya que no interfiere tanto el comportamiento del sistema, es la del muestreo, que consiste en interrumpir el sistema a intervalos regulares o aleatorios para detectar el estado de alguno de sus componentes. Si el número de muestras recogidas es suficientemente grande, este tipo de medidas puede ser suficientemente preciso.

2.2.1. Detección de acontecimientos

Se ha definido un acontecimiento como un cambio de estado del sistema. Se puede definir el estado de un sistema informático como el establecido por los

valores de todas sus memorias (en un sentido amplio, es decir, todo soporte capaz de almacenar alguna información visible o no desde el nivel del programa). Evidentemente, el seguimiento del estado de todas ellas constituiría una tarea imposible que requeriría un sistema tanto o más grande que el que se está observando. Cuando se observa un sistema con ánimo de evaluar su comportamiento, normalmente se restringe el estado del sistema a un conjunto limitado de memorias del sistema, que englobe aquellas que reflejen los cambios en el comportamiento del sistema que interesa retener, de acuerdo con los objetivos del estudio que se hayan planteado.

Cuando un acontecimiento está asociado con una función de un programa se dice que se está frente a un acontecimiento software, como, por ejemplo, el hecho de iniciar un programa una operación de E/S. De forma similar un acontecimiento hardware consiste en la aparición de una o más señales en los circuitos de un componente del sistema y es independiente del programa que se esté ejecutando en ese momento, como, por ejemplo, los movimientos del brazo de un disco.

No obstante, muchos acontecimientos hardware pueden ser reconocidos por el software ya que van acompañados de la modificación de alguna memoria accesible al software, es decir, llevan asociados un acontecimiento software. En la discusión posterior, cuando se efectúan medidas (observaciones) por detección de acontecimientos, se supondrá que se trata con acontecimientos reconocibles por el software. Por consiguiente, se ilustrará esta técnica haciendo referencia a acontecimientos software, aunque, conceptualmente se puede usar en medidas hardware.

El principio de detección de acontecimientos software es el de insertar un código suplementario (*traps*, es decir, interrupciones controladas por el programa) en lugares determinados del sistema operativo. Cuando se produce un acontecimiento que debe detectarse, este código transferirá el control a la rutina de tratamiento, que almacenará sus datos significativos, junto con el instante de aparición, en un área tampón, para posteriormente grabarlos en disco y devolver el control al sistema operativo. El conjunto de todos los datos registrados de esta forma constituye lo que se denomina una traza de acontecimientos. No obstante, hay que ser prudente a la hora de seleccionar la cantidad de acontecimientos que se desea registrar, ya que se corre el riesgo de que la observación de los acontecimientos software provoque una distorsión notable en el comportamiento del sistema.

En sistemas conducidos por interrupciones, se puede adoptar una técnica ligeramente distinta para detectar la aparición de determinados acontecimientos. En este caso, la dirección de la rutina de tratamiento de una interrupción determinada debe sustituirse por la dirección de inicio de la rutina de medición. Esto permite interceptar cada interrupción de este tipo y leer, cuando se produce la interrupción, los contenidos de determinadas posiciones de memoria o tablas. Estos datos, junto con el instante de aparición, se registrarán en cinta o

disco antes de devolver el control al sistema operativo para el tratamiento de la interrupción.

Algunos sistemas, normalmente grandes, no obstante, están equipados de hardware especial para facilitar la captación de acontecimientos por parte del software y disminuir su coste, especialmente en necesidades de CPU.

2.2.2. Muestreo

La técnica de detección de acontecimientos tratada en el apartado anterior se basa en la intercepción y registro de todos los acontecimientos de tipos determinados. Se trata, pues, de una técnica que se deberá utilizar sólo cuando sea necesario conocer la secuencia de estos acontecimientos o el número de veces que se han producido en un intervalo determinado. Cuando el conocimiento de estos elementos no es esencial para el estudio de un problema, puede adoptarse el método de muestreo.

El *muestreo* es una técnica estadística que se puede usar siempre que el conocimiento de todos los componentes, objetos o acontecimientos de una población sea imposible, poco práctica o demasiado cara. Entonces, en vez de examinar todos los elementos de la población completa, este método analiza solamente una parte de ellos, denominada una muestra. A partir de esta muestra es posible estimar, normalmente con un grado de precisión elevado, algunos de los parámetros estadísticos que caracterizan la población.

En la evaluación del comportamiento de un sistema informático, esta técnica presenta la ventaja de producir un volumen de datos mucho menor, reduciendo y simplificando, por lo tanto, su análisis. Además, el proceso de recogida de datos causa unos disturbios inferiores al sistema y esta reducción de interferencia hace que la alteración de la evaluación del comportamiento debida al instrumento de medida (observación) sea a menudo irrelevante y siempre más fácilmente controlable.

El muestreo puede usarse con dos objetivos distintos:

1. Evaluar las fracciones de un intervalo de tiempo dado que cada componente del sistema ha permanecido en distintos estados. Los datos recogidos durante el intervalo de medida están sujetos a un análisis posterior para determinar qué sucedió durante dicho intervalo.
2. Seguir la evolución de un sistema y predecir su comportamiento futuro de forma que puedan tomarse las decisiones que tendrán una influencia positiva en su comportamiento.

El primer objetivo se alcanza con herramientas de medida específicas, utilizadas normalmente por los usuarios de un sistema para evaluar la eficiencia de uso de los recursos. La precisión de los resultados viene determinada por el tamaño de la muestra. Debe observarse que, puesto que las cantidades muestreadas son función del tiempo, será necesario suponer que la carga es estacionaria durante largos períodos de tiempo.

No obstante, pueden obtenerse situaciones razonablemente estacionarias subdividiendo el intervalo de medida en períodos de tiempo relativamente cortos (de fracciones de minuto a algunos minutos) y agrupando los bloques de datos homogéneos.

El segundo objetivo se alcanza cuando la medida es una parte integrante del proceso de control de la actividad de un sistema informático. Muchas de las funciones del sistema operativo deben adaptarse dinámicamente a las variaciones de la carga o, de forma más general, a las variaciones de las condiciones del sistema. Las estimaciones de los parámetros, sobre las que se basan determinadas decisiones, consideran la historia reciente de sus valores. Puesto que la evolución del sistema no puede predecirse de forma determinista, es necesario crear un modelo basado en la estructura estocástica subyacente. En particular, en un sistema informático, las variables que representan los estados de los componentes del sistema son constantes por intervalos con saltos en instantes aleatorios; es decir son variables aleatorias discretas. No obstante su importancia, este tipo de modelos no se considerará en este texto ya que tienen mayor influencia en el diseño de sistemas que no en la evaluación del comportamiento.

2.3. Estructura del monitor

El esquema conceptual del monitor de un sistema computador es el que aparece en la figura 2.1.

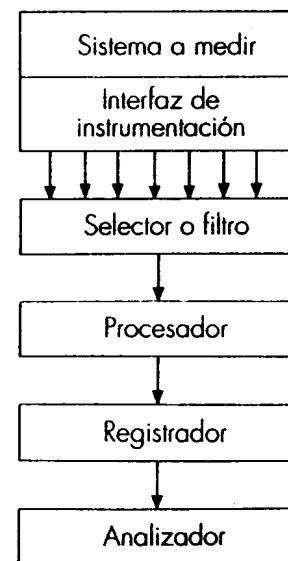


FIGURA 2.1.

La conexión entre el monitor y el sistema se realiza a través de la interfaz de instrumentación. Esta interfaz incluye todos aquellos elementos que permiten acceder a los puntos del sistema que contienen información relevante para la monitorización, tales como, por ejemplo, secciones de código incluido en el sistema operativo, temporizadores, puntos de sondeo, etc.

El elemento selector o de filtrado permite una captura selectiva de las actividades sondeadas. Esto le permite al usuario orientar la medida hacia los aspectos específicos que desea explorar. Además tiene por misión generar aquellas variables que no tienen existencia física directa en el sistema, pueden generarse en función de otras que son directamente observables.

El elemento procesador realiza las comprobaciones de los elementos del sistema que van a ser analizados o medidos. El resultado de las comprobaciones se graba en un medio de almacenamiento para poder ser analizado e interpretado y así obtener los resultados deseados.

Algunas veces la fase de análisis e interpretación se realiza en paralelo con la detección y captura de eventos, mientras que otras veces se hace después de recoger toda la información. En el primer caso (que presenta interesantes problemas estadísticos de estimación) se dice que se tiene un sistema de medida en tiempo real. Los sistemas de medida en tiempo real tienen la posibilidad de ser utilizados para el control dinámico de las prestaciones del sistema, mientras que el análisis diferido se usa para la planificación y ajuste del sistema a largo plazo.

2.4. Clasificación de los monitores

2.4.1. Introducción

La monitorización de un sistema informático no es una tarea habitual, por lo que estos sistemas no se diseñan (generalmente) para ser monitorizados, lo que suele añadir una complejidad adicional a la hora de diseñar un monitor. No obstante, en sistemas grandes y en muchos de los multiprocesadores que se diseñan actualmente, el monitor está integrado en el sistema y se diseña al mismo tiempo que el sistema informático.

El desarrollo de herramientas de medida para computadores, bajo una carga dada, ha seguido caminos diferentes, separados según las técnicas empleadas para su realización.

De este modo, los monitores se clasifican, según su implantación en:

- *Monitores software*, que son programas o ampliaciones del sistema operativo que acceden al estado del sistema, informando al usuario (ya sea de forma continua, o cuando éste lo requiere) sobre dicho estado.
- *Monitores hardware*, que son dispositivos electrónicos que se conectan a determinados puntos del sistema, donde se encargan de detectar determinados niveles o señales eléctricas que caracterizan la evolución del sistema.

- *Monitores híbridos*, que son una combinación de las dos técnicas anteriores, intentando combinar las ventajas de una y otra.

Según su mecanismo de activación se clasifican en:

- *Monitores de eventos o acontecimientos*, que son aquellos que se activan por la aparición de ciertos eventos. Si el evento se da con frecuencia, la sobrecarga que producen es elevada. Este evento puede ser también una decisión externa del responsable del sistema.
- *Monitores de muestreo*, que son aquellos que se activan a intervalos de tiempo fijos o aleatorios mediante interrupciones de reloj. La frecuencia de muestreo viene determinada por la frecuencia de aparición del estado que se desea analizar y por la resolución deseada.

Y según su forma de mostrar los resultados se clasifican en:

- *Monitores en tiempo real*, que constan de un módulo analizador que procesa los datos recogidos a medida que los recibe. En consecuencia, el módulo de proceso debe ejecutarse de forma continua, o con gran frecuencia, para poder presentar resultados parciales al analizador que informará de la evolución del sistema.
- *Monitores batch*, que difieren de los anteriores en que primero se recoge la totalidad de la información y, una vez terminado el período de recolección, se procesa dicha información.

Las tres clasificaciones pueden utilizarse de forma conjunta, así pues, un monitor puede clasificarse como software, conducido por muestreo y con análisis en batch de los resultados. La clasificación según el tipo de interfaz es la más habitual.

En la siguiente tabla se muestra cómo realizan cada una de las funciones los monitores software, hardware e híbrido.

Función	Monitor software	Monitor hardware	Monitor híbrido
Instrumentación	Traps Interceptación Muestreo	Sondas Conexión permanente	Traps Sondas
Selector	Programa	Programa Lógica	Programa Lógica
Procesador	Programa	Lógica Programa	Lógica Programa
Registrador	Memoria Disco	Contador Memoria Disco	Contador Memoria Disco
Analizador	Programa	Programa	Programa

Hay que señalar que, en la tabla precedente, mientras las referencias a programa de los monitores software son ejecutadas por la propia CPU del sistema que se está analizando, en los monitores hardware corresponden a una CPU distinta y en los híbridos pueden darse los dos casos.

2.4.2. Monitores software

Los monitores software son los más fáciles de encontrar en un sistema informático y aunque son utilizados de forma general para estudiar el comportamiento de un sistema son, sobre todo, adecuados para monitorizar los sistemas operativos, las redes y las bases de datos así como las aplicaciones que las utilizan.

Cada activación del monitor implica la ejecución de varias instrucciones por parte de la CPU del sistema que se está analizando, lo que puede provocar una gran sobrecarga en el sistema si la causa de la activación se produce con gran frecuencia.

Los monitores software pueden extraer información del sistema a través del seguimiento de los acontecimientos o eventos (mediante traps o interceptación) o muestrear los estados en que se halla. Las dos técnicas para la obtención de datos no son mutuamente excluyentes, sino que pueden ser utilizadas a la vez por una misma herramienta.

La aportación de código que necesitan los monitores software se puede realizar de tres formas distintas:

- Adición de un nuevo programa.
- Modificación del software que se va a medir.
- Modificación del sistema operativo.

El primer método es el más fácil de utilizar, a la vez que puede ser eliminado fácilmente, cuando ya no se necesite tomar medidas. Además se garantiza la integridad del sistema operativo y del propio programa que se va a medir. Generalmente las funciones del programa monitor se limitan a tomar medidas del sistema así como de los programas que se van a medir.

El segundo método está basado en el uso de sondas software, las cuales no son más que conjuntos de instrucciones insertadas en puntos clave del programa que debe observarse. Cada sonda debe detectar la llegada del flujo de control al punto crítico donde está situada, con lo que se puede determinar el número de veces y los instantes que se pasa por ese lugar, a la vez que el contenido de ciertas variables cuando se está ejecutando esa zona de código.

El tercer método se utiliza cuando determinadas variables que se deben medir sólo están disponibles para el sistema operativo, como pueden ser relojes de tiempo real, estado de colas del planificador, etc.

Como el monitor software necesita recursos del sistema, puesto que ocupa memoria, consume tiempo de CPU y realiza operaciones de E/S, se debe vigilar que se cumplan los siguientes requerimientos:

- La extracción de datos del sistema debe realizarse de forma cualitativa y cuantitativa.
- Las modificaciones del sistema operativo deben ser las mínimas posibles.
- La alteración de las características de carga del sistema, y, por lo tanto, de sus prestaciones no debe ser apreciable.
- Los recursos requeridos deben ser los mínimos posibles.

Existe un juego de intereses contrarios en cuanto a la precisión de las medidas y a la interferencia que éstas puedan ocasionar, ya que si se requiere una gran precisión de resultados, se tendrán que realizar un gran número de observaciones, lo que provocará una sobrecarga del sistema que hará que las medidas obtenidas no sean fiables.

Por otro lado, si no se quiere sobrecargar mucho el sistema y se realizan pocas observaciones, éstas pueden no recoger todo lo que realmente está sucediendo en el sistema, por tanto, no tendrán validez (o no serán significativas).

El monitor debe ofrecer la posibilidad de habilitar/deshabilitar, con facilidad, la monitorización, total o de determinadas partes, para evitar sobrecargar el sistema cuando no se utilice o no se utilicen determinadas partes del monitor.

En cuanto a su prioridad, si el monitor se ejecuta asincrónicamente, la prioridad debe ser baja para no afectar a las operaciones importantes del sistema, pero si es necesario un registro minucioso del funcionamiento del sistema, su prioridad deberá ser alta. En general, su prioridad deberá ser superior a la de los acontecimientos que debe observar.

En cualquier caso, puesto que para su funcionamiento requieren de la ejecución de instrucciones del propio código del sistema analizado, habrá que tener presente en todo momento que teóricamente no pueden detectar acontecimientos que se produzcan por debajo del tiempo de ejecución de una instrucción ni acceder a acontecimientos que no se reflejen en alguna posición de memoria del sistema que sea accesible por la ejecución de una instrucción.

La mayoría de los monitores están escritos en lenguajes de programación de bajo nivel, tales como ensamblador o C, para mantener al mínimo la sobrecarga. Ya que un monitor software forma parte, generalmente, del sistema operativo, es mejor escribir ambos en el mismo lenguaje de programación.

A) Monitores software conducidos por acontecimientos

Existen dos tipos de instrumentación en los monitores conducidos por eventos o acontecimientos:

- Por *traps*, o segmentos de código que se insertan en determinadas partes del programa. Cuando el flujo de control llega a ellos, se ejecutan, realizando determinadas acciones encaminadas a señalar la aparición de ese evento y a extraer la información que lleva asociada.
- Por interceptación de las interrupciones, que desvían el tratamiento de la interrupción, redireccionando dicho tratamiento a una zona de código

donde se halla el monitor, el cual contabiliza el tipo de interrupción y recopila la información asociada al acontecimiento que ha provocado la interrupción.

B) Monitores software de muestreo

Un monitor de muestreo interfiere mucho menos en el funcionamiento del sistema que un monitor conducido por eventos. Además, es posible variar sus dos principales características con el fin de ajustar la interferencia a unos niveles adecuados. Así, se puede variar la frecuencia de muestreo y el número de variables que hay que medir en cada uno de estos instantes de muestreo, teniendo en cuenta que la interferencia aumenta con el número de medidas que se toman y con la frecuencia elegida.

El procedimiento de extracción de datos utiliza un reloj, normalmente un temporizador interno, que provoca una interrupción del sistema a intervalos regulares de tiempo, para la extracción de los datos concernientes al estado del mismo.

Otras veces se procede a la recolección de datos a intervalos aleatorios aunque con una longitud media fija, para evitar que un posible sincronismo entre la evolución de los estados y los intervalos fijos impida al monitor detectar todos los estados del sistema. Otra solución para evitar el sincronismo se basa en elegir un período de muestreo que sea un submúltiplo del que utiliza el sistema operativo en el lanzamiento de aquellas funciones que se realizan a intervalos periódicos. La sincronización entre muestreo y eventos puede ser detectada fácilmente.

La figura 2.2 muestra el esquema funcional de un muestreador en un sistema conducido por interrupciones provocadas por una señal de reloj.

Lógicamente, el programa de extracción debe ser el de mayor prioridad, y debe ejecutarse con las interrupciones inhibidas. Por tanto, es necesario que el programa sea muy rápido (poca demanda de CPU), con el fin de no interferir en las prestaciones del sistema.

Los datos que se deben tomar son de dos tipos:

- Acerca de las actividades suspendidas cuando se provocó el disparo del temporizador.
- Acerca de las actividades en progreso, como el estado de canales, contenido de ciertas áreas de memoria, etc.

Los datos recolectados se deben almacenar de la forma más compacta posible. Generalmente, se subdividen en grupos temporales, formando lo que se denominan *registros lógicos*, que no son más que un conjunto de datos recogidos durante un período de tiempo.

Este período de tiempo puede ser de unos pocos minutos o, en el caso de tiempos de muestreo cortos, de segundos.

Si se supone que se desea conocer el estado de la CPU y del canal cada 333 ms, suponiendo un período de extracción de 8 horas (un día), se tendrá que el

número de muestras se eleva a $8 \times 60 \times 60 \times (1000/333) = 86486$ muestras. Considerando registros lógicos de 5 minutos, este número se ve reducido a $8 \times 12 = 96$ datos, conteniendo cada uno la media de $5 \times 60 \times (1000/333) = 900$ muestras.

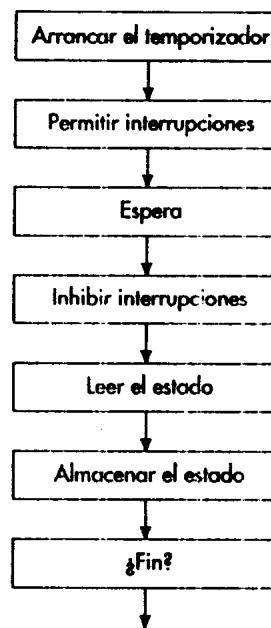


FIGURA 2.2. Esquema funcional de un muestreador en un sistema conducido por interrupción provocada por una señal de reloj.

a) Consideraciones temporales

Un monitor software debe ser capaz, entre otras cosas, de dar una duración de los fenómenos observados; es decir, de dar medidas temporales.

Esto lo puede realizar una herramienta software de dos formas: por muestreo o realizando medidas directamente. El primer método da medidas estadísticas en cuanto a tiempos medios de utilización de recursos, a partir de los cuales se pueden realizar medidas indirectas de otros fenómenos, como fallos de página, etc.

El método de tomar medidas directas es más interesante, por cuanto es capaz de dar la duración exacta de los fenómenos (instante de finalización menos instante de comienzo), así como la traza de acontecimientos del sistema, pero tiene el inconveniente de ser más difícil de implantar, sobre todo si se tiene en cuenta que algunos procesos entran con un alto grado de prioridad.

Ambos tipos de monitores software necesitan de un reloj en el sistema. Y en el caso de muestreadores, este reloj debe ser capaz, además, de realizar interrupciones en el sistema.

Una forma muy usual de realizar este reloj es utilizar una posición de la memoria principal o un registro interno del procesador como un contador binario que es incrementado (o decrementado) por un dispositivo hardware capaz de generar pulsos a intervalos regulares (generalmente un cristal de cuarzo). La frecuencia de este reloj da la resolución del sistema, es decir, la mínima cantidad de tiempo medible por él.

Muchas veces existen relojes paralelos que se inicializan cuando se arranca un proceso, salvando el estado de estos relojes cuando el proceso se suspende y recargándolo cuando éste se carga de nuevo. Cuando el proceso concluye, en este reloj se tiene el tiempo de ejecución del mismo. Lógicamente, el manejo de estos relojes corre a cargo del sistema operativo.

Existen algunas causas que hacen que este método no sea del todo exacto. Las principales son:

- Existe un retardo variable entre el comienzo y el final de la actividad y el arranque y la parada del reloj lógico asociado.
- El sistema operativo puede no ser totalmente honesto en cuanto a la adjudicación de los tiempos, y así el tiempo de tratamiento de determinadas interrupciones es cargado al proceso que tiene la CPU en ese momento.
- El acceso al reloj es normalmente una tarea del supervisor, y el tiempo de entrada de éste es aleatorio, cargando este tiempo al reloj lógico del proceso.

Otro elemento importante en la precisión de las medidas directas es la resolución del reloj. Si la resolución es demasiado pequeña (el tiempo entre dos ticks consecutivos es demasiado grande), es posible que el reloj no sea adecuado para la medida de la duración de determinadas actividades, o bien que los resultados no tengan la precisión necesaria. Se puede comprobar con un ejemplo en que se supone que hay s milisegundos entre dos ticks consecutivos (figura 2.3).

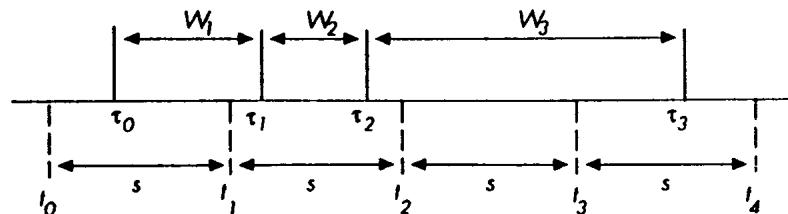


FIGURA 2.3.

W_i , cuya duración real es $T_i = \tau_i - \tau_0$ se considerará que tiene una longitud igual a $s = (t_i - t_0)$, es decir, de menor longitud que la real.

W_2 , cuya duración real es $T_2 = \tau_2 - \tau_1$ se considerará que tiene una longitud igual a $0 = (\tau_1 - \tau_1)$, es decir, evidentemente de mayor longitud que la real.

W_3 , cuya duración real es $T_3 = \tau_3 - \tau_2$ se considerará que tiene una longitud igual a $2s = (\tau_3 - \tau_1)$, es decir, de mayor longitud que la real.

De manera general, se consulta el valor del reloj cuando comienza la actividad (instante T_i). Se supone que el valor real en ese instante es T_a , y que T es la duración del intervalo entre dos ticks del reloj (figura 2.4). Si la actividad termina en T_f y se consulta el reloj en ese momento, éste contendrá el valor T_b . La duración real de la actividad es de:

$$T_{real} = T_f - T_i$$

mientras que el valor medido es de:

$$T_{medido} = T_b - T_a$$

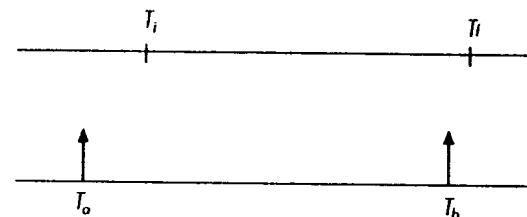


FIGURA 2.4.

Considerando el error como la diferencia entre lo que realmente dura y lo que se ha medido, se obtiene:

$$Error = T_{real} - T_{medido} = (T_f - T_i) - (T_b - T_a) = (T_f - T_b) - (T_i - T_a)$$

Tanto $(T_f - T_b)$ como $(T_i - T_a)$ pueden variar entre 0 y T , por lo que el error podrá variar entre $-T$ y T , o lo que es lo mismo

$$0 \leq |Error| \leq T$$

Es decir, que el error depende de T , por lo que la resolución debe de ser grande para que el error cometido en las medidas sea pequeño.

En caso de utilizar la técnica de muestreo existe un problema similar con el intervalo de muestreo, como muestra la figura 2.5.

Como puede verse, los resultados obtenidos serán los siguientes:

- A W_1 se le asigna 1 tick de reloj. Mucho más de lo que le corresponde.
- W_2 no se detecta.
- A W_3 se le asignan 2 ticks, único caso válido.

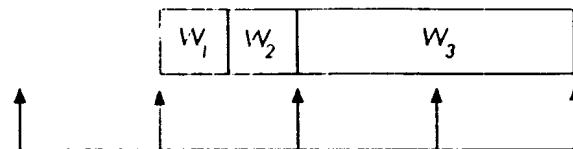


FIGURA 2.5.

Lo anterior se puede plantear de una manera más formal. De esta forma si se considera un muestreo genérico, como el de la figura 2.6, se tiene:

$$T_{real} = T_f - T_i$$

$$T_{estimado} = T_b - T_a$$

y por tanto:

$$Error = T_{real} - T_{estimado}$$

$$0 \leq |Error| \leq T$$

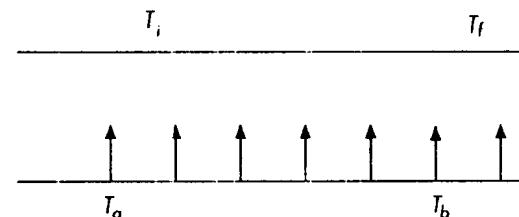


FIGURA 2.6.

Es decir, que cuanto mayor sea T , tanto mayor podrá ser el error, por lo que se deberán tomar intervalos de muestreo pequeños para disminuir el error. Lógicamente, se provocará una mayor sobrecarga tal y como se comentó anteriormente.

2.4.3. Monitores hardware

Son dispositivos para medir las prestaciones de sistemas computadores y que se conectan al hardware del sistema que se va a monitorizar por medio de sondas electrónicas, que son elementos capaces de detectar eventos de tipo eléctrico (flancos, cambios de nivel, etc.). Con ellos se pueden deducir los índices de prestaciones más importantes, sobre todo, aquellos de más bajo nivel.

Su principal característica es que son externos al sistema que van a medir, lo que implica que:

- No utilizan recursos del sistema que se va a monitorizar.
- No producen interferencias con éste.
- Son muy rápidos.

Por tanto, en un principio parece que sean mejores que los software, pero tienen algunas desventajas frente a estos últimos. Las principales son:

- Son más difíciles de instalar.
- Existen magnitudes a las que sólo se puede acceder desde el software.
- Requieren, para su operación y para el análisis de sus resultados, de personal más especializado.
- Pueden interaccionar a nivel eléctrico con el sistema que se va a monitorizar, provocando perturbaciones que resulten en un funcionamiento anómalo del sistema monitorizado. No se deben utilizar nunca en medidas sobre sistemas críticos en funcionamiento real.

El esquema más simple de un monitor hardware está representado en la figura 2.7.

En ese monitor, los módulos lógicos tienen la misión de filtrar las señales, de forma que sólo se obtengan disparos cuando se produce una determinada combinación de las señales de entrada, lo que equivale a decir que se está produciendo un determinado evento.

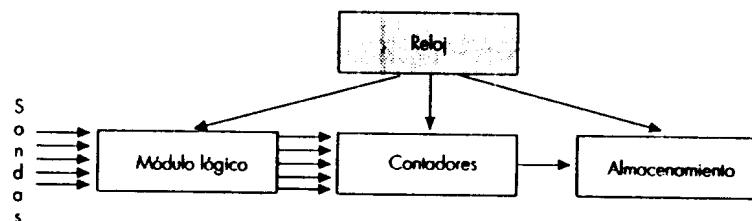


FIGURA 2.7. Esquema simple de un monitor hardware.

Los contadores se encargan de registrar el número de veces que se produce un determinado fenómeno. La parte de almacenamiento tiene la misión de memorizar de forma permanente el valor de los contadores. Puede ser opcional. El reloj se encarga de dar la señal de sincronismo con la que se deben captar los eventos.

Una realización muy simple del esquema anterior se puede ver en la figura 2.8. Se pretende determinar el tiempo de utilización de un determinado recurso. Este tiempo se caracteriza por un nivel alto en la salida de un biestable. Esta salida es interceptada por las sondas, las cuales llevan el nivel lógico de salida al módulo lógico, que en este sencillo caso no es más que una puerta AND cuya

otra entrada está conectada a la señal de reloj del monitor. De esta forma, la salida del módulo lógico será un tren de pulsos si la entrada está a nivel alto.

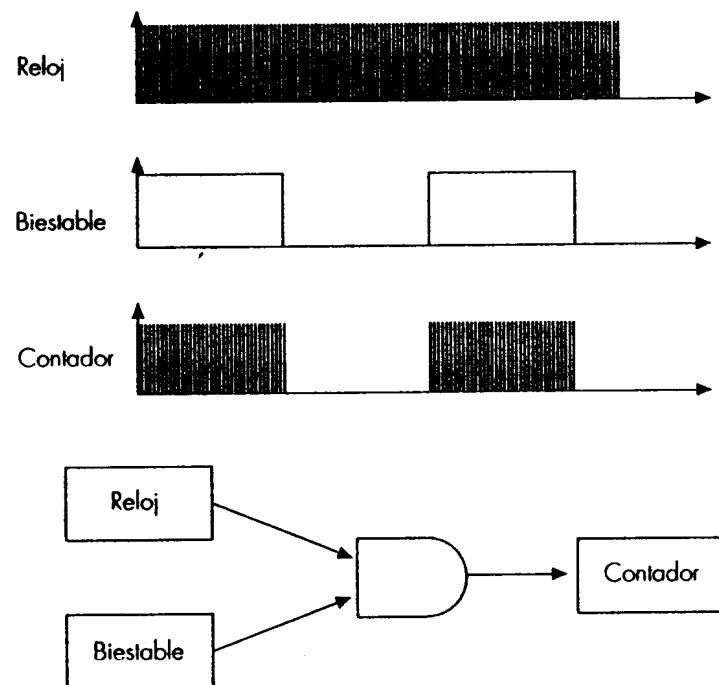


FIGURA 2.8.

La salida de este módulo lógico se lleva a un contador para que registre el número de pulsos recibidos. En otro contador se tiene directamente la señal de reloj.

La salida del monitor son las cuentas de los contadores, ya que la razón entre éstas da directamente el tanto por ciento de ocupación del dispositivo, que es lo que se pretendía conocer.

De este sistema tan simple se puede ir, por adición de módulos, complicando al monitor, con el fin de dotarle de más potencia.

El siguiente paso consiste en sustituir parte de la batería de contadores por una memoria RAM encargada de registrar de forma secuencial los eventos que se van produciendo, así como el instante de su aparición. En la figura 2.9 se puede ver un esquema que responde a estas características donde, de hecho, se han sustituido los distintos elementos por un microprocesador.

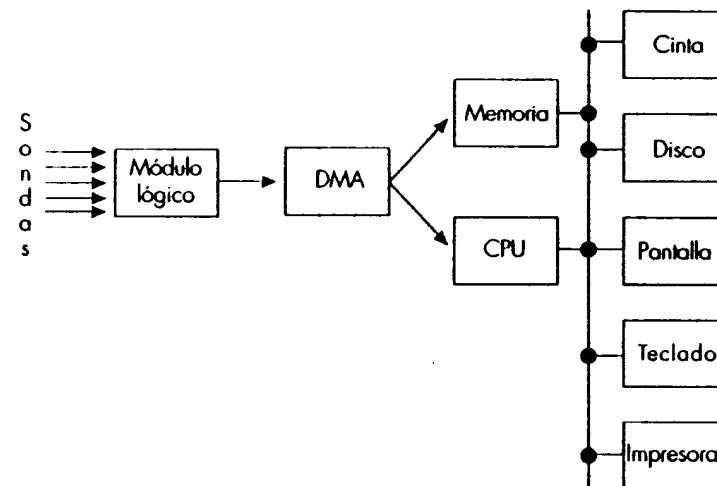


FIGURA 2.9.

Desde la aparición del microprocesador se han sustituido la mayor parte de los módulos por uno de estos componentes, que se puede encargar de realizar las operaciones de filtrado, cuenta y memorización. Los contadores son ahora actualizados y almacenados por el propio microprocesador, que, además, puede interpretar los resultados y visualizarlos.

La ventaja es evidente ya que los módulos lógicos, que en un principio eran cableados, pasan a ser programados, con las ventajas (aumento de flexibilidad, etc.) que ello conlleva.

El problema que aparece es que, al ser el microprocesador el encargado de detectar, filtrar y almacenar la información, la frecuencia de muestreo se ralentiza, con lo que se está perdiendo una de las ventajas de los monitores hardware respecto de los monitores software, la rapidez.

Para soslayar este problema, se deja parte de la lógica de filtrado a la vez que se dispone de un módulo de registradores donde se almacena temporalmente la información hasta que el procesador esté disponible para utilizarla.

En resumen, puede decirse que un monitor hardware podrá reconocer todos aquellos acontecimientos que se reflejen en puntos fijos del sistema (donde se conectarán las sondas) o produzcan secuencias fijas de señales en puntos determinados.

2.4.4. Monitores híbridos

Los monitores híbridos son una mezcla de los monitores software y hardware que tratan de aunar las ventajas de ambos monitores y evitar sus inconvenientes.

Estos monitores añaden sobre el sistema operativo una parte muy pequeña de código que envía información al monitor hardware que está observando de forma continua la ejecución del procesador principal, según se puede ver en la figura 2.10. La parte de código que se añade puede consistir en la ejecución de una instrucción de entrada/salida a la que se añade una palabra de 16 ó 32 bits que hace referencia al suceso que se desea monitorizar.

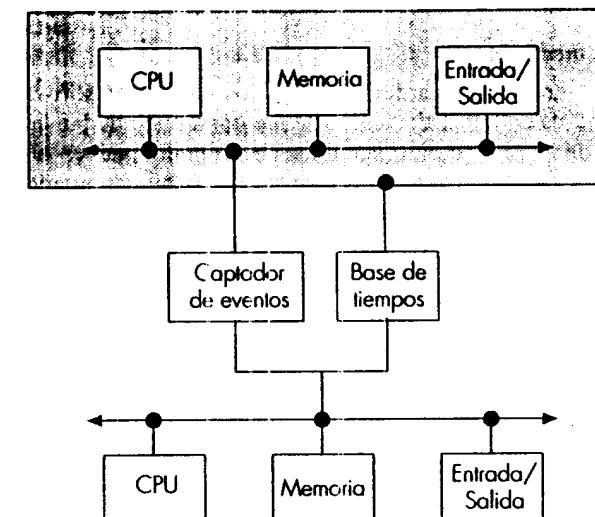


FIGURA 2.10.

La parte hardware de un monitor híbrido es un sistema con microprocesador con una base de tiempos que cada vez que la parte software envía una información añade el tiempo en que tiene lugar la captura. El tiempo y la información que se captura son almacenados y procesados por el procesador del monitor. Toda la información monitorizada se representa al final o de forma concurrente, en la pantalla de un ordenador personal o estación de trabajo (que constituye el monitor), en forma de gráficas o tablas de datos, además de almacenarse en un archivo.

Al reducirse generalmente la parte software a la ejecución de una instrucción de entrada/salida, y al procesarse y almacenarse en otro computador, se elimina una parte de la sobrecarga que tenía un monitor software.

Estos monitores son los más usados en el diseño de nuevas arquitecturas de computadores, y vienen integrados en algunos computadores modernos. También son muy usados en el estudio del comportamiento de prototipos de nuevas máquinas.

2.4.5. Comparación entre monitores

Dado un problema de monitorización, la elección del tipo de monitor que se debe utilizar no es difícil de resolver, ya que en la mayoría de las aplicaciones sólo uno de ellos es el adecuado. La base de comparación está constituida por todas las características que se exigen a un monitor (véase apartado 2.1.2.).

De ellas, las que principalmente deben tomarse en cuenta a la hora de seleccionar el tipo de monitor son las siguientes:

a) Dominio de medición

Como se ha mencionado anteriormente, los monitores software tienen el acceso controlado a aquellos elementos de memoria que pueden leerse mediante la ejecución de una instrucción máquina; por lo tanto, pueden observar acontecimientos relacionados con el hardware sólo si están acompañados por una transferencia de control a una dirección conocida (interrupción) o si almacenan la información identificadora de forma que pueda ser investigada posteriormente. Los monitores hardware pueden observar cualquier elemento de memoria, siempre que no sea necesaria ninguna señal generada por el sistema para recuperar la información del estado. Un monitor hardware no puede observar directamente el contenido de una memoria de acceso directo; tiene acceso a la información almacenada sólo cuando entra o sale de ella. Estrictamente sucede lo mismo en un monitor software, pero éste puede decidir qué datos deben ser accedidos para comprobarlos o procesarlos, mientras que el monitor hardware es sólo un observador pasivo sobre el sistema de memorias.

En resumen, mientras el monitor hardware sólo es capaz de seguir el comportamiento de acontecimientos que se reflejen en posiciones fijas de memoria o que provoquen transferencias de control a direcciones fijas o la ejecución de instrucciones determinadas (como llamadas al supervisor), el monitor software puede seguir los acontecimientos que se reflejen en posiciones de memoria susceptibles de ser leídas por la ejecución de una instrucción. Así, por ejemplo, un monitor software no será capaz de contar el número de instrucciones máquina ejecutadas en un período determinado, mientras que un monitor hardware sí lo será. En otro caso típico, toda información ligada a los nombres de trabajos, programas o archivos, que pueden hallarse en posiciones variables de la memoria, no podrá ser seguida con un monitor hardware y deberá serlo por uno software. Aparte de esos ejemplos, hay numerosas magnitudes que tanto las puede seguir un monitor hardware como uno software, como, por ejemplo, la utilización de los dispositivos (CPU, canales, discos, etc.).

b) Resolución

Los monitores hardware tienen una capacidad de resolver acontecimientos a frecuencias elevadas (de 10 a 50 MHz, por ejemplo) y es una magnitud absoluta determinada por la velocidad de las sondas y la lógica del monitor. El

monitor software no puede descender por debajo de su acontecimiento elemental: la ejecución de una instrucción. Por lo tanto, la máxima frecuencia de entrada del monitor software viene fijada por la máxima frecuencia de ejecución de instrucciones, una vez descontadas las necesarias para la ejecución del monitor. Es pues una magnitud relativa. Ahora bien, el monitor software tiene la posibilidad de "parar" el sistema y permitir el análisis de todos los acontecimientos que se han producido hasta que desaparece el posible atasco; bien es verdad que ello provoca una distorsión, tal vez importante, en el funcionamiento del sistema.

c) Anchura de entrada

Un monitor software puede detectar los acontecimientos sólo secuencialmente. Sin embargo, como se ha dicho, es capaz de "parar" el sistema hasta extraer toda la información necesaria. En este sentido la anchura de entrada es, teóricamente, ilimitada, con la perturbación del *overhead* introducido. El monitor hardware permite la detección de acontecimientos en paralelo, pero su anchura está limitada por el número de sondas disponibles.

d) Interferencia

El monitor hardware prácticamente no introduce ninguna perturbación en el sistema observado (si las sondas han sido correctamente elegidas), mientras que el monitor software provoca una perturbación que puede ser apreciable, ya que utiliza los propios recursos del sistema.

e) Facilidad de uso

Utilizar un monitor hardware requiere un buen conocimiento del hardware del sistema y, por tanto, un mayor grado de especialización que el monitor software, que de hecho es una extensión del sistema operativo utilizado.

f) Coste

Los monitores hardware son más caros que los software tanto de instalación como de funcionamiento. Este hecho es suficiente en algunos casos para determinar la elección.

g) Resumen

Si la cantidad de datos que se espera obtener es muy elevada, lo apropiado es utilizar un monitor hardware que incorpore almacenamiento secundario, evitando de esta forma sobrecargar el sistema.

Los monitores software consumen recursos del sistema que deberían estar disponibles para los usuarios. Por contra, los monitores hardware consumen muy pocos recursos, siendo incluso invisibles al sistema.

Los monitores hardware suelen diseñarse para realizar medidas sobre sistemas de distintos vendedores, o con distintos sistemas operativos. Los monitores software se desarrollan para un hardware y software particular, luego no son portables a sistemas de distintos vendedores.

En momentos de mal funcionamiento del sistema, el monitor hardware sigue tomando medidas que pueden ser de utilidad para resolver el problema, mientras que el monitor software puede no funcionar adecuadamente en este caso.

Ambos monitores pueden introducir errores en los datos medidos.

En la siguiente tabla, se muestra a modo de resumen una comparación entre los diferentes tipos de monitores:

Característica	Monitor hardware	Monitor software	Monitor híbrido
Dominio	Bajo nivel (fijo)	Alto nivel	Todos
Resolución	Alta (fija)	Baja (variable)	Alta
Anchura	Finita	Infinita	Infinita
Interferencia	Baja	Alta	Media
Coste	Alto	Bajo	Alto
Reducción	No	Sí	No/Sí
Portabilidad	Sí	No	No

2.4.6. Monitores de ejecución de programas

Proporcionan información de un programa durante su ejecución y sirven para saber:

- Dónde pasa un programa su tiempo de ejecución.
- Cuántas veces se ejecuta una línea de código del programa.
- Cuántas veces se llama a un procedimiento y desde dónde.
- A cuántos archivos accede un programa y durante cuánto tiempo.

Toda esta información con otra general sobre consumos de recursos del sistema como tiempo de CPU, accesos a disco, actividad de interrupciones, etc., serán de suma importancia para optimizar y mejorar el comportamiento y tiempo de ejecución de un programa.

En la siguiente tabla se pueden ver los resultados obtenidos ejecutando un programa de simulación escrito en lenguaje C.

Núm. Función	Función	Tiempo (seg)	Núm. llamadas
1	ranf	6.9912	180059
2	main	3.4515	1
3	enlist	2.9586	87514
4	schedule	2.5681	70041
5	erlang	2.4798	10003
6	cause	2.1048	82140
7	release	1.4832	30005
8	getelm	0.9882	84832
9	putelm	0.8001	84822
10	request	0.6639	27465
11	enqueue	0.5969	14791
12	preempt	0.5771	14661
13	expnll	0.4867	10006
14	replpage	0.2770	1
15	random	0.2605	10005
16	time	0.1840	20008
17	suspend	0.0552	2682
18	endpage	0.0500	1
19	mpl	0.0095	1
20	IJ	0.0005	6

El programa se ejecutó en un PC 486/33 Mhz y el tiempo total de ejecución fue de 26.929 seg. Así, por ejemplo, según la tabla, la función *ranf*, que genera números aleatorios en el intervalo 0-1, se ejecutó 180059 veces y en total tardó 6.9912 seg.

2.4.7. Monitor de contabilidad

El objetivo de un monitor de contabilidad es obtener el consumo de recursos del sistema que realiza cada usuario. Su origen se halla en la necesidad de algunas instalaciones de facturar a los usuarios el consumo de recursos realizado en la ejecución de las aplicaciones correspondientes. Además, existen algunas empresas y universidades donde cada usuario tiene limitado el consumo de

recursos del sistema, y superar el límite adjudicado, supone pagar los recursos extra consumidos.

La tarea de usar este monitor para contabilizar los recursos que consumen los usuarios suele llevarla a cabo el administrador o personal del departamento de explotación del sistema informático.

Los monitores de contabilidad permiten realizar las siguientes operaciones:

- Monitorizar la utilización del espacio de disco para usuarios individuales, como número de escrituras y lecturas, número de bytes escritos y leídos, etc.
- Registrar datos de sesiones, como número de sesiones realizadas y su duración.
- Generar resúmenes de informes y archivos que pueden ser utilizados para analizar el trabajo del sistema y el consumo de recursos de cada usuario, como tiempo de CPU, nombre de los programas ejecutados, número de escrituras y lecturas desde cada terminal, bytes leídos y escritos, etc.

Para cumplir estas tareas, el monitor de contabilidad proporciona una serie de órdenes que permiten crear datos, borrarlos, visualizarlos, mezclarlos, resumirlos y crear informes. Estas órdenes actúan de forma conjunta en el proceso de la información monitorizada, es decir, la salida de una orden puede ser utilizada como entrada para otra.

Los monitores de contabilidad también registran la información referente a condiciones anormales, como violaciones de seguridad, terminación anormal de trabajos, errores en dispositivos, rearranques del sistema, etc.

La unidad de trabajo de un monitor de contabilidad es el programa y la información registrada se realiza para cada usuario y para cada programa ejecutado.

2.5. Presentación de los resultados

2.5.1. Introducción

En los apartados anteriores se vieron los instrumentos de medida utilizados para la obtención de los datos en un estudio de evaluación. Uno de los pasos más importantes en cualquier estudio es precisamente la presentación de los resultados, ya que la persona encargada de la toma de decisiones tendrá que basarlas en la interpretación de aquéllos. Por ello la presentación de los resultados debe ser clara, concisa y simple, para facilitar al máximo su interpretación.

En muchos casos (seguimiento de un sistema, presentación a dirección, etc.) es preferible utilizar una presentación gráfica de los resultados en vez de una explicación textual, debido a que un gráfico puede ser interpretado más rápida y fácilmente dado que permite una presentación más concisa de los datos.

Debido a la gran cantidad de variables y de objetivos diferentes que pueden considerarse, no existe un método estándar para la presentación de los resultados. Además de gráficas de líneas, barras, histogramas, etc., se suelen utilizar

también gráficos que se han desarrollado específicamente para el análisis de prestaciones de sistemas de computadores. Ejemplos de estos últimos son los diagramas de Gantt y de Kiviat que se verán en este apartado.

Entre los gráficos de tipo general, las formas que se adoptan con mayor frecuencia son las de histograma y la de evolución de una variable respecto a otra. Los histogramas permiten representar con facilidad la distribución estadística de una variable, como, por ejemplo, el tiempo de respuesta de una transacción. Cuando se representa la evolución de una variable con respecto a otra, la variable independiente acostumbra a ser el tiempo, y la función que se representa es, con frecuencia, la utilización de los dispositivos, su solape, la frecuencia de aparición de un acontecimiento, etc. También se acostumbra a representar en el mismo gráfico más de una variable función, para poner de manifiesto las concordanacias y oposiciones que aparecen entre ellas en el funcionamiento de un sistema.

En ocasiones es conveniente hacer intervenir en la presentación tanto los aspectos de la carga como los del sistema operativo. Dentro de este último, no sólo hay que ver el consumo de CPU que provoca, sino también el tipo de funciones que la carga requiere de él (llamadas al supervisor, interrupciones, etc.), los módulos transitorios que llama y que carga en memoria, el resultado de la gestión de memoria (paginación, asignación de memoria, etc.), etc. La carga deberá presentarse, a ser posible, desglosada, por lo menos, en batch, conversacional y transaccional. Los dos primeros tipos se definen por el número de trabajos existentes en el sistema y la última por la frecuencia de llegada de transacciones. Conviene, por tanto, poder relacionar los elementos de la carga con las magnitudes que definen el comportamiento.

2.5.2. Tipos de variables

La elección del tipo de gráfica a utilizar es función de la variable que se va a representar. Las variables pueden ser de dos tipos (figura 2.11):

- **Variables cualitativas o categóricas:** Se definen por un conjunto de subclases mutuamente exclusivas que se expresan normalmente con palabras. Pueden estar ordenadas (por ejemplo, el tipo de computador: supercomputador, minicomputador y microcomputador) o no (tipos de carga: científica, de ingeniería o educacional, por ejemplo).
- **Variables cuantitativas:** Sus distintos niveles se expresan numéricamente. Pueden ser de dos tipos:
 - **Discretas:** Pueden adoptar un número finito o infinito de valores, pero en todo caso será numerable (número de procesadores en un sistema multiprocesador, tamaño del bloque de disco, etc.).
 - **Continuas:** Pueden adoptar un número infinito y no contable de valores diferentes (variables reales como el tiempo de respuesta de un trabajo en el sistema).

Para expresar la relación entre dos variables cuantitativas continuas se utilizará una gráfica de líneas, mientras que si la variable del eje de abscisas es cualitativa, o cuantitativa pero discreta, será conveniente la utilización de un gráfico de barras.

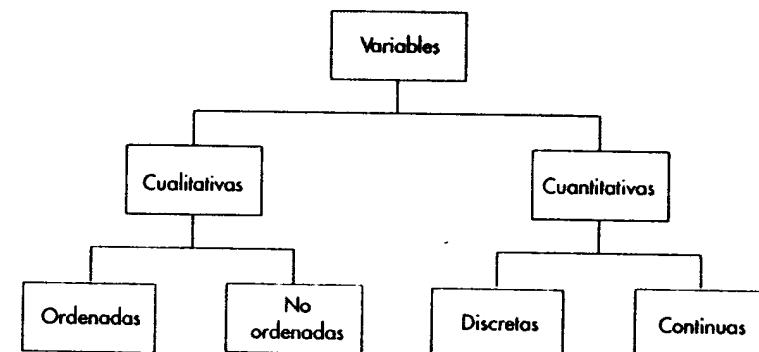


FIGURA 2.11.

2.5.3. Recomendaciones para la construcción de gráficas

Cuando se presentan los resultados en forma gráfica, se recomienda:

- *Minimizar el esfuerzo del lector:* Dadas dos representaciones de los mismos datos, la mejor de ellas será la que requiera un menor esfuerzo para su interpretación por parte del lector.
- *Maximizar la información:* Evitar los símbolos, indicar las unidades, etc.
- *Minimizar el empleo de "tinta":* Evitar la información innecesaria que dificultaría la interpretación del gráfico.
- *Seguir la práctica habitual de construcción de gráficas:* Situar, siempre que sea posible, el origen en el (0,0), procurar que las divisiones en los ejes sean todas iguales, etc.
- *Evitar la ambigüedad:* Identificar cada curva o barra, no representar en un mismo gráfico varias variables distintas, etc.

Hay que tener en cuenta que se trata de recomendaciones y no de reglas. En ocasiones será necesario prescindir de alguna de ellas para lograr un propósito determinado.

En la figura 2.12 se presentan algunos ejemplos.

Si se observan las gráficas de la figura 2.12, se puede comprobar que la primera resulta más sencilla de interpretar por el etiquetado directo de las curvas. Por tanto, siguiendo la primera recomendación, será más conveniente utilizar la primera de ellas porque requiere menor esfuerzo por parte del lector.

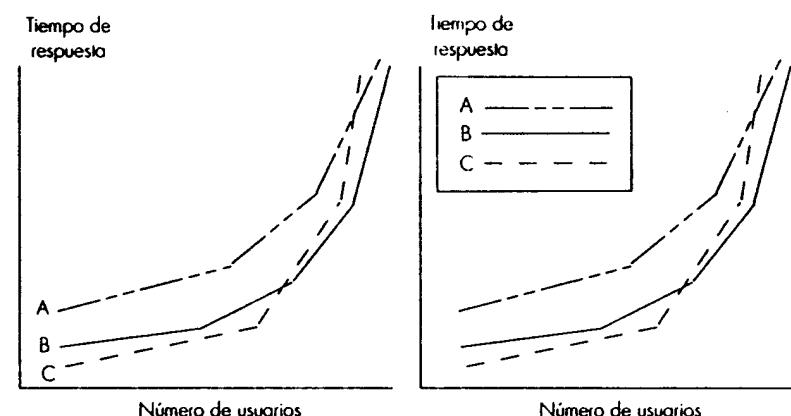


FIGURA 2.12.

En la figura 2.13 se han dibujado demasiadas variables en un mismo gráfico dificultando la interpretación del mismo. Lo conveniente, en este caso, es realizar tres gráficas diferentes obteniendo así una presentación más clara y lógica a no ser que interese efectuar un estudio acerca de la simultaneidad de determinados acontecimientos o circunstancias.

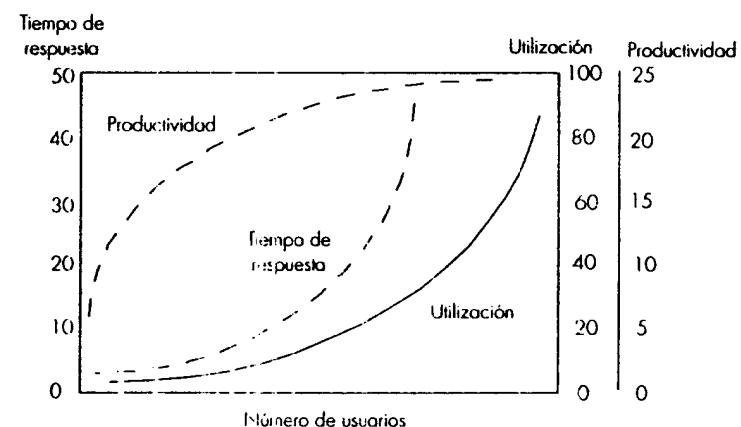


FIGURA 2.13.

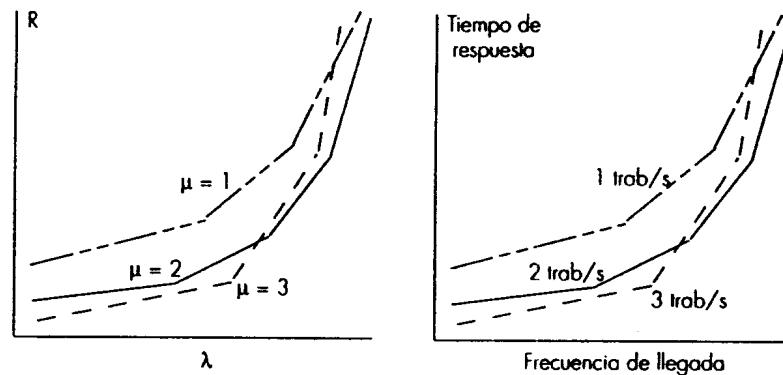


FIGURA 2.14.

En la figura 2.14, el segundo gráfico resulta más sencillo de interpretar porque no utiliza símbolos. Para interpretar el primero de ellos el lector tendrá que repasar el texto que acompaña a la presentación de los gráficos para conocer el significado de los símbolos.

El fallo que presenta el gráfico de la figura 2.15 consiste en que se ha utilizado un gráfico de líneas a pesar de que no se pueden interpolar los valores intermedios. Lo aconsejable hubiese sido utilizar uno de barras.

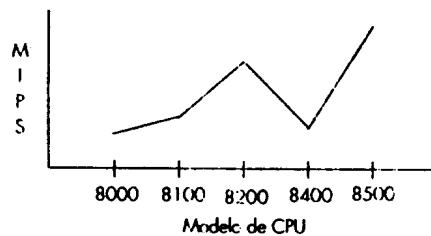


FIGURA 2.15.

2.5.4. Diagramas de Gantt

A la hora de evaluar las prestaciones de un sistema puede resultar interesante asegurar la óptima utilización de los recursos. Si la utilización de un recurso es demasiado alta, éste puede ser la causa de un *cuello de botella* en el sistema, degradándose las prestaciones del mismo. Una utilización muy baja, por el contrario, implica también una falta de eficiencia del sistema por desaprovechamiento de los recursos del mismo.

Para una utilización adecuada de todos los recursos, la carga debe constar de una mezcla de trabajos que hagan uso de ellos y que exista solapamiento en su utilización. Los diagramas de Gantt permiten representar este solapamiento en la utilización de los recursos. Este tipo de diagramas se utiliza generalmente para mostrar la duración relativa de cualquier condición booleana (condición que puede ser verdadera o falsa). Ejemplos de condiciones booleanas podrían ser la utilización de un recurso o el solapamiento en la utilización de dos recursos (la CPU y un canal, por ejemplo).

En sistemas donde se realizan monitorizaciones periódicas, se suelen utilizar los diagramas de Gantt para observar cómo varían las utilizaciones de los recursos a medida que evoluciona la carga en el sistema.

Se presenta, a continuación, un ejemplo que muestra la utilización de tres recursos: la CPU, un canal de E/S y un enlace de red (figura 2.16).

A partir de este diagrama se observa que la CPU estuvo ocupada durante el 60 % del intervalo de observación, el canal durante un 40 % y el enlace durante un 60 %. También se puede deducir el solapamiento en la actividad de los distintos recursos: la CPU y el canal estuvieron ocupados simultáneamente durante un 20 % de la sesión; CPU y red, durante un 40 %; canal y red durante un 15 %, etc. Además, se observa que el sistema estuvo desocupado durante un 5 %.

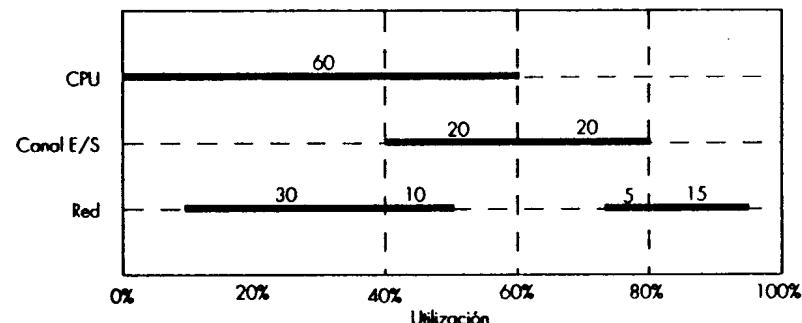


FIGURA 2.16.

En el siguiente ejemplo se verá cómo construir un diagrama de este tipo a partir de los datos de una evaluación.

Ejemplo

Para un sistema con cuatro recursos (CPU, dos canales de E/S y un enlace de red) se han obtenido los siguientes datos de las utilizaciones para los 16 estados posibles. Los recursos, etiquetados como A, B, C y D, se utilizan cuando la entrada en la tabla es 1, y un 0 indicará la no utilización de los mismos.

A	B	C	D	Tiempo utilizado (%)
0	0	0	0	5
0	0	0	1	5
0	0	1	0	0
0	0	1	1	5
0	1	0	0	10
0	1	0	1	5
0	1	1	0	10
0	1	1	1	5
1	0	0	0	10
1	0	0	1	5
1	0	1	0	0
1	0	1	1	5
1	1	0	0	10
1	1	0	1	10
1	1	1	0	5
1	1	1	1	10

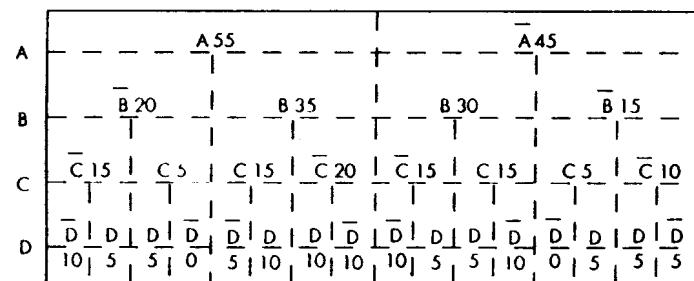


FIGURA 2.17.

Los cuatro recursos están representados mediante cuatro niveles en el diagrama de Gantt (figura 2.17). El primer nivel se divide en dos partes que corresponden a $A = 1$ (recurso activo) y a $A = 0$ (recurso inactivo). Cada uno de estos segmentos se dividen, a su vez, en dos correspondientes a $B = 0$ y $B = 1$, y así sucesivamente.

El diagrama final se dibuja a partir del anterior, escalando los segmentos (figura 2.18).

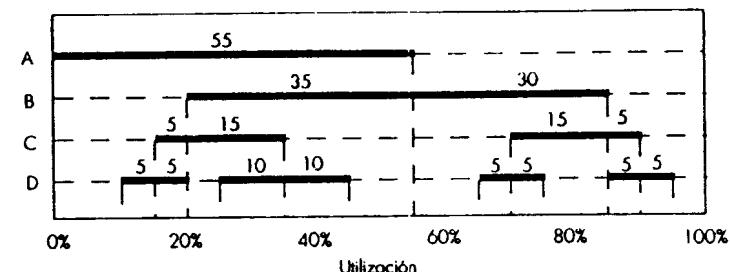


FIGURA 2.18.

2.5.5. Gráficos de Kiviat

Este tipo de gráfico fue introducido por Kolence y Kiviat en 1973 y adquirió gran popularidad para los estudios de prestaciones ya que permite a los administradores de sistema un reconocimiento rápido de los problemas de prestaciones.

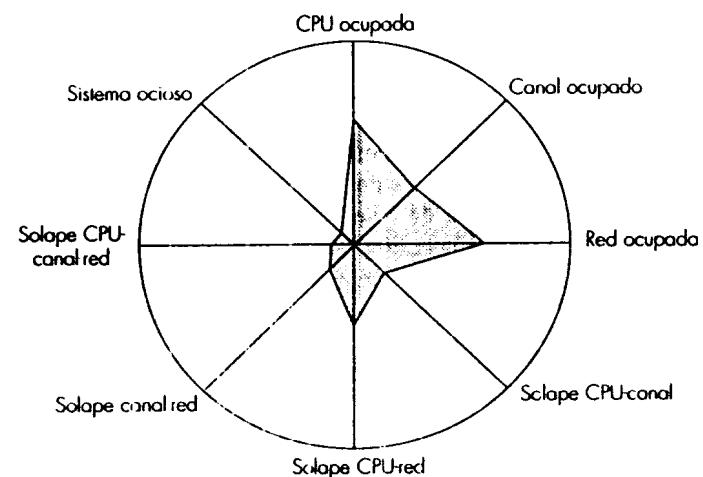


FIGURA 2.19.

Se trata de un gráfico circular en cuyos ejes radiales se representan diferentes índices de prestaciones. Las intersecciones entre los radios y la circunferencia representan los valores máximos que pueden alcanzar las variables representadas en los mismos.

En el ejemplo de la figura 2.19 se representa el gráfico de Kiviat correspondiente al gráfico de Gantt de la figura 2.16.

En este caso, la situación óptima, que correspondería al caso hipotético de que tanto la CPU como el canal y la red estuviesen siempre activos (solapamiento total), se representaría mediante un gráfico de Kiviat como el de la figura 2.20.

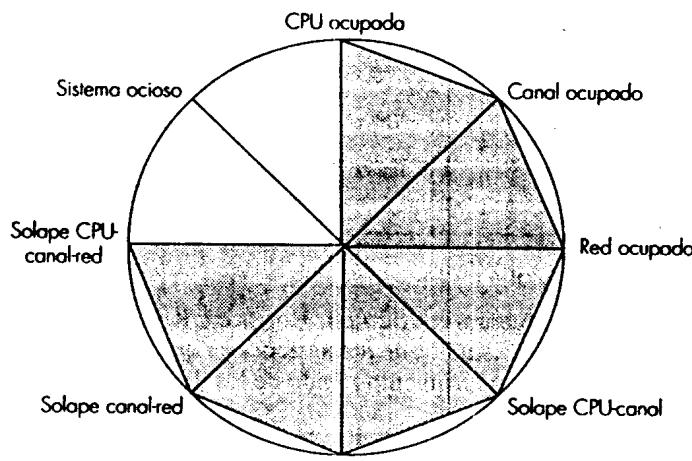


FIGURA 2.20.

Los gráficos de Kiviat son más fáciles de interpretar porque están basados en la capacidad humana de reconocer formas. Para este ejemplo particular (CPU, un canal y una red), dados los gráficos de Kiviat correspondientes a dos situaciones diferentes se puede ver cuál de ellas representaría una mejor situación en cuanto a los índices de prestaciones simplemente observando su mayor o menor semejanza con la figura 2.20.

Aunque, en principio, el número de ejes que puede tener un gráfico de este tipo es arbitrario y depende de los datos que se van a representar, se suelen seguir unos convenios de presentación. El más popular de ellos es el conocido como la *versión de Kent*, basado en las siguientes convenciones:

1. Se selecciona un número par, frecuentemente ocho, de variables que hay que estudiar la mitad de ellas deben ser buenos índices de prestaciones (a mayor valor, mejores prestaciones) y la otra mitad malos (mejores prestaciones a menor valor de los mismos).

2. Se subdivide el círculo en tantos sectores como variables hay que representar.
3. Se numeran los semiejes secuencialmente (preferiblemente en sentido horario) comenzando por el semieje vertical superior.
4. Se asocian los buenos índices de prestaciones a los semiejes impares y los malos a los pares.

En un sistema ideal, todos los índices buenos tendrían valores altos y los malos, valores muy bajos. El gráfico de Kiviat de ese sistema, construido siguiendo estas indicaciones, tendría forma de estrella.

Se considera, por ejemplo, la siguiente asignación de índices de prestaciones:

1. CPU ocupada o activa.
2. Sólo CPU ocupada.
3. Solapamiento de CPU y canal.
4. Sólo canal ocupado sin solape en la CPU.
5. Cualquier canal ocupado.
6. CPU en estado de espera.
7. CPU en estado usuario o atendiendo a programas de usuario.
8. CPU en estado supervisor.

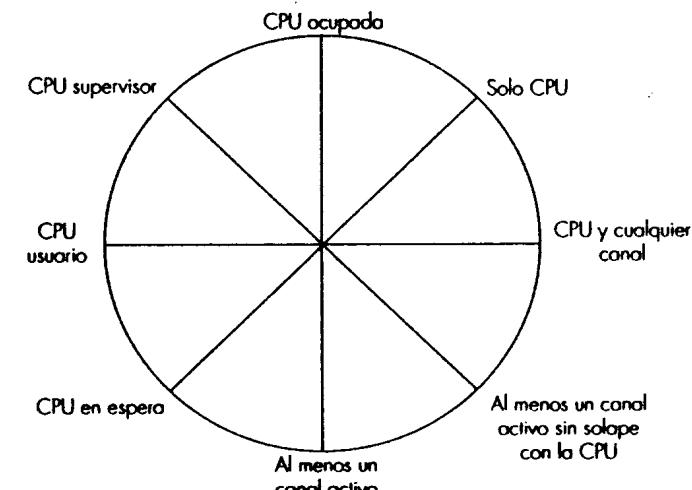


FIGURA 2.21.

El primer índice (CPU ocupada) es bueno en el sentido de que se utiliza el recurso y éste no se desperdicia. Además es preferible que estén simultáneamente ocupados cuantos más recursos mejor para que exista un elevado paralelismo en la utilización de los mismos. Por ello, el segundo índice (sólo CPU ocu-

pada) no es bueno porque indica que no existe solapamiento con los canales siendo, por tanto, preferible un valor bajo para este índice. Por razones similares, el tercer índice (solapamiento CPU-canal) es bueno y el cuarto (sólo canal ocupado) malo. La CPU en estado de espera indica que está, por ejemplo, esperando a la finalización de una operación de E/S y, por tanto, se está malgastando ese recurso (índice malo). Si la CPU se encuentra en estado supervisor, está ejecutando código de sistema operativo. Esto representa precisamente el *overhead* del sistema operativo y se considera un índice malo mientras que se considera bueno que la CPU esté atendiendo a los usuarios.

El gráfico de Kiviat de estos índices siguiendo las convenciones antes enunciadas es el de la figura 2.21 y en la figura 2.22 se muestra el gráfico de Kiviat de una instalación de gestión normal.

El gráfico de Kiviat correspondiente a una situación ideal tendría forma de *estrella de Kiviat* (figura 2.23.A).

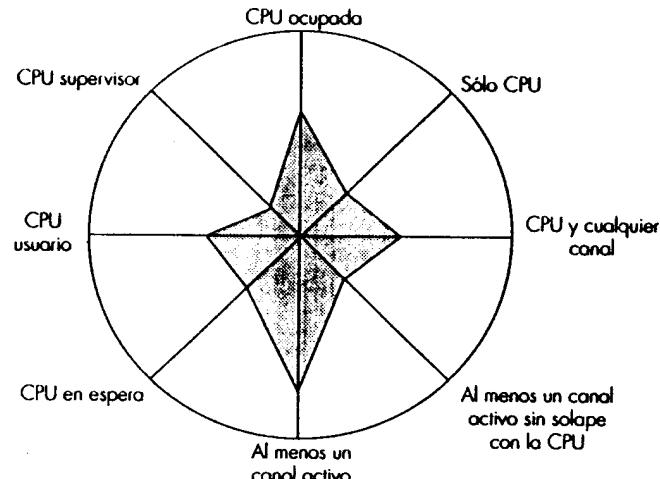


FIGURA 2.22.

Este conjunto de índices es uno de los más utilizados y extendidos desde que se introdujo este tipo de gráficos.

2.5.6. Formas típicas de gráficos de Kiviat

Con el conjunto de índices enunciado en el apartado anterior se han identificado una serie de formas típicas correspondientes a determinadas situaciones.

En la figura 2.23 se observan las siguientes formas: de estrella de kiviat, de vela de barco, de cuña o Iceberg, de flecha de E/S y la de Sistema saturado por paginación

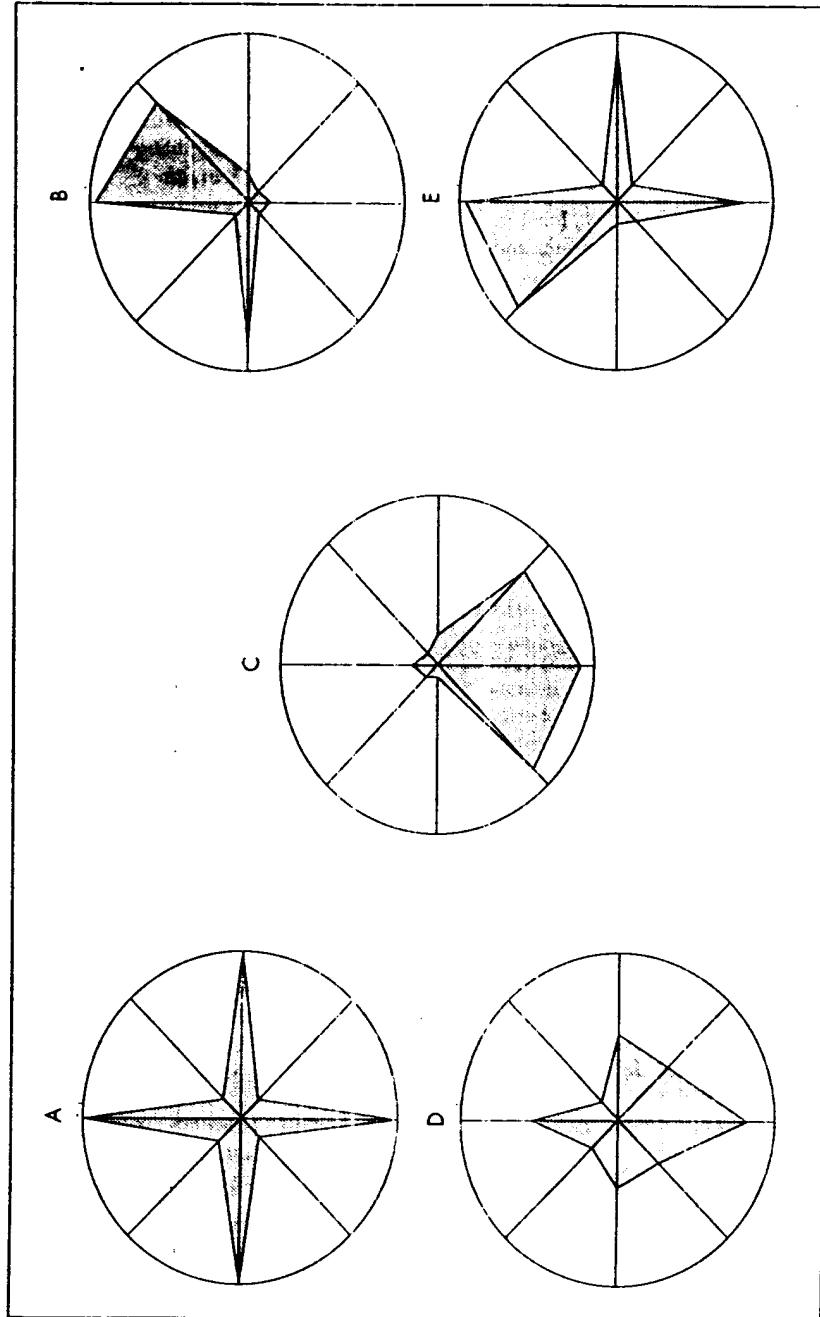


FIGURA 2.23. El gráfico de la figura A, en forma de estrella de kiviat, corresponde a una situación ideal. La forma de vela de barco (figura B) corresponde a sistemas limitados por la CPU (sistemas con mucho demanda de CPU y relativamente poco utilización de los canales). El gráfico de la figura C con forma de cuña o iceberg corresponde a sistemas limitados (y muy dimensionados) por la E/S y bajo de la CPU. La forma denominada 'flecha de E/S' (figura D) es la que corresponde a sistemas limitados por la E/S que además tienen una elevada utilización de la CPU (es decir, bien dimensionados). Finalmente la figura E muestra cuál sería la forma de un sistema saturado por la paginación (thrashing).

Anexo 2.A.

MONITORES DE REDES DE ÁREA LOCAL

2.A.1. Introducción

Cada vez más las redes de área local son el eje bajo el cual se vertebran los sistemas informáticos de muchas organizaciones, industrias, etc.

La administración de estas redes es una responsabilidad que conlleva el buen funcionamiento de éstas de modo continuo y con unas buenas prestaciones.

La monitorización de una red es esencial para administrarse bien ya que proporciona todos los datos necesarios para caracterizar o estudiar sus prestaciones.

La organización OSI define cinco dominios de administración de una red de área local.

1. Manejo de fallos.
2. Administración de cuentas.
3. Manejo de configuraciones.
4. Manejo de las prestaciones.
5. Manejo del control de acceso.

Los sistemas de monitorización de una LAN dan información con mucha aplicación a los puntos 1, 2, 3 y 4.

El punto primero, manejo de fallos, puede resumirse como el esfuerzo para minimizar el tiempo medio requerido para detectar y reparar un problema en la red. Una rápida diagnosis y resolución de los problemas de la red mejora la fiabilidad de la red y puede conducir a la necesidad de una menor redundancia.

El segundo punto, administración de cuentas, se lleva acabo mediante un tipo de monitor conocido como "monitor de contabilidad", que informa de los recursos utilizados por los usuarios del sistema con el fin, entre otras cosas, de proceder a la facturación de estos consumos.

El punto de manejo de prestaciones puede resumirse como el esfuerzo para mejorar las prestaciones de la red y detectar prestaciones degradadas por colección y análisis de estadísticas de la red, y modificación de los parámetros apropiados de la red.

El punto de manejo de configuraciones se puede resumir como el esfuerzo para definir y monitorizar las configuraciones física y lógica de la red.

2.A.2. Características del monitor de una LAN

Los principales puntos a tener en cuenta en el monitor de una LAN son los siguientes:

- *Interferencia:* Corresponde a la interferencia producida en el sistema por la introducción de un dispositivo monitor. Además de minimizar la interferencia, esta magnitud debe ser conocida, para que pueda eliminarse de las medidas. Esto último asegura una indicación no sesgada de las prestaciones.

- *Generación de tráfico artificial:* Algunos monitores tienen la capacidad de generar tráfico con características conocidas, para facilitar la comprobación y la depuración de una red. Los generadores de tráfico artificial pueden ser usados para emular condiciones de altas cargas, producir patrones de tráfico variable y repetitivo, y extraer información temporal. Por ejemplo, un generador de tráfico puede realizarse con tasas de llegadas periódicas y de Poisson, lo cual facilita la comparación de investigaciones. También, el estampar el tiempo en los paquetes sirve para medir la velocidad del tráfico en la red y tiempo de respuesta de los nodos. Si los generadores comunican con un sistema de monitorización, la generación de tráfico y la toma de datos pueden estar sincronizadas.
- *Ánalisis en tiempo real:* Los datos recogidos por el monitor pueden ser analizados inmediatamente (en tiempo real) o almacenados para un análisis posterior. Cuando los paquetes son transmitidos a una velocidad rápida, el tiempo de análisis y de recogida de los datos está limitado lo cual hace difícil el análisis en tiempo real.
- *Parámetros de interés:* La elección de un tipo de monitorización puede ser muy dependiente de los parámetros que interese conocer de la red. Posteriormente, en una tabla, se verán los parámetros más usuales que se obtienen en los distintos tipos de monitores.

2.A.3. Tipos de monitores de LAN

Se pueden considerar en general tres tipos de monitores de LANs:

- Centralizados.
- Distribuidos.
- Híbridos.

2.A.3.1. Monitores centralizados

Es el más usual en las redes de difusión (*broadcast*). A su vez, estos monitores se pueden dividir en dos tipos, monitores de prueba y monitores espía.

El *monitor de prueba* inyecta paquetes a la red a intervalos específicos y puede grabar parámetros de la red para cada paquete inyectado, tales como retraso en la adquisición del canal y número de colisiones. Este es un monitor activo que produce una sobrecarga en la red.

El *monitor espía* es un nodo especial dedicado a la monitorización de la red de modo pasivo, por lo que no introduce ninguna sobrecarga. Este monitor debe ser capaz de procesar los paquetes a medida que los ve pasar, por lo que este monitor debe estar equipado con suficiente potencia de proceso y espacio de almacenamiento.

Los monitores centralizados son simples, baratos y dan toda la información deseada. Con estos monitores no se puede obtener información de un nodo manera de los tiempos de llegada de un paquete a la red.

Un esquema de un monitor centralizado se puede ver en la figura 2.24.

2.A.3.2. Monitores distribuidos

El esquema general de un monitor distribuido se puede ver en la figura 2.25.

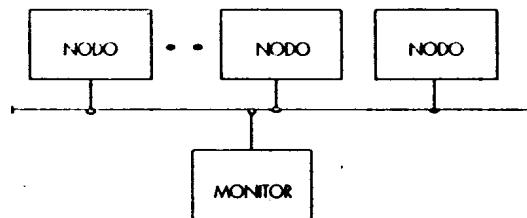


FIGURA 2.24.

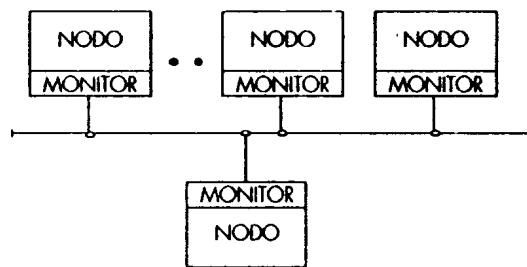


FIGURA 2.25.

En este caso se tiene información de todos los nodos de la red. Cada nodo captura datos y periódicamente transmite esta información al analizador central. El analizador central sólo analiza los datos recibidos desde los monitores distribuidos, puesto que éste no monitoriza la red.

La monitorización en los sistemas distribuidos se puede basar en monitores hardware y software. La monitorización hardware requiere que cada nodo monitorice el tráfico. Este método es más factible en los diseños de redes futuras, puesto que es caro modificar las interfaces de los nodos actuales.

La monitorización software, por otro lado, es más flexible al ser éste modificable. Sin embargo, puesto que los nodos son de tipos diferentes, se necesita diferente software para cada nodo. También hay que tener en cuenta que los monitores software pueden producir una sobrecarga en el procesador del nodo, así como unos mayores requerimientos de memoria.

Los monitores distribuidos, en general, tienen ciertos inconvenientes. Existe una gran sobrecarga de comunicación cuando se envían grandes cantidades de datos a la estación central, a no ser que se utilice un canal dedicado para pasar la información recogida por los nodos. Cada nodo debe de enviar datos sobre la red al analizador central, lo que introduce interferencias y, además, el utilizar la red a medir como parte integral del sistema de monitorización puede causar pérdidas de las funciones de monitorización si la red falla.

2.A.3.3. Monitores híbridos

El método híbrido combina las características esenciales de los métodos centralizado y distribuido. Este método permite medidas precisas y comprensivas mientras necesita

menos modificaciones en las interfaces de los nodos y reduce la cantidad de información que hay que enviar al monitor central. El monitor central directamente observa la red para recoger sus propios datos y también recibe datos desde las interfaces de cada nodo. El monitor central analiza todo; los datos para calcular los parámetros de interés. En la figura 2.26, se puede ver el esquema correspondiente a este tipo de monitor. En este monitor también se introducen interferencias en la red si no se utiliza un canal dedicado. Las interferencias son menores que en el método distribuido puesto que el monitor central recoge localmente alguno de los datos necesarios.

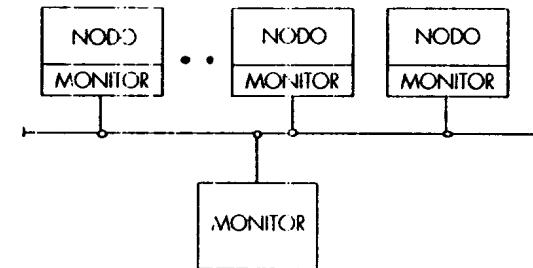


FIGURA 2.26.

2.A.3.4. Comparación de los distintos tipos de monitores

En la siguiente tabla, se comparan los tres tipos de monitores, analizando las características de cada uno.

Característica	Centralizado	Distribuido	Híbrido
Sobrecarga	No	Grande	Media
Tiempo de llegada entre paquetes	Sí	Sí	Sí
Distribución de la longitud de los paquetes	Sí	Sí	Sí
Colisiones en el medio	Sí	Sí	Sí
Colisiones en el nodo	No	Sí	Sí
Número de retransmisiones	No	Sí	Sí
Distribución de los retardos	No	Sí	Sí
Utilización del canal	Sí	Sí	Sí
Utilización de los nodos	No	Sí	Sí
Modificaciones en el hardware	No	Sí	Sí
Modificaciones en el software	No	Sí	Sí
Coste	Pequeño	Grande	Grande

3. Caracterización de la carga

3.1. Introducción

Se denomina *carga* a todas las demandas que realizan los usuarios de un sistema (programas, datos, órdenes, etc.) durante un intervalo de tiempo.

En muchas ocasiones se puede estar interesado en obtener determinados índices de prestaciones de un sistema con el fin de ver el grado de explotación a que está sometido, prever su compra, etc. Para ello es preciso tomar una serie de medidas sobre este sistema. Las herramientas necesarias para ello han sido estudiadas en el capítulo anterior. Ahora bien, para realizar estas mediciones es necesario someter al sistema a una carga determinada.

El resultado de las medidas de prestaciones de un sistema es función y, por tanto, es inseparable de la carga bajo la cual fue determinado. Por ello, cuando se da un índice de prestaciones de un sistema, debe especificarse la carga bajo la que fue obtenido dicho índice. Así, la comparación entre sistemas o configuraciones, sólo tiene sentido cuando las cargas procesadas para su comparación son las mismas.

En caso contrario, las diferencias en los índices de prestaciones reflejarán, además de las diferencias en las prestaciones reales, las diferencias en las cargas, siendo imposible separar ambas diferencias. Un ejemplo son las prestaciones que los fabricantes dan de sus equipos, que normalmente están tomadas bajo cargas que favorecen las características de sus productos y penalizan fuertemente los de sus competidores.

Otro aspecto a tener en cuenta, y que es fuente de problemas, es la variación de la carga en el tiempo. Así, la carga que procesa cualquier sistema está continuamente variando en el tiempo, lo que dificulta su modelización (figura 3.1).

Por último, otro aspecto importante es que la propia carga está interaccionando con el sistema que la procesa, estableciéndose distintas realimentaciones. Por consiguiente, se suelen tener dos realimentaciones principales en un sistema: la interna y la externa (figura 3.2).

12 EVALUACIÓN Y EXPLOTACIÓN DE SISTEMAS INFORMÁTICOS

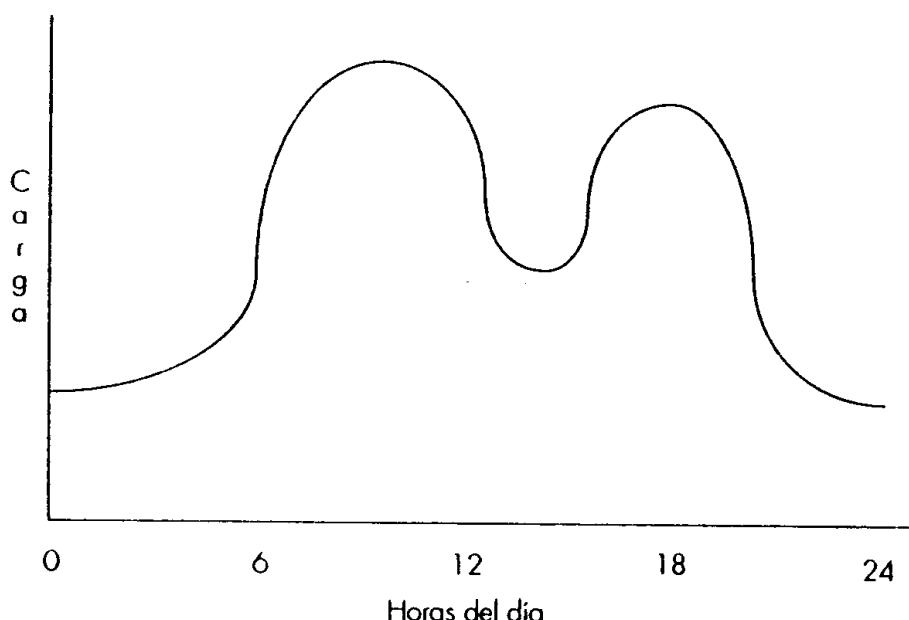


FIGURA 3.1.

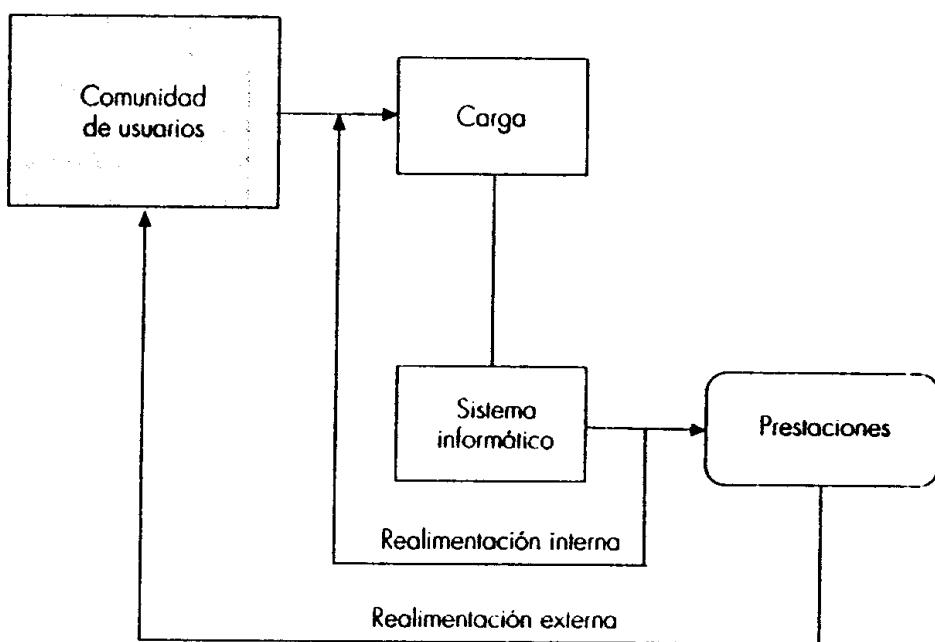


FIGURA 3.2.

El bucle interior de realimentación pone de manifiesto las variaciones que las modificaciones de parámetros del sistema operativo (del asignador dinámico de memoria, de situación de archivos en disco, etc.) inducen en la carga (y más exactamente en la coincidencia de trabajos en el sistema). Por ejemplo, el tiempo de servicio de los discos es función de la carga del sistema, pues depende del tiempo que necesita la unidad de control para resolver los conflictos; de esta forma, el consumo de recursos del sistema informático (tiempo de servicio de los discos, en este caso) puede ser función de la carga.

El bucle exterior de realimentación es el que intenta poner de manifiesto la incidencia del comportamiento del sistema (tiempos de respuesta, principalmente) sobre los hábitos y comportamientos de los usuarios, los cuales, a fin de cuentas, son los que provocan la carga.

En general, se ve, pues, que las relaciones entre el sistema, la carga y las prestaciones pueden ser muy complejas. Las principales causas de esta complejidad son:

- La carga está continuamente fluctuando.
- La carga interacciona con su entorno (el propio sistema, los usuarios, etc.).

Estas consideraciones justifican la necesidad de llegar a definir cuantitativamente la carga de un sistema, es decir, llegar a establecer un modelo de la carga. En general, esta caracterización acostumbra a efectuarse en función de aquellos parámetros que pueden afectar al comportamiento del sistema. Es decir, una carga está correctamente caracterizada si, y sólo si, su resultado es un conjunto de parámetros cuantitativos seleccionados de acuerdo con los objetivos de la operación de caracterización.

La definición de estos objetivos, es decir, del uso del modelo de la carga que se va a caracterizar, es el primer paso de cualquier estudio de evaluación del comportamiento. La especificación de estos objetivos permite determinar los parámetros que se deben usar y su representación.

Una vez establecidos los objetivos, deben determinarse las herramientas que manipularán (modelos analíticos, simulación, el sistema real, etc.) el modelo de la carga. Este hecho permitirá establecer los parámetros necesarios y su forma de representación, que, como se vio en el apartado 1.4., pueden referirse a cada componente de la carga o a su conjunto. Algunos de los parámetros que se utilizan en la caracterización de la carga son los siguientes:

- tiempo total de CPU, tiempo medio de CPU entre dos E/S y sus distribuciones
- número de operaciones de E/S
- características (o tiempo) de operaciones de E/S y sus distribuciones
- espacio en memoria
- líneas impresoras
- ficheros en disco
- cintas leídas
- mezcla de instrucciones

- frecuencia de llegada o tiempo entre llegadas consecutivas y sus distribuciones
- número de usuarios
- tiempo de reflexión y su distribución
- etcétera.

Independientemente de la forma como se represente, *la carga de prueba o de test* (el modelo de la carga) se define como la carga que debe procesar un sistema mientras se está realizando un experimento de medición determinado.

Uno de los principales problemas que puede tener la carga de prueba es que no sea fácilmente reproducible. Por tanto, la carga real es difícilmente reproducible aunque se ejecuten los mismos programas en el mismo sistema.

Por consiguiente, la repetibilidad o reproductibilidad será una característica importante de las cargas de prueba, sobre todo si se están comparando distintos sistemas. Otras veces, la carga de prueba (en estudios predictivos) sólo se conoce hipotéticamente y de forma teórica.

La mayoría de las veces la carga de prueba será un modelo de la carga por los siguientes motivos:

- Satisfacer la exigencia de reproductibilidad de los experimentos, que haga significativas las comparaciones de los mismos índices de comportamiento; estas comparaciones son esenciales en muchos tipos de estudios, como, por ejemplo, cuando se evalúa la eficacia de distintas acciones de sintonización.
- Reducir sustancialmente la duración de cada sesión de medición con respecto a la que se requeriría para ejecutar la carga real total.
- Obtener una representación de la carga consistente con su uso (compatibilidad), ya que muchas veces se están utilizando simuladores o modelos analíticos, donde, evidentemente, no se puede utilizar la carga real.
- Evitar problemas de privacidad y seguridad, que a veces limitan el uso de programas y datos reales en los estudios de evaluación del comportamiento.

Por tanto, cualidades como:

- reproductibilidad
- compacidad
- compatibilidad

son características deseables en la carga de prueba. Estas características en ocasiones sólo se pueden obtener con modelos de carga que se utilizarán como carga de prueba para reemplazar la carga real.

Otras características que debe presentar el modelo de carga son:

- Representatividad, que indica los aspectos de la carga real retenidos en el modelo.

- **Flexibilidad**, que indica si es posible y poco costoso variar el modelo para ajustarlo a las variaciones que se produzcan en la carga real.
- **Independencia del sistema**, importante en los problemas de selección, que consiste en que la representatividad del modelo no varíe al ir cambiando el sistema sobre el que se procesa.

Se llama *sesión de medida* al intervalo de tiempo en que se realiza un experimento. Una sesión de medida no tiene por qué ser continua en el tiempo, sino que puede estar compuesta por varias subsesiones que se realicen en distintos días y a distintas horas.

Ejemplos de ello serían cuando interesaría medir las prestaciones del sistema, cuando su carga sobrepasara cierto nivel (para estudios de saturación), o cuando se produjera la conexión a-determinada red remota (para estudiar la carga que ésta produce).

3.2. Representatividad de un modelo de carga

El apartado anterior ha puesto de manifiesto la necesidad de construir cargas de prueba o modelos de la carga, como resultado del proceso de su caracterización. La precisión del modelo de la carga es claramente una característica esencial para su credibilidad y, por tanto, para su utilidad práctica. Cuando se usa un modelo para representar la carga real de un sistema, su precisión se conoce a menudo como la representatividad del modelo.

La carga puede representarse a distintos niveles, correspondientes a aquéllos en que puede describirse un sistema informático. Es decir, no hay un único criterio para evaluar la representatividad de un modelo; dependiendo del nivel de modelado adoptado y de los objetivos del estudio, se seleccionará una definición adecuada de representatividad y un criterio consistente con su evaluación.

Tres de los más importantes niveles de modelado de un sistema informático y, por lo tanto, de su carga son el físico, el virtual y el funcional. Esto significa que no hay un criterio único para evaluar la representatividad de un modelo, sino que, en función del nivel de modelado adoptado, se tendrán unas características para modelar y unos criterios de representatividad distintos.

3.2.1. Representatividad a nivel físico

A nivel físico, el modelo de la carga se basa en los consumos absolutos o unitarios de los recursos hardware y software. Por ejemplo, cada componente básico de la carga puede caracterizarse por el tiempo de CPU consumido o por el número de instrucciones máquina ejecutadas, por los espacios requeridos de memoria principal y secundaria, por el tiempo total de E/S consumido, por el número de archivos utilizados, por el número y duración de los accesos físicos de E/S a canal y a disco, etc. Este nivel de modelado, que se basa en una caracterización orientada a recursos, depende fuertemente del sistema empleado y, por

consiguiente, puede usarse significativamente en los estudios de comportamiento que no alteren sustancialmente la configuración del sistema existente. En particular, es útil cuando se pretende evaluar la eficiencia de la sintonización de un sistema o determinar la capacidad residual de un sistema.

La mayoría de los parámetros usados en la modelización a este nivel están recopilados por las rutinas de contabilidad (*logging* o *accounting*) presentes en la práctica totalidad de los sistemas operativos. Por la facilidad de acceso a los datos, este tipo de modelado es muy popular.

Como consecuencia de todo ello, se dice que un modelo de la carga W' representa perfectamente la carga W si solicita los mismos recursos físicos en las mismas proporciones que W . Es decir, de acuerdo con este criterio de representatividad, el modelo de la carga deberá usar los mismos recursos (CPU, E/S, etc.) en la misma proporción que en la carga real. Además, el modelo de carga será mejor en cuanto utilice los mismos recursos físicos y en las mismas proporciones, por lo que será muy fácil de construir utilizando un método de proporcionalidad.

Las características de estos modelos son:

- Orientación al consumo de recursos físicos.
- Gran dependencia del sistema.
- Relativamente fáciles de construir (los datos que se necesitan para construir estos modelos son fácilmente obtenibles de los sistemas operativos de los equipos, o pueden obtenerse fácilmente con monitores).

Las mayores aplicaciones de estos modelos son:

- Sintonización del sistema (modificación del consumo de recursos para un mejor funcionamiento del sistema).
- Planificación de la capacidad residual (qué tanto por ciento de la potencia del equipo no está siendo utilizada).
- Diseño.

3.2.2. Representatividad a nivel virtual

En este nivel se consideran los recursos del sistema a nivel lógico. Por ejemplo, cada componente básico de la carga puede caracterizarse por el número de sentencias de cada tipo del lenguaje de alto nivel, el número de accesos lógicos a cada archivo o base de datos, el número y tipo de órdenes interactivas, etc. A causa de este tipo de parámetros, los modelos de este nivel están próximos al punto de vista del programador y son menos dependientes del sistema que los modelos a nivel físico. Estos modelos son adecuados para aquellos estudios en que la modificación de la configuración o del sistema pueden hacer que alguno de los parámetros del anterior nivel de representación pierda significación. Los parámetros de este nivel de representación son bastante más difíciles de obtener que en el tipo de modelo anterior y son necesarios instrumentos especiales para medir muchos de ellos. Este tipo de modelos

acostumbra a utilizarse en estudios de ampliación, de determinación de la capacidad de un sistema, etc.

Como consecuencia de ello, se dice que un modelo de la carga W' representa perfectamente la carga W si solicita los mismos recursos físicos con la misma frecuencia que W . Es decir, de acuerdo con este criterio de representatividad, el modelo de la carga deberá provocar por unidad de tiempo el mismo número de ráfagas de CPU y de E/S de la misma duración que en el sistema real.

Las principales características de estos modelos son:

- Orientación a recursos lógicos.
- Menor dependencia del sistema que en el modelo anterior.
- Mayor proximidad al punto de vista del usuario que en el modelo anterior.
- Mayor dificultad para obtener los parámetros para la construcción del modelo que en el modelo anterior.

Estos modelos son adecuados para:

- Estudios de ampliación, en los que se quiere prever el funcionamiento del sistema después de añadir nuevas unidades.

3.2.3. Representatividad a nivel funcional

En este nivel la carga viene determinada por las aplicaciones que la componen. En el modelo deben aparecer programas que efectúen las mismas funciones que en la carga original. Así pues, las funciones que componen la carga deben describirse de forma independiente del sistema. Los modelos de este tipo se usan en estudios de adquisición, donde deben compararse sistemas de arquitecturas distintas o cuando las diferencias en las configuraciones, en los modelos del tipo anterior, dejan sin sentido alguno de los parámetros.

Como consecuencia de ello, se dice que un modelo de la carga W' representa perfectamente la carga W si realiza las mismas funciones con las mismas proporciones que W .

Por ejemplo, si un determinado sistema realiza 400 horas de compilaciones, 250 horas de pruebas de trabajos, 700 horas de explotación de trabajos y 300 horas de cálculo científico, un modelo funcional de la carga sería el que usara 40 minutos de compilaciones, 25 minutos de pruebas de trabajos, 70 minutos de explotación de trabajos y 30 minutos de cálculo científico.

Las principales características de estos modelos son:

- Orientación a aplicaciones.
- Independencia del sistema.
- Dificultad de diseñar sistemáticamente.

Y las principales aplicaciones de estos modelos son:

- Selección de un computador.
- Planificación de la capacidad.

3.2.4. Representatividad a nivel de comportamiento

Puesto que los modelos de la carga se usan normalmente para la evaluación del comportamiento del sistema, parece evidente usar, para la caracterización sobre la que se basa el modelo, parámetros cuyas relaciones con el comportamiento se conozcan completamente. Los modelos que satisfacen alguno de los criterios de representatividad precedentes pueden estar basados en parámetros cuya relación con los índices de comportamiento sea desconocida.

Si, por ejemplo, se está interesado en el tiempo de respuesta y la productividad de un sistema *batch*, puede no ser conocido, y además difícil de determinar, el impacto de parámetros, tales como el tiempo medio de CPU por programa, el número medio de operaciones de E/S, el número medio de tarjetas leídas, etc., en los valores de los índices del comportamiento considerados.

Además, normalmente se ignoran los aspectos dinámicos de la carga. Por tanto, no se intentan reproducir las mezclas de programas y su evolución en el tiempo. La carga se considera como un conjunto de programas. El criterio virtual permite la evaluación de algunos aspectos que se ignoran totalmente en el criterio físico. Por ejemplo, el cálculo de la ráfaga media de CPU requiere recoger más información dinámica acerca del comportamiento del programa que el del tiempo total de CPU consumido por un programa.

Por consiguiente, un criterio que permite evitar alguno de los inconvenientes anteriores es el de la evaluación del comportamiento, en el que se dice que un modelo W' representa perfectamente la carga W , si produce los mismos valores de los índices de comportamiento que W , cuando se ejecuta en el mismo sistema.

Es decir, se considera como criterio de la precisión del modelo la diferencia que pueda existir entre los valores de los índices de comportamiento que produzcan la carga y su modelo.

El uso de este tipo de criterio permite evaluar la validez del modelo por los efectos que produce en el sistema y no solamente por sus posibles causas. Desafortunadamente este criterio produce modelos dependientes del sistema.

3.2.5. Compacidad del modelo

Esta propiedad de los modelos de la carga tiene en cuenta el factor por el que se reduce su tiempo de ejecución respecto a la carga real. De esta forma, la construcción de modelos de la carga relativamente compactos satisfaciendo los criterios físico o virtual es relativamente directa, ya que, a grosso modo, consiste en reducir proporcionalmente el número de programas que hay que ejecutar en cada uno de los grupos homogéneos de programas que se pueden considerar en la carga real.

La obtención de un modelo compacto que tenga las mismas características funcionales que la carga original ya no es tan simple, puesto que no resulta claro lo que puede significar la reducción proporcional de varias aplicaciones. En algunos casos puede lograrse por reducción del volumen de datos que hay que procesar; en otros casos, debe reducirse el tamaño de los programas; a veces

deben seleccionarse unos datos de entrada elegidos para el caso; o también por una astuta mezcla de los métodos citados. Además, cuando se selecciona un criterio cuantitativo para la representación proporcional, se introducen dependencias no deseadas del sistema, ya que la caracterización funcional es cualitativa por naturaleza y es natural usar tiempos ligados a recursos como medidas de los factores de reducción.

El enfoque orientado al comportamiento puede producir generalmente modelos que satisfagan razonables exigencias de compacidad. Una ventaja de estos modelos es que las mezclas de programas y sus efectos en el comportamiento del sistema se toman en cuenta implícita y automáticamente por el método usado en su construcción. Sin embargo, las mezclas reales presentes en el modelo y su dinámica tienen ciertamente un impacto no despreciable en la precisión del modelo cuando se usa para evaluar un sistema modificado. Es decir, incluso en un modelo orientado al comportamiento, el realismo en la reproducción de las mezclas y de su dinámica es una exigencia importante.

En conclusión, no hay un criterio único y absoluto para construir y evaluar modelos de la carga por lo que el nivel de caracterización y sus parámetros deberán seleccionarse de acuerdo con los objetivos del estudio y los datos disponibles.

No obstante, debe observarse que la elección de un criterio no excluye la toma en consideración de los demás, y es frecuente considerar la construcción de modelos basados en la combinación de varios de ellos, como, por ejemplo, cuando se combina el orientado al comportamiento con alguno de los demás.

3.3. Cargas de test o de prueba

La carga que procesa el sistema durante la sesión de medida, tal y como se ha comentado anteriormente, puede ser o bien la carga real de ese sistema informático o bien un modelo de esa carga.

En la siguiente tabla están resumidos los principales tipos de cargas de test:

Cargas de prueba	Real	
	Sintética	Natural Híbrida
Artificial	Ejecutable	Mix Kernel Programa sintético Secuencias transaccionales Secuencias conversacionales Benchmarks
	No ejecutable	Distribuciones estadísticas Trazas para simulación

3.3.1. Cargas de test reales

La carga de test que consista en la carga que se está procesando realmente en el sistema durante un sesión de medición es, al menos potencialmente, la más representativa y la más barata de implantar. Se puede considerar como un modelo de la carga ya que la duración de su ejecución es normalmente mucho más corta que la de la carga real a la que representa. La única elección que debe realizar el encargado de modelar la carga es decidir qué porción de tiempo de ejecución de la carga real es el que debe usarse en los experimentos de medición.

La duración de una sesión de medida es función de los objetivos del estudio, de la naturaleza de las aplicaciones, etc. En algunos casos, los objetivos del estudio imponen ciertas limitaciones en cuanto al instante de comienzo y a la duración. Por ejemplo, si se quiere estudiar el sistema cuando está muy cargado, cuando se produce la conexión a una red remota, etc.

Si los objetivos del estudio no imponen determinadas elecciones, la duración de la sesión debe elegirse utilizando técnicas estadísticas. En el caso de que las variables medidas (es decir, las observaciones) sean estadísticamente independientes, se puede determinar el mínimo número de observaciones (el tamaño de la muestra) que garantiza que las estimaciones están afectadas de un error máximo dado (el intervalo de confianza) con una probabilidad mínima dada (el nivel de confianza). Se trata, pues, del clásico problema de determinar el mínimo tamaño de la muestra que corresponde a una precisión dada de los resultados.

La suposición de un prerrequisito, del que todavía no se ha hecho mención, es que la secuencia de observaciones de todas las variables es estacionaria; es decir, su distribución estadística no varía a lo largo del tiempo. Esta suposición es difícil de comprobar y demostrar, aunque en general se admite como válida, a falta de mejor hipótesis.

Una de las principales dificultades de usar cargas de test reales es la de reproducir situaciones cuando se quiere repetir un experimento, debido a los imponderables (especialmente humanos) que influyen en la dinámica del sistema. Las principales razones que limitan el uso de cargas de test reales en experimentos que deban repetirse en las mismas condiciones son, pues:

- *Falta de flexibilidad*, debido a la imposibilidad de modificar los programas y sus consumos de recursos.
- *Necesidad de reutilizar los datos originales* (archivos, bases de datos, etc.) cuando se vuelven a ejecutar los programas reales; todos estos datos deberán, por consiguiente, copiarse en una memoria secundaria para su restauración posterior, con los costes económicos e interferencia subsiguientes.
- *Confidencialidad de ciertos programas y datos*, que pueden llegar a prohibir su duplicación y forzar su sustitución por programas y datos que tengan similares características de comportamiento.
- *Diferencias en la organización del hardware y el software* de los distintos sistemas o de las distintas versiones del mismo sistema que hacen uso de la carga de test real, especialmente en los estudios de selección.

4. Introducción a las técnicas analíticas: Análisis operacional

4.1. Introducción

En los sistemas de computadores, muchos trabajos tienen que compartir los mismos recursos (la CPU, los discos,...). Como generalmente sólo un trabajo puede utilizar un recurso en un instante de tiempo dado, todos los demás trabajos que deseen utilizarlo tendrán que esperar en cola.

La *teoría de colas* permite determinar el tiempo que un trabajo pasa esperando en las distintas colas del sistema. Combinando estos tiempos se podrá predecir el tiempo de respuesta, que básicamente no es más que el tiempo total que un trabajo pasa en el sistema y que incluye el tiempo durante el cual el trabajo es ejecutado (tiempo de servicio) más el tiempo que el trabajo espera en las colas para recibir los servicios solicitados.

Por todo ello, los modelos de colas han alcanzado una gran difusión entre los analistas de prestaciones de sistemas informáticos. Una red de colas es un conjunto de estaciones de servicio y de clientes. Las estaciones de servicio representan los recursos del sistema y los clientes son los usuarios, es decir, transacciones o trabajos sometidos al sistema por aquéllos.

Una estación de servicio consta de un servidor más una cola de espera asociada, estableciéndose la siguiente correspondencia entre los modelos y los sistemas reales:

$$\begin{aligned} \text{Servidor (modelo)} &\Leftrightarrow \text{Recurso del sistema (hardware)} \\ \text{Cola (modelo)} &\Leftrightarrow \text{Cola (software) asociada al recurso} \end{aligned}$$

La obtención de los índices de prestaciones a partir de un modelo de colas se denomina *resolución* o *evaluación analítica* y consiste en la resolución de un conjunto de ecuaciones que se deducen a partir del modelo y de sus parámetros. Estos sistemas de ecuaciones son complejos pero se han desarrollado algoritmos muy eficientes para ello.

El objetivo del análisis operacional es llegar a establecer relaciones entre las variables que caracterizan la carga y las que miden el comportamiento. Estas relaciones deducidas mediante el análisis operacional tienen por objeto llegar a las mismas relaciones que las deducidas a partir de la teoría de colas, pero partiendo de hipótesis operacionales, es decir, comprobables experimentalmente por medida. El análisis operacional prescinde de hipótesis de difícil comprobación usadas en teoría de colas, tales como la aleatoriedad, la estacionaridad o la ergodicidad necesarias en la teoría de redes de colas para llegar a lo que se conoce como modelo estocástico.

El modelo operacional parte de hipótesis cuya veracidad puede establecerse sin duda por medida, es decir, que sean comprobables operacionalmente. Además es importante tener presente que las magnitudes que caracterizan el comportamiento de un sistema y que permiten su control están relacionadas entre sí. El análisis operacional permite poner de manifiesto de forma simple algunas relaciones fundamentales entre dichas magnitudes.

El término operacional implica que el sistema es directamente medible. Por tanto, una suposición o hipótesis operacional será aquella que puede ser comprobada o verificada mediante la medida. Por ejemplo, la suposición según la cual se producen igual número de llegadas que de salidas en un sistema determinado puede ser fácilmente comprobada. Por tanto, esta suposición, que se conoce generalmente con el nombre de suposición de equilibrio de flujo, es una suposición operacional.

En este tema se verán las variables operacionales, que son aquellas que pueden ser medidas directamente durante un intervalo de observación finito, y las leyes operacionales, que son las relaciones que mantienen estas variables entre sí.

4.2. Estación de servicio: Variables operacionales

La figura 4.1 muestra el esquema de una estación de servicio.

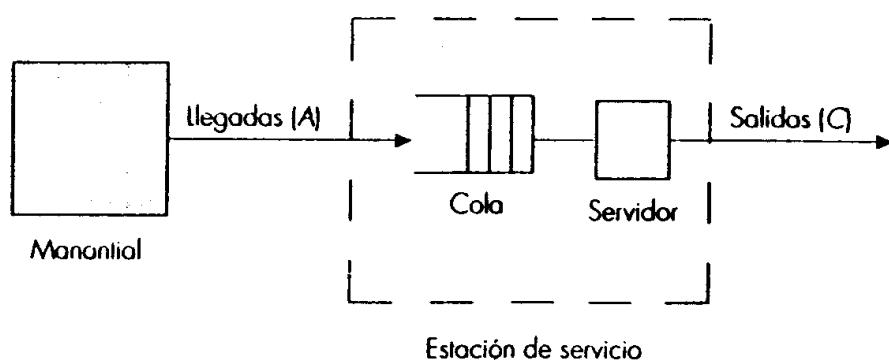


FIGURA 4.1.

Cuando un cliente llega a una estación de servicio y el servidor ya está ocupado por otro cliente, tendrá que esperar en cola hasta que el servidor quede libre. Una vez recibido servicio, abandona la estación.

El conjunto estación de servicio más clientes que llegan constituye la versión más simple de un modelo de red de colas. Este modelo tendrá dos parámetros:

- La intensidad de carga, que es la tasa de llegada de los clientes (por ejemplo, que llegue un cliente cada 2 segundos, o que lleguen 0.5 clientes/s).
- La demanda de servicio, que es el tiempo de servicio que, por término medio, requiere cada cliente (por ejemplo, 1.25 s).

A partir de estos parámetros es posible evaluar el sistema a partir del modelo, obteniendo los índices de prestaciones ya definidos, tales como:

- *Utilización*: proporción de tiempo que el servidor está ocupado.
- *Tiempo de residencia o de respuesta*: tiempo que, por término medio, un cliente permanece en la estación. Engloba tiempo de espera en cola y tiempo de servicio.
- *Longitud de cola*: número medio de clientes en la estación, tanto en espera como en servicio.
- *Productividad o throughput*: tasa de salida de los clientes de la estación de servicio.

Para los parámetros del ejemplo, 0.5 clientes/s y una demanda de servicio de 1.25 s, se obtendría:

- Utilización = 0.625
- Tiempo de residencia o de respuesta = 3.33 s
- Longitud de cola = 1.67 clientes
- Productividad o throughput = 0.5 clientes/s

En los gráficos de la figura 4.2, la intensidad de carga o la tasa de llegadas de los clientes varía de 0 a 0.8 llegadas/s, ya que, por una parte no tiene sentido una tasa de llegada menor que 0 y, por otra, si la demanda media de servicio es de 1.25 s, la tasa máxima a la que la estación puede atender a los clientes es de uno cada 1.25 s ($1/1.15 = 0.8$). Con una tasa de llegada mayor se dice que la estación está saturada.

Las gráficas obtenidas responden a los resultados esperados. Por ejemplo, en el caso del tiempo de residencia, es lógico pensar que cuando la intensidad de carga es baja, un cliente que llegue difícilmente encontrará competencia por el servidor, por lo que podrá pasar inmediatamente al mismo y obtener un tiempo de residencia próximo a su demanda de servicio. A medida que aumenta la intensidad de carga, se incrementará la congestión y la competencia por el servidor, aumentando el tiempo de espera y, por tanto, el tiempo medio de residencia. Inicialmente este aumento será gradual, para ser cada vez mayor a medida que la tasa de llegada se aproxima a la que satura la estación de servicio.

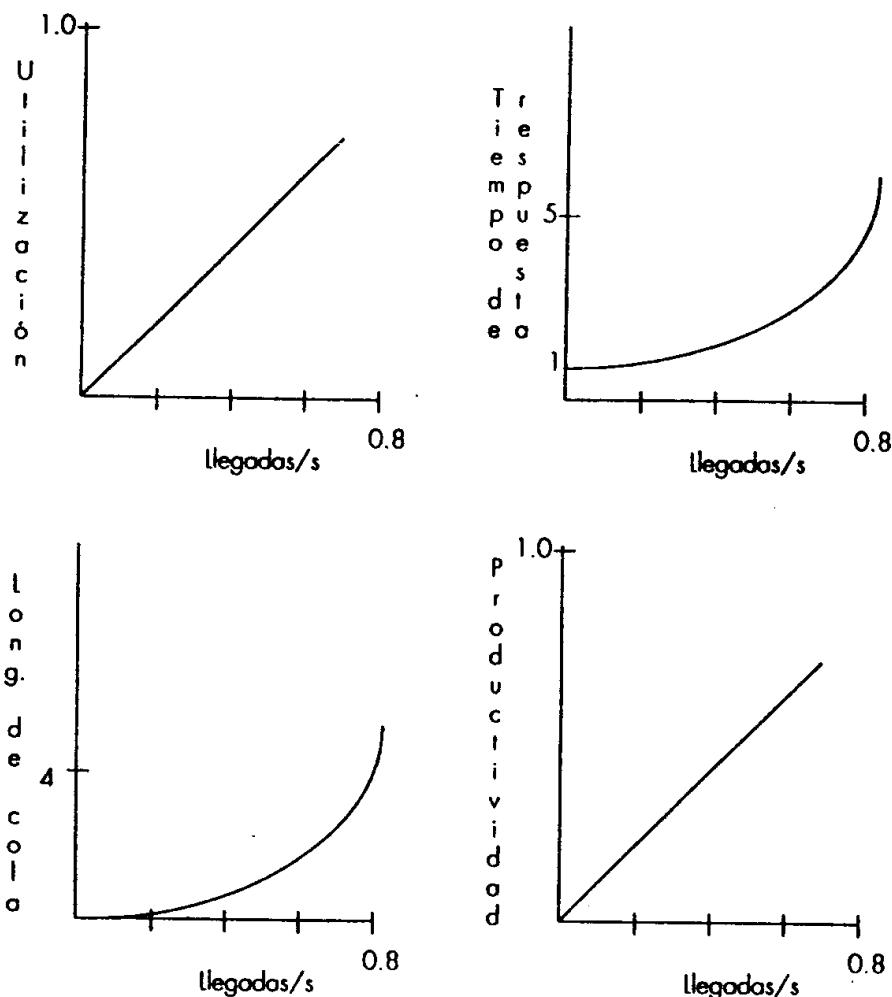


FIGURA 4.2.

4.2.1. Variables operacionales en una estación de servicio

a) Variables operacionales básicas

Las variables operacionales básicas son las que se pueden medir directamente sobre el sistema durante un intervalo de observación finito. Son las siguientes:

- T , intervalo de observación o de medida del sistema.
- Λ (*Arrivals*), número de peticiones o llegadas de clientes durante el intervalo T .
- C (*Completions*), número de peticiones completadas o servidas durante el intervalo T .
- B (*Busy*), tiempo durante el cual el recurso observado (la estación de servicio) ha estado ocupado.

b) Variables operacionales deducidas

Las variables operacionales deducidas se obtienen a partir de las básicas y son:

- λ , tasa de llegada, que mide las llegadas al sistema o a la estación de los clientes o peticiones por unidad de tiempo.
Si se producen 8 llegadas durante un intervalo de observación de $T = 4$ minutos, entonces la tasa de llegada es de $8/4=2$ peticiones/minuto.
- X , productividad, que es el número de peticiones completadas por unidad de tiempo.

$$X = \frac{C}{T}$$

Si se observan 8 terminaciones durante $T = 4$ minutos, entonces la productividad es de $8/4 = 2$ peticiones/minuto.

- U , utilización, que mide la proporción de tiempo durante la cual el servidor ha estado ocupado.

$$U = \frac{B}{T}$$

Si se observa que el recurso está ocupado 2 minutos durante el intervalo $T = 4$ minutos, entonces la utilización es de $2/4 = 0.5$, es decir del 50 por 100.

- S , tiempo medio de servicio, que mide el tiempo durante el cual un cliente ocupa el servidor.

$$S = \frac{B}{C}$$

Si se observan 8 terminaciones durante el intervalo $T = 4$ minutos y la estación estuvo ocupada durante 2 minutos, entonces el tiempo medio de servicio que requiere cada petición es de $2/8 = 0.25$ minutos.

c) Leyes y teoremas operacionales

Se puede comprobar, por tanto, que las magnitudes deducidas satisfacen la ecuación

$$U = XS$$

que constituye la ley operacional de la utilización.

Se supone, que el número de llegadas es igual al número de salidas durante el periodo de observación (la realidad dice que si el sistema es capaz de atender a los clientes que llegan, esas dos cantidades deberán ser iguales o casi, si el periodo de observación es suficientemente largo, pues, de lo contrario, el sistema iría acumulando clientes hasta reventar. En cualquier caso, esta hipótesis se puede comprobar experimentalmente), es decir,

$$A = C$$

Esta suposición determina el equilibrio del flujo de trabajos e implica que

$$\lambda = X$$

y, en consecuencia, que

$$U = \lambda S$$

que constituye un ejemplo de teorema operacional.

4.3. Redes de colas

Para representar o modelar un sistema o subsistema informático se suele utilizar un conjunto de estaciones de servicio relacionadas entre sí, constituyendo lo que se denomina una *red de colas*.

Las estaciones de servicio se corresponderán normalmente a los recursos del hardware y sus colas software en los sistemas informáticos. Los clientes serán los usuarios o las transacciones que llegan al sistema.

Puesto que el objetivo de este estudio es poner de manifiesto la existencia de relaciones entre las distintas variables que miden las prestaciones y que describen la carga, se trabajará con valores medios solamente, o, lo que es equivalente, se considerará que todos los trabajos son del mismo tipo, es decir, solicitan los mismos recursos el mismo número de veces por trabajo y cada vez solicitan el mismo servicio del recurso.

Una hipótesis frecuente en el estudio de los sistemas informáticos mediante análisis operacional es que en un instante dado, un cliente puede estar presente únicamente en una estación de servicio (ya sea en la cola o en el servidor).

4.3.1. Tipos de redes

a) Redes cerradas

No tienen ni llegadas ni salidas externas. Por lo tanto, los trabajos permanecerán en el sistema circulando de una estación a otra, es decir, toda petición de entrada a una estación será a su vez salida de otra. En este tipo de redes el número de trabajos o clientes en el sistema permanece constante.

En el estudio de este tipo de redes se parte del número de trabajos (N) y se trata de determinar la productividad (o la tasa a la cual se completan los trabajos).

Este tipo de redes se utiliza para modelar sistemas *batch* y sistemas interactivos. En un sistema *batch* se puede considerar que, cuando termina un trabajo, es inmediatamente reemplazado por otro que estaba en espera para ser lanzado. Por tanto, para especificar la intensidad de carga en un sistema *batch* bastará con indicar el número medio de trabajos activos (N) o, lo que es lo mismo, el factor de multiprogramación (figura 4.3).

En los sistemas interactivos, para especificar la intensidad de carga se tendrán que especificar dos parámetros:

- N , que es el número de terminales activos (clientes).
- Z , que es el tiempo de reflexión, es decir, tiempo que, por término medio, los usuarios utilizan los terminales entre dos interacciones consecutivas.

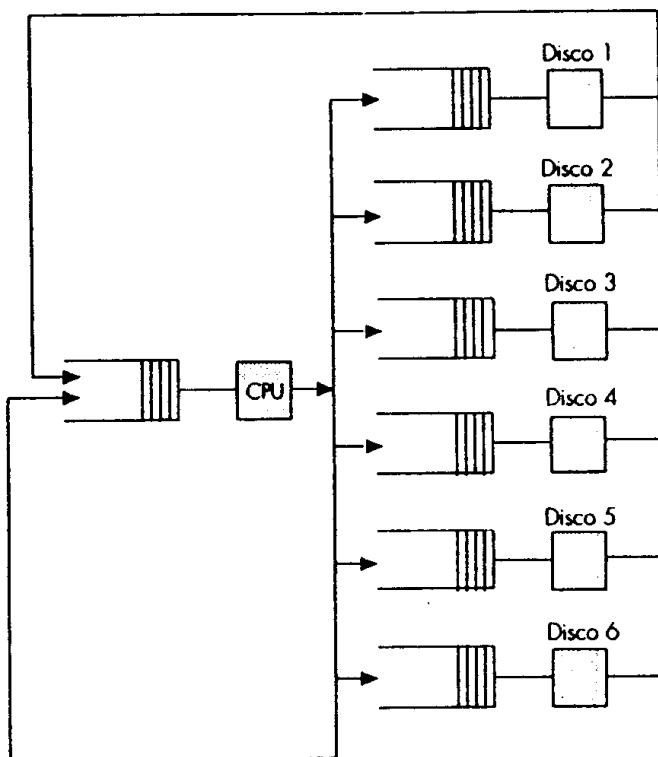


FIGURA 4.3.

Se puede decir que cada terminal está manejado por un usuario que alterna períodos de reflexión y de espera. En un período de reflexión, el usuario contempla el resultado de la interacción anterior y prepara la siguiente y, por tanto, el subsistema central (que se considera formado por la CPU y las unidades de E/S) no efectúa ningún trabajo para él. Cuando transmite una petición, el usuario entra en un período de espera en el que permanece mientras el subsistema central efectúa el trabajo solicitado (figura 4.4).

Para modelar los terminales, se introduce un tipo de estación de servicio que es la estación de retardo.

Los clientes en una estación de cola compiten por el servidor. Por ello, el tiempo de residencia estará compuesto por un posible tiempo de espera y un

tiempo de servicio (figura 4.5). Se utilizan para representar recursos del sistema en los cuales los clientes compiten por el servicio (CPU, dispositivos de E/S,...).

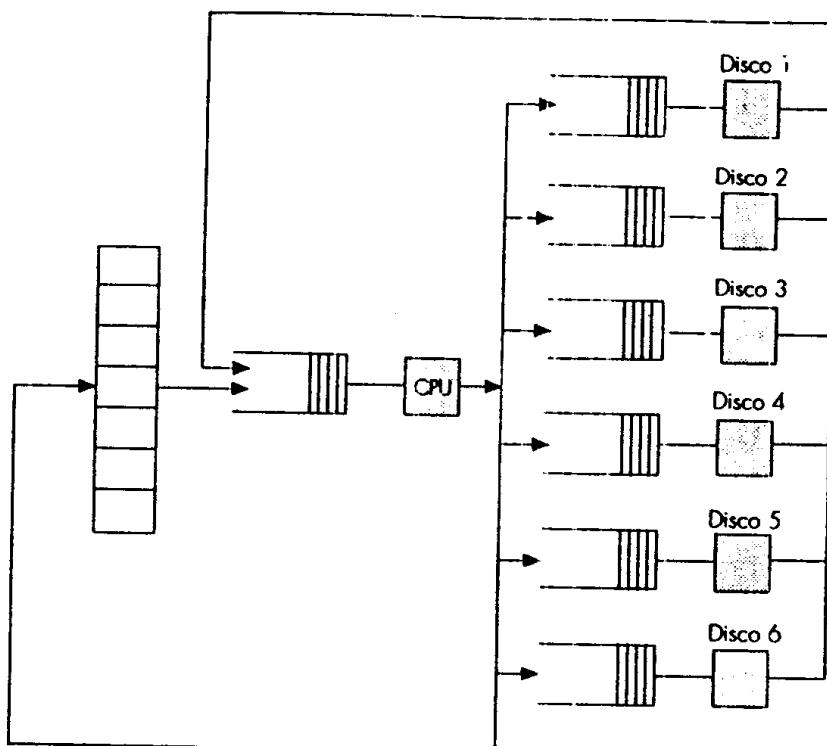


FIGURA 4.4.

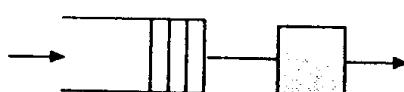


FIGURA 4.5.

En una estación de retardo no hay competencia por el servicio (figura 4.6) ya que siempre que llega un cliente hay un servidor disponible. El tiempo de residencia es equivalente a la demanda de servicio del cliente. Se utilizan, sobre todo, para representar tiempos de reflexión en cargas de tipo interactivo. Son útiles siempre que haya que introducir un tiempo de retardo de media conocida.

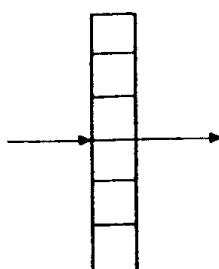


FIGURA 4.6.

b) Redes abiertas

Tienen llegadas y salidas externas. En consecuencia, el número de trabajos en el sistema varía a lo largo del tiempo.

Para el análisis de los sistemas representados por este tipo de redes, se parte de una productividad conocida, a partir de la cual se trata de obtener los índices de prestaciones tales como el tiempo de respuesta, entre otros.

Este tipo de redes se utiliza, por ejemplo, para modelar sistemas transaccionales (figura 4.7).

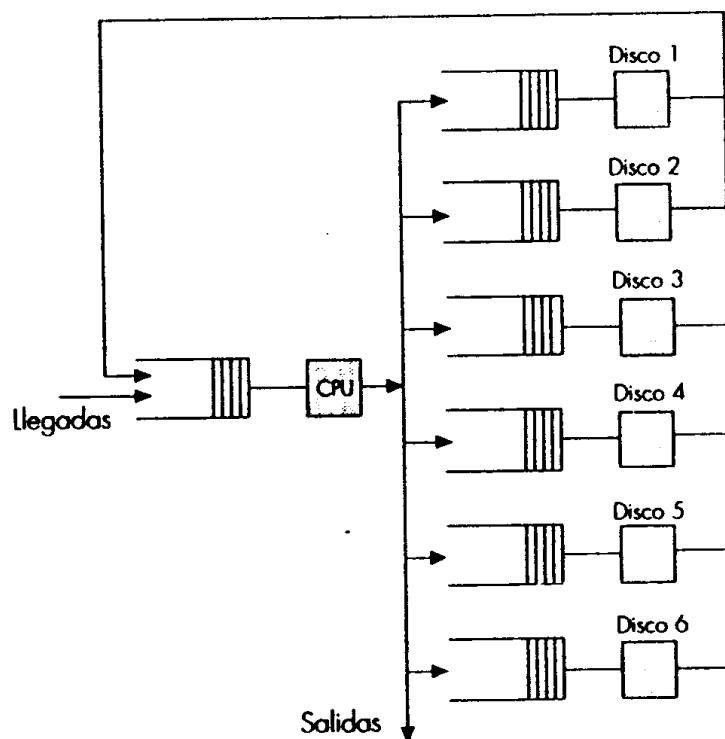


FIGURA 4.7.

Para especificar la intensidad de carga se utiliza la tasa de llegada λ de los clientes (transacciones) al sistema. El número de clientes en el sistema transaccional varía con el tiempo. Los clientes que completan su servicio abandonan el sistema.

c) Redes mixtas

Se trata de modelos abiertos para unos tipos de cargas y cerrados para otros. En el sistema del ejemplo de la figura 4.8 se muestran dos tipos de cargas o dos clases de clientes o trabajos. El sistema es cerrado para trabajos interactivos y abierto para trabajos de tipo transaccional.

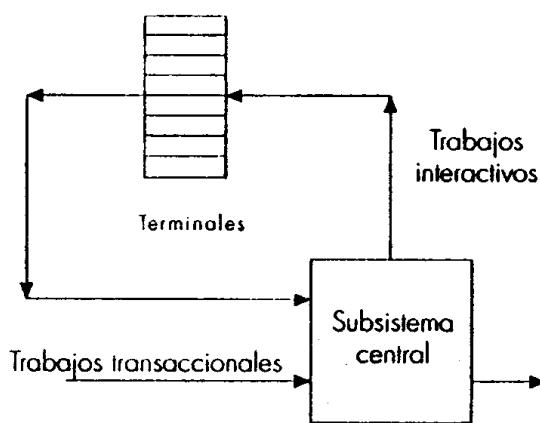


FIGURA 4.8.

d) Resumen

- Las cargas interactivas y *batch* tienen una población total fija (número de clientes en el sistema). Una carga interactiva con un tiempo de reflexión $Z = 0$ será totalmente equivalente a una carga *batch*.
- Tanto en los sistemas interactivos como en los transaccionales, la población en el subsistema central (sistema sin terminales) varía, siempre que en el sistema interactivo se cumpla $Z \neq 0$. El número total de clientes en el subsistema central de un sistema interactivo estará limitado por N , el número de terminales activos. En los sistemas transaccionales no existe este límite superior.

4.3.2. Medidas operacionales en redes

a) Variables operacionales básicas

Se supone una red de colas de K dispositivos o estaciones de servicio. Para un período de observación T se obtienen las siguientes medidas para cada estación i , para $i = 1, 2, \dots, K$:

- A_i , número de llegadas a la estación i ,
- B_i , tiempo de ocupación de la estación i ,
- C_{ij} , número de veces que un trabajo requiere servicio del dispositivo o estación j inmediatamente después de terminar el servicio en el dispositivo i .

Si se considera el mundo exterior como dispositivo o estación 0, se puede definir también:

- A_{0j} , número de trabajos cuya primera petición de servicio se produce en la estación j .
- C_{ij} , número de trabajos cuya última petición de servicio se produce en la estación i .

Se supone $C_{00} = 0$. Por el contrario, C_{ii} puede ser mayor que 0 para $i \neq 0$. El número de trabajos terminados en un dispositivo o estación será:

$$C_i = \sum_{j=1}^K C_{ij}$$

donde, $i = 1, 2, \dots, K$.

Los números de llegadas y salidas del sistema serán, respectivamente:

$$A_0 = \sum_{j=1}^K A_{0j}$$

$$C_0 = \sum_{i=1}^K C_{i0}$$

Si el sistema es cerrado se cumplirá que $A_0 = C_0$.

b) Variables operacionales deducidas

- $U_i = B_i/T$, utilización del dispositivo i .
- $S_i = B_i/C_i$, tiempo medio de servicio del dispositivo i .
- $X_i = C_i/T$, productividad o tasa de salida del dispositivo i .
- $q_{ij} = C_{ij}/C_i$, si $i = 1, 2, \dots, K$, o
- $q_{ij} = A_{0j}/A_0$, si $i = 0$, frecuencia de encaminamiento, es decir la fracción de trabajos que se dirigen al dispositivo j al terminar el servicio en el dispositivo i .

'Se observa que

$$\sum_{j=0}^K q_{ij} = 1$$

y que q_{ij} es la frecuencia de encaminamiento hacia la salida desde la estación i y que q_{ji} es la de encaminamiento de entrada a la estación j .

4.4. Leyes operacionales

4.4.1. Ley del flujo de salida

Puesto que la frecuencia de salida o productividad del sistema es

$$X_0 = C_0/T$$

es fácil deducir la ley operacional

$$X_0 = \sum_{i=1}^K X_i q_{i0}$$

4.4.2. Ley de la utilización

Dado el número de terminaciones C_i y el tiempo de ocupación B_i de una estación o dispositivo i durante un período de observación T , se puede deducir fácilmente la siguiente relación algebraica entre las mismas:

$$\frac{B_i}{T} = \frac{C_i}{T} \cdot \frac{B_i}{C_i}$$

de donde:

$$U_i = X_i S_i$$

que constituye la ley de la utilización.

La utilización de un recurso es igual al producto de la productividad de ese recurso por su tiempo medio de servicio.

Ejemplo: Se supone un disco que sirve 40 accesos/s, y cada acceso requiere un servicio de 0.0225 s.

Según esta ley, la utilización del disco será de $40 \times 0.0225 = 90\%$.

4.4.3. Ley del equilibrio del flujo de trabajos

De nuevo se hace esta hipótesis de flujo de trabajos equilibrados, aceptable para períodos de observación suficientemente largos en sistemas capaces de atender todo el trabajo que llega a ellos. En general, pues, se admite que $A_i - C_i$ es pequeño frente a C_i y sólo será exactamente cero cuando $n_i(0) = n_i(T)$, es decir, eligiendo como punto final del período de observación un instante en que cada servidor se halle en el mismo estado que en el instante inicial (que es la misma idea que subyace en el método de los puntos de regeneración en la realización de simulaciones).

Expresando en forma de ecuación este principio, se puede escribir que para cada estación todo lo que entra en ella debe salir, es decir,

$$C_j = A_j = \sum_{i=0}^K C_{ij}, \text{ para } j = 0, 1, \dots, K$$

Teniendo en cuenta la definición de $q_{ij} = C_{ij}/C_i$

$$C_j = \sum_{i=0}^K C_i q_{ij}$$

y empleando la definición de $X_i = C_i/T$, se obtiene

$$X_j = \sum_{i=0}^K X_i q_{ij}, \text{ para } j = 0, 1, \dots, K$$

que son las ecuaciones que describen el equilibrio del flujo de trabajos.

Si la red es abierta, el valor de X_0 se especifica externamente y las ecuaciones tienen una solución única para las incógnitas X_i . Sin embargo, si la red es cerrada, X_0 es inexistente y las ecuaciones no tienen una solución única, puesto que puede comprobarse que hay K ecuaciones independientes con $K + I$ incógnitas, es decir, se está frente a un sistema homogéneo, puesto que el segundo miembro del mismo es idénticamente nulo. A pesar de ello, cualquiera de las soluciones que se obtenga (todas ellas son proporcionales entre sí) contiene información de considerable valor para el estudio de los métodos analíticos de resolución de redes de colas. Estas soluciones describen todos los posibles flujos equilibrados entre las distintas estaciones; la elección del que describe el estado del sistema dependerá del número total de trabajos en el sistema.

4.4.4. Ley de Little

Como ya se ha dicho, n_i indica el número de trabajos en el dispositivo i que varía a lo largo del período de observación y, para ponerlo de manifiesto, se representará por $n_i(t)$. La figura 4.9 muestra un ejemplo de tal evolución. Si se representa por W_i el área comprendida por la función $n_i(t)$, se puede escribir que el número medio de trabajos en el dispositivo i , N_i , será

$$N_i = W_i/T$$

El tiempo medio de respuesta del dispositivo i , R_i , está relacionado también con la magnitud W_i , que representa la permanencia total en el dispositivo i de los trabajos que han pasado por él. Si se considera R_i como el tiempo medio de presencia de los trabajos completados, se tendrá

$$R_i = W_i/C_i$$

Una consecuencia inmediata de las ecuaciones anteriores es la ley operacional

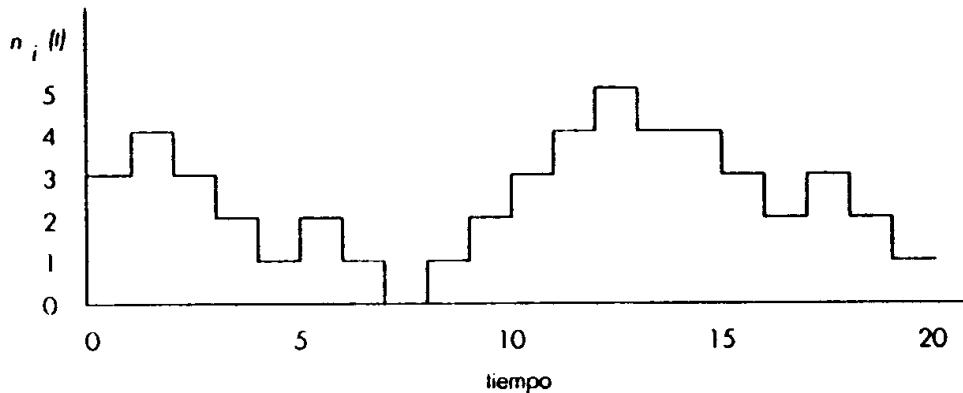


FIGURA 4.9.

$$N_i = X_i R_i$$

conocida como ley de Little, que dice que el número medio de peticiones en el sistema es igual a la productividad del sistema por el tiempo medio de residencia de cada petición en el mismo.

Dado un sistema informático, la ley de Little puede aplicarse a muchos niveles distintos: a un solo recurso, a un subsistema o a todo el sistema en conjunto.

La ley de Little puede aplicarse a un sistema a distintos niveles. Así, por ejemplo, en el sistema en tiempo compartido de la figura 4.10 y, en general, en cualquier sistema, se puede aplicar a 4 niveles distintos:

Nivel 1: Aplicación de la ley de Little a un único recurso, sin incluir la cola del mismo.

En este nivel, el recurso es utilizado siempre que haya una petición presente; por ello la utilización del recurso es igual a la proporción de tiempo durante la cual está presente una petición, que también equivale al número medio de peticiones presentes.

- Población: utilización del recurso (en cada instante de tiempo hay o 1 ó 0 peticiones presentes).
- Productividad: tasa de satisfacción de peticiones.
- Tiempo de residencia: tiempo medio de servicio requerido por una petición en el recurso (ya que el retardo en la cola de espera no está incluido).

Esta aplicación de la ley de Little constituye una derivación alternativa de la ley de la utilización.

Nivel 2: Aplicación de la ley de Little al mismo recurso incluyendo su cola de espera.

En este nivel, corresponde la aplicación típica de la ley de Little a un dispositivo.

- Población: número total de peticiones en la cola o en servicio.
- Productividad: tasa de satisfacción o terminación de servicios o peticiones.

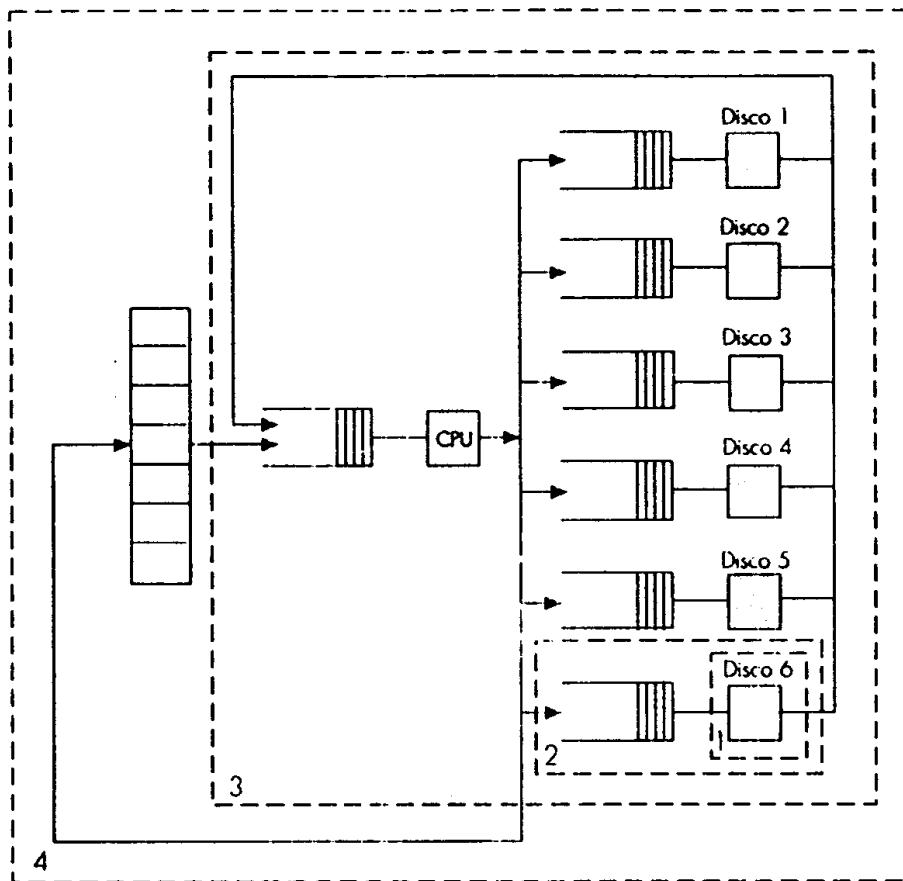


FIGURA 4.10.

- Tiempo de residencia: tiempo medio que una petición pasa en el recurso (tiempo de espera en cola + tiempo de servicio).

Nivel 3: Aplicación de la ley de Little al subsistema central (el sistema sin los terminales).

En este nivel cambia la definición de petición; ya no se está interesado en las visitas a un recurso en particular, sino en interacciones a nivel de sistema.

- Población: número de usuarios en el subsistema central (aquellos usuarios que no están pensando).
- Productividad: tasa a la que las interacciones fluyen entre los terminales y el subsistema central.
- Tiempo de residencia: noción convencional de tiempo de respuesta, es decir, el período de tiempo desde que un usuario envía una petición hasta que la respuesta es devuelta al usuario.

Nivel 4: Aplicación de la ley de Little a todo el sistema, incluyendo los terminales.

A este nivel cambia la definición de petición; ya no se está interesado en las interacciones a nivel de sistema, sino en el ciclo completo de una interacción entre sistema y terminales.

- Población: número total de usuarios interactivos.
- Productividad: tasa a la que fluyen las interacciones entre los terminales y el sistema.
- Tiempo de residencia: suma del tiempo de respuesta más el tiempo de reflexión de los usuarios.

Si se denomina Z al tiempo medio de reflexión, se puede escribir la ley de Little a este nivel de la siguiente manera:

$$N = X(R + Z)$$

que se suele expresar como la ley del tiempo de respuesta de un sistema interactivo:

$$R = \frac{N}{X} - Z$$

Ejemplo

En un entorno interactivo como el de la figura 4.10 se dispone de las siguientes medidas:

- 10 terminales activos.
- Por término medio hay 7.5 terminales trabajando.
- Productividad del sistema igual a 0.5 interacciones/s.
- Datos del disco A:
 - Sirve 40 peticiones/s.
 - Cada petición requiere una media de 0.0225 s de servicio.
 - Número medio de peticiones presentes igual a 4.

Se pide:

1. Número medio de peticiones recibiendo servicio en el disco A.
2. Tiempo medio de residencia de una petición al disco A.
3. Tiempo medio de espera en la cola de una petición al disco A.
4. Número medio de peticiones en la cola de espera del disco A.
5. Tiempo medio de respuesta percibido por el usuario.
6. Tiempo de reflexión.

Solución:

- La productividad del disco es de $X_{DISCO} = 40$ accesos/s y el tiempo medio de servicio por petición $S_{DISCO} = 0.0225$ s; por tanto, aplicando la ley de Little, se tiene:

$$U_{DISCO} = 40 \times 0.0225 = 0.9 = 90\%$$

- Por tanto, el número medio de peticiones recibiendo servicio en el disco es de 0.9.
- Como el número medio de peticiones presentes es $N = 4$ y el disco atiende 40 accesos/s ($X_{DISCO} = 40$), el tiempo medio de residencia de una petición en el disco será de:

$$R_{DISCO} = \frac{4}{40} = 0.1 \text{ s}$$

- Si el tiempo de espera más el tiempo de servicio es igual a 0.1s y el tiempo de servicio es igual a 0.0225s, entonces el tiempo medio de espera en la cola de una petición al disco será de:

$$T_{COLA} = 0.1 - 0.0225 = 0.0775 \text{ s}$$

- El número medio de peticiones en la cola de espera del disco se puede deducir del siguiente modo:

Las peticiones en servicio más las peticiones en espera en la cola son iguales a 4, y como en promedio (utilización) 0.9 peticiones están recibiendo servicio, el número medio de peticiones en espera en la cola es de 3.1 peticiones.

- Se tiene una productividad de $X = 0.5$ interacciones/s, y restán trabajando por término medio $N = 7.5$ usuarios. Aplicando la ley de Little se obtiene un tiempo medio de respuesta de:

$$R = \frac{N}{X} = \frac{7.5}{0.5} = 15 \text{ s}$$

- Aplicando la ley del tiempo de respuesta interactivo, se obtiene que:

$$Z = \frac{N}{X} - R = \frac{10}{0.5} - 15 = 5 \text{ s}$$

donde N es ahora el número total de usuarios.

4.4.5. Ley del flujo forzado

Durante la discusión de la ley de Little, el campo de visión iba desde un recurso individual a considerar todo el sistema. Para cada nivel de detalle, se necesitan definiciones diferentes de petición. Por ejemplo, si se considera un disco, es habitual definir una petición como un acceso a disco, y medir productividad y tiempo de residencia a partir de esa base.

La ley del flujo forzado expresa la relación entre estas dos visiones del sistema. Esta ley relaciona la productividad del sistema con las productividades individuales de los dispositivos y establece que los flujos (productividades) tienen que ser proporcionales uno respecto al otro en todas las partes del sistema.

En un modelo abierto, el número de trabajos que abandonan el sistema por unidad de tiempo define la productividad del mismo. En un modelo cerrado, ningún trabajo abandona el sistema. No obstante, se puede considerar que atravesar el enlace exterior equivale a abandonar el sistema e inmediatamente volver a entrar en el mismo. La productividad del sistema se define, entonces, como el número de trabajos que atraviesan ese enlace por unidad de tiempo (figura 4.11).

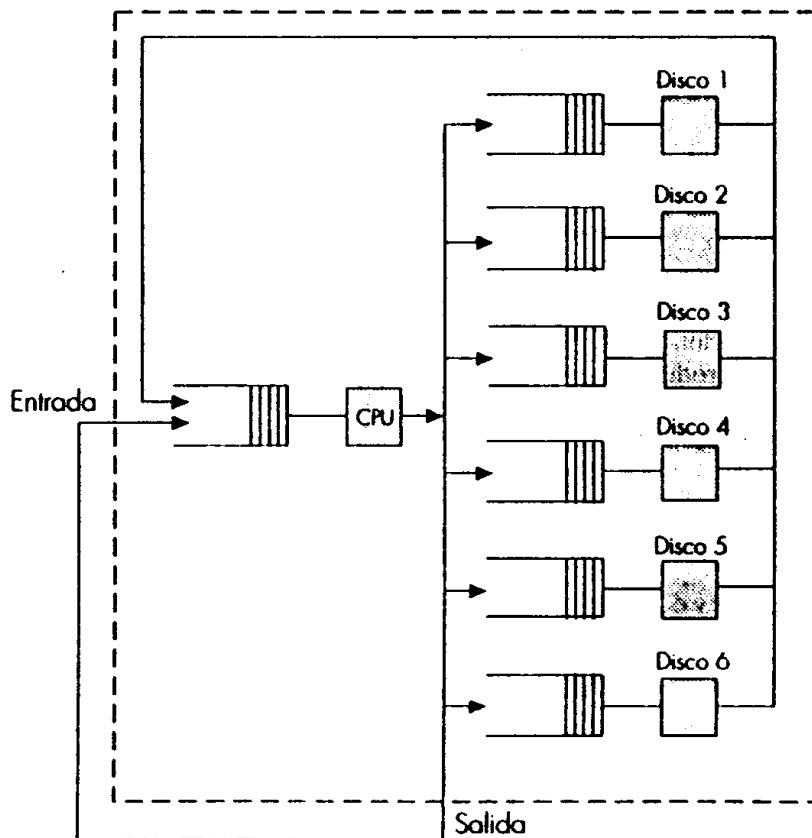


FIGURA 4.11.

Se supone que durante un intervalo de observación se contabilizan no sólo las terminaciones a nivel de sistema sino también el número de ellas en cada recurso. Se define la tasa de visita a un recurso como la relación entre el número de terminaciones en ese recurso y las que ha habido en el sistema en el mismo

periodo, o más intuitivamente, el número medio de visitas que una petición a nivel de sistema hace a ese recurso. Se usará el subíndice k para hacer referencia al recurso k -ésimo (una variable sin subíndice seguirá refiriéndose al sistema global). Así se puede definir V_k , la tasa de visita al recurso k como:

$$V_k = \frac{C_k}{C}$$

Si durante un intervalo de observación se miden 10 terminaciones a nivel de sistema y 150 en un disco concreto, entonces como media, cada petición a nivel de sistema requiere $150/10 = 15$ operaciones en disco.

Si se reescribe esta definición como $C_k = CV_k$ y se recuerda que $C/T = X$ (productividad), entonces la productividad del recurso k viene dada por:

$$X_k = X V_k$$

y esta relación se conoce como la ley del flujo forzado.

Sustituyendo las ecuaciones de la ley del flujo forzado en las ecuaciones del equilibrio del flujo, se obtiene las ecuaciones que permiten determinar las tasas de visita a partir del conocimiento del comportamiento del sistema,

$$V_0 = 1$$

$$V_j = q_{0j} + \sum_{i=1}^K V_i q_{ij}, \text{ para } j = 1, 2, \dots, K$$

que son $K + 1$ ecuaciones con $K + 1$ incógnitas de las que siempre se puede obtener una solución única y en las que siempre se tiene derecho a hacer $V_0 = 1$, ya que en el punto donde se mide el flujo de trabajos se tendrá forzosamente una visita por trabajo.

Ejemplo. Si cada trabajo en un sistema *batch* requiere una media de 6 accesos a un disco específico, y el disco atiende 12 peticiones de los trabajos *batch* por segundo, entonces ¿cuál será la productividad de trabajos *batch* del sistema?

La productividad será de:

$$X = \frac{X_k}{V_k} = \frac{12}{6} = 2 \text{ trabajos/s.}$$

Si además, se sabe que otro disco sirve 18 peticiones de trabajos *batch* por segundo, ¿cuántos accesos requiere un trabajo *batch*, por término medio, a ese segundo disco?

$$V_k = \frac{X_k}{X} = \frac{18}{2} = 9 \text{ accesos}$$

La ley de Little resulta especialmente eficaz si se combina con la ley del flujo forzado.

Ejemplo

Hay que determinar el tiempo medio de respuesta para un sistema interactivo con las siguientes características:

- 25 terminales ($N = 25$).
- Tiempo medio de reflexión de 18 s ($Z = 18$ s).
- Se realizan 20 visitas a un disco específico por interacción ($V_{DISCO} = 20$).
- El disco presenta una utilización del 30 % ($U_{DISCO} = 0.3$).
- Tiempo medio de servicio por visita a ese disco es de 25 ms ($S_{DISCO} = 0.025$ s).

Interesa aplicar la ley del tiempo respuesta de un sistema interactivo, pero no se conoce la productividad X .

Según la ley del flujo forzado aplicada al disco del sistema, se tiene:

$$X = \frac{X_{DISCO}}{V_{DISCO}}$$

y según la ley de la utilización, se tiene:

$$X_{DISCO} = \frac{U_{DISCO}}{S_{DISCO}} = \frac{0.3}{0.025} = 12 \text{ accesos/s}$$

de donde

$$X = \frac{X_{DISCO}}{V_{DISCO}} = \frac{12}{20} = 0.6 \text{ interacciones/s}$$

y por la ley del tiempo de respuesta de un sistema interactivo

$$R = \frac{N}{X} - Z = \frac{25}{0.6} - 18 = 23.7 \text{ s}$$

Se puede describir el requerimiento de servicio en disco de una interacción de dos maneras:

- Indicando que la interacción realiza un cierto número de visitas al disco y requiere una cierta cantidad de servicio en cada visita.
- Especificando la cantidad total de servicio requerida por una interacción.

Estos dos puntos de vista son equivalentes, y en cada momento se escogerá el más conveniente. Así se puede definir la demanda de servicio, D_k , a un recurso k , como:

$$D_k = V_k S_k$$

Ejemplo: Si un trabajo hace una media de 20 visitas a un disco y requiere una media de 25 ms de servicio en cada visita, entonces el trabajo requiere un total de $20 \times 25 = 500$ ms de servicio de disco, es decir, su demanda de servicio es de 500 ms en ese disco.

A partir de ahora se utilizará S_k para hacer referencia al requerimiento de servicio por visita al recurso k , y D_k para hacer referencia a la petición o requerimiento total a ese recurso. D será la suma de las demandas de servicio D_k y se denominará demanda total de servicio de un trabajo en todos los recursos del sistema.

Por tanto, a partir de las leyes de la utilización y del flujo forzado, se puede deducir que

$$U_k = X_k S_k = X V_k S_k = X D_k$$

4.4.6. EJEMPLOS

Ejemplo 1

En un sistema en tiempo compartido (figura 4.12) se han obtenido las siguientes medidas

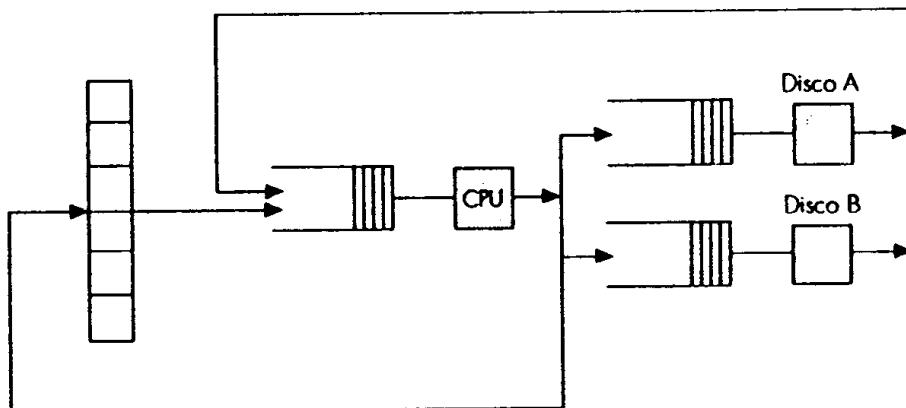


FIGURA 4.12.

- Cada programa requiere una media de 5 s de tiempo de CPU y hace 80 peticiones de E/S al disco A y 100 peticiones al disco B.
- El tiempo medio de reflexión de los usuarios es de 18 s.
- De las especificaciones de los dispositivos se sabe que el disco A requiere 50 ms para satisfacer un acceso de E/S y el disco B 30 ms por acceso.
- Con 17 terminales activos, la productividad del disco A es de 15.7 accesos de E/S por segundo.

¿Cuál es la productividad del sistema y la utilización de los distintos dispositivos?

Solución:

- Datos:
 - $D_{CPU} = 5 \text{ s}$
 - $V_A = 80$
 - $V_B = 100$
 - $S_A = 0.05 \text{ s}$
 - $S_B = 0.03 \text{ s}$
 - $Z = 18 \text{ s}$
 - $N = 17 \text{ terminales}$
 - $X_A = 15.7 \text{ accesos/s}$

Como los trabajos deben visitar la CPU antes de ir a uno de los discos o volver al terminal, la tasa de visita a la CPU es:

$$V_{CPU} = V_A + V_B + 1 = 181$$

El primer paso en el análisis operacional suele ser, generalmente, determinar la demanda de servicio total D_k de todos los dispositivos. En este caso se tiene:

- $D_{CPU} = 5 \text{ s}$
- $D_A = V_A S_A = 0.05 \times 80 = 4 \text{ s}$
- $D_B = V_B S_B = 0.03 \times 100 = 3 \text{ s}$

Utilizando la ley del flujo forzado, las productividades son:

- $X = \frac{X_A}{V_A} = \frac{15.7}{80} = 0.1963 \text{ trabajos/s}$
- $X_{CPU} = X V_{CPU} = 0.1963 \times 181 = 35.48 \text{ interacciones/s}$
- $X_B = X V_B = 0.1963 \times 100 = 19.6 \text{ accesos/s}$

Usando la ley de la utilización, las utilizaciones de los dispositivos son:

- $U_{CPU} = X D_{CPU} = 0.1963 \times 5 = 98 \%$
- $U_A = X D_A = 0.1963 \times 4 = 78.4 \%$
- $U_B = X D_B = 0.1963 \times 3 = 58.8 \%$

Ejemplo 2

La figura 4.13 representa un sistema en tiempo compartido con una restricción de memoria: se puede dar una acción de *swapping* entre dos interacciones. Así una petición puede ser forzada a estar en cola para conseguir una partición de memoria antes de competir por los recursos del subsistema central.

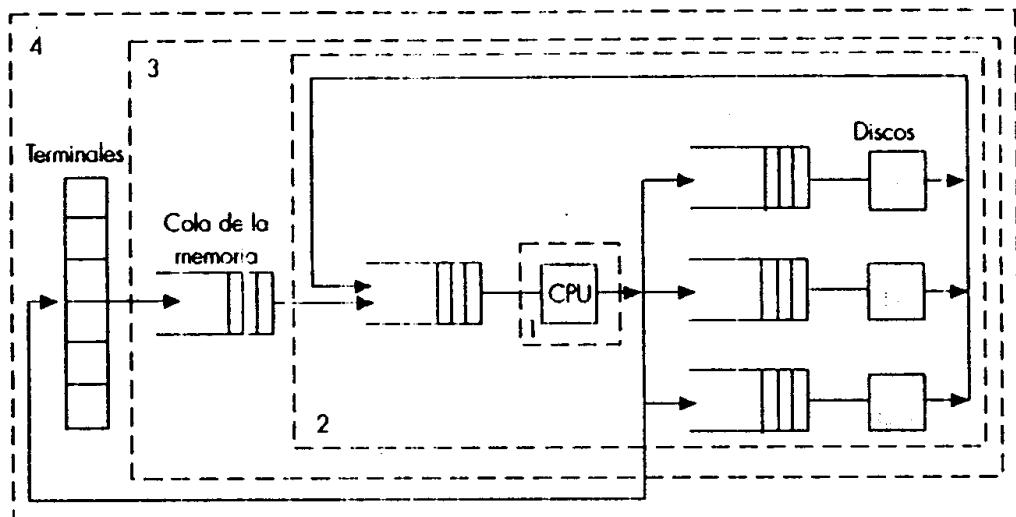


FIGURA 4.13.

De nuevo se aplicará la ley de Little a diferentes niveles.

Se obtuvieron los siguientes datos medidos observando la carga de tiempo compartido de un sistema con diferentes cargas:

- Número medio de usuarios de tiempo compartido igual a 23 ($N = 23$).
- Tiempo medio de respuesta percibido por el usuario igual a 30 s ($R = 30$ s).
- Productividad de los usuarios de tiempo compartido igual a 0.45 interacciones/s ($X = 0.45$ interacciones/s).
- Número medio de peticiones de tiempo compartido ocupando memoria igual a 1.9 ($N_{cuerda} = 1.9$).
- Tiempo medio de servicio de CPU requerido por interacción igual a 0.63 s ($D_{CPU} = 0.63$ s).

Considerar las siguientes cuestiones:

1. ¿Cuál fue el tiempo medio de reflexión de un usuario de tiempo compartido?

Aplicando la ley del tiempo de respuesta de tiempo compartido al nivel 4 de la figura 4.13, se obtiene:

$$Z = \frac{N}{X} - Z = \frac{23}{0.45} - 30 = 21 \text{ s}$$

2. ¿Cuántos usuarios, como media, estuvieron intentando obtener servicio?, es decir, ¿cuántos usuarios no estaban "pensando" en sus terminales?

Se aplica la ley de Little a nivel 3:

$$N_{\text{quierenmem}} = XR = 0.45 \times 30 = 13.5 \text{ usuarios}$$

De los 23 usuarios del sistema, una media de 13.5 estaban intentando obtener servicio.

Se sabe de las medidas que sólo 1.9 usuarios, como media, estaban ocupando memoria; por tanto, los restantes 11.6 debían estar en la cola para conseguir memoria.

3. ¿Cuánto tiempo, como media, pasaba entre la adquisición de memoria y la terminación de la interacción?

Aplicando la ley de Little al nivel 2:

$$N_{\text{enmem}} = XR_{\text{enmem}}$$

Lo cual implica que

$$R_{\text{enmem}} = \frac{1.9}{0.45} = 4.2 \text{ s}$$

Es decir, de los 30 s de tiempo de respuesta percibidos por el usuario, cerca de 26 s transcurren en espera de acceder a la memoria.

4. ¿Cuál es la contribución a la utilización de la CPU de la carga en tiempo compartido?

Aplicando la ley de Little a nivel 1, la CPU (Ley de la utilización):

$$U_{CPU} = XD_{CPU} = 0.45 \times 0.63 = 28 \% \text{ de la capacidad de la CPU.}$$

Hay que tener en cuenta en esta aplicación de la ley de la utilización que X estaba definido a nivel del sistema y la petición de servicio en base a petición por interacción.

Ejemplo 3

En este ejemplo se intentarán combinar las leyes estudiadas junto con la hipótesis de equilibrio del flujo.

Dado el modelo de colas correspondiente al sistema transaccional de la figura 4.14, en el que hay 3 dispositivos (una CPU y dos discos) y 3 clases de transacciones con las siguientes características:

Tipo de transacción	Frecuencia de llegada (trans/h)	D_{CPU} s/trans	D_{Disco1} s/trans	D_{Disco2} s/trans
Compilación	480	2.0	0.75	0.25
Ejecución	120	11.9	5.00	5.70
Sesión de edición	600	0.5	0.20	0.60

¿Cuál será la utilización de los distintos dispositivos de este sistema?

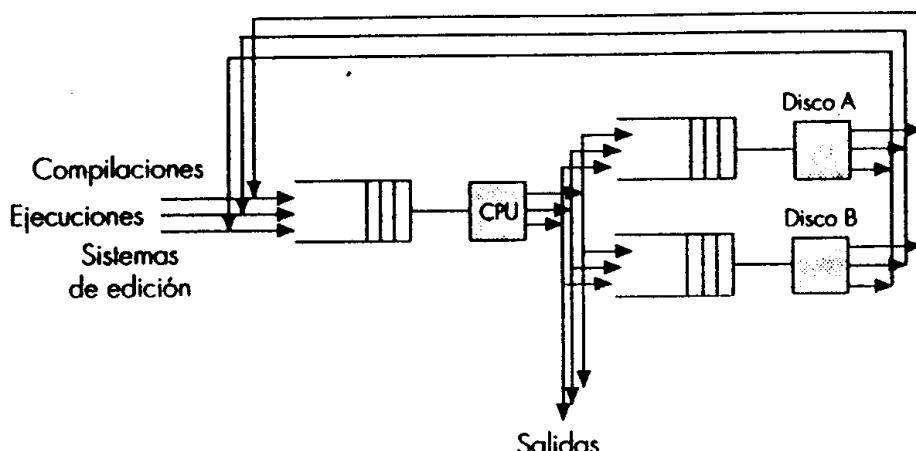


FIGURA 4.14.

Solución:

Para calcular la utilización de un dispositivo se aplicará la ley de la utilización por separado a cada tipo de transacción, y luego se sumarán los resultados.

Por ejemplo, si se considera la CPU: las transacciones de compilación llegan al sistema con una frecuencia de 480 trans/hora y cada una necesita 2.0 s de trabajo en la CPU. La utilización de la CPU debida a transacciones de compilación debe ser:

$$U_{CPU_{Comp}} = \frac{480}{3600} \times 2.0 = 27 \%$$

Argumentos similares llevan a

$$U_{CPU_{Eje}} = \frac{120}{3600} \times 11.9 = 40 \%$$

y

$$U_{CPU_{Ed}} = \frac{600}{3600} \times 0.5 = 8 \%$$

Por tanto, en total $U_{CPU} = 75 \%$.

Igualmente se podría proceder con los discos.

4.5. Distribuciones en las colas

En este apartado se considerará de nuevo el sistema general descrito en la figura 4.1, sin ninguna de las hipótesis que se han incorporado en los apartados precedentes.

Asociadas a cualquier sistema de servicio general se pueden definir tres distribuciones de la longitud de cola:

- La *distribución global*, $p(n)$, que da la proporción de tiempo que hay n trabajos en el sistema.
- La *distribución a la llegada*, $p_A(n)$, que da la fracción de llegadas que encuentran n trabajos en el sistema.
- La *distribución a la salida*, $p_C(n)$, que da la fracción de salidas que dejan n trabajos en el sistema.

Cada distribución corresponde, evidentemente, a un distinto modo de observación de la cola.

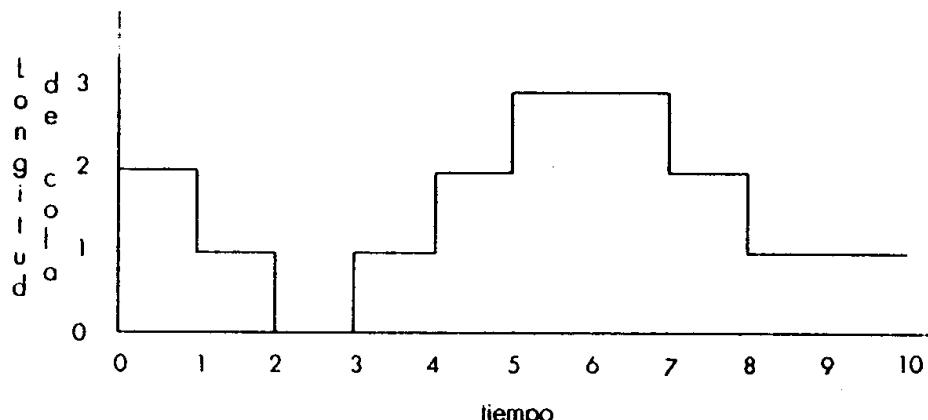


FIGURA 4.15.

La figura 4.15 muestra la longitud $n(t)$ de una cola durante un período de 10 segundos en el que se aprecian 3 llegadas y 4 salidas. En consecuencia, la tabla que representa las tres distribuciones es la siguiente:

n	$p(n)$	$p_A(n)$	$p_C(n)$
0	1/10	1/3	1/4
1	4/10	1/3	2/4
2	3/10	1/3	1/4
3	2/10	—	—

que, como se observa, pueden ser distintas. En general, las características de la cola se refieren a la distribución global, aunque en el estudio de determinados tipos de cola se consideran las restantes distribuciones como la $p_C(n)$ en la M/G/1 y la $p_A(n)$ en la G/M/1.

Se considera un solo recurso y se observa, tal como se ha hecho en el apartado 4.2., durante el intervalo $[0, T]$. $n(t)$ representa el estado de la cola, indicado por el número de trabajos en espera o recibiendo servicio en el instante t , y que varía entre 0 y N durante ese intervalo. El registro de $n(t)$ es la secuencia de comportamiento o, simplemente, el comportamiento de la cola. Se definen las siguientes magnitudes operacionales para un comportamiento dado:

- $A(n)$, número de llegadas que encuentran $n(t) = n$, variando n entre 0 y $N - 1$.
- $C(n)$, número de salidas que se producen cuando $n(t) = n$, variando n entre 1 y N .
- $T(n)$, tiempo total durante el cual $n(t) = n$, variando n entre 0 y N .

A partir de estas magnitudes observadas se pueden hallar las magnitudes observadas en el apartado 4.2., es decir,

$$A = A(0) + A(1) + \dots + A(N-1)$$

$$C = C(1) + C(2) + \dots + C(N)$$

$$T = T(0) + T(1) + \dots + T(N)$$

Dadas estas magnitudes se pueden determinar las tres distribuciones de la cola:

$$p(n) = T(n)/T, \text{ para } n = 0, \dots, N;$$

$$p_A(n) = A(n)/A, \text{ para } n = 0, \dots, N - 1;$$

$$p_C(n) = C(n+1)/C, \text{ para } n = 0, \dots, N - 1.$$

Se observa que $C(n+1)$ se usa para definir $p_C(n)$, que se refiere a los elementos que hay en la cola justo después de producirse la salida, mientras que $C(n)$ está contado justo antes de producirse. Se pueden también calcular las siguientes magnitudes:

- $S(n) = T(n)/C(n)$, tiempo medio entre salidas cuando $n(t) = n$ (definido sólo si $C(n)$ es mayor que cero).
- $\lambda(n) = A(n)/T(n)$, frecuencia de llegada cuando $n(t) = n$ (definida sólo si $A(n)$ es mayor que cero).
- $B = T(1) + T(2) + \dots + T(N)$, tiempo total de ocupación del servidor.
- $S = B/C$, tiempo medio entre salidas.
- $\lambda_0 = A/T$, frecuencia de llegada.
- $\lambda = A/[T - T(N)]$, frecuencia de llegada restringida (definida sólo si $T(N)$ es menor que T).
- $W = T(1) + 2T(2) + \dots + NT(N)$, trabajos \times segundo de tiempo de espera acumulado.

- $Q = W/T$, longitud media de la cola.
- $R = W/C$, tiempo medio de respuesta de los trabajos terminados.

Por simple sustitución se pueden justificar las siguientes nuevas leyes operacionales:

- $p_A(n) = p(n)\lambda(n)/\lambda_0$, (si $\lambda(n)$ está definida).
- $\lambda/\lambda_0 = 1/[1-p(N)]$, (si $T(N)$ es menor que T).
- $\lambda_0 = p(0)\lambda(0) + p(1)\lambda(1) + \dots + p(N-1)\lambda(N-1)$, (para las $\lambda(n)$ definidas).
- $S = p_C(0)S(1) + p_C(1)S(2) + \dots + p_C(N-1)S(N)$, (para las $S(n)$ definidas).
- $X = p(1)/S(1) + p(2)/S(2) + \dots + p(N)/S(N)$, (para las $S(n)$ definidas).
- $R = S(1)p_C(0) + 2S(2)p_C(1) + \dots + NS(N)p_C(N-1)$, (ley del tiempo de respuesta).

4.5.1. Flujo equilibrado y comportamiento unitario

La hipótesis de flujo equilibrado establece que la frecuencia de llegada global, λ_0 , coincide con la productividad, X , lo cual es equivalente a suponer que, en el período de observación T , se han producido igual número de llegadas, A , que de salidas, C , o que el estado inicial $n(0)$ coincide con el final $n(T)$.

El comportamiento unitario significa que la longitud de la cola, $n(t)$, sólo puede variar en pasos unitarios, es decir, que en un instante sólo puede haber una llegada o una salida, sin que en ningún instante una llegada coincida con una salida o se produzcan dos llegadas o dos salidas simultáneamente.

Si $n(0) = n(T)$ y si $n(t)$ sólo puede variar en pasos unitarios, entonces $A = C$ y también el número de transiciones del estado n al $n+1$ es igual a las del estado $n+1$ al n , por lo cual:

$$A(n) = C(n+1), \text{ para } n = 0, 1, \dots, N-1$$

Combinándolo con las definiciones anteriores

$$p_A(n) = p_C(n), \text{ para } n = 0, 1, \dots, N-1$$

Es decir, la distribución de llegada y de salida son idénticas siempre y cuando se cumplan las hipótesis de flujo equilibrado y de comportamiento unitario. De esta forma, se permite dejar de lado, por ejemplo, la distribución de salida, $p_C(n)$.

Para calcular $p(n)$ y $p_A(n)$ se puede establecer un procedimiento de cálculo iterativo. Cuando $n = 1, 2, \dots, N$:

$$\begin{aligned} p(n) &= T(n)/T = (C(n)/T(n-1)) (T(n)/C(n)) (T(n-1)/T) = \\ &= (A(n-1)/T(n-1)) (T(n)/C(n)) (T(n-1)/T) = \lambda(n-1)S(n)p(n-1) \end{aligned}$$

y cuando $n = 1, 2, \dots, N-1$:

$$\begin{aligned} p_A(n) &= A(n)/A = (A(n)/T(n)) (T(n)/C(n)) (C(n)/A) = \\ &= (A(n)/T(n)) (T(n)/C(n)) (A(n-1)/A) = \lambda(n)S(n)p_A(n-1) \end{aligned}$$

que permiten el cálculo de $p(n)$ y de $p_A(n)$ a partir de medidas o estimaciones de $\lambda(n)$ y $S(n)$, junto con las condiciones respectivas

$$\sum_{n=0}^N p(n) = 1$$

y

$$\sum_{n=0}^{N-1} p(n) = 1$$

Es conveniente observar que las ecuaciones correspondientes a $p(n)$ son equivalentes a las que rigen los procesos de nacimiento-muerte con llegadas poissonianas dependientes del estado y servicio exponencial también dependiente del estado y que se ha llegado a ellas sin hipótesis markovianas de ningún tipo.

4.5.2. Homogeneidad

Para el cálculo de $p(n)$ y de $p_A(n)$ es preciso medir o estimar los $2N$ valores correspondientes a $\lambda(n)$ para $n = 0, 1, \dots, N - 1$ y a $S(n)$ para $n = 1, 2, \dots, N$. En ciertos casos se simplifica el problema haciendo una de las dos o ambas simplificaciones siguientes:

$$\lambda(0) = \lambda(1) = \dots = \lambda(N - 1) = \text{constante} = \lambda$$

$$S(1) = S(2) = \dots = S(N) = \text{constante} = S$$

La primera de ellas consiste en suponer la homogeneidad en las llegadas y la segunda la homogeneidad en el servicio, es decir, hacer ambas magnitudes independientes del estado, circunstancia que se puede admitir con frecuencia en numerosos sistemas reales y que requieren el conocimiento de un número inferior de variables (2 frente a $2N$), que, por lo tanto, pueden calcularse con un mayor grado de confianza.

a) Homogeneidad de llegadas

Se considera un sistema cuyo comportamiento permita suponer que las llegadas se producen de forma homogénea. Puesto que $\lambda(n) = \lambda$, para toda n , se tiene

$$p_A(n) = p(n)\lambda/\lambda_n = p(n)/[1 - p(N)], \text{ para } n = 0, 1, \dots, N - 1.$$

En el límite, cuando N tiende a infinito, $p(N)$ tiende a cero, y la distribución global y la encontrada a la llegada coinciden.

b) Homogeneidad de servicio

Se considera un sistema cuyo comportamiento permita suponer que el servicio es independiente del estado, es decir, $S(n) = S$. La longitud media de la cola es

$$Q = p(1) + 2p(2) + \dots + Np(N)$$

y la longitud media de la cola vista por los clientes a su llegada es

$$Q_A = p_A(1) + 2p_A(2) + \dots + (N-1)p_A(N-1)$$

Si se tiene en cuenta que

$$p_A(n) = p(n)\lambda(n)/\lambda_0$$

y que

$$p(n+1) = \lambda(n)p(n)S$$

se obtiene

$$p_A(n) = p(n+1)/(S\lambda_0)$$

Puesto que el flujo es equilibrado, $\lambda_0 = X$,

$$p_A(n) = p(n+1)/(SX) = p(n+1)/U$$

y sustituyendo esta expresión en la que da la longitud de cola vista por los clientes al llegar, se obtiene

$$\begin{aligned} Q_A &= [p(2) + 2p(3) + \dots + (N-1)p(N)]/U = \\ &= [p(1) + 2p(2) + \dots + Np(N)]/U - [p(1) + p(2) + \dots + p(N)]/U = Q/U - 1 \end{aligned}$$

Teniendo en cuenta la ley de Little, $Q = RX$, se obtiene

$$Q_A = Q/U - 1 = RX/(RS) - 1 = R/S - 1$$

de donde

$$R = S(1 + Q_A)$$

que dice que el tiempo medio de respuesta, R , es el tiempo necesario para completar el servicio de los Q_A clientes que se hallan en el sistema al llegar un cliente, más el propio tiempo de servicio de ese cliente.

c) Homogeneidad simultánea de llegadas y servicios

Se considera ahora un sistema cuyo comportamiento permita suponer que tanto la llegada como el servicio son independientes del estado. Si se utilizan los resultados de los dos apartados anteriores, se obtendrá

$$\begin{aligned} Q_A &= p_A(1) + 2p_A(2) + \dots + (N-1)p_A(N-1) = \\ &= [p(1) + 2p(2) + \dots + (N-1)p(N-1)]\lambda/\lambda_0 = [Q - Np(N)]\lambda/\lambda_0 = Q/U - 1 \end{aligned}$$

de donde

$$Q = U[1 - (N+1)p(N)]/[1 - U - p(n)]$$

equivalente a los resultados, que se hallan para la cola M/M/1/N. Si la cola no está limitada, $Np(N)$ tenderá a cero cuando N tienda a infinito y se encontrará la bien conocida expresión

$$Q = U/(1 - U)$$

que da la longitud media de la cola en las colas M/M/1.

4.6. Conclusión

El análisis operacional ha permitido primeramente establecer las relaciones básicas entre las magnitudes operacionales que caracterizan la carga y el comportamiento de un sistema informático. Además se ha utilizado para poner de manifiesto el paralelismo existente con la teoría de colas, reproduciendo resultados clásicos de ésta, aunque sin necesidad de recurrir a las hipótesis difícilmente comprobables en que se basa la teoría de colas.

Ejercicios

1. Un monitor software de un sistema interactivo indica una utilización de la CPU de un 75 %, una demanda de servicio de CPU de 3 s, un tiempo de respuesta de 15 segundos y 10 usuarios activos.
¿Cuál es el tiempo medio de pensar de estos usuarios?
2. Un sistema interactivo con 80 terminales activos presenta un tiempo medio de reflexión de 12 s. Cada interacción causa una media de 15 accesos a disco por paginación. El tiempo de servicio de acceso al disco por paginación es de 30 ms y el disco está ocupado en un 60 %.
¿Cuál es el tiempo medio de respuesta?
3. Se supone un sistema interactivo que soporta a 100 usuarios con 15 s de tiempo de reflexión y una productividad de 5 interacciones/s.
¿Cuál es el tiempo de respuesta del sistema?
Se supone que la demanda de servicio evoluciona con el tiempo, de manera que la productividad del sistema baja un 50 % de su valor anterior (es decir, pasa a valer 2.5 interacciones/s). Se supone el mismo número de usuarios con un tiempo de reflexión de 15 segundos,
¿Cuál sería su tiempo de respuesta?
4. Se considera el sistema cuyo modelo aparece en la figura 4.16. Una petición de usuario que llega al sistema debe esperar en cola en memoria y puede comenzar su proceso (en el subsistema central) sólo cuando haya obtenido una partición de memoria.

Si hay 100 usuarios activos con un tiempo de reflexión de 20 s, y el tiempo de respuesta del sistema (suma de espera en cola y tiempo de residencia en el subsistema central) es de 10 s, ¿cuántos usuarios, por término medio, están compitiendo por la memoria?

Si el tiempo de espera en cola es de 8 s, ¿cuál es el número medio de usuarios cargados en memoria?

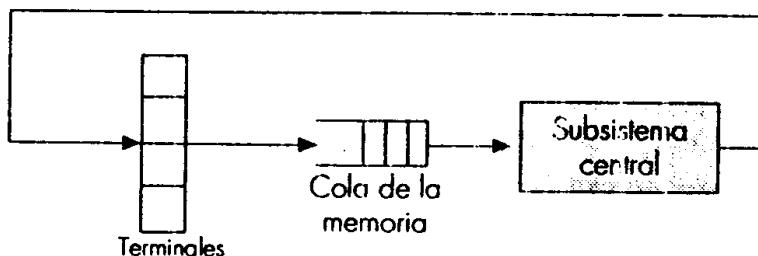


FIGURA 4.16.

5. En un intervalo de observación de 30 minutos, un disco estuvo ocupado durante 12 minutos. Si se sabe que un trabajo requiere, como media, 320 accesos a ese disco, y que el tiempo medio de servicio por acceso es de 25 ms, ¿cuál es la productividad del sistema (en trabajos/segundo)?
6. Se consideran los siguientes datos medidos en un sistema interactivo con restricción de memoria:
 - Longitud del intervalo de medida = 1 hora.
 - Número medio de usuarios = 80.
 - Tiempo medio de respuesta = 1 segundo.
 - Número medio de peticiones residentes en memoria = 6.
 - Número de peticiones terminadas = 36000.

Utilizaciones de :

- CPU = 75 %
- Disco1 = 50 %
- Disco2 = 50 %
- Disco3 = 25 %

- a) ¿Cuál es la productividad (en peticiones/s)?
- b) ¿Cuál es el tiempo medio de reflexión?
- c) ¿Cuántos usuarios, en promedio, estaban intentando obtener servicio (es decir, no estaban pensando)?
- d) ¿Cuánto tiempo, en promedio, pasa un usuario esperando para obtener memoria (es decir, no está residente en memoria)?
- e) ¿Cuál es la demanda media de servicio en el Disco1?

7. En un sistema mixto (*batch-interactivo*) se han realizado las siguientes medidas:

- Número de terminales activos, 40.
- Tiempo de reflexión de los usuarios interactivos, 15 s.
- Tiempo de respuesta de los usuarios interactivos, 5 s.
- Tiempo medio de servicio del disco, 40 ms, y su utilización, 90 %
- Cada interacción realiza una media de 10 accesos al disco.
- Cada programa *batch*, una media de 5 accesos al disco.

Se pregunta:

- ¿Cuál es la productividad interactiva?
 - ¿Cuál es la productividad *batch*?
 - Si se triplicara la productividad de trabajos *batch* del sistema, ¿cuál sería la cota inferior del tiempo de respuesta interactivo?
-

5. Análisis de los cuellos de botella

5.1. Introducción

Una parte considerable de las actividades para mejorar las prestaciones de los sistemas informáticos están relacionadas con la detección y eliminación de cuellos de botella.

Un *cuello de botella* no es más que una limitación de las prestaciones del sistema que puede ser debida a varias causas:

- Un componente hardware.
- Un componente software.
- La organización del sistema.

En ocasiones, se utiliza el término cuello de botella para indicar el componente o la parte del sistema causante del mismo. De esta forma, se dice que un disco o un compilador es un cuello de botella en lugar de decir que es su causa.

Un cuello de botella causa una ralentización considerable del tráfico de los procesos en un área del sistema. Cuando la demanda de servicio de un determinado componente excede en frecuencia e intensidad a la capacidad de servicio de ese componente, se dan las condiciones para la aparición de un cuello de botella. No obstante, éste aparecerá solamente si los demás componentes del sistema tienen una demanda relativamente inferior y trabajan a menor ritmo. Debido a la naturaleza de las peticiones de los procesos, que suelen ser secuenciales y no afectar simultáneamente a más de un componente, cuando aparece un cuello de botella en una parte del sistema, las demás partes se ven mucho menos cargadas. Muchos procesos activos estarán en las colas de espera de los dispositivos sobrecargados y no podrán contribuir a las demás colas.

El término cuello de botella sólo es apropiado cuando el problema en las prestaciones puede ser atribuido a uno o dos recursos del sistema. En un sistema en el que todos o casi todos los componentes están sobrecargados no se pueden encontrar cuellos de botella concretos, y se habla de un sistema sobrecargado o

saturado. Se trata de sistemas demasiado pequeños para la carga. Si se quisieran mejorar sus prestaciones, se tendría que reemplazar el sistema por otro más potente o bien reducir la carga.

Cuando se ha detectado la presencia de un cuello de botella, cualquier intento de eliminarlo será inútil, si no va encaminado directamente al dispositivo o recurso causante del mismo. Si en un sistema en el que el cuello de botella es debido a la velocidad insuficiente del canal, se reemplaza la CPU por una más rápida (lo que supone además un importante desembolso económico), no se apreciará ninguna mejora en el sistema. En vez de mejorar la productividad y el tiempo de presencia de los trabajos en el sistema, lo que se hace es incrementar la aglomeración de trabajos en el canal.

Por todo ello, la detección de cuellos de botella resulta de gran importancia. Y el coste de las mejoras en el sistema sólo se verá compensado si se actúa sobre los componentes causantes del cuello de botella.

5.2. Detección y eliminación de cuellos de botella

Existen distintas aproximaciones para la detección de cuellos de botella. Todas ellas son conceptualmente similares, aunque están basadas en técnicas diferentes (simulación, modelos analíticos, medidas sobre el sistema...). El método más común es el basado en la interpretación a posteriori (*off-line*) de las medidas realizadas sobre el sistema.

Conceptualmente, el procedimiento que sigue cualquiera de estos métodos se muestra en la figura 5.1.

Como consecuencia de síntomas de ineficiencia o de uno de los estudios de evaluación que se suelen realizar con regularidad en el sistema, se sospecha de la existencia de un cuello de botella. El siguiente paso será preguntarse por su localización y por la disponibilidad de métodos de eliminación que sean económicamente justificables.

Una vez formulada una hipótesis acerca de la causa del cuello de botella, habrá que proceder a validarla. Para ello se recurre a los datos medidos sobre el sistema o bien, si son insuficientes, a la recogida de más datos y a su análisis.

Cuando se confirma la hipótesis, se plantea el problema de eliminar el cuello de botella o de reducir, al menos, sus efectos. En ocasiones, la eliminación de un cuello de botella hace que aparezca otro distinto (se dice que un cuello de botella esconde otro). Este nuevo cuello de botella puede, a su vez, ser estudiado siguiendo el mismo procedimiento, el cual se repetirá hasta que el sistema esté equilibrado (*balanced*), es decir, libre de cuellos de botella.

Hay que tener en cuenta que los cuellos de botella no están directamente ligados a una configuración dada, sino que son función, en gran medida, de la carga. Así dados dos sistemas con idéntica configuración pero diferente carga, uno de ellos puede presentar un cuello de botella en un recurso dado, mientras que el otro no presenta ninguno o lo tiene en un recurso diferente. Además,

como la carga en los sistemas suele variar en el tiempo, pueden aparecer cuellos de botella temporales, que son aquellos que aparecen por un espacio de tiempo relativamente corto respecto a la sesión de medida. Este tipo de cuellos de botella aparece como consecuencia de incrementos temporales en la carga y suele ser muy difícil determinar su causa si lo que se hace es una interpretación a posteriori de las medidas realizadas (interpretación que puede llevarse a cabo horas o incluso días después de la toma de datos). En este caso, un método de interpretación de las medidas en tiempo real (*on-line*) resulta más eficaz. Su principal ventaja, que hace que este método sea especialmente interesante y útil, es la velocidad a la cual se pueden detectar síntomas importantes, la facilidad con la que se suelen identificar sus causas y la posibilidad de eliminar rápidamente los cuellos de botella temporales.

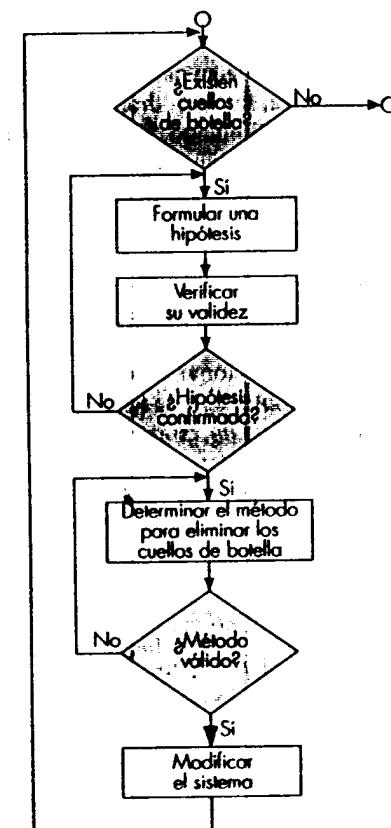


FIGURA 5.1.

Por ejemplo, si se dispone en un sistema de un monitor capaz de proporcionar en tiempo real información acerca del mismo, y en un momento dado se detectara una ralentización en el sistema, se podría pedir información al monitor acerca del funcionamiento de la CPU. Si hay un trabajo de elevada prioridad y con mucho consumo de procesador, se podría plantear la posibilidad de cancelar ese trabajo y relanzarlo con menor prioridad, solucionando así el problema o la causa de esa lentitud. En caso de no ser posible esa cancelación, habría que procurar lanzarlo la próxima vez con menor prioridad. Otra causa posible es que un trabajo haya entrado en un bucle infinito (de nuevo habría que intentar cancelarlo).

Si no se detectaran problemas en la CPU, se seguiría pidiendo información acerca de los canales. Si hubiera un trabajo con una elevada actividad de canal podría ser conveniente redistribuir sus ficheros entre los distintos canales, etc.

A continuación se tratará con más detalle la cuestión de la eliminación de cuellos de botella.

Lo habitual en un estudio de evaluación es intentar eliminar un cuello de botella tan pronto como se averigua su causa probable. Si lo que se utiliza son, por ejemplo, técnicas de medida, no se podrán detectar otros cuellos de botella hasta que el principal haya sido eliminado. Y si la detección se realiza en tiempo real, se podrán ir eliminando algunos cuellos de botella (sobre todo los temporales) de manera inmediata. Lo que no suele ser posible es detectar primero todos los cuellos de botella o todas las ineficiencias y eliminarlos, posteriormente, de manera conjunta.

Las posibles acciones que se pueden emprender para la eliminación de cuellos de botella son conceptualmente las mismas con independencia de si se utilizan técnicas de monitorización (ya sea *on-line* u *off-line*) o de modelado.

En primer lugar habrá que seleccionar los cambios que sean precisos realizar. Si se ha verificado que, por ejemplo, uno de los discos del sistema es la causa del principal cuello de botella, no por ello están resueltas las acciones que se deberán emprender. En este caso, se pueden mejorar las prestaciones del sistema de muchas maneras distintas, como por ejemplo:

- Reemplazando el disco por uno más rápido.
- Desplazando alguno de los ficheros que se utilizan con mayor frecuencia a otros discos menos cargados.
- Reorganizando el disco para reducir los desplazamientos del cabezal.

Estas acciones no son, además, mutuamente excluyentes. Todo lo contrario, el cuello de botella causado por un disco, por ejemplo, puede ser eliminado por la aplicación conjunta o simultánea de más de una terapia.

Los riesgos y el coste de esta eliminación dependen de la cura realizada. Por tanto, habrá que seleccionar con mucho cuidado las posibles terapias, evaluando la relación coste/beneficio que reportaría cada una de ellas.

En general, se pueden distinguir dos tipos de modificaciones del sistema para la eliminación de cuellos de botella:

- Modificaciones del hardware: añadir, reemplazar o incluso eliminar en ocasiones uno o más componentes hardware, es decir, *terapias de reposición (upgrading)*.
- Modificaciones que no alteran la configuración pero de alguna manera tienen efecto sobre la organización del sistema, es decir, *terapias de sintonización (tuning)*. Por ejemplo, cambiar ficheros de un disco a otro, cambiar un disco de canal, etc.

Las terapias de sintonización son, en general, más económicas y menos radicales, por lo que se les suele dar prioridad frente a las acciones de reposición.

Los cambios que suponen un desembolso económico importante deberían llevarse a cabo sólo cuando exista la suficiente seguridad acerca de su conveniencia. Es decir, habrá que intentar predecir el efecto que tendría ese cambio sobre las prestaciones. Para este tipo de estudios, lo más apropiado suele ser realizar un modelo simulado o analítico.

5.3. Análisis de cuellos de botella

5.3.1. Límites asintóticos

El análisis de límites es una aproximación muy simple y, a la vez, de gran utilidad al análisis de sistemas informáticos mediante modelos de redes de colas.

Esta técnica analítica permite obtener, con muy poco cálculo, los límites superior e inferior de la productividad y del tiempo de respuesta del sistema como funciones de la intensidad de carga (número o frecuencia de llegada de clientes al sistema).

Las técnicas de límites se utilizan sobre todo en estudios preliminares que conllevan la consideración de un gran número de configuraciones candidatas. También se pueden utilizar para estimar la mejora potencial de prestaciones que reportarían diferentes alternativas de reposición (*upgrades*). Un ejemplo mostrará cómo las gráficas de límites proporcionan una idea de qué reducción de la demanda de servicio en la estación cuello de botella es necesaria para alcanzar unos determinados objetivos de prestaciones (la demanda de servicio en una estación puede reducirse bien desplazando algún trabajo de estación o sustituyendo el servidor por uno más rápido).

Nota: Se restringirá el análisis de límites al caso de clase única. Existe una generalización para varias clases, pero no está demasiado extendida. Una de las razones es que las técnicas de límites son, sobre todo, útiles para estudios de capacidad de la estación cuello de botella, para la cual modelos de clase única son suficientes. Además, el mayor atractivo de estas técnicas se encuentra en su simplicidad, la cual se perdería si se incluyeran múltiples clases en los modelos.

Los modelos que se van a considerar se pueden describir según los siguientes parámetros:

- K , número de estaciones de servicio.
- D_{\max} , la mayor de las demandas de servicio en las estaciones del sistema.
- D , la suma de las demandas de servicio en las estaciones.
- El tipo de clientes (*batch*, conversacional o transaccional).
- Z , tiempo medio de reflexión (tipo conversacional).

Para modelos con cargas de tipo transaccional, los límites de la productividad indican la frecuencia de llegada máxima que puede procesar el sistema, mientras que los límites del tiempo de respuesta reflejan el tiempo máximo y mínimo posible que esos clientes pueden experimentar como función de la tasa de llegada al sistema.

Para modelos con un tipo de carga *batch* o conversacional, los límites indican la productividad y el tiempo de respuesta máximos y mínimos posibles como función del número de clientes en el sistema.

Se hará referencia a los límites superior de la productividad e inferior del tiempo de respuesta como límites optimistas (ya que indican las mejores prestaciones posibles) y a los límites inferior de la productividad y superior del tiempo de respuesta como límites pesimistas (ya que indican las peores prestaciones posibles). Estos límites se derivan de considerar condiciones extremas de carga (cargas ligeras y pesadas).

La validez de los límites depende de un supuesto: que la demanda de servicio de un cliente en una estación no dependa de cuántos clientes haya concurrentemente en el sistema, o en qué estaciones estén localizados.

5.3.2. Carga transaccional

En este tipo de carga, el límite de la productividad indica la frecuencia máxima de llegada de clientes que el sistema puede procesar con éxito.

Si la tasa de llegada excede este límite, los clientes sin procesar se irán acumulando cada vez en mayor cantidad. En este caso se dice que el sistema está saturado. El límite de la productividad es, por tanto, la frecuencia de llegada que separa el proceso factible de la saturación.

La clave para determinar el límite de X es la ley de la utilización:

$$U_k = X_k \cdot S_k, \text{ para cada estación de servicio } k.$$

Si se denomina λ a la frecuencia de llegada, entonces

$$X_k = \lambda \cdot V_k, \text{ por la ley del flujo forzado}$$

y se puede reescribir la ley de la utilización como

$$U_k = \lambda \cdot D_k$$

donde D_k es la demanda de servicio de la estación k .

Para derivar el límite de λ , hay que darse cuenta que mientras que a todas las estaciones les quede capacidad sin utilizar (es decir, que su utilización sea menor que 1), se puede acomodar una frecuencia de llegada mayor.

Por el contrario, si cualquiera de las estaciones se satura (es decir, su utilización llega a 1), se satura todo el sistema, ya que no puede tratar con éxito un incremento de la frecuencia de llegada.

Por tanto, el límite de λ es la menor frecuencia de llegada (λ_{\min}) que satura alguna de las estaciones.

La estación que se satura a la menor frecuencia de llegada, es la estación cuello de botella que será la estación con la mayor demanda de servicio. Se indicará la estación cuello de botella como estación *max*. Entonces:

$$U_{\max}(\lambda) = \lambda \cdot D_{\max} \leq 1$$

y

$$\lambda_{\min} = \frac{1}{D_{\max}}$$

Así, para frecuencias de llegada superiores a la que se acaba de calcular, el sistema está saturado, mientras que frecuencias inferiores es capaz de procesarlas.

Los límites asintóticos del tiempo de respuesta indican los tiempos de respuesta máximo y mínimo posibles, que los usuarios pueden experimentar con una frecuencia de llegada al sistema λ .

Como el sistema no es estable para $\lambda > \lambda_{\min}$, el estudio se limitará al caso en que la frecuencia de llegada sea inferior al límite de λ .

Se pueden considerar dos situaciones extremas:

- El mejor de los casos es que ningún cliente interfiera nunca con otro, de manera que no haya retardos de espera. En este caso, el tiempo de respuesta del sistema para cada cliente es simplemente la suma de sus demandas de servicio, D .
- En el peor de los casos, n clientes llegan juntos cada n/λ unidades de tiempo. Los clientes al final del lote tendrán que esperar en cola, experimentando grandes tiempos de respuesta. A medida que se incrementa el tamaño del lote, más y más clientes estarán esperando cada vez mayor cantidad de tiempo. Por consiguiente, dada una frecuencia λ , por cada límite pesimista del tiempo de respuesta es posible encontrar un tamaño de lote lo suficientemente grande que haga que se exceda ese límite. En conclusión, por muy baja que sea la frecuencia de llegada λ , no existe límite pesimista para el tiempo de respuesta.

Es fácil ver que la productividad para una frecuencia de llegada λ deberá ser tal que:

$$X(\lambda) \leq \frac{1}{D_{\max}}$$

y el tiempo de respuesta:

$$D \leq R(\lambda)$$

5.3.3. Cargas batch y conversacional

Las figuras 5.2 y 5.3 muestran la productividad y el tiempo de respuesta de un sistema *batch* en función del factor de multiprogramación y las figuras 5.4 y 5.5 las mismas magnitudes para un sistema conversacional.

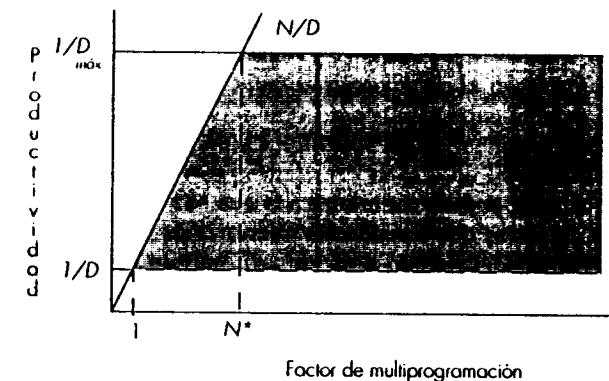


FIGURA 5.2.

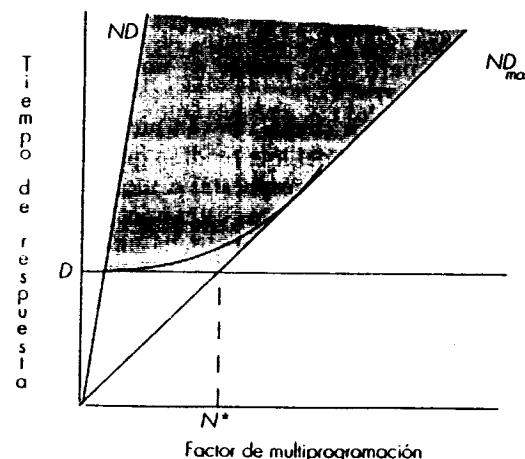


FIGURA 5.3.

Los límites indican que los valores precisos de la productividad y del tiempo de respuesta deben estar en las porciones sombreadas de las figuras.

Para obtener estos límites, se considerarán primero los límites de productividad, y luego se utilizará la Ley de Little para transformarlos en los límites correspondientes del tiempo de respuesta. El análisis se ha hecho en base a cargas conversacionales. Con $Z = 0$ se obtienen los límites para cargas *batch*.

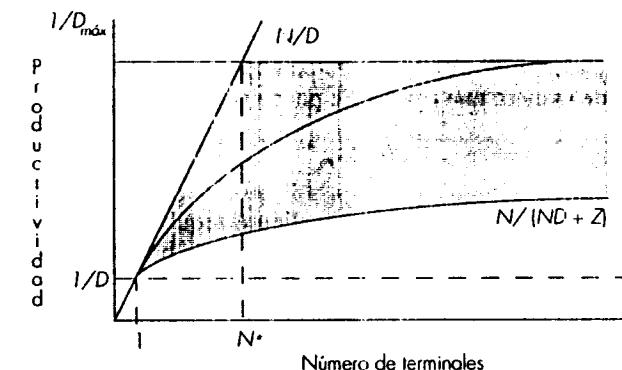


FIGURA 5.4.

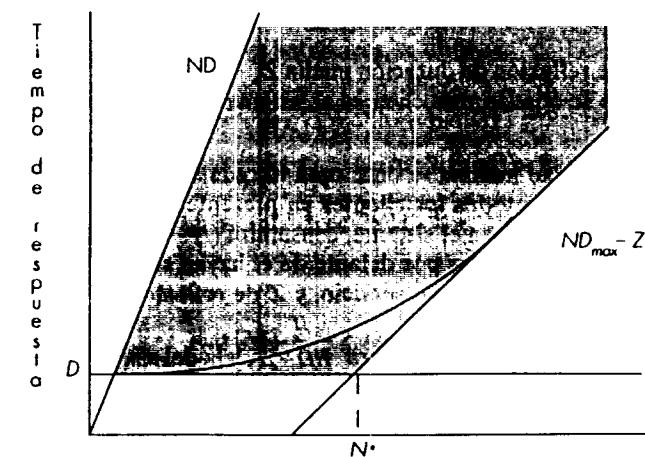


FIGURA 5.5.

Se considera en primer lugar una situación con mucha carga en el sistema (lo que se denominará carga pesada). A medida que aumenta el número de clientes (N) en el sistema, la utilización de todas las estaciones irá aumentando. No obstante existe un límite a ese incremento, ya que ninguna utilización puede ser mayor que 1. De la ley de la utilización se establece que para cada estación k :

$$U_k(N) = X(N)D_k = 1$$

Por tanto, cada estación limitará la productividad máxima que puede conseguir el sistema. Y como la estación cuello de botella, aquélla para la cual D_k es máxima, será la primera en saturarse, será también la que restrinja la productividad del sistema de manera más severa. Se puede concluir, por tanto, que:

$$X(N) \leq \frac{1}{D_{\max}}$$

Intuitivamente está claro porque cada cliente requiere una media de D_{\max} unidades de tiempo de servicio en la estación cuello de botella, por lo que no es posible una frecuencia de terminación superior a un cliente cada D_{\max} unidades de tiempo.

Se considera ahora una situación de carga ligera. Como caso extremo, un único cliente en el sistema obtendrá una productividad de $\frac{1}{D+Z}$, ya que cada interacción consta de un período de servicio, cuya duración media es:

$$D = \sum_{k=1}^K D_k$$

y de un tiempo de reflexión de duración media Z .

A medida que se añadan más clientes al sistema se pueden dar dos situaciones límite:

- La productividad mínima se dará cuando cada cliente adicional sea colocado en cola tras todos los clientes ya presentes en el sistema. En este caso, con N clientes en el sistema, consumirá $(N-1)D$ unidades de tiempo hasta que los clientes por delante de él hayan sido servidos, D unidades de tiempo en su propio servicio, y Z de reflexión, de manera que la

productividad de cada cliente será $\frac{1}{ND+Z}$, y la del sistema $\frac{N}{ND+Z}$.

- La productividad máxima posible se dará cuando cada cliente adicional no sea retrasado por ningún otro cliente del sistema. En este caso, no se consumirá tiempo alguno en cola, se emplearán D unidades de tiempo en servicio y Z en reflexión. Así la productividad de cada cliente será

$\frac{1}{D+Z}$, y la del sistema $\frac{N}{D+Z}$.

Las observaciones anteriores se pueden considerar como los límites asintóticos de la productividad del sistema:

$$\frac{N}{ND+Z} \leq X(N) \leq \min\left(\frac{1}{D_{\max}}, \frac{N}{D+Z}\right)$$

El límite optimista consta de dos componentes, el primero de los cuales se aplica bajo carga pesada y el segundo bajo carga ligera.

Como se vio en las figuras 5.2, 5.3, 5.4 y 5.5, hay un tamaño particular de población N^* tal que para toda N superior a ese valor se aplicará el límite de carga pesada y cuando sea inferior el de carga ligera. El punto de intersección se dará cuando los valores de los dos límites sean iguales:

$$\frac{N^*}{D+Z} = \frac{1}{D_{\max}}$$

lo cual implica que

$$N^* = \frac{D+Z}{D_{\max}}$$

N^* se denomina punto de saturación del sistema y representa para cargas batch el número teórico de trabajos que pueden procesarse sin colas. Cuando N supera ese valor se formarán colas de espera al menos en la estación cuello de botella. Para cargas de tipo conversacional, N^* indica el número de terminales a partir del cual al menos un recurso o estación estará saturado.

Se pueden obtener los límites del tiempo de respuesta $R(N)$, transformando los de la productividad mediante la ley de Little.

Reescribiendo las ecuaciones anteriores, se obtiene:

$$\frac{N}{ND+Z} \leq \frac{N}{R(N)+Z} \leq \min\left(\frac{1}{D_{\max}}, \frac{N}{D+Z}\right)$$

Invertiendo cada componente para expresar límites de $R(N)$:

$$\max\left(D_{\max}, \frac{D+Z}{N}\right) \leq \frac{R(N)+Z}{N} \leq \frac{ND+Z}{N}$$

de donde despejando $R(N)$:

$$\max(ND_{\max} - Z, D) \leq R(N) \leq ND$$

5.3.4. Resumen de límites asintóticos

Todos los límites son líneas rectas a excepción del límite pesimista de la productividad en cargas transaccionales.

Una vez conocidos D y D_{\max} , los cálculos de los límites asintóticos, expresados como una función del número de clientes en el sistema, sólo requieren unas

pocas operaciones aritméticas. La cantidad de cómputo es independiente tanto del número de estaciones del modelo como del tamaño de la población de clientes.

5.3.5. Ejemplos de aplicación de los límites asintóticos

Ejemplo 1

Terminales de puesto de trabajo

Una compañía de seguros disponía de 20 nodos distribuidos geográficamente basados en IBM 3790 que proporcionaban unos tiempos de respuesta inaceptables. La compañía decidió entrar en un ciclo de tres años de selección, adquisición y conversión, pero durante ese periodo se requería una reposición temporal. Tanto los IBM 8130 como los 8140 se mostraban capaces de ejecutar las aplicaciones existentes y entraron, por tanto, en consideración para su uso durante ese periodo de transición de tres años. Según el distribuidor, la utilización de los 8130 iba a producir una mejora de prestaciones respecto a los 3790 de un factor del 1.5 al 2, mientras que la utilización de los 8140 mejoraría las prestaciones en un factor de 2 a 3.5. (No se precisó el significado de esa mejora de prestaciones.)

Se inició un estudio de modelado para determinar en qué lugares bastaba con la instalación del sistema más económico, el 8130. Se sabía, además, que ambos sistemas, 8130 y 8140, incluían un disco sustancialmente más rápido que el del 3790. Con respecto a la velocidad de la CPU, el procesador del 8130 era ligeramente más lento que el del 3790, mientras que el del 8140 era, aproximadamente, 1.5 veces más veloz. Combinando esta información con la obtenida a partir de medidas sobre los sistemas 3790 existentes y de experimentos de *benchmarking* sobre los sistemas 8130 y 8140, se determinaron las siguientes demandas de servicio (en segundos):

Sistema	CPU	Discos
3790 (medidas)	4.6	4.0
8130 (estimaciones)	5.1	1.9
8140 (estimaciones)	3.1	1.9

Establecidas las demandas de servicio, se utilizó la técnica de los límites asintóticos para calcular las prestaciones que se podían esperar de cada uno de los tres sistemas.

El modelo de redes de colas (figura 5.6) sólo presenta una estación de servicio de disco. Algunos nodos del sistema disponían de dos dispositivos físicos de disco pero el controlador de disco no permitía que estuvieran activos simultáneamente. Por ello, es apropiado tener una única estación de servicio de disco.

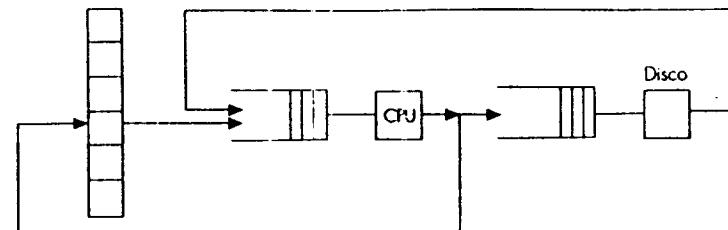


FIGURA 5.6.

Los parámetros del modelo son los siguientes:

- K , el número de estaciones de servicio (2).
- D_{max} , la mayor demanda de servicio (4.6 s en el 3790, 5.1 s en el 8130 y 3.1 s en el 8140).
- D , la suma de las demandas de servicio (8.6 s, 7.0 s y 5.0 s, respectivamente).
- El tipo o la clase de los clientes (conversacional).
- Z , el tiempo medio de retención (se estimó en 60 s).

Aplicando, para los tres sistemas, los cálculos que se vieron en el apartado 5.3., se obtienen los límites asintóticos optimistas que aparecen en las figuras 5.7 y 5.8 (los límites pesimistas se han obviado para mayor claridad).

Las gráficas revelan que con carga pesada las prestaciones del 8130 serán inferiores a las del 3790. Esto es consecuencia del hecho que el 8130 tiene una CPU más lenta, estación que es precisamente el cuello de botella del sistema. Por tanto, en lugar de obtener la prometida mejora de prestaciones (en un factor de 1.5 a 2), el reemplazar los 3790 por 8130 conllevaría una degradación de las mismas en cuanto el número de terminales activos sobrepasara un determinado valor (N^*).

A partir de la figura también se puede deducir una mejora de prestaciones si se reemplazan los 3790 por 8140, pero esta mejora no alcanza el factor prometido de dos o más.

Basados en este estudio se realizaron unos test adicionales de *benchmarking*. Estos estudios confirmaron que las prestaciones de los 8130 resultaban peores que las de los 3790 cuando el número de terminales activos era mayor o igual a quince, y que la ganancia de prestaciones de los 8130 sobre los 3790 para carga ligera resultaba despreciable. Consecuentemente, no había razón alguna para invertir en los 8130, y la compañía decidió instalar durante el periodo de transición los sistemas 8140.

Sin este estudio tan simple, la compañía podría haber ordenado la compra de los 8130 sin haber realizado los tests de *benchmarking*.

Hay que remarcar, no obstante, que las conclusiones obtenidas en este estudio están directamente relacionadas con la carga. Una carga diferente podría haber llevado a resultados distintos.

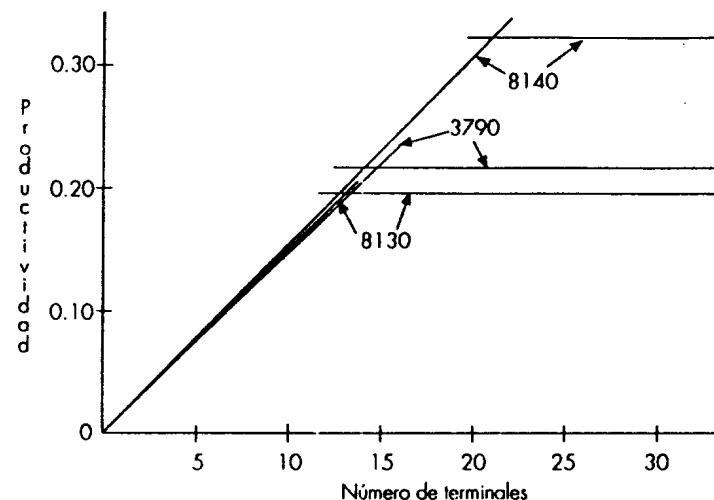


FIGURA 5.7.

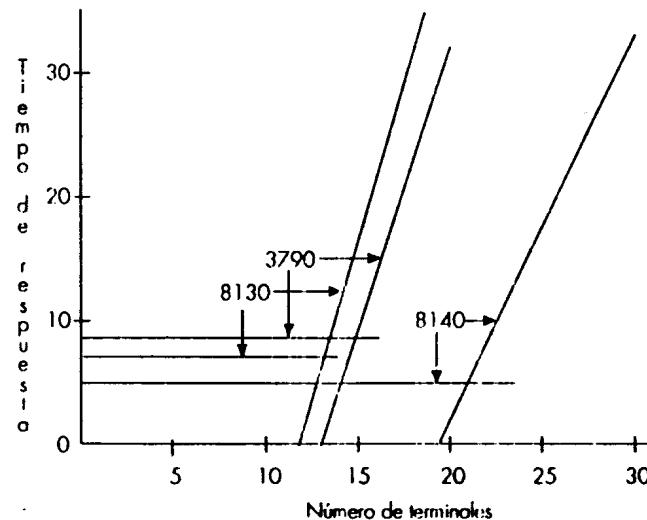


FIGURA 5.8.

Ejemplo 2**Eliminación de cuellos de botella**

Ya se ha visto que la estación cuello de botella restringe la productividad del sistema a $1/D_{\text{max}}$ como máximo. Pero, ¿qué ocurre si se alivia ese cuello de botella reemplazando el dispositivo por uno más rápido o desplazando trabajo a otro dispositivo? En cualquier caso se reduciría D_{max} y la asintota optimista de la productividad, $1/D_{\text{max}}$, se incrementaría. El límite a esa mejora lo impondrá la estación que originalmente presentaba la segunda demanda de servicio más alta. A esa estación se la denomina cuello de botella secundario, frente al cuello de botella primario que será la de mayor demanda.

Se considera, a modo de ejemplo, un modelo con tres estaciones de servicio ($K = 3$), una carga conversacional con un tiempo medio de reflexión de 15 s ($Z = 15$) y demandas de servicio de 5, 4 y 3 segundos en las estaciones ($D_1 = 5$, $D_2 = 4$ y $D_3 = 3$).

En las figuras 5.9 y 5.10 están representados los límites asintóticos optimistas de productividad y tiempo de respuesta que se obtendrían para este ejemplo.

Se han representado, además, en líneas discontinuas, los límites optimistas para carga pesada correspondientes a cada una de las estaciones. Estos gráficos proporcionan una visión de la mejora de prestaciones que reportaría la eliminación del cuello de botella primario. Cuando se reduce la carga en la estación cuello de botella, se desplaza hacia arriba el límite optimista para carga pesada de la productividad y hacia abajo el del tiempo de respuesta. Los límites para carga ligera, aunque también sufren una modificación, son menos sensibles a la demanda de servicio en las estaciones que los de carga pesada.

Una conclusión que se puede obtener de este estudio es que cualquier mejora que no afecte al cuello de botella no producirá mejora alguna en las prestaciones del sistema. La reducción de las demandas de servicio en las estaciones que no sean cuello de botella sólo mejora las asintotas de carga ligera, y esta mejora suele ser insignificante.

En las siguientes figuras 5.11 y 5.12 se compara el efecto sobre las asintotas de dos acciones: por una parte, duplicar la velocidad (reduciendo la demanda de servicio) de la estación cuello de botella primario y, por otra, la del secundario en el sistema del ejemplo.

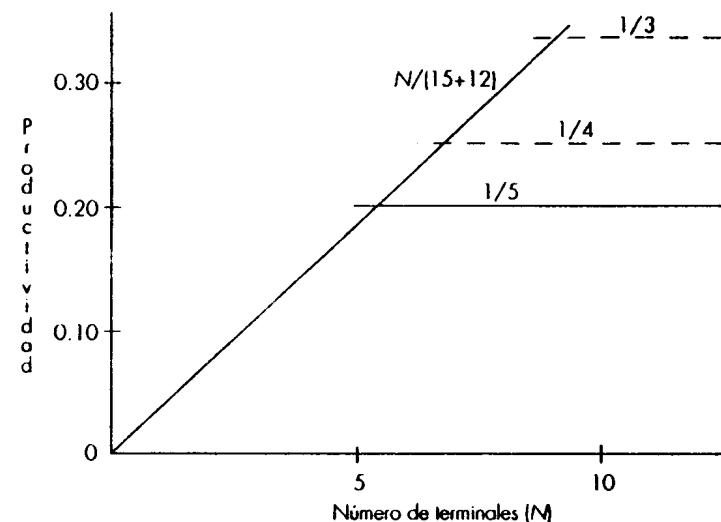


FIGURA 5.9.

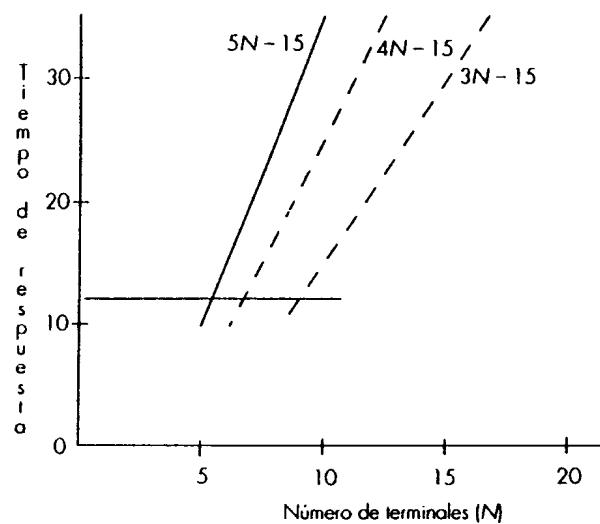


FIGURA 5.10.

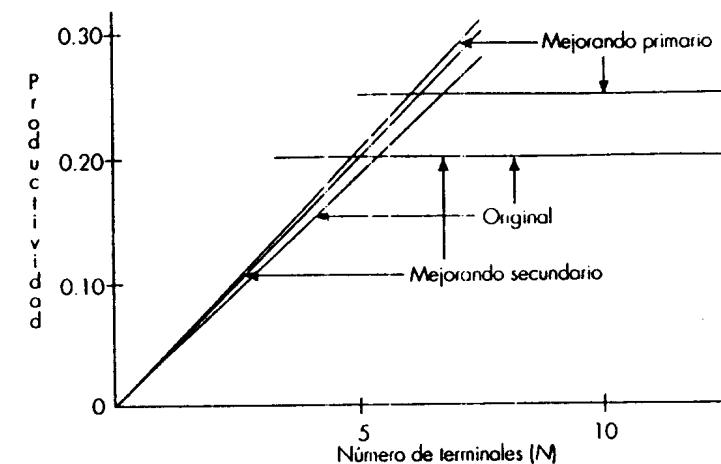


FIGURA 5.11.

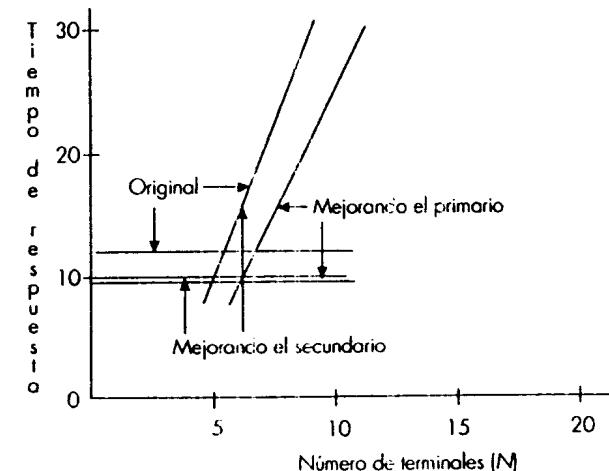


FIGURA 5.12.

Ejemplo 3**Análisis de modificación**

En este ejemplo se verá cómo utilizar los límites asintóticos para analizar el impacto que tendrían sobre el sistema una serie de modificaciones. Se considerará el sistema interactivo simplificado de la figura 5.13, para el cual se han obtenido las siguientes medidas:

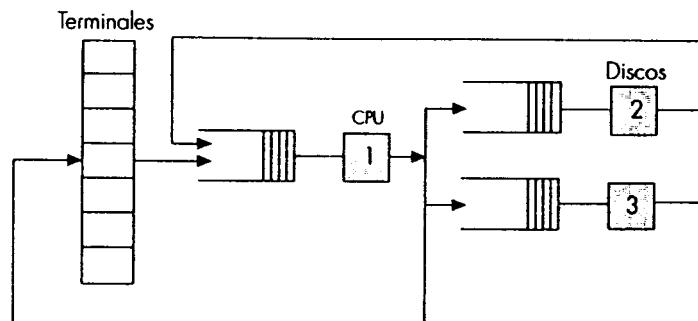


FIGURA 5.13.

- $T = 900$ segundos, longitud del intervalo de medida.
- $B_1 = 400$ segundos, ocupación de la CPU.
- $B_2 = 100$ segundos, ocupación del disco "lento".
- $B_3 = 600$ segundos, ocupación del disco "rápido".
- $C = 200$ trabajos, trabajos completados o terminados.
- $C_2 = 2000$, operaciones en el disco "lento".
- $C_3 = 20000$, operaciones en el disco "rápido".
- $Z = 15$ segundos, tiempo de reflexión.

Para obtener las asintóticas del sistema se deben obtener en primer lugar las tasas de visita a las estaciones y los requerimientos de servicio:

- $V_2 = C_2/C_0 = 2000/200 = 10$ accesos.
- $V_3 = C_3/C_0 = 20000/200 = 100$ accesos.
- $V_1 = 111$ accesos.
- $S_1 = B_1/C_1 = B_1/(V_1 C_0) = 400/(200 \times 111) = 0.018$ s.
- $S_2 = B_2/C_2 = 100/2000 = 0.05$ s.
- $S_3 = B_3/C_3 = 600/20000 = 0.03$ s.

Ahora ya se pueden obtener las demandas de servicio en las estaciones:

- $V_1 S_1 = D_1 = 2.0$ s.

- $V_2 S_2 = D_2 = 0.5$ s.
- $V_3 S_3 = D_3 = 3.0$ s.

Por tanto, el cuello de botella es la estación 3, el disco rápido.

$$D_3 = D_{\max} = 3.0 \text{ s}$$

y

$$D = \sum_{i=1}^3 D_i = 5.5 \text{ s}$$

A partir de estos datos se obtienen las siguientes asintóticas:

$$\frac{N}{5.5N+15} \leq X(N) \leq \min\left(\frac{N}{20.5}, \frac{1}{3}\right)$$

$$\max(5.5, 3N - 15) \leq R(N) \leq 5.5N$$

Se consideran cuatro posibles mejoras del sistema y se determinará para cada uno de ellas las asintóticas correspondientes.

1. Reemplazar la CPU por una dos veces más rápida. Es decir cambiar $D_1 = 2.0$ s por $D_1 = 1.0$.

Como $D_1 = 1.0$ se obtiene $D = 4.5$, y las siguientes asintóticas:

$$\frac{N}{4.5N+15} \leq X(N) \leq \min\left(\frac{N}{19.5}, \frac{1}{3}\right)$$

$$\max(4.5, 3N - 15) \leq R(N) \leq 4.5N$$

2. Equilibrar las demandas en los discos, desplazando algunos ficheros del disco rápido al lento. Se considerará sólo el efecto primario, el cambio en la velocidad del disco, y se ignorarán los posibles efectos secundarios tales como el hecho que el tamaño medio de bloque puede ser diferente en los dos discos.

La nueva demanda de servicio se deriva de la siguiente manera:

$$V_2 + V_3 = 110,$$

y como

$$S_2 = 0.05 \text{ s} \text{ y } S_3 = 0.03 \text{ s}$$

se puede decir que:

$$\frac{V_2 S_2}{S_1} + \frac{V_3 S_3}{S_1} = 110$$

y como se quiere que $D_2 = V_2 S_2 = V_3 S_3 = D_3$, entonces

$$D_2 \left(\frac{1}{0.05} + \frac{1}{0.03} \right) = 110$$

de donde $D_2 = D_3 = 2.06$ s. Dividiendo por los tiempos de servicio, se obtienen las nuevas tasas de visita: $V_2 = 41$ y $V_3 = 69$.

3. Añadir un segundo disco rápido (estación 4) para tratar la mitad de la carga del disco que esté más ocupado de los dos ya existentes. De nuevo, se considerarán únicamente los efectos primarios del cambio.

$$K = 4, D_3 = 1.5 \text{ s}, D_4 = 1.5 \text{ s}$$

4. Aplicar los tres cambios de manera conjunta: la CPU más rápida, la carga equilibrada entre los dos discos rápidos y el disco lento. Las demandas de servicio serán las siguientes:

$$K = 4, D_1 = 1 \text{ s}, D_2 = 1.27 \text{ s}, D_3 = 1.27 \text{ s}, D_4 = 1.27 \text{ s}$$

Éstas se han derivado de manera similar a como se hizo en los párrafos anteriores.

Las figuras 5.14 y 5.15 muestran las asíntotas optimistas para el sistema original (etiquetado como "(0)", para cada modificación individual (etiquetadas como "(1)", "(2)" y "(3)", respectivamente) y para la combinación de las tres (etiquetada como "(4)").

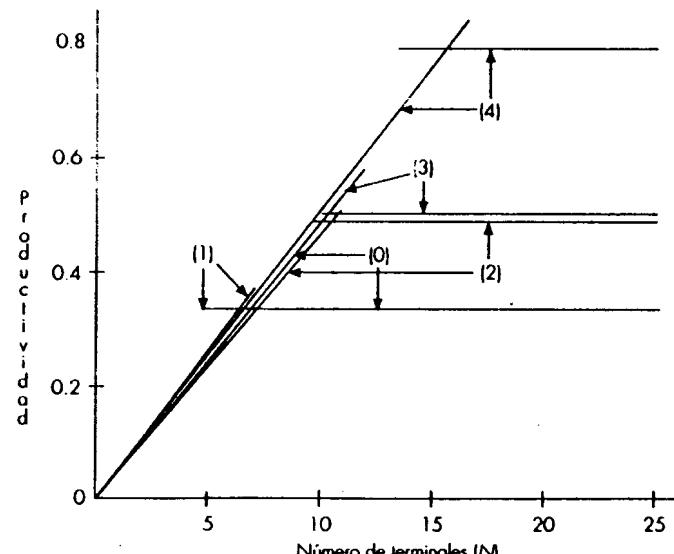


FIGURA 5.14.

Aunque intuitivamente parezca que el primer cambio debe ser el más significativo, las gráficas demuestran que no es así. Como el disco rápido es originalmente el cuello de botella, los cambios (2) y (3) tienen mucha mayor influencia. El segundo cambio proporciona casi tanta mejora como el tercero a pesar de no requerir ningún hardware adicional. La combinación de las tres modificaciones es la que produce los resultados más significativos.

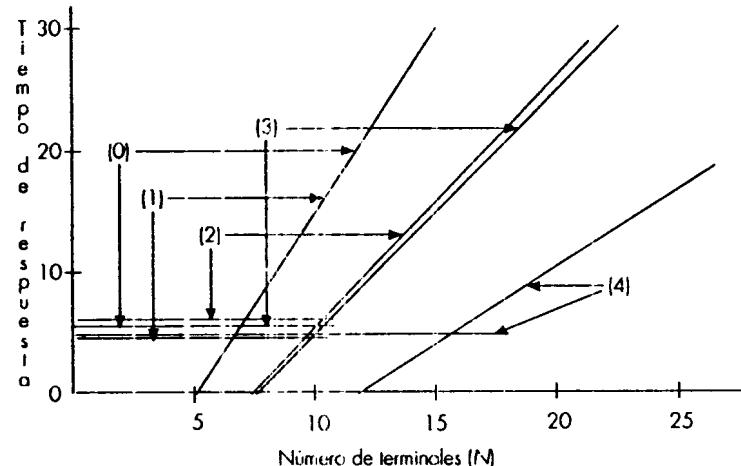


FIGURA 5.15.

5.4. Ejercicios propuestos

Ejercicio 1: Dado el modelo de sistema interactivo de la figura 5.16 se dispone de los siguientes datos:

- $S_1 = 0.03 \text{ s}, S_2 = 0.03 \text{ s}, S_3 = 0.035 \text{ s}$
- $Z = 10 \text{ s}$
- Las probabilidades de encaminamiento son $q_{10} = 0.04, q_{13} = 0.64$

1. ¿Cuál es el dispositivo cuello de botella?
2. Dibujar las asíntotas correspondientes al tiempo de respuesta en función del número de terminales conectados.
3. Si durante un intervalo de observación se midió una productividad de 0.8 int/s, y un tiempo de respuesta de 5 s, ¿cuál era el número de terminales activos durante ese intervalo?
4. ¿Sería posible obtener un tiempo de respuesta de 4 segundos con 20 terminales conectados? Si no lo fuera, ¿en qué porcentaje debería aumentar la velocidad de la CPU para permitirlo?
5. ¿Sería posible obtener un tiempo de respuesta de 8 segundos con 40 terminales conectados? Si no lo fuera, ¿en qué porcentaje debería aumentar la velocidad de la CPU para permitirlo?

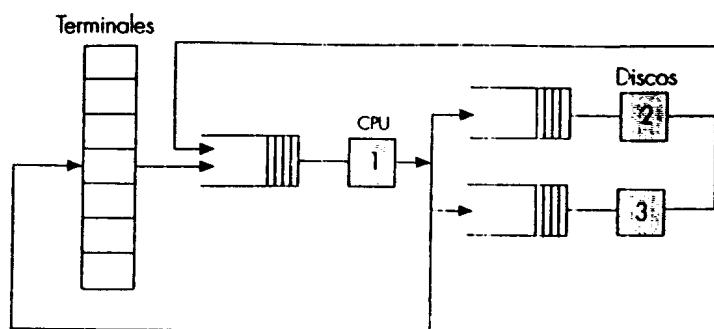


FIGURA 5.16.

Ejercicio 2: Se dispone de un sistema interactivo con una CPU y dos discos del que se han obtenido las siguientes medidas:

- Intervalo de observación de 30 minutos.
- 30 terminales activos.
- Tiempo de reflexión de 12 segundos.
- 1600 transacciones completadas en el sistema.
- 32000 accesos al disco rápido.
- 12000 accesos al disco lento.
- La CPU estuvo ocupada durante 1080 segundos.
- El disco rápido estuvo ocupado durante 400 segundos.
- El disco lento estuvo ocupado durante 600 segundos.

1. Determinar las tasas de visita, tiempo de servicio por visita y demandas de servicio de cada estación.
2. Determinar las asintotas optimistas y pesimistas de la productividad y del tiempo de respuesta para 5, 10, 20 y 40 terminales activos.
Se consideran las siguientes modificaciones en el sistema:

- Desplazar todos los ficheros al disco rápido.
 - Reemplazar el disco lento por un segundo disco rápido.
 - Incrementar la velocidad de la CPU en un 50% (manteniendo los discos originales).
 - Incrementar en un 50% la velocidad de la CPU y equilibrar la carga entre los dos discos rápidos.
3. Dibujar, para el sistema original y para las cuatro modificaciones propuestas, las asintotas optimistas y pesimistas de la productividad y del tiempo de respuesta como función del número de terminales activos.

4. Especificar, para el sistema original y para la tercera modificación, el número máximo de terminales que pueden estar conectados de manera que sea posible un tiempo medio de respuesta de 8 segundos.
5. Si en el sistema original hay 40 terminales conectados, ¿en cuánto debería incrementarse la velocidad de la CPU para posibilitar un tiempo medio de respuesta de 10 segundos?
6. Si en el sistema original hay 80 terminales conectados, ¿qué modificaciones mínimas son necesarias para lograr un tiempo medio de respuesta de 15 segundos?

6. Selección y configuración de computadores: *Benchmarking*

6.1. Introducción

Aunque en el tercer capítulo se consideró como un tipo de carga de test artificial ejecutable, hoy en día se utiliza el término *benchmark* casi como un sinónimo de carga de test. Se llama *benchmark* a *kernels*, a programas sintéticos y cargas del nivel de aplicación. Algunos autores han intentado restringir el término a un subconjunto de programas extraídos de la carga real (lo que en el capítulo 3 se denominó carga de test sintética natural) aunque esta denominación no se ha impuesto finalmente.

Por tanto, en general se puede decir que los *benchmarks* son programas utilizados para medir el rendimiento de un sistema informático o de alguna de sus partes. Y la finalidad de este tipo de estudios puede ser muy variada, como por ejemplo, la comparación de sistemas, su sintonización, la planificación de su capacidad, la comparación de compiladores en la generación de código eficiente, el diseño de sistemas o procesadores, etc.

Al proceso de comparar dos o más sistemas mediante la obtención de medidas se le denomina *benchmarking*.

Para conseguir un buen paquete de programas *benchmark* se debe seguir una serie de pasos, siendo los principales:

- Determinar los objetivos de utilización del *benchmark*. Es decir, saber qué es lo que se quiere estudiar y por qué.
- Escoger los mejores programas *benchmark* según los objetivos determinados en el paso anterior. Esto sirve para que los programas que se vayan a usar sean realmente significativos en el estudio que se desea realizar.

Éste es un paso fundamental, ya que, a partir de aquí, se va a desarrollar todo el trabajo y según como se realice esta parte, los resultados que se obtengan del *benchmark* tendrán o no algún significado.

Por ejemplo, si se van a comparar sistemas en general, se tomarán unos *benchmarks* estándar; pero si la comparación es para un usuario concreto se deberá realizar una caracterización de la carga de trabajo y extraer los programas que sean representativos de esta carga, para utilizarlos como *benchmark*.

Por ejemplo, si se desea estudiar el rendimiento de E/S de un sistema, se elegirán programas con un consumo importante de E/S y no programas con consumo intensivo de CPU.

- Se deben comprobar los aspectos del sistema bajo estudio que influyen en el rendimiento, como pueden ser el sistema operativo, los compiladores y su nivel de optimización, el uso de memorias cache, etc. Junto con los resultados obtenidos de la ejecución de los *benchmarks* se debe incluir toda la información referente a la prueba, así como la fecha en que se realizó.

Además se debe comprobar que los programas, versiones y datos usados como *benchmark* sean los mismos en todas las pruebas.

Todo esto es fundamental para que las comparaciones tengan significado, ya que la carga que se somete a los distintos sistemas a comparar deben ser lo más parecida posible y, para ello, las distintas pruebas deben realizarse con los mismos programas y datos. Aunque en algunos casos también es útil permitir optimizar los programas para cada sistema, sobre todo en los supercomputadores, debido a sus arquitecturas complicadas y elevado precio, aunque esto supone un esfuerzo considerable y sólo es realizado en ciertos casos.

- Finalmente, obtenidos los resultados y entendiendo perfectamente qué hace cada programa *benchmark*, se debe intentar estudiar la causa que produce variaciones en los resultados de los distintos sistemas.

Como programa *benchmark* se puede usar prácticamente cualquier programa ejecutable, escrito en cualquier lenguaje de programación, incluso en lenguaje máquina.

6.2. Definiciones y necesidad de los *benchmarks*

La palabra *benchmark* se puede definir de dos formas, que describen la misma entidad, aunque a diferente nivel de generalización.

La primera definición de *benchmark* es: Los *benchmarks* son una forma de evaluar las prestaciones de un sistema informático, en conjunto o sobre partes determinadas. Además, si el *benchmark* está estandarizado, los resultados se podrán comparar con los obtenidos en distintos sistemas.

Es preciso recordar que la evaluación de las prestaciones de un sistema informático es la medida de cómo un software está utilizando un hardware para atender una carga compuesta por una combinación de programas. Es decir, se estudian las interrelaciones entre el software y el hardware en presencia de una carga.

La segunda definición de *benchmark* se obtiene particularizando y creando una clasificación en la que se reparten los distintos programas usados para evaluar sistemas informáticos; en este caso los *benchmarks* se pueden definir como conjuntos de programas completos escritos en lenguaje de alto nivel extraídos de la carga real y que se consideran representativos de esta carga real.

Una vez dadas estas definiciones de *benchmark* y considerando principalmente la primera, la más general, conviene ver cuáles son los principales motivos que han provocado la aparición y amplio uso de este tipo de programas:

- En primer lugar la *adquisición de equipos*. La comparación del rendimiento de sistemas es hoy en día muy importante debido a la gran cantidad de sistemas informáticos existentes a todos los niveles, cubriendo todas las posibles demandas, y la gran cantidad de fabricantes.

Por su precio y amplia oferta se debe comparar muy bien un sistema frente a sus rivales para poder seleccionar el más apropiado para un usuario. Esta comparación se realiza usando un paquete de *benchmark*, que, a ser posible, refleje las necesidades del usuario. Esto es especialmente importante en la adquisición de equipos por parte de la empresa pública que podrá usar los resultados de los *benchmarks* como argumento válido para justificar una compra, frente a la empresa privada que puede estar influida por otros aspectos, como facilidades de financiación, por ejemplo.

Asimismo se debe remarcar que, en la selección de sistemas, este tipo de pruebas será tanto más importante cuanto mayores y más complejos sean el sistema y las aplicaciones que debe soportar. De esta forma será más importante en máquinas de tipo UNIX (multitarea y multiusuario) trabajando sobre arquitecturas diversas que en máquinas MS-DOS (monopuesto y monotarea) trabajando sobre la base de una misma familia de *chips*. Dentro de este apartado se puede incluir la comparación de sistemas, pero de forma más general, como lo pueden hacer las revistas especializadas o los fabricantes; en este caso se usarán paquetes de *benchmark* estándar que normalmente se encuentran disponibles en librerías preparadas para este fin.

Resumiendo, en este apartado los *benchmarks* estarán formados por programas extraídos de la carga real, programas estándar o una mezcla de ambos, y servirán para comparar el rendimiento de sistemas informáticos. Finalmente, al comparar el rendimiento de sistemas informáticos, se debe tener en cuenta que los programas ejecutados sean exactamente los mismos y que se ejecuten en condiciones lo más parecidas posible respecto a

sistema operativo, compiladores y sus niveles de optimización, sistemas de gestión de bases de datos, uso de memorias cache, etc.

- Los *benchmarks* se pueden usar en la *sintonización de sistemas*, es decir, cuando se quiere mejorar el rendimiento de un sistema informático que ya está en funcionamiento.

En este apartado se necesitan programas *benchmark* que permitan detectar qué partes del sistema deben mejorarse, o, una vez introducidas las modificaciones, comprobar si realmente ha aumentado el rendimiento del sistema.

En este caso los *benchmarks* pueden emplearse para evaluar el rendimiento total del sistema, y en este caso los programas se extraen de la carga real del sistema, o para comprobar sólo los aspectos mejorados en la sintonización, eligiendo *benchmarks* específicos para este fin.

Será especialmente importante conocer a fondo los programas *benchmark* y así saber con exactitud las causas que producen las variaciones en los resultados obtenidos.

- En la *planificación de la capacidad* de un sistema informático también se usan los *benchmarks* para conocer la capacidad que le queda disponible en previsión de posibles ampliaciones.

En este apartado los *benchmarks* usados deberán llevar el rendimiento del sistema hasta el límite, viendo así las carencias que tendrá el sistema en el futuro. Para este fin se utilizan programas artificiales que disponen de parámetros regulables que permiten modificar la cantidad de consumo de recursos en el sistema.

- Los *benchmarks* también son un método sumamente eficaz para *comparar compiladores* desde el punto de vista de la generación de código.

Los programas elegidos para la comparación de compiladores pueden ser estándar o extraídos de la carga real. Es conveniente indicar que ciertos compiladores sofisticados pueden detectar algunos *benchmarks* estándar y sustituirlos por algoritmos optimizados manualmente por lo que se aconseja que se compruebe el código generado por el compilador. Esto se puede hacer tomando de la carga real un programa similar al *benchmark* estándar y comprobar para este caso el código generado.

En este apartado se debe indicar sobre qué arquitectura y sistema operativo se realizan las pruebas.

- Finalmente los *benchmarks* son ampliamente usados en el *diseño de sistemas informáticos o de procesadores*.

En este caso se establece un sistema general de partida y, mediante simulación, tomando como entrada los *benchmarks* elegidos, se obtienen unos resultados, a partir de los cuales se intenta ir mejorando paulatinamente el diseño del sistema.

En el diseño se debe tener en cuenta el tipo de compilador que se emplea, ya que deben ser tan independientes como sea posible de la

arquitectura. También se pueden estudiar las interrelaciones existentes entre las distintas arquitecturas, sus implantaciones, los lenguajes de programación y los algoritmos que ejecutan.

Debido a la importancia que tienen los distintos campos en que se usan los *benchmarks*, el mundo informático está sumamente interesado en su estudio, desarrollo y utilización. A los usuarios les permite seleccionar el sistema más apropiado y los fabricantes intentan, con su uso a todos los niveles, demostrar que su sistema cubre de la mejor manera posible las necesidades de los usuarios.

6.3. Factores que influyen en el *benchmarking*

Aspectos importantes que deben tenerse en cuenta cuando se usa cualquier modelo de carga para evaluar el rendimiento de un sistema informático son los siguientes:

- El rendimiento puede depender del tipo y versión del sistema operativo que tenga instalado el sistema estudiado. Una forma idónea de realizar las pruebas de medida consiste en usar en los distintos sistemas que se van a comparar la misma copia de sistema operativo; aunque esto es inviable en la mayoría de ocasiones, sí se aconseja que las características y parámetros de los sistemas operativos sean lo más parecidos posible.
- El compilador usado en la prueba es realmente un elemento determinante del rendimiento obtenido en el sistema, ya que, según la facilidad que tenga el compilador en generar código eficiente, el modelo de carga se ejecutará más o menos rápidamente. Además en la compilación se debe tener en cuenta qué nivel de optimización se está usando. Por tanto, cuando se muestren los resultados de la prueba, éstos deberán ir acompañados de las características del compilador.

Usando el mismo modelo de carga con diferentes compiladores y diferentes grados de optimización en un mismo sistema, se pueden ver grandes diferencias en los resultados; es por ello que los *benchmarks* también se utilizan con mucha frecuencia en la comparación de compiladores para la generación de código eficiente.

- Los lenguajes de programación también influyen sobre los tests realizados con *benchmarks*, ya que tienen facilidades diferentes en secuencias de llamadas, punteros, manejo de tiras de caracteres, etc., lo que repercute directamente sobre los tiempos de ejecución, aunque los algoritmos parezcan similares.
- También se debe tener en cuenta el sistema de librerías de ejecución, ya que, según sean éstas, las funciones que implantan serán más eficientes, disminuyendo el tiempo de ejecución.

Normalmente hay dos tipos de librerías para coma flotante: una cumple con el estándar marcado por la IEEE, y otra, que es más rápida, pero

tiene menor precisión. Esto debe tenerse en cuenta sobre todo cuando el usuario necesita una buena precisión, y, por lo tanto, es necesario saber qué tipo de librería se ha usado en la prueba.

- El tamaño de la memoria cache es un factor muy importante en los resultados de pruebas de medición del rendimiento, sobre todo cuando se usan programas de pequeño tamaño. Los tiempos de ejecución varían notablemente dependiendo de si el programa cabe entero en la cache o si no cabe y debe acceder a la memoria principal.
- Es conveniente que los programas usados como modelo de carga realicen una verificación de los resultados obtenidos, para que el usuario sepa que el *benchmark* se ha ejecutado correctamente. Además, este aspecto ha tomado importancia ya que han aparecido nuevos compiladores capaces de reconocer ciertos paquetes de *benchmark* ampliamente usados, y lo que hacen es limitarse a dar el resultado esperado, o sustituyen los programas por otros optimizados a mano mucho más eficientes.

6.4. Errores comunes en el *benchmarking*

La ejecución de *benchmarks* es un método muy utilizado para comparar las prestaciones de sistemas a la hora de realizar estudios de selección o de planificación de la capacidad. A continuación se presenta una lista de errores que se observan con frecuencia en estos estudios.

- *Representar en la carga de test sólo los comportamientos medios.* Como ya se ha indicado, las cargas de test o de prueba se diseñan para que sean representativas de la carga real. Los diseñadores de carga se aseguran que la demanda relativa de recursos sea similar a la observada en el entorno real. Pero se suele representar sólo el comportamiento medio ignorando la varianza. Así, por ejemplo, el número medio de operaciones de E/S o el tiempo medio de CPU en la carga de test suelen ser similares a los de la carga real. Estos valores constantes no son deseables ya que pueden causar sincronizaciones llevando a conclusiones erróneas. En ocasiones, será necesario introducir la varianza o ir a una representación más detallada de las demandas de recurso.
- *Controlar de manera inadecuada el nivel de carga.* La carga de test dispone de varios parámetros que se pueden modificar para incrementar el nivel de carga en el sistema. Se puede, por ejemplo, incrementar el número de usuarios, decrementar el tiempo de reflexión de los mismos o aumentar la demanda de servicio por usuario. Estas tres opciones no producen los mismos resultados. Lo más ajustado a la realidad sería incrementar el número de usuarios, pero esto representaría emplear más recursos (si se están realizando medidas, emplear más emuladores de terminal remoto (RTEs) o bien más espacio de almacenamiento si lo que se está haciendo es una

simulación), por lo que se suele recurrir a las dos opciones restantes. Lo más sencillo es modificar el tiempo de reflexión, pero esto no es equivalente a incrementar los usuarios, ya que no se modifica el orden de las peticiones a los dispositivos, por lo que la probabilidad de fallo de cache será inferior a la que se obtendría en un sistema con más usuarios. Modificar la demanda de servicio cambiaría considerablemente la carga por lo que ya no resultaría representativa de la real.

- *Ignorar los efectos de la cache.* Las memorias cache son muy sensibles al orden de las peticiones. Y en la mayoría de los estudios de caracterización de la carga se pierde totalmente la información relativa al orden. La coincidencia de los valores medios y de la varianza de las demandas de recurso no asegura unas prestaciones similares de la cache. En los sistemas modernos se emplean las caches en los accesos a memoria, a los discos e, incluso, a las redes. Por ello cada vez es más necesario modelar de manera precisa el orden de las llegadas.
- *Ignorar el overhead del monitor.* Los mecanismos de recolección de datos o los monitores software que se utilizan en las medidas presentan un *overhead* que introduce error en los valores medidos. Consumen procesador, memoria y recursos de E/S para mantener y almacenar las medidas.
- *No validar las medidas.* Se trata de un error bastante común. Mientras que los modelos analíticos y simulados se suelen validar antes de su utilización, pocas veces se plantea hacer lo mismo con los datos medidos. Así cualquier error cometido en el experimento de medida puede pasar totalmente desapercibido. En los monitores hardware, por ejemplo, no es nada difícil que se coloque una sonda en un lugar equivocado o que quede suelta. Será necesario hacer comprobaciones de las medidas y cualquier valor que no pueda ser explicado debe ser investigado. Se deben incluir en las medidas comprobaciones o tests automáticos; como, por ejemplo, comprobar que el número total de paquetes enviados por todos los nodos de una red sea próximo al número total de paquetes recibido.
- *No asegurar las mismas condiciones iniciales.* Cada ejecución del *benchmark* altera el estado del sistema. El espacio en disco, por ejemplo, se ve reducido. Los registros de datos pueden ver modificando su contenido. El experimento puede ser no repetible si se vuelve a ejecutar el *benchmark* sobre el sistema modificado. Una solución sería asegurar que se borren todos los ficheros que hayan sido creados por la carga y cargar con su estado original todos aquellos que hayan sido modificados.
- *No medir las prestaciones del transitorio.* La mayoría de los experimentos de medida, simulaciones y modelos analíticos se diseñan para predecir las prestaciones en condiciones estables. Así, durante las medidas, se permite al sistema alcanzar el estado estable antes de tomar las medidas. Se permite, por ejemplo, el llenado de las caches antes de empezar a tomar valores. Esta aproximación es válida en la mayoría de los casos. Aunque, si al sis-

tema le cuesta un largo período de tiempo alcanzar el estado estable puede ocurrir que el sistema esté más tiempo en estado transitorio que en estado estable. En este caso, sería más realista estudiar las prestaciones en el transitorio que aquellas que se alcanzarían en el estado estable o régimen permanente.

- *Usar las utilizaciones de los dispositivos para comparar las prestaciones.* Las utilizaciones de los dispositivos son índices de prestaciones en el sentido que, dada la misma carga, es preferible la utilización menor. Sin embargo, emplearlas para comparar dos sistemas puede no ser adecuado. En un entorno cerrado, por ejemplo, se generan las peticiones después de un tiempo de reflexión fijo una vez atendidas las peticiones anteriores. Dados dos sistemas para este entorno, el sistema que presente un tiempo de respuesta menor tendrá más peticiones generadas por unidad de tiempo y, por tanto, también tendrá una utilización mayor de los dispositivos. Utilizaciones menores para el mismo número de usuarios no deberían interpretarse como mejores prestaciones. El índice adecuado para comparar dos sistemas en este caso sería la productividad expresada en peticiones por segundo. Otro error muy común consiste en usar las utilizaciones para la validación de modelos. Que coincidan las utilizaciones que se obtienen a partir de un modelo con las medidas en el entorno real no valida realmente el modelo.
- *Recoger muchos datos pero realizar poco análisis.* También es un error muy común. La recogida de datos es el primer paso en el *benchmarking*. El paso siguiente, el análisis de los datos, no suele recibir la atención adecuada por varias razones. En primer lugar, porque la persona que realiza las medidas no suele tener demasiada práctica en la utilización de técnicas estadísticas. Y en segundo lugar, porque la toma de datos suele ocupar tanto tiempo del proyecto que ya no queda para el análisis. Una manera de evitarlo es la formación de equipos de analistas para el análisis de las medidas, asignar al proyecto el tiempo adecuado de análisis a la hora de la planificación del mismo, e intercalar las actividades de medida y análisis, ya que a menudo se pueden modificar los planes de medida basándose en los resultados del análisis.

Los errores que se han visto son los que los analistas suelen cometer de manera involuntaria debido a su inexperiencia. Pero también se da el caso de analistas experimentados que cometen errores a propósito para demostrar así la superioridad de sus sistemas.

6.5. **Benchmarking games**

Como ya se ha comentado, el *benchmarking* consiste en comparar dos sistemas utilizando *benchmarks* conocidos. Pero este proceso no siempre se

lleva a cabo de manera honesta. A continuación se proponen algunos ejemplos que pueden llevar a que el resultado de un estudio de *benchmarking* sea erróneo.

- *Utilizar configuraciones diferentes para ejecutar la misma carga en dos sistemas.* Las configuraciones pueden tener distinta cantidad de memoria, discos diferentes o un número de discos diferente.
- *Los compiladores pueden estar diseñados para optimizar la carga.* Se ha llegado al extremo que un compilador eliminaba totalmente el bucle principal de un programa sintético, dando, por tanto, unas prestaciones infinitamente mejores que otros sistemas.
- *Utilizar una secuencia de trabajos sincronizada.* Es posible manipular la secuencia de trabajos para que las etapas limitadas por el consumo de CPU y las limitadas por E/S se sincronicen para obtener así mejores prestaciones globales.
- *Escoger la carga de manera arbitraria.* Escoger benchmarks sin verificar su representatividad de las aplicaciones reales del sistema.
- *Utilizar benchmarks muy pequeños.* Estos benchmarks dan un 100% de aciertos de cache, ignorando las posibles ineficiencias de memoria y de organización de cache. La mayoría de los sistemas reales utilizan una gran variedad de cargas. Será entonces interesante utilizar tantas cargas como sea posible para comparar varios sistemas. Si se utilizan sólo unas pocas cargas seleccionadas, se podrán decantar los resultados hacia donde se quiera.
- *Realizar una traducción manual de los benchmarks para optimizar las prestaciones.* A menudo es necesario traducir los benchmarks de manera manual para que sean ejecutables en los distintos sistemas. Las prestaciones pueden entonces depender más de la habilidad del traductor que del sistema bajo prueba.

6.6. Descripción de algunos benchmarks

En este apartado se van a nombrar y describir las características de algunos de los programas que se usan como *benchmarks*. En cuanto a las características de los programas, se menciona tanto la descripción del trabajo desarrollado por el programa, como el tipo de consumo de recursos que realiza el programa cuando se ejecuta en un sistema. Este tipo de programas se puede dividir en dos grandes grupos. Por una parte, están los programas generales, es decir programas pensados y diseñados para estudios de evaluación de prestaciones. Y por otro, los programas de aplicación, que son programas extraídos de las cargas reales de sistemas o programas tomados como *benchmarks* pero que se pueden encontrar como aplicaciones en los sistemas.

6.6.1. *Benchmarks* generales

a) *Programas LINPACK*

El *benchmark* Linpack fue desarrollado en el Argonne National Laboratory por Jack Dongarra en 1976. Es uno de los *benchmarks* más usados en estudios sobre sistemas científicos y de ingeniería. Estos programas sirven funcionalmente para resolver sistemas densos de ecuaciones lineales, y los resultados de este *benchmark* deben interpretarse de acuerdo con la función que desarrollan. Estos *benchmarks* ponen particular énfasis en el cálculo en coma flotante y, por tanto, son muy sensibles a la capacidad que tiene el hardware para realizar este tipo de operaciones. Se trata de programas fácilmente vectorizables y dependen mucho de las librerías en tiempo de ejecución porque están gran parte del tiempo ejecutando unas rutinas llamadas BLAS (*Basic Linear Algebra Subroutines*) de las que hay dos tipos, las codificadas en ensamblador y las que lo están en FORTRAN; los sistemas resueltos por estos programas, son:

- Dos sistemas de orden 100, uno de simple precisión y otro de doble precisión.
- Dos sistemas de orden 1000, uno de simple precisión y otro de doble precisión. Aunque se reduce en algunos casos el tamaño del problema a orden 500 debido al elevado tiempo de ejecución.

b) *Programas BYTE, PCW y EDN*

Estos *benchmarks* son algunos de los programas propuestos por las publicaciones BYTE, PCW y EDN; estos programas se han escrito principalmente en C y alguno en FORTRAN. De entre los propuestos por la revista BYTE, hay alguno de los *benchmarks* históricamente más usados, como son los Dhrystone, Sieve, Ackerman o Whetstone.

La característica principal de estos programas es su pequeño tamaño, por lo que suelen caber en memoria cache. En cuanto a las operaciones que realiza este conjunto son de lo más variadas, incluyendo programas de cálculo, test de disco, manejo de pantallas, ordenación, etc.

A continuación se describen los distintos programas y sus características.

— Función de Ackerman

Es un programa escrito de forma recursiva, y sirve fundamentalmente para evaluar la facilidad que tiene un lenguaje para realizar llamadas a procedimientos. Para comparar sistemas se pueden usar informaciones tales como: tiempo medio de ejecución por llamada, número de instrucciones ejecutadas por llamada, o cantidad de espacio de pila necesario para cada llamada.

— Criba (Sieve)

Este programa se basa en el algoritmo de la criba de Eratóstenes y se usa para calcular una serie de números primos hasta un número determinado. Se

7. Mejora de las prestaciones de un sistema: Sintonización

7.1. Introducción

El primer paso para mejorar las prestaciones y la eficiencia de un sistema informático es analizar su funcionamiento. Es decir, llevar a cabo un estudio de evaluación de las prestaciones del sistema.

El diagrama de la figura 7.1 muestra las etapas de un método de mejora de prestaciones. Las operaciones que se llevarán a cabo se pueden agrupar en las siguientes fases:

1. Definición de los objetivos.
2. Caracterización de la carga.
3. Selección de la instrumentación.
4. Diseño del experimento.
5. Validación.

7.2. Definición de objetivos

La definición de los objetivos del estudio es una fase fundamental, ya que en función de los mismos se determinará el método que se utilizará para analizar las prestaciones del sistema, la cantidad de recursos que es preciso emplear y la forma de justificar la inversión necesaria. Si no se tienen claros los objetivos, difícilmente se podrá dar respuesta a estas cuestiones. La figura 7.2 muestra las etapas de la definición de objetivos.

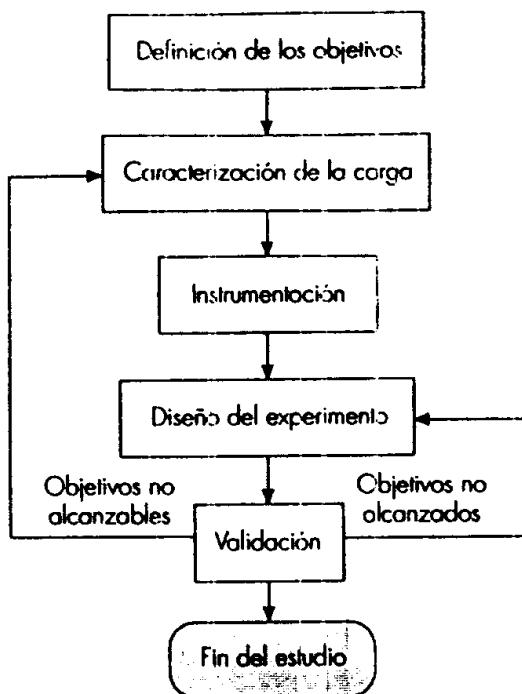


FIGURA 7.1. Etapas de un método de mejora de prestaciones.

Básicamente, hay que determinar los objetivos del estudio y el tipo de datos que se desea obtener. También es interesante especificar la importancia que tiene la adquisición de un determinado tipo de información en función de los objetivos del estudio para poder considerar así las prioridades que hay que asignar a los distintos aspectos y etapas del estudio y poder seleccionar la metodología y las herramientas que conviene emplear.

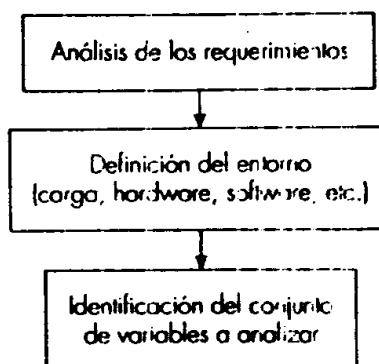


FIGURA 7.2. Etapas de la definición de objetivos

En la mayoría de los casos se plantea la necesidad de realizar un estudio de evaluación de prestaciones, más que por un análisis de la situación, por la sospecha de que existan ineficiencias, o porque las prestaciones están por debajo de lo que se esperaba. A partir de una información tan vaga no se puede pretender que simplemente adquiriendo cualquier instrumentación de medida se vaya a obtener toda la información necesaria acerca de la actividad del sistema para que, en un plazo corto, se pueda solucionar el posible problema. Los objetivos iniciales deben ser más modestos. Se suele comenzar con un análisis de los datos recogidos por las rutinas de contabilidad que, a menudo, proporcionan los puntos de partida del estudio y pueden revelar la existencia de problemas que no se habían percibido. Estos problemas se pueden subdividir en clases (análisis de las utilizaciones de los dispositivos del hardware, de la eficiencia de los programas, de la carga, de la localización de posibles cuellos de botella, etc.). Cada una de estas áreas de estudio requerirá unas herramientas y unas técnicas específicas. También será conveniente fijar el ámbito del estudio, es decir, si se va a tratar un problema o aspecto concreto, o si se trata de un estudio a nivel más global.

Ejemplos de objetivos de estudio a nivel global del sistema son los siguientes:

- Verificar la posibilidad de evitar o de, al menos, posponer por algún tiempo la adquisición de nuevo hardware (memoria, periféricos, CPU, ...).
- Reducir el *overhead* del sistema y, en general, todas las actividades que no sean productivas.
- Reducir la carga actual.

La consecución del segundo o tercer objetivo aumentará la capacidad residual del sistema y resultará de interés si lo que se pretende es encontrar espacio para nuevas aplicaciones sin tener que expandir la configuración del sistema. Aunque ambos objetivos parezcan similares, requieren para su consecución aproximaciones totalmente distintas. Para reducir las actividades no productivas se tendrá que actuar sobre el sistema operativo y sobre los programas del sistema fundamentalmente. Por el contrario, si lo que se pretende es reducir la carga, lo que se suele hacer es intentar mejorar las prestaciones de los programas existentes. Esto implica un análisis y optimización de los mismos para lograr la reducción de su consumo de recursos. Esta aproximación resulta más efectiva si no se ha realizado una optimización de los programas de aplicación con anterioridad verificando principalmente si las hipótesis que justificaron en su día el diseño adoptado, siguen siendo válidas. En algunos casos, esta reducción también puede conseguirse desplazando parte de la carga a otro sistema o dentro del mismo sistema a distintas horas. El primero, y especialmente, el segundo de los objetivos suelen requerir una localización de posibles cuellos de botella del sistema.

Ejemplos de objetivos específicos son los siguientes:

- Reducir el tiempo medio de respuesta en un porcentaje dado.

- Determinar si la tasa de paginación es tan elevada que pueda ser necesario algún tipo de intervención (como por ejemplo, la ampliación de la memoria principal).
- Determinar la mejor distribución de los archivos en los discos conectados a los diversos canales.
- Determinar la relación existente entre las utilizaciones de memoria y de CPU y el número de usuarios conectados.
- Equilibrar y optimizar la actividad de los canales.

En ocasiones para lograr un determinado objetivo hay que analizar y resolver previamente otros problemas. Para lograr, por ejemplo, el quinto de los objetivos que se acaban de citar (equilibrar y optimizar la actividad de los canales), a menudo será necesario examinar la actividad de cada canal, así como los posibles solapamientos en su actividad que presentan entre ellos y con respecto a la CPU. A continuación habrá que analizar las utilizaciones de los dispositivos conectados a los canales, para evaluar así la contribución de cada uno de ellos a la carga total del canal. También será conveniente conocer la frecuencia de los accesos a los archivos.

Por tanto, para lograr el objetivo propuesto, se deberán llevar a cabo las siguientes acciones:

- Mejora de la asignación de archivos en los dispositivos.
- Distribución óptima de los archivos entre los distintos dispositivos.
- Posible reconfiguración de las conexiones entre canales y periféricos.

La primera acción ya alcanza un objetivo por sí mismo, reducir el tiempo de acceso a disco.

Los objetivos que se han propuesto sólo son algunos de los posibles. Cuando se trata de buscar posibles soluciones a un problema concreto, es mejor no confiar demasiado en las más obvias, aunque en ocasiones la intuición proporciona resultados inmediatos con muy poco esfuerzo.

En ocasiones no será posible definir los objetivos hasta que la primera fase de comprensión del sistema se haya llevado a cabo. Esto es debido a la posible existencia de situaciones contradictorias que dificultan la definición precisa del problema. En esos casos, resulta conveniente proseguir con la investigación preliminar y recoger más información que pueda proporcionar una formulación inicial de las hipótesis de trabajo. El siguiente paso será analizar con detalle la carga.

7.3. Caracterización de la carga

Este tema ya fue estudiado en el capítulo 3 y aquí se ofrece sólo un pequeño recordatorio.

Para caracterizar la carga se suele intentar fragmentar el tiempo total transcurrido en los tiempos que pasa el sistema ocupado en las distintas actividades. A partir de este análisis inicial de la carga se podrán determinar sus principales características (su tendencia a cargar el subsistema de E/S más que la CPU, el tamaño medio de los programas, las tasas de llegada, ...) y buscar sus similitudes.

La figura 7.3 muestra las principales actividades de esta fase.

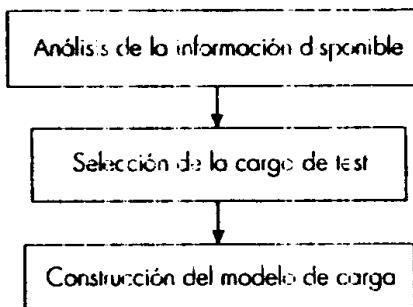


FIGURA 7.3.

7.4. Selección de la instrumentación

Una vez definidos los objetivos del estudio, el siguiente paso será medir todo aquello que pueda tener influencia sobre los mismos. Medir implica recoger datos acerca de la actividad del sistema empleando para ello las técnicas y la instrumentación adecuada. Pero, ¿cómo se saben cuáles son las técnicas adecuadas? Para responder a esta pregunta es preciso hacerlo previamente a otras tales como ¿Son medibles todas las variables que se quieren analizar? ¿Cómo se pueden realizar las medidas con la suficiente precisión?

El esquema de las actividades de esta fase está recogido en la figura 7.4.

A continuación se discutirán algunos aspectos generales de un estudio de evaluación de prestaciones, como son los criterios para la selección de la instrumentación de medida y la capacidad de medida de las diferentes variables que tienen estas herramientas.

El estudio se centra en las siguientes áreas:

- Los componentes hardware del sistema.
- El software del sistema (sistema operativo y los programas de sistema y de usuario).
- La carga, es decir, el conjunto de programas que el sistema tiene que procesar durante un período dado.

Las actividades hardware y software pueden medirse mediante monitores hardware y software. En función del tipo de medida que haya que realizar resultará más adecuado uno u otro tipo de monitor e incluso, en ocasiones, sólo será

posible detectar ciertos eventos con uno solo de los dos. También se pueden recopilar informaciones de los recursos consumidos mediante las rutinas de contabilidad del correspondiente sistema operativo.

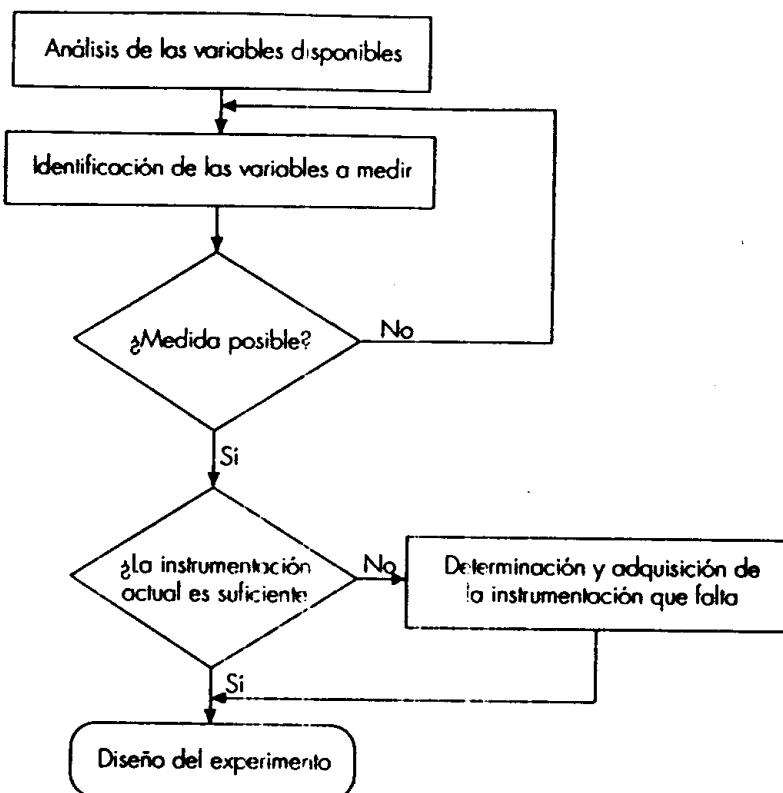


FIGURA 7.4. Actividades de la selección de la instrumentación.

7.4.1. Hardware

Es preciso conocer la utilización de cada componente hardware para poder determinar si está sobrecargado, o si está sometido a una carga por debajo de su capacidad o si presenta una utilización adecuada. Es importante detectar los componentes que causan los cuellos de botella, pero también aquellos que están prácticamente inactivos o infrautilizados. Para mejorar las prestaciones será necesario eliminar los cuellos de botella y la consecución de una distribución más equilibrada de la carga entre los componentes hardware.

La determinación de las utilizaciones de los componentes también es importante para saber qué porción de la capacidad del sistema queda sin utilizar. Así se sabrá hasta qué extremo se puede incrementar la carga sin que se produzca una degradación acusada de las prestaciones del sistema.

A continuación se presenta una lista en la que se indica, para cada componente hardware, las variables que hay que medir de mayor utilidad, así como las herramientas que resultan para su obtención.

a) *CPU*

1. *CPU ocupada*: es decir, la utilización de la CPU.
Se emplean tanto monitores hardware como software, y también se puede obtener, aunque con menor fiabilidad, a partir de las rutinas de contabilidad.
2. *Sólo CPU*: esta variable permite cuantificar el grado de solapamiento entre la CPU y otras actividades.
Monitor hardware o software.
3. *CPU ocupada en estado supervisor*: según los monitores utilizados puede indicar la ocupación de la CPU dedicada a ejecutar código del supervisor o, simplemente, el *overhead* del sistema.
Monitor software y, en algunos casos, también hardware.
4. *CPU ocupada en estado usuario*: indica cuánto tiempo pasa la CPU ejecutando programas de usuario.
Monitor software y, en algunos casos, también hardware.
5. *Ocupación de la CPU por cada usuario*: representa la contribución de cada usuario a la ocupación de la CPU.
Monitor software o rutinas de contabilidad.

b) *Canales*

1. *Canal ocupado*: indica la utilización de cada canal.
Monitor hardware o software.
2. *Equilibrio de los canales*: indica la demanda relativa de los canales y si las actividades de E/S están equilibradas entre ellos; se deriva de la utilización de todos los canales.
Monitor hardware o software.
3. *Solapamiento de canal*: indica el grado de solapamiento de los canales entre sí, y entre los canales y la CPU; se pueden definir tantos factores como combinaciones existen de dos o más componentes.
Monitor hardware o software.
4. *Longitud de cola en el canal*: indica la cantidad de contención en cada canal, o lo que es lo mismo, hasta qué punto es responsable de la existencia de un cuello de botella. Se suelen obtener la longitud media y máxima de las colas, así como la cantidad de tiempo durante la cual las colas no están vacías.
Sólo se puede medir mediante un monitor software.

c) *Unidades de control*

1. *Unidad de control ocupada*: siempre será superior al tiempo de ocupación del canal o canales que acceden a la unidad, ya que el controlador estará

ocupado durante el tiempo de la transferencia de datos y además durante la transmisión de órdenes a los dispositivos y a los canales; si no dispone de memoria cache local, la diferencia será generalmente muy pequeña, pero si la tiene, puede llegar a ser bastante apreciable.

Una elevada utilización de un controlador puede indicar la existencia de una cola de espera de peticiones de E/S para una de las unidades conectadas al mismo. Si un controlador atiende a canales conectados a diferentes CPUs, puede haber competencia también por el controlador. La utilización de un controlador sólo se puede obtener a partir de una herramienta hardware, mientras que la contención provocada por él sí es medible mediante herramientas software.

d) Dispositivos de acceso directo (discos)

1. *Ocupación del dispositivo*: indica cuánto tiempo necesita el disco para las operaciones de E/S, que incluyen los tiempos de movimiento del brazo, de latencia, de transferencia y de resolución de los conflictos en el camino hacia la memoria central. Hay que observar que el tiempo de servicio de los discos no es independiente de la carga, ya que ésta influye en el tiempo de resolución de los conflictos en el camino hacia la memoria central.
Monitor software, si sólo se necesita el tiempo total, y hardware si además se quiere la descomposición del tiempo total en sus componentes.
2. *Frecuencia de acceso al dispositivo*: indica la frecuencia de las operaciones de posicionamiento del brazo del disco.
Monitor software o hardware
3. *Tiempo de movimiento del brazo*: indica la duración de las operaciones de posicionamiento del brazo del disco. También proporciona información acerca de si la posición relativa de los archivos en disco es la adecuada.
Monitor software si sólo interesa la longitud del desplazamiento y hardware si además se quiere conocer su valoración en tiempo.
4. *Cola de espera en el dispositivo*: indica el tiempo durante el cual la cola de espera no está vacía y la longitud media de la misma. Estos índices indican la contención existente en el dispositivo.
Sólo se puede obtener a partir de herramientas software.

e) Dispositivos de acceso secuencial (cintas)

1. *Ocupación del dispositivo*: mide los tiempos de lectura-escritura, posicionamiento y rebobinado de cada unidad de cinta.
Monitor software o hardware.
2. *Solapamiento en la actividad de las cintas*: se refiere al solapamiento con la actividad de otras unidades de cinta. Dará una idea de si el número de unidades de cinta es el adecuado.
Monitor software o rutinas de contabilidad.

f) Impresoras de líneas

1. *Disponibilidad*: hace referencia al tiempo durante el cual la impresora está disponible y tiene una notable influencia en la productividad del sistema. La existencia de frecuentes períodos de tiempo de "impresora no disponible" puede degradar las prestaciones del sistema.
Monitor software o hardware.
2. *Ocupación del dispositivo*: normalmente se expresa como el número de líneas impresas en un período determinado. En un entorno *batch*, la velocidad de impresión puede restringir la productividad del sistema de manera importante.
Monitor software o hardware o rutinas de contabilidad.

7.4.2. Software*a) Sistema operativo*

El *overhead* del sistema influye tanto en las prestaciones como en la capacidad del sistema. El *overhead* puede variar de manera considerable en función de las características de la configuración, sobre todo por la cantidad de memoria disponible y por la propia configuración del sistema operativo. El conocimiento de la utilización de cada uno de los módulos del sistema operativo puede ser útil a la hora de seleccionar los parámetros de generación. También puede ser interesante conocer cómo utiliza el sistema operativo sus áreas de trabajo en disco. La existencia de un cuello de botella en el sistema operativo también puede ser responsable de una limitación de las prestaciones del sistema.

Mediante la utilización de herramientas hardware se pueden obtener medidas bastante ajustadas acerca de la utilización de la CPU por parte del sistema operativo, pero si se quiere conocer también la utilización de la misma por parte de cada módulo individual del sistema operativo, así como la longitud media de las colas de espera en los recursos, se tendrá que recurrir al uso de herramientas software.

b) Programas de sistema y de usuario

A diferencia de los módulos del sistema operativo, puede modificarse un programa de usuario o de sistema cuando se comprueba su falta de eficiencia. Será importante conocer la utilización de la CPU y el número de operaciones de E/S de cada uno de los programas, de manera global y de cada uno de sus segmentos.

Suele ser posible obtener esta información a partir de las rutinas de contabilidad, aunque en ocasiones no proporcionan las medidas de CPU y del número de operaciones de E/S de los programas de sistema o no lo hacen con suficiente exactitud. En este caso, habrá que recurrir a la utilización de herramientas software aunque en algún caso también pueden dar información herramientas hardware.

Un monitor software permite, por ejemplo, determinar la frecuencia de ejecución de las distintas sentencias de un programa, así como el número de llamadas al sistema que genera.

7.4.3. Carga

La información de la carga de programas se obtiene normalmente mediante las rutinas de contabilidad del sistema. Los programas que procesan los datos recopilados por estas rutinas suelen actuar siguiendo el esquema de la figura 7.5.

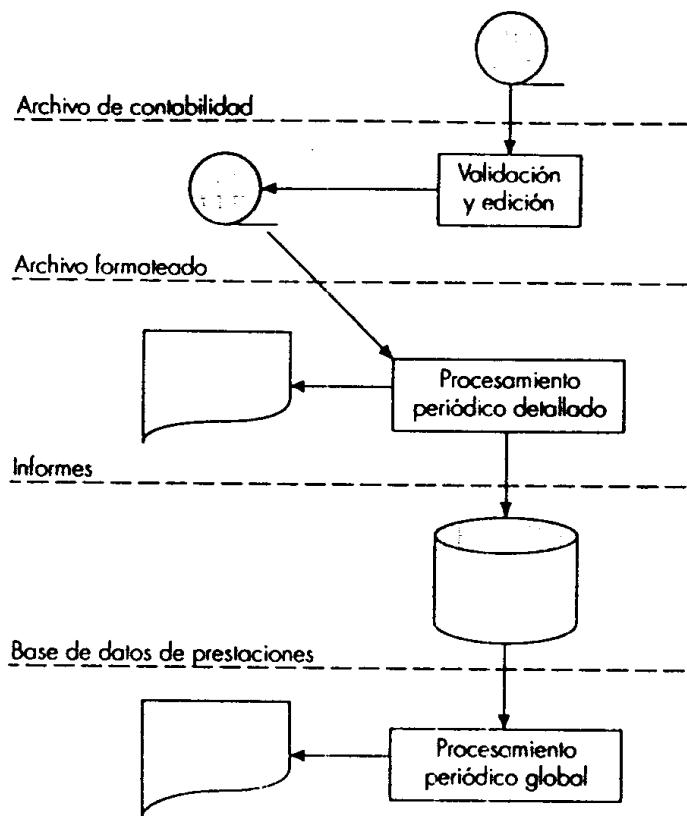


FIGURA 7.5.

Inicialmente, se validan los datos, ya que pueden existir inconsistencias debidas a anomalías en el funcionamiento del sistema. Una vez validados, se adaptan a un formato adecuado para el programa que los procesará.

Después puede haber dos tipos de reducción: una detallada, que abarca períodos de tiempo variables (unas pocas horas, un día o más tiempo) seleccionados por el usuario para analizar situaciones específicas de la carga, y una reducción global de los datos, que, basándose en los contenidos de una base de datos de prestaciones, produce informes acerca de la carga medida y de las actividades del sistema a lo largo de períodos de tiempo de longitud fija (un mes, un cuatrimestre, un semestre, etc.). Esta base de datos es actualizada durante la reducción detallada de los datos.

Los informes se elaboran a menudo indicando utilizaciones de CPU y de los demás dispositivos para cada usuario o grupos de ellos. Así se podrán deducir los importes que se deben cargar por la utilización de cada recurso. De ahí el nombre de rutinas de contabilidad, ya que en su origen fueron concebidas con este objetivo y luego se aprovecharon para evaluar las prestaciones.

La utilización de estas herramientas permite obtener gran cantidad de información acerca de la carga del sistema, como por ejemplo:

- La actividad de cada componente del sistema durante un período de tiempo determinado.
- La existencia de picos de carga durante uno o más días o en horas determinadas de cada día.
- Número total de operaciones de E/S llevadas a cabo por cada programa en los distintos dispositivos.
- La contribución a la actividad de la CPU y de los dispositivos de E/S de los trabajos *batch* e interactivos.
- La paginación debida a cada usuario (en entornos de memoria virtual).

La precisión y fiabilidad de las medidas depende de las herramientas utilizadas para su obtención, así como del empleo realizado de las mismas y de la experiencia de sus usuarios.

Las herramientas hardware, como ya se vio en el segundo capítulo, tienen la ventaja de no depender del sistema y de no interferir en él mismo (es decir, no consumir recursos durante el intervalo de medida). Esto es especialmente cierto para la utilización de la CPU, que podrá obtenerse con un monitor hardware con mayor exactitud que con uno software, ya que no se ve afectado porque ciertas actividades del sistema operativo no sean interrumpibles.

Este problema limita la precisión de las herramientas software. Por otra parte, las herramientas hardware no son capaces de detectar las condiciones que pueden causar contención en los recursos. No suele ser posible inspeccionar con este tipo de herramienta las colas de espera de los distintos recursos, por lo que, en general, no resultan adecuadas para la medida directa de la contención.

Otra característica favorable de los monitores hardware es su flexibilidad, ya que las señales procedentes de los sensores se pueden combinar de muchas maneras, permitiendo la observación simultánea de un gran número de actividades concurrentes y, por tanto, la medida de sus solapamientos.

Las herramientas software, menos flexibles, tienden a medir de manera secuencial, pero se adaptan mejor que los monitores hardware a las modificaciones en la configuración del sistema. No requieren la modificación de las conexiones existentes entre monitor y sistema (como se tendría que hacer con un monitor hardware), y sólo es necesario alterar ligeramente las órdenes para cambiar la extracción de datos. El analista dispone de un conjunto estándar de posibles medidas que puede combinar, lo que reduce de manera considerable el tiempo necesario para preparar la sesión de medida.

7.5. Diseño y planificación de la sesión de medida

El esquema global de las operaciones implicadas en esta fase aparece en la figura 7.6.

Una vez definidos los objetivos del estudio, detectados los posibles problemas que hay que resolver para alcanzar estos objetivos y seleccionadas las herramientas adecuadas, el paso siguiente será planificar de manera cuidadosa el experimento de medida antes de llevarlo a cabo.

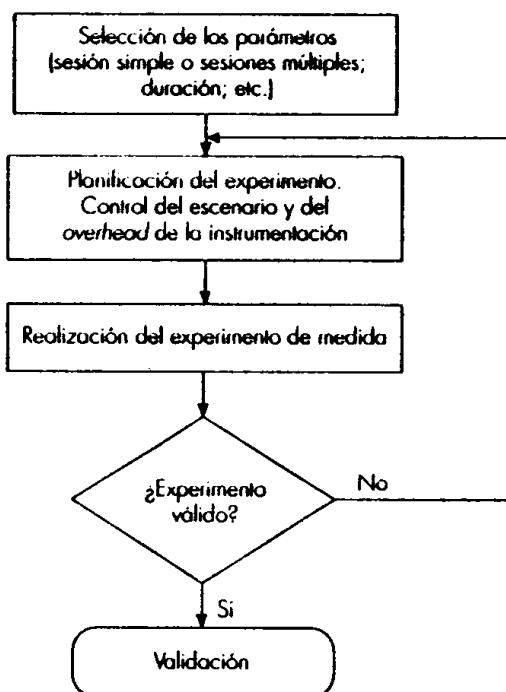


FIGURA 7.6.

Para poder planificar una sesión de medida debe conocerse, al menos someramente, la carga y su evolución en el tiempo. Las medidas deben llevarse a cabo en un entorno controlado con carga conocida y no se deben hacerse modificaciones en el hardware ni cambios en los procedimientos habituales de operación durante su realización.

Conocer la carga implica tener información acerca de sus características en las distintas franjas horarias y de sus variaciones a lo largo de cada día, semana o mes. Se deberá saber, al menos de manera aproximada, cuándo se producen los mayores incrementos de carga (picos de carga) para poder tenerlo en cuenta a la hora de planificar la sesión. Las sesiones suelen incluir medidas correspondientes a, precisamente, esos períodos de incremento de carga. Por ejemplo, un sistema en el que los mayores tiempos de respuesta se producen durante la mañana,

no tiene sentido realizar las medidas por la tarde si se desea obtener información para poder mejorar esos tiempos.

Cuando se planifican medidas con un monitor software, hay que tener en cuenta que este monitor es un programa, por lo que tendrá que residir, total o parcialmente, en la memoria principal para poder ejecutarlo. Un monitor software puede ser o un programa de aplicación o formar parte del sistema operativo. En cualquier caso, quedará menos memoria disponible para el usuario cuando esté activa la herramienta de medida.

La demanda habitual de memoria de un monitor software suele ser reducida (del orden de unos pocos Kbytes). Pero si el sistema operativo subdivide la memoria en un número fijo n de particiones, al menos una de ellas tendrá que estar dedicada al monitor, a no ser que forme parte del espacio de memoria del mismo sistema operativo. En el primer caso, habrá una partición menos y, por tanto, se podrán planificar como máximo $n - 1$ ejecuciones de programas durante la sesión de medida, y en el segundo, habrá al menos una pequeña reducción de la memoria disponible para los usuarios. En entornos de memoria virtual este problema suele ser mucho menos importante. Por el contrario, el problema del *overhead* de CPU que causa el monitor cuando está activo suele ser, en casi todos los sistemas, un problema bastante más delicado.

Si se selecciona de manera adecuada el período de muestreo, se podrá mantener prácticamente constante el consumo total de tiempo de CPU por parte de la herramienta. Esto es debido a que el nivel de confianza que proporciona un monitor software de muestreo depende más del número de elementos de la muestra que de la frecuencia de muestreo. Por tanto, habrá que seleccionar la longitud de la sesión de medida (y por tanto, la frecuencia de muestreo) de manera que, dado un nivel de confianza, la interferencia con el sistema sea la mínima posible.

Los aspectos que hay que tener en cuenta cuando se prepara una sesión de medida con una herramienta hardware son quizás más complejos que los que surgen con una herramienta software, aunque presentan la ventaja de no tener que considerar el problema de la minimización de la interferencia. Es necesario conocer la lista de las direcciones de conexión a los sensores de los distintos dispositivos hardware. En la práctica, es muy fácil que se produzca una conexión incorrecta a los sensores, por lo que es imprescindible que esa lista de direcciones sea correcta y completa y que, una vez realizada la conexión, ésta se verifique para disminuir sus errores.

Otros problemas que hay que tener en cuenta son los siguientes:

- La longitud de los cables: Pueden aparecer señales retardadas debido a la diferente longitud de los cables de las mismas.
- La selección de las señales: A menudo se dispone de más de una señal para detectar el mismo evento o medir la misma cantidad.

- En distintos modelos del mismo componente hardware se pueden tener señales distintas que representan la misma variable. Es muy fácil cometer errores si se cambia el modelo sin verificar, de nuevo, la validez de las señales.

Es habitual que las herramientas hardware permanezcan conectadas al sistema durante largos períodos de tiempo. De ahí la importancia de una cuidadosa verificación de los resultados.

7.6. Validación

El esquema global de las operaciones implicadas en esta fase aparece en la figura 7.7.

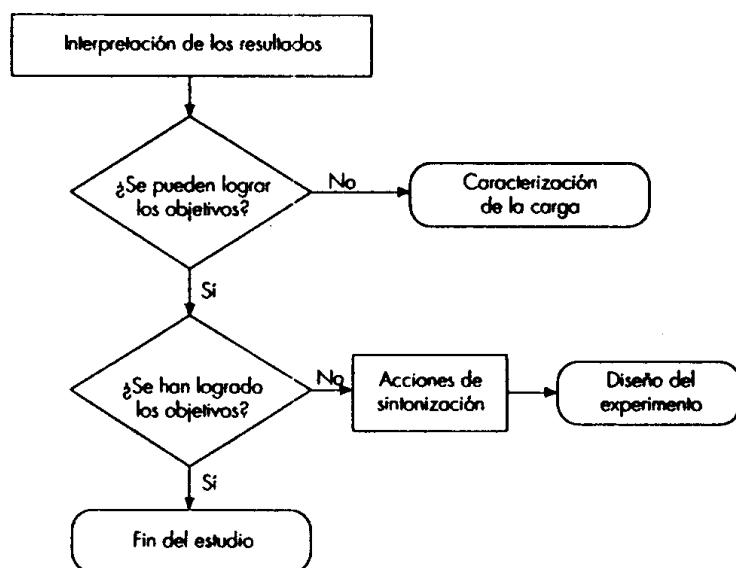


FIGURA 7.7.

7.7. Estudios de sintonización

En los ejemplos planteados en los apartados 7.7.1 y 7.7.2 se siguen los siguientes pasos:

- Planteamiento del problema:* Se plantean los síntomas externos de inefficiencia que han motivado el estudio, así como los objetivos que se pretenden alcanzar.
- Medidas e interpretaciones.*
- Acciones emprendidas y verificación de sus efectos.*

7.7.1. Equilibrar un sistema multiprogramado

El primer paso para mejorar las prestaciones de un sistema multiprogramado es determinar la existencia y localización de cuellos de botella. Después de la fase de diagnóstico, se emprende la terapia, que consiste en la eliminación o reducción de los cuellos de botella. A menudo se revela la existencia de un cuello de botella por los desequilibrios en la actividad de los componentes del sistema. Pero esto no siempre es cierto, dado que un sistema puede estar relativamente bien equilibrado y presentar aún cuellos de botella. Por otra parte, se ha comprobado que un sistema perfectamente equilibrado no tiene por qué presentar las mejores prestaciones posibles.

Se considera, por ejemplo, el caso de un sistema con dos unidades de disco con unos tiempos medios de servicio de 10 y 20 ms, respectivamente. Si la tasa total de peticiones para los dos discos es de 45 peticiones por segundo, parece razonable dividir la tasa de manera que las utilizaciones de los dos discos resulten idénticas. Esto se consigue enviando 30 peticiones por segundo al disco más rápido y 15 al lento. Si se supone que los tiempos entre llegadas y los tiempos de servicio se distribuyen exponencialmente, se obtienen unos tiempos medios de respuesta para cada disco de:

$$\bar{R}_i = \frac{S_i}{1 - \lambda_i S_i}$$

donde S_i es el tiempo medio de servicio y λ_i la tasa de llegada al dispositivo i . Según esta expresión, se obtiene un tiempo medio de respuesta de 14.3 ms para el disco rápido y 28.6 ms para el lento. El tiempo medio de respuesta del subsistema formado por los dos discos será:

$$\frac{\lambda_1}{\lambda} R_1 + \frac{\lambda_2}{\lambda} R_2 = 19 \text{ ms}$$

Pero estas prestaciones, que se corresponden con una situación en la que los dos discos presentan una utilización del 30 por 100 y carga equilibrada, pueden ser mejoradas. Por ejemplo, si se envían al disco más rápido 38 peticiones por segundo y sólo 7 por segundo al lento, se obtiene un tiempo medio de respuesta de 17.2 ms. Se ha obtenido una mejora de alrededor de un 10 por 100 con respecto a la situación de carga equilibrada.

Por tanto, se obtienen mejores prestaciones si los componentes más rápidos (a excepción de la CPU) están más cargados que los lentos. El estado de equilibrio del sistema puede dar idea de la posible existencia de cuellos de botella, pero no de su ausencia, ya que no puede ser considerado como un indicador preciso de su "proximidad" al punto de máximas prestaciones.

Cuando un diagnóstico revela que el sistema está desequilibrado, o que, al menos, es mejorable, es necesario emprender una fase de terapia una vez que se

haya evaluado la conveniencia de las acciones encaminadas a la mejora de prestaciones.

Ejemplo 1

Cuellos de botella en discos y canales

— Planteamiento del problema

Se considera un sistema multiusuario *batch* como el de la figura 7.8.

La memoria del sistema *M* está subdividida en siete particiones; cinco de ellas se dedican a la ejecución de los programas que se van a evaluar, una la utiliza el sistema y otra se reserva para programas de elevada prioridad. El 80 % de la carga lo constituyen programas administrativos y el 20 % restante, programas de aplicación científica.

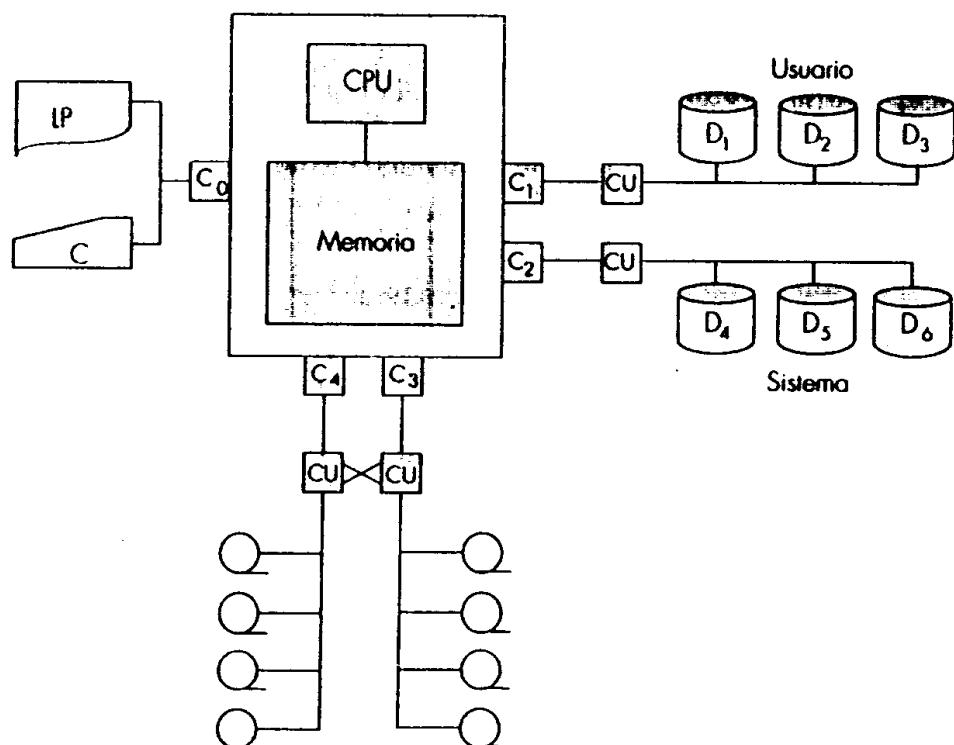


FIGURA 7.8.

Las unidades de disco conectadas al canal C_1 contienen áreas de trabajo, bibliotecas y archivos de usuario. Los discos del canal C_2 se dedican al sistema operativo y contienen el sistema, sus bibliotecas y sus espacios de trabajo.

El objetivo general del estudio es ver si los parámetros de generación del sistema son los adecuados y obtener información acerca de la utilización de los componentes para plantear posibles reposiciones.

— *Las medidas y su interpretación.*

Las medidas se tomaron con un monitor software de muestreo, y las sesiones de medida, de una duración media de tres horas, tuvieron lugar:

- Durante períodos en los cuales se había observado una disminución apreciable de la productividad.
- Durante períodos de elevada carga del sistema, lo cual solía ocurrir en días laborables de 10 de la mañana a 4 de la tarde.
- Después de cada modificación de la configuración del sistema y de los parámetros de generación del mismo. Las medidas iniciales llevaron a la obtención de los resultados que aparecen en las figuras 7.9 y 7.10.

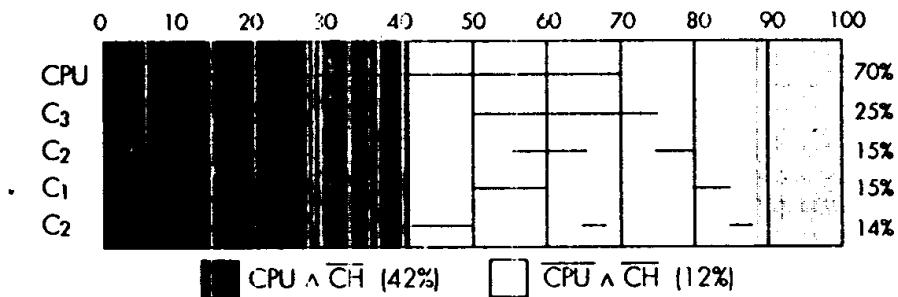


FIGURA 7.9.

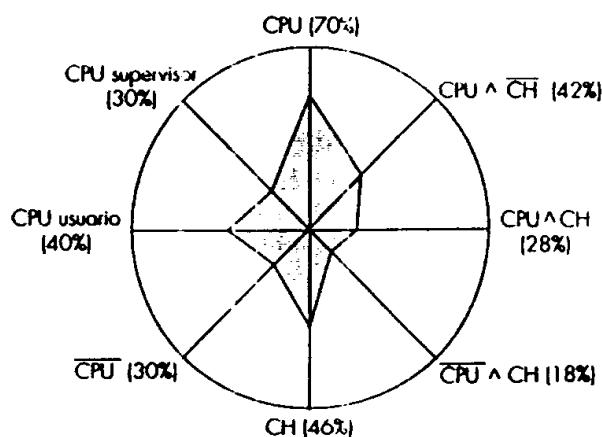


FIGURA 7.10.

La utilización media de la CPU es de un 70 %, con oscilaciones a lo largo del día entre el 95 y el 60 por 100. Un 30 % del tiempo, como media, se dedica a funciones de supervisor. El análisis de la actividad de los canales muestra que mientras la carga en los canales C_1 y C_2 (canales a los que están conectados los discos) está equilibrada, la carga en C_3 y C_4 (canales responsables de las unidades de cinta) no lo está. La figura 7.11 muestra las utilizaciones de C_3 y C_4 durante una punta de carga del sistema.

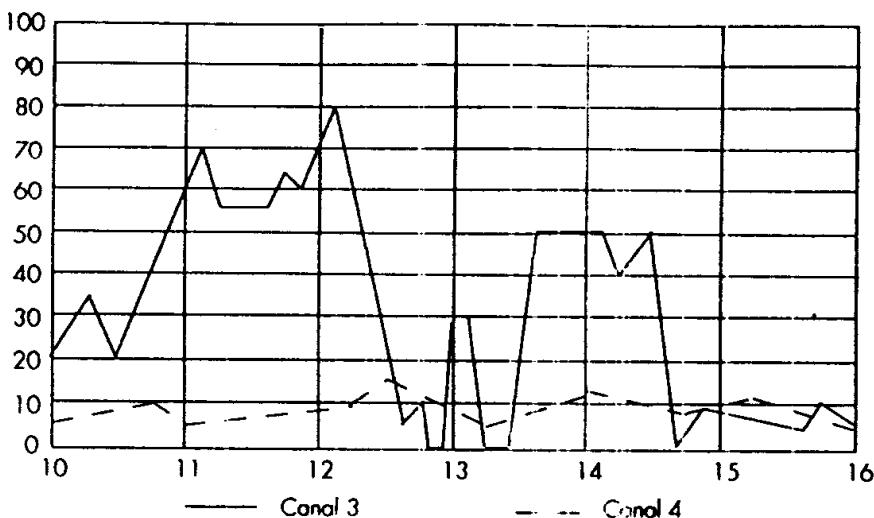


FIGURA 7.11.

De todos los discos, el que presenta mayor utilización es el D_4 , atendido por C_2 , y que almacena las librerías del sistema. Un estudio más detallado de la utilización de sus contenidos muestra que la librería de los módulos del supervisor es la más solicitada. Algunos de los módulos de supervisor más utilizados se cargan, además, en disco cada vez que se requieren, en vez de mantenerlos en memoria principal. Por tanto, sería conveniente revisar la lista de los módulos residentes.

— *Acciones que hay que emprender y sus efectos*

Basándose en los datos medidos sobre el sistema, se plantearon las siguientes acciones para lograr una mayor disponibilidad de la CPU para los programas de usuario:

1. La reducción de la carga en D_4 , optimizando, por una parte, la lista de módulos de supervisor residentes en memoria y por otra la ubicación de los archivos en el disco.
2. Equilibrar las cargas en C_3 y C_4 .

La revisión de la lista de módulos de supervisor residentes a partir de la frecuencia de acceso produce una reducción considerable de la carga en disco. La tasa de

acceso a la librería correspondiente se vio reducida en un 70 %. La figura 7.12 muestra la utilización de la CPU por parte de los programas de usuario y de supervisor antes y después de la revisión de la lista de módulos residentes.

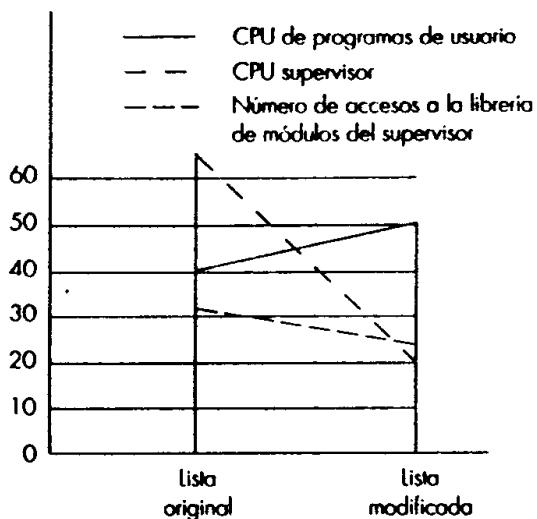


FIGURA 7.12.

La mejora en la ubicación de los archivos en el disco D_4 se consigue a partir de los datos acerca del movimiento del cabezal, los cuales permiten inferir la secuencia de acceso. A partir de esta secuencia se intentará minimizar la distancia entre los archivos más utilizados.

Una reducción en el movimiento del cabezal reduce también el tiempo de CPU desocupada.

Como consecuencia de las acciones emprendidas, la utilización de la CPU debida a programas de usuario se vio incrementada en un 17 %. Un 8 % de este incremento se debió a la reducción de la frecuencia de acceso a la librería y un 9 % a la reducción de los tiempos de espera en D_4 y al equilibrio de carga en C_3 y C_4 . La utilización global de la CPU se incrementó en un 8 % y la variable cualquier canal ocupado (CH) se redujo a un 40 % (de un 46 %).

Este aumento en la utilización de la CPU por parte de los programas de usuario mejoró la productividad del sistema en un 10 % aproximadamente.

La figura 7.13 muestra el gráfico de Kiviat del sistema después de las acciones de sintonización.

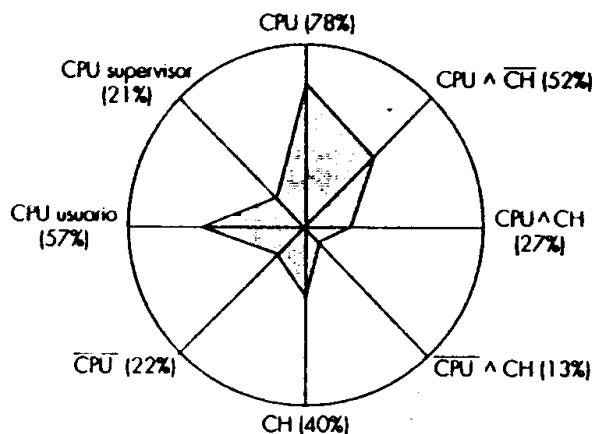


FIGURA 7.13.

Ejemplo 2

Memoria insuficiente

— *Planteamiento del problema*

El sistema de la figura 7.14 presenta dispositivos de almacenamiento de diferente velocidad. Un disco rápido en el canal C_1 , discos de mayor tiempo de acceso en los canales C_2 y C_3 , y unidades de cinta en C_4 .

La productividad del sistema resultaba insuficiente y había muchos trabajos en espera de conseguir espacio en memoria. A menudo el sistema se mostraba incapaz de procesar su carga diaria en el mismo día y programas largos solían procesarse incluso a lo largo de varios días.

Por otra parte, la capacidad del sistema era tal que era muy improbable una saturación del mismo con el nivel de carga actual. Por tanto, el propósito del estudio consiste en determinar la posible existencia de un cuello de botella o ver si había que rediseñar la configuración del sistema para adaptarlo a la carga actual.

— *Las medidas y su interpretación*

En este estudio se utilizaron las medidas recogidas por las rutinas de contabilidad. Estas rutinas, que forman parte del sistema, funcionan como un monitor software conducido por eventos con un coste prácticamente imposible de suprimir en el funcionamiento normal de cualquier sistema. Se utilizó una carga de test para que así se pudieran comparar de una manera más sencilla los resultados obtenidos después de las acciones de sintonización.

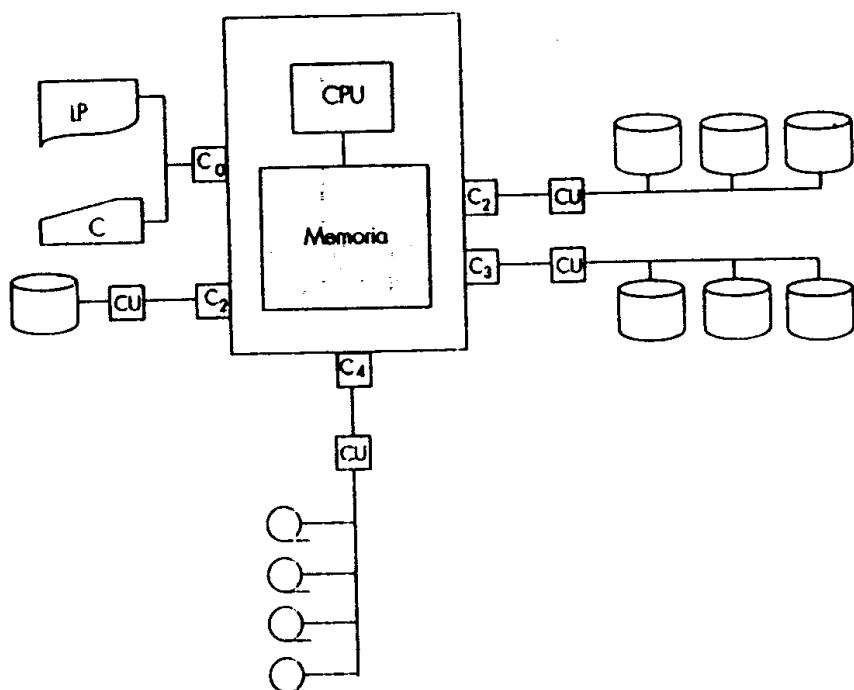


FIGURA 7.14.

La carga de test estaba constituida por programas de la carga real escogidos según un criterio de representatividad. Se trataba, por tanto, de una carga de test sintética natural con características similares a las de la carga real. De las medidas tomadas durante la ejecución de la carga de test se obtuvieron las utilizaciones de la CPU y de los canales. Las rutinas de contabilidad son un tipo de herramienta de medida que raramente suele proporcionar información acerca del grado de solapamiento entre los dispositivos. En la tabla siguiente se han resumido los resultados antes y después de la sintonización.

Componente	Utilización antes	Utilización después
CPU	58	76
Canal 1	27	40
Canal 2	25	37
Canal 3	45	62
Canal 4	25	35

El nivel de utilización indica el porcentaje de tiempo de ejecución de la carga de test antes y después de las acciones de sintonización que se comentarán más adelante.

No se puede deducir de estas medidas que exista un dispositivo causante de un cuello de botella, ya que todas las utilizaciones son bastante bajas. Basándose en estos datos, se formularon las siguientes hipótesis:

1. La carga no es lo suficientemente grande para poder cargar de forma apreciable los componentes del sistema.
2. La planificación de los recursos o de las colas de entrada no es eficiente (cuello de botella causado por el planificador).
3. No hay suficiente memoria (el factor de multiprogramación es demasiado bajo para la capacidad del sistema).

La primera hipótesis quedaba descartada tanto para la carga de test como para la real porque ambas suponían un nivel de carga considerable. La segunda hipótesis también se podía descartar porque en el mismo tipo de sistema y en instalaciones muy similares se habían observado niveles de utilización muy superiores. Por tanto, sólo quedaba profundizar en el estudio de la tercera hipótesis.

— *Acciones que hay que emprender y sus efectos*

Se dobló el espacio de memoria disponible para los usuarios, obteniéndose los niveles de utilización que se observan en la segunda columna de la tabla. Estos datos demuestran la corrección de la tercera hipótesis: el cuello de botella era causado por un espacio de memoria insuficiente.

Con un nivel de multiprogramación mayor, se redujo el tiempo de ejecución de la carga de test a un 75 % con respecto al obtenido antes de incrementar el tamaño de la memoria. La productividad, que es proporcional a la inversa de este tiempo, se incrementó en un 33 %.

7.7.2. Sintonización de un sistema interactivo

La carga en un sistema interactivo suele estar compuesta por dos o tres subcargas de diferentes características. Junto a la carga generada por los usuarios en línea (*on-line*), locales o remotos, suele haber también carga *batch* (o *background*). Se trata, por tanto, más que de sistemas puramente interactivos, de sistemas mixtos o híbridos.

Los usuarios en línea, a su vez, pueden dividirse en dos categorías: aquellos que son atendidos siguiendo una técnica de tiempo compartido (*time-sharing*) y las órdenes o interacciones atendidas en tiempo real.

La aplicación principal de la primera clase de usuarios es el desarrollo y ejecución interactivo de programas. El usuario puede verificar interactivamente, incluso sentencia a sentencia, la existencia de errores sintácticos en su programa, y analizar en su terminal los resultados de su ejecución, evitando así abundantes listados y reduciendo de manera apreciable los tiempos de espera. El tiempo medio de residencia se reduce considerablemente y la productividad de los programadores aumenta con respecto a la que se obtenía en un entorno *batch*.

Las órdenes utilizadas por este tipo de usuario suelen ser generalmente de tipo "ligero" (inserción y borrado de texto, sustitución de caracteres, impresión de listados de programas, etc.). Por tanto, este tipo de carga puede caracterizarse por un considerable consumo de E/S junto con una escasa demanda de CPU.

Por otra parte, hay que mencionar los usuarios que interaccionan con el sistema para ejecutar uno o más programas, o bien responden a las peticiones de estos programas durante su ejecución. La carga que genera este tipo de usuario es muy variable. Estas aplicaciones suelen requerir la introducción de datos por parte del usuario, desde un terminal local o remoto, para procesarlos posteriormente. Este tipo de carga suele estar caracterizado por un gran número de operaciones de E/S y una demanda variable de CPU en función de la complejidad de los cálculos.

Para poder atender la demanda de estos diferentes tipos de usuarios, en ocasiones se suele acudir, especialmente en grandes instalaciones, a sistemas "dobles" (*twins*) en los que se cuenta con un sistema dedicado a los usuarios interactivos y con otro para ejecutar programas *batch*.

En principio, esta solución se muestra capaz de satisfacer los requerimientos de los usuarios en términos de tiempos de respuesta y de continuidad del servicio, pero, por otra parte, suele tener un coste muy alto si se compara con la cantidad de trabajo realizado. Para que en un sistema interactivo, durante un incremento temporal de carga, los tiempos de respuesta no sobrepasen unos valores límite, se suele sobredimensionar el sistema, incrementando por tanto su coste que ya de por sí acostumbra a ser alto.

También es posible emplear para los tipos de carga descritos, un único sistema, con una o más CPUs. En este caso será más difícil equilibrar el sistema ya que, además de un elevado número de componentes, se tiene el factor añadido de la coexistencia de diferentes tipos de carga.

De esta forma, por ejemplo, un incremento de la carga *batch* puede degradar los tiempos de respuesta interactivos hasta límites no tolerables, mientras que, por otra parte, si se incrementa el número de usuarios conectados, puede empeorar de manera considerable el tiempo de presencia en el sistema de los trabajos *batch*.

En este tipo de sistemas, al tener un elevado número de programas en ejecución, será más difícil determinar la contribución de los distintos componentes de la carga a la utilización de los componentes del sistema. Una vez fijados unos límites al tiempo de respuesta y de presencia, habrá que seleccionar de manera adecuada las políticas de planificación que se emplean para cada tipo de carga. Esta selección será más compleja debido a esta coexistencia de las cargas.

Ejemplo 3

Canales sobrecargados

— Planteamiento del problema

En este caso se parte de un sistema interactivo multiprogramado que responde al esquema global de la figura 7.15.

Se trata de un sistema multiprocesador con dos CPUs que comparten los periféricos de almacenamiento secundario. El sistema operativo reside en los discos conectados al canal C_2 . En el canal C_1 están conectados dos tipos de discos, unos comparativamente más rápidos que otros. La carga está compuesta por programas procedentes de usuarios interactivos, que se ejecutan en tiempo compartido, y por programas batch.

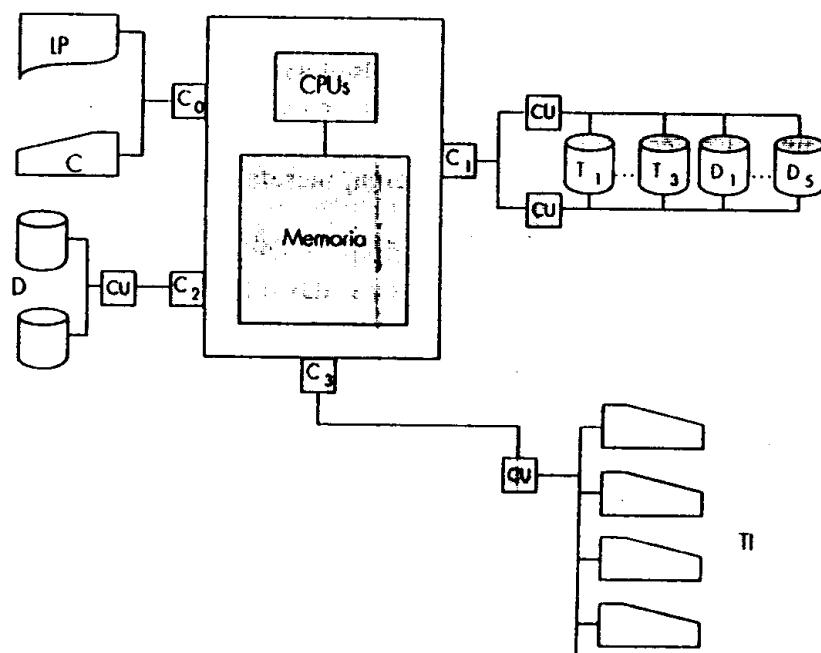


FIGURA 7.15.

En algún momento, los principales índices de prestaciones del sistema, tiempos de respuesta y de presencia, experimentaron un empeoramiento progresivo hasta alcanzar límites no tolerables. Por tanto, durante períodos de mucha carga, un usuario interactivo podía estar de 1 a 2 minutos esperando por una orden "ligera", y un usuario batch, con un programa de tamaño mediano de tiempo de eje-

cución inferior a 5 minutos, podía tener que esperar de 3 a 4 horas para obtener un listado de su ejecución.

Por todo ello, se decidió proceder a un análisis del sistema para establecer si se podía mejorar la eficiencia del mismo eliminando cuellos de botella, o si por el contrario era imprescindible acudir a terapias de reposición. Si esto último fuera necesario, habría que indicar también qué componentes debían añadirse o sustituirse en el sistema.

— *Las medidas y su interpretación*

Se observó la actividad del sistema durante períodos de elevada carga en los cuales el tiempo de respuesta era particularmente largo. El intervalo de medida tuvo una duración total de 1h y 11 minutos (4260 s). Se empleó para la obtención de las medidas un monitor software conducido por eventos, que presentó un *overhead* de sólo 104 s, es decir de un 2.1 % de la duración de la sesión de medida.

Mientras que la utilización del canal C_2 parecía normal, no ocurría lo mismo con la de los canales C_1' y C_1'' . Por ello se midió la utilización de las unidades conectadas a C_1' y C_1'' . T_1 , T_2 y T_3 son los discos más rápidos y es donde el sistema aloja los archivos temporales (para el programa en ejecución, para el compilador, para *swapping*, etc.), las librerías del programa, los compiladores más utilizados, los archivos de programas de utilidades y otros archivos que el sistema necesita para tratar los usuarios interactivos. Los discos más lentos, D_1 , D_2 , D_3 , D_4 y D_5 , se dedican a los usuarios y contienen bases de datos, áreas de trabajo privadas, etc.

El sistema envía sus peticiones de E/S a los periféricos prioritariamente a través del canal C_1' (que es el canal principal). Si está ocupado, es decir, si ya está transfiriendo datos, la CPU envía las peticiones al canal C_1'' , canal que se considera secundario.

A partir de las medidas se observó una utilización elevada del canal C_1' , cercana al 70 % (una media de un 69 %; 2942 s), límite de utilización que se ha demostrado con la experiencia que no es recomendable rebasar. El canal C_1'' , por el contrario, presentaba una utilización baja (una media de un 33 %; unos 1390 s).

Un estudio de mayor profundidad de las operaciones de E/S en los dos canales antes mencionados reveló la existencia de un cuello de botella debido a la competencia de las peticiones de acceso a T_1 , T_2 y T_3 . De hecho, de 28 peticiones/s, 25.9 (un 92 %) eran peticiones a las unidades T_1 , T_2 y T_3 , y sólo un 2.1 % eran peticiones a D_1 , D_2 , D_3 , D_4 y D_5 . La siguiente tabla muestra las utilizaciones de cada unidad.

Unidad	Accesos/s	Peticiones en cola
T_1	15.3	85%
T_2	3.4	89%
T_3	6.9	90%
T_{m}	25.6	—
		(.../...)

Unidad	Accesos/s	Peticiones en cola
D_1	0.5	—
D_2	0.1	—
D_3	0.4	—
D_4	0.7	—
D_5	0.4	—
D_{tot}	2.1	—
Total	27.7	—

En esta tabla se puede observar que los porcentajes de peticiones a las unidades T_1 , T_2 y T_3 , que tienen que esperar, eran muy elevados, un 85, 89 y 90 % respectivamente. El 28 % de las peticiones que tienen que esperar son, a su vez, colocadas en colas que ya contienen al menos tres peticiones.

Una petición se pone en cola por una de las siguientes razones:

- Canales ocupados y unidades libres: 16 %.
- Canales y unidades ocupadas: 43 %.
- Canales libres y unidades ocupadas: 41 %.

El 84 % de las peticiones eran, por tanto, puestas en cola porque la unidad estaba ocupada.

La espera en cola de las peticiones produce un incremento del tiempo de respuesta y de presencia, ya que incrementa el tiempo medio de acceso a T_1 , T_2 y T_3 . Además, el manejo de esas colas de espera causa un *overhead* del sistema que añade un retardo aún mayor.

Un análisis de las causas de tal cantidad de actividad de E/S en C_1' y C_1'' reveló que el sistema tenía un nivel de multiprogramación demasiado alto. En la siguiente tabla se puede ver el número medio de programas en ejecución, de programas cargados en memoria y de programas en espera de ejecución.

	Batch	Interactivo	Total
Programas en ejecución	15.1	29.8	44.9
Programas en memoria principal	6.5	15.6	22.1
Programas en espera de ejecución	43.2	—	48.2

Como se puede observar en la tabla, que sólo la mitad de los programas en ejecución están también cargados en memoria. Esto produce un elevado tráfico entre memoria principal y secundaria (en este caso el disco T_1), lo que se denomina *swapping*. El número medio de usuarios interactivos con programas en ejecución (30) es demasiado elevado. Como los programas interactivos tienen una prioridad de ejecución mayor que los *batch*, las prestaciones para éstos se ven muy

degradadas al ser este tipo de programas frecuentemente interrumpidos en su ejecución. Por otra parte, la longitud de la cola de programas en espera de ejecución demuestra que el sistema no es adecuado para la carga que tiene que procesar.

— *Acciones que hay que emprender y sus efectos*

Una acción posible para reducir la actividad de *swapping* consiste en incrementar el tamaño de la memoria principal. Como el nivel de utilización de la CPU puede incrementarse, se podría tener así un mayor número de programas simultáneamente en memoria, reduciendo el tráfico entre memoria principal y secundaria. Esta solución reduciría la carga en el disco T_1 (disco de *swapping*), pero no eliminaría el cuello de botella debido a T_2 y T_3 . Es probable, incluso, que empeore el problema.

Por tanto, se debe examinar la posibilidad de reducir las cargas en las unidades con las mayores colas de espera. Después de analizar cada uno de los archivos en T_1 , T_2 y T_3 , se decidió expandir el subsistema C_1 , como se ve en la figura 7.16. Ahora se contará con cuatro discos rápidos y ocho más lentos disponibles para los usuarios.

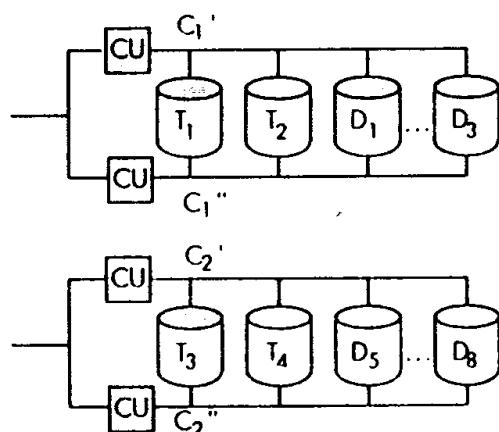


FIGURA 7.16

A continuación se investigó la causa de la elevada utilización de los terminales. Se vio que aproximadamente un 10 % de los usuarios interactivos utilizaban estos dispositivos únicamente para ejecutar sus programas con una prioridad mayor que los usuarios *batch*. Por todo ello, se decidió reducir el número de terminales.

Como consecuencia de estas acciones, las prestaciones del sistema mejoraron considerablemente. El tiempo de respuesta máximo para órdenes ligeras pasó a 30 s y el tiempo de presencia se redujo en media de un 25 %.

8. Explotación de sistemas informáticos

8.1. Introducción

Está generalmente aceptado que es necesaria una cierta metodología para el diseño de aplicaciones informáticas, pero se tiende a olvidar que el éxito de la aplicación reside en que funcione correctamente en su etapa de explotación (y, por lo tanto, en competencia por el uso de recursos con las otras aplicaciones existentes).

La organización de la función de explotación del ordenador debe basarse en una determinada metodología de trabajo, pues, de lo contrario, la organización será consecuencia de los usos y costumbres arraigadas en la empresa en que se desenvuelve la Informática, siendo causa de frecuentes fricciones y de utilización ineficiente de los recursos.

Se comentará, en primer lugar, los puntos de coincidencia entre las funciones de explotación en Informática con la funciones de producción de una empresa industrial.

Si se considera la explotación del ordenador como una fábrica en la que entra información que, tras pasar por diversas manipulaciones, es transformada finalmente en otra clase de información (listado), que es el producto final de dicha elaboración, se puede establecer la siguiente analogía:

- *Información-Material*: La información llega en un cierto estado y se va transformando en diversos pasos de trabajo.
- *Programas-Matrices*: Al igual que una matriz proporciona una forma nueva a la materia, un programa genera un nuevo estado en la información que le llega.
- *Archivos-Stocks*: La información entre fases de trabajo se almacena en archivos temporales, durante la realización del trabajo, o de larga dura-

- ción, al igual que se almacena el material en curso de fabricación o los productos acabados.
- *Grabadoras-Fundición*: Los datos de base son traspasados a un soporte adecuado para su tratamiento, al igual que los metales se convierten en barras, planchas y otras formas susceptibles de ser mecanizadas.
 - *Ordenador-Taller*: En monoprogramación, actuación con una sola máquina; en multiprogramación, efectuando varios trabajos como si fueran máquinas diferentes.
 - *Acabados-Embalaje*: Los listados, una vez producidos, son separados, cortados, encuadrados y preparados para su envío, al igual que los productos acabados se embalan y envían.
 - *Hoja de manipulación-Hoja de ruta*: Al igual que la hoja de ruta indica en un taller la secuencia de operaciones que hay que realizar para producir un artículo, la hoja de manipulaciones indica el conjunto de pasos (programas) que es preciso ejecutar.

Esta analogía conduce a las siguientes conclusiones de orden metodológico:

- El operador debe actuar como el encargado de una o más máquinas, gestionando la cola de trabajos encomendada y vigilando la buena marcha del equipo.
- Debe existir, como en la oficina de producción, una función de lanzamiento y control que, de acuerdo con la planificación de trabajos, prepara y pone en cola de operación los trabajos que hay que realizar y controla el seguimiento de su realización.
- Debe existir, por lo tanto, y análogamente, una función de planificación que, de acuerdo con la demanda de información en un cierto plazo, confeccione la planificación correspondiente.
- Los trabajos realizados, al igual que los artículos fabricados, requieren una verificación o control de calidad que mantenga a ésta en los límites aceptados.

Existen también trabajos de captura de datos y de acabados que, al igual que la recepción, embalaje y expedición, exigen tratamientos aparte.

Esta organización es típica de los sistemas de producción de artículos de catálogo, es decir, de artículos de características preestablecidas (cadenas de explotación) y que se producen principalmente en serie, es decir, de forma repetitiva (al final de mes, la nómina y la contabilidad; cada semana, el almacén, etc.). Y todo esto aunque existan varios trabajos sobre pedido que deban compaginarse con los anteriores (un listado especial, compilaciones, etc.).

La fuerte semejanza de un sistema productivo de estas características con la explotación de aplicaciones informáticas ha conducido, desde sus inicios, a estructurar los departamentos de explotación con organizaciones funcionales análogas sobre todo en centros de proceso de datos importantes. No se considerarán aquí ni los pequeños departamentos de explotación cuyo símil sería más el de un taller

(predominio de lo artesanal) que el de una fábrica, ni los ordenadores dedicados a aplicar su potencia de cálculo a la resolución de problemas técnicos o científicos.

Sin embargo, la evolución de la Informática en los últimos años ha seguido un rumbo que debilita las semejanzas antes expuestas. En efecto, en la situación descrita anteriormente el producto final es un listado que se envía a un usuario y el departamento de explotación es responsable de los procesos que hacen posible su confección. En los ordenadores primitivos el operador era el auténtico artífice de que los trabajos llegaran a buen término. Los sistemas operativos en multiprogramación han restado protagonismo al operador, pero su responsabilidad continúa siendo fundamental en el proceso.

Posteriormente, dos fenómenos cambian el panorama. El primero es la utilización en gran escala del teleproceso, en el que el usuario solicita y recibe información sin que intervenga directamente el departamento de explotación. El segundo es un cierto cambio en el diseño de aplicaciones: se distinguen claramente dos tipos de procesos, los procesos encaminados a actualizar la información almacenada y los procesos dedicados a obtener información, con mayor o menor grado de elaboración, a partir de la información almacenada (esto último, principalmente, en entornos con teleproceso).

El enfoque deriva hacia la información. La imagen del operador artesano, que a partir de tarjetas obtiene listados con ayuda de ciertos archivos maestros, pasa a ser la del que asegura la correcta custodia de los datos almacenados y garantiza el buen servicio para la obtención de dicha información, obtención que ya no dirige ni controla él mismo.

El producto ya no es tanto un conjunto de listados sino, principalmente, un servicio de custodia de datos y de disponibilidad del equipo. Estas circunstancias hacen que el símil de la Explotación como Fabricación vaya derivando hacia el de Explotación como Servicio.

El artífice de la obtención de información es ahora el usuario (mediante terminales, utilizando lenguajes de alto nivel, etc.). El operador tiene su responsabilidad orientada hacia la vigilancia de la disponibilidad del equipo. La informática distribuida, en cuanto tiene de acercamiento de los medios informáticos al usuario, contribuye aún más a acelerar este enfoque.

Evidentemente, coexistirán en casi todas las instalaciones ambos enfoques. Esto implica que, además de las funciones orientadas hacia la producción, aparezcan una serie de funciones orientadas hacia la atención al usuario (cliente del servicio); pero no en cuanto a la situación de sus trabajos sino en cuanto a la situación de las disponibilidades de los equipos (terminales, paros, mantenimiento, etc.).

8.2. Metodología de explotación

La forma de estructurar la organización y funcionamiento de la explotación del ordenador, que se concreta en una serie de normas de actuación, constituye

un *modelo de explotación* que debe responder a los objetivos y enfoques con que se desea abordar esta problemática. Este modelo será diferente en cada caso, puesto que las circunstancias de cada empresa son distintas y también lo son los objetivos y restricciones del entorno.

La determinación de una *metodología de explotación* consistirá, pues, en diseñar el modelo más acorde con las necesidades expresadas. Tal modelo es notablemente complejo, por lo que es conveniente analizarlo bajo diferentes puntos de vista que se denominarán dimensiones del modelo o simplemente, modelos específicos. A continuación se describen estos modelos específicos.

8.2.1. Modelo decisional

Una dimensión del modelo de explotación es la estructura de las decisiones y funciones que deben realizarse en los diversos niveles para que puedan controlarse efectivamente las operaciones de explotación.

Bajo este punto de vista, y en lo que respecta a las operaciones de explotación relativas a los trabajos *batch* encomendados, es muy útil, siguiendo el símil de fabricación, distinguir tres niveles:

- Planificación
- Lanzamiento
- Control

cuya relación queda de manifiesto en la figura 8.1.

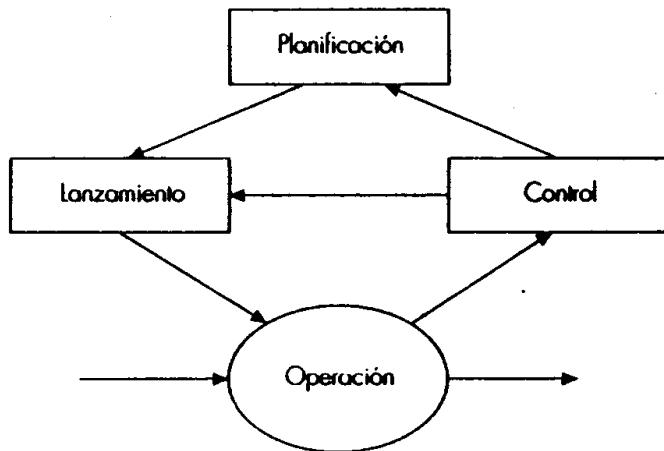


FIGURA 8.1.

La planificación a largo plazo comprende las decisiones de previsión de cargas y necesidades de capacidad de acuerdo con las exigencias de los planes de desarrollo. Normalmente se realizarán, o al menos se revisarán, anualmente en el marco de la confección de los presupuestos para Informática.

A corto plazo, la planificación comprende la ordenación en el tiempo de los trabajos que se pretende realizar. El plan resultante suele contemplar diversos períodos de temporales:

- plan mensual
- plan semanal
- plan diario

El objetivo fundamental de la planificación es resolver conflictos de capacidad limitada, por una parte, y mantener una equilibrada utilización de los equipos, por otra.

El lanzamiento consiste en la preparación de todos los requisitos necesarios para la ejecución de los trabajos contenidos en el plan diario y en la confección de la orden de explotación de estos trabajos o la colocación directamente de ellos en las colas de entrada del ordenador.

Naturalmente hay que tener en cuenta en los diferentes planes la aparición segura de trabajos urgentes no previstos (como relanzamientos, por ejemplo), a los que deberá asegurárseles también un lugar en la cola de trabajos que hay que realizar. En concreto, y por lo que se refiere al lanzamiento, éste es el momento en el que se deciden los últimos ajustes sobre el plan diario, añadiendo las urgencias, repeticiones y otras incidencias que puedan afectar.

El objetivo en el lanzamiento consiste en asegurar la existencia de todos los requisitos necesarios para ejecutar los trabajos y procurar una mejor distribución de los trabajos en entornos con multiprogramación (y/o multitarea) que mejore el rendimiento del equipo.

El control consiste en verificar los resultados obtenidos y, si procede, realizar, según las normas establecidas, un relanzamiento. Del control se obtienen también una serie de medidas sobre el rendimiento del equipo y características de los trabajos, que son fundamentales para la elaboración de nuevos planes.

El conjunto de planificación, lanzamiento y control garantiza el adecuado flujo de realización de trabajos y el aprovechamiento eficaz del equipo.

8.2.2. Modelo organizacional

Las decisiones y elaboraciones descritas en el punto anterior deben realizarse por personas encuadradas en una organización. Se entiende por modelo organizacional el que muestra la asignación de responsabilidades a los elementos de la organización así como a la propia estructura de esta organización.

En este aspecto pueden darse gran cantidad de soluciones dependientes, no sólo de la complejidad y del volumen de trabajos que se van a realizar sino también de las personas disponibles para constituir esa organización.

8.2.3. Modelo relacional

Otro aspecto importante que hay que tener en cuenta en el establecimiento de una metodología de explotación es la fijación de las relaciones del departamento de explotación con el resto de la organización de la empresa donde está inserto.

La gestión de los recursos de hardware y de software y de las normas de trabajo, que forman el núcleo de la Informática, marca las principales relaciones entre las tres componentes internas del servicio de informática: explotación, sistemas y proyectos.

La prestación de servicios, bien como gestionador de procesos (*usuarios batch*) o de medios (*usuarios de teleproceso y autónomos*), marcan las relaciones con el exterior a la Informática, dentro de la propia empresa. De esta forma deben establecerse normas muy estrictas que definen claramente las relaciones con:

- *Usuarios batch*: Cómo se entregan y obtienen datos.
- *Usuarios teleproceso*: Cómo se coordinan las incidencias y averías.
- *Usuarios autónomos (programadores técnicos)*: Cómo se vigila la utilización del equipo.
- *Desarrollo de aplicaciones*: Cómo se entregan los programas a explotación. Cómo se coordinan las incidencias.
- *Sistemas*: Cómo se coordinan cambios de configuración o de software del sistema. Cómo se establece el seguimiento de incidencias.

8.2.4. Modelo informacional

Las relaciones anteriormente expuestas y las decisiones internas precisan de soportes documentales (vales, solicitudes de explotación, catálogos, etc.), que constituyen otro capítulo importante en el establecimiento de una metodología.

Se pueden distinguir tres grupos de información:

- *Documentación de procedimientos*, correspondiente a la información necesaria para poder explotar los programas, guardar información, controlar resultados, etc., de cada aplicación.
- *Documentación de trabajos*: Cada trabajo que deba ser realizado ha de ir acompañado de la información necesaria para su ejecución y control (parámetros, fecha de entrada, etc.).
- *Documentación de control*: Para la gestión de la explotación es necesario conocer, periódicamente, la situación de catálogos de archivos, estado de los terminales, registro de incidencias, etc.

El primer grupo suele materializarse en forma de una carpeta de explotación que se confecciona al entregar a explotación la aplicación correspondiente, y se actualiza durante toda la vida de la misma. Su contenido debe extenderse a:

- *Ámbito de aplicación*: esquema de cadenas de programas.
- *Normas de lanzamiento*: cómo preparar los trabajos a partir de cada solicitud.
- *Normas de explotación*: cómo operar (hojas de manipulación), cómo controlar la calidad, cómo relanzar el trabajo.
- *Normas de archivo*: cómo guardar archivos resultantes y copias de seguridad.

La documentación de trabajos incluye todas las informaciones para poder seguir los trabajos desde su solicitud hasta su entrega. Por lo tanto incluirá:

- *Solicitud de trabajo*, con las características especiales para cada caso.
- *Orden de trabajo*, que permite, mediante anotaciones sucesivas, conocer la situación del trabajo.
- *Hojas de control*, o resúmenes de la realización del trabajo, avisos, etc. (suele proporcionarlo el propio sistema operativo).
- *Vales de entrega* de los resultados obtenidos.

La documentación de control es muy variada y básicamente está constituida por:

- *Planificación de trabajos*.
- *Situación de librerías*.
- *Relación de incidencias*.
- *Partes de averías*.
- *Etc.*

Desde el punto de vista metodológico debe resaltarse que el sistema de información para la explotación es, en muchos aspectos, mecanizable y que debe procurarse, en lo posible, incrementar el uso de ayudas mecanizadas disminuyendo la, a veces muy cargada, burocracia que se encuentra en grandes instalaciones.

8.2.5. Incidencia sobre el diseño de aplicaciones

La metodología establecida para la explotación tiene una incidencia muy fuerte sobre el diseño tecnológico de las aplicaciones.

a) Utilización de recursos

En los entornos de multiprogramación, debe existir una norma sobre la utilización de recursos. Por ejemplo:

- Número de archivos de cinta ocupados simultáneamente por un programa.
- Espacio en disco ocupable por archivos de un solo trabajo.
- Duración máxima.
- Tamaño máximo de la ocupación de un programa en memoria.

Estas restricciones condicionan fuertemente el diseño de los programas.

b) Seguridad

Las aplicaciones deben diseñarse no sólo para funcionar correctamente según su especificación, sino también para poder resistir frente a los incidentes que puedan presentarse. Es decir, deben prever malos funcionamientos del equipo, fallos humanos de operación, etc., y, por lo tanto, prever la reversibilidad del trabajo (copias de seguridad, relanzamiento, etc.).

De la misma manera los sistemas de seguridad de la información almacenada (cintoteca, copias de seguridad, etc.) obligan a prever la obtención de copias en discos o cintas en diversos puntos de la aplicación.

c) *Relanzamientos*

La forma de realizar los relanzamientos afecta también al diseño de cadenas de explotación limitando, por ejemplo, la duración de los mismos y obligando a prever los puntos de relanzamiento.

Los relanzamientos, cuando las aplicaciones afectan a archivos accesibles por teleproceso, se complican por la necesidad de recurrir a la restauración de las bases de datos en función del estado en que han quedado tras la realización de los trabajos precedentes, y por tanto, obligan a que los programas informen sobre dicho estado.

d) *Seguimiento y control*

Las cadenas y programas deben, además, estar diseñados de forma que sea posible diagnosticar la correcta o incorrecta finalización del trabajo mediante mensajes, estadísticas, etc.

8.2.6. Resumen

En resumen, una vez determinada la metodología de explotación, ésta se materializará en:

- una estructura organizativa
- unas normas de actuación
- unas normas de documentación

8.3. Organización de un departamento de explotación

La organización del departamento de explotación viene determinada por una serie de factores lo suficientemente importantes como para que su conjunción dé origen a su propia estructura.

Algunos de estos factores más relevantes, son los siguientes:

- Tamaño de la instalación.
- Recursos de que dispone el departamento de explotación.
- Características específicas del sistema operativo de la instalación.
- Características específicas del personal existente.
- Situación del departamento de explotación, dentro del centro de proceso de datos de que se trata.
- Situación del centro de proceso de datos, dentro del marco general de la empresa u organización.
- Características propias de esta empresa u organización.

- Plan de Informática a corto y largo plazo de la misma.
- Características básicas de la política informática interna de la empresa: centralización o descentralización informáticas, etc.

Todos estos factores son elementos que acabarán configurando una organización específica.

En este punto se hará un planteamiento general sobre la organización de un departamento de explotación, tratándose únicamente de un intento de síntesis de elementos que suelen aparecer como comunes en distintos departamentos de explotación.

De los planteamientos generales sobre la organización de un departamento de explotación parece deducirse demasiadas veces la necesidad de cubrir una amplia gama de puestos de trabajo distintos para realizar correctamente las tareas afectadas al departamento de explotación. Esto supone implícitamente una amplitud de recursos sólo disponibles en el caso de las grandes instalaciones y, por contrapartida, el que la carencia de tales recursos aconseje a las instalaciones pequeñas y medias olvidarse de realizar un análisis *a priori* de las necesidades y funciones que hay que cubrir por un departamento de explotación como base para establecer una organización racional del mismo.

Todo esto procede de la confusión entre lo que son las tareas funcionales básicas a cubrir por todo departamento de explotación y la posterior concreción de las mismas en puestos de trabajo. Los conocidos como *job description* de la literatura anglosajona pueden reconvertirse fácilmente en las tareas funcionales básicas, dejando para una segunda etapa la concreción de los puestos de trabajo en que se realicen tales tareas funcionales. Por tanto se relega a esta segunda etapa la intervención de los recursos posibles, sin alterar la generalidad del estudio. Es en esta segunda etapa en la que cabe hallar la diferenciación entre grandes y pequeñas instalaciones, centrada principalmente en el número de puestos de trabajo concretos que cubren cada una de las tareas funcionales básicas. Es obvio que, en muchas instalaciones pequeñas o medias, una sola persona deberá desempeñar más de una de estas tareas funcionales básicas, potenciando la necesaria capacitación de las personas que formen parte de este tipo de departamento de explotación por la necesaria multiplicidad de las funciones a ellos encomendadas.

8.3.1. El departamento de explotación como unidad de producción y su organización funcional

La idea fundamental del departamento de explotación reside en ser el soporte central y físico del proceso de datos y tal actividad es el objetivo fundamental del departamento.

Para conseguir este objetivo es necesario realizar y distinguir cuatro grandes funciones:

- Procesar datos.

- Promover y desarrollar procedimientos estándares.
- Promover, investigar y desarrollar nuevas técnicas y materiales de proceso de datos.
- Promover, formar y desarrollar al personal.

El departamento de explotación comparte con el resto de los departamentos informáticos (estudios, análisis y programación, dirección informática, etc.) la mayor parte de las funciones reseñadas, quedando como específica del departamento de explotación el proceso de los datos.

Procesar datos es una función productiva en sus diversos aspectos como producir información, producir impresos, etc. Éste es el punto básico que debe caracterizar a la organización de un departamento de explotación. De forma tradicional, se considera que una organización funcional para cualquier tipo de actividad de producción debe contemplar las siguientes funciones:

- Producción.
- Control de procesos de producción.
- Gestión de stocks (materias primas, productos intermedios, productos acabados).
- Soporte.

Análogamente, un departamento de explotación presenta el siguiente aspecto, atendiendo a la correspondencia función-área:

FUNCTION	AREA
Producción	Producción (sala del computador)
Control — flujo de datos del proceso antes durante después — calidad	Control-planificación
Stocks de — suministros — datos — programas	Biblioteca-almacén
Soporte	Soporte técnico (propio del departamento de explotación)

Una observación aparte que merece la pena resaltar es la temática concerniente a la conversión de datos. Tradicionalmente se ha considerado dependien-

te del departamento de explotación, pero cada vez más la descentralización de la recogida de datos y el teleproceso alejan del departamento de explotación tales funciones.

Con todo lo mencionado hasta el momento se configura una primera imagen de la estructura de un departamento de explotación, que es la que se puede observar en la figura 8.2.

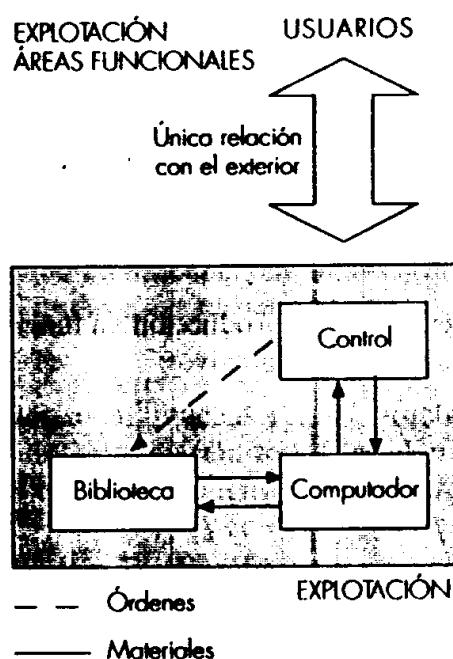


FIGURA 8.2.

En la figura 8.2 se ha incluido ya la idea de que el soporte al usuario, e incluso toda la comunicación y relación con éste, debe de estar centralizada en una sola área funcional, pareciendo más adecuada la de control. Debe insistirse en la necesidad de unificar los contactos con los distintos usuarios bajo una única responsabilidad. De esta manera se logran evitar las interferencias que puede causar al funcionamiento de un departamento de explotación el camino directo entre el usuario y el operador, lo que acaba indefectiblemente perjudicando al conjunto de usuarios.

8.3.2. Tareas funcionales básicas del departamento de explotación

Dentro de las áreas funcionales que se han visto anteriormente cabe especificar con mayor detalle lo que serán las tareas funcionales básicas. Hay que tener muy en cuenta que no se debería establecer, todavía, ninguna relación entre éstas y los puestos de trabajo concretos que las materializarán. En la práctica, cuando las dimensiones del departamento de explotación lo exijan, una misma

persona (puesto de trabajo en sentido estricto) podrá desempeñar tareas funcionales diversas, pero el propio crecimiento del departamento de explotación lleva indefectiblemente a que algunas de estas tareas funcionales básicas (si no todas ellas) tengan que ser cubiertas por una o más personas configurando verdaderos sub-departamentos en la organización del departamento de explotación.

Son precisamente estos sub-departamentos funcionales los que se pretende delimitar aquí, indicando brevemente los componentes más destacados de cada tarea:

a) *Área de control-planificación*

1. *Recepción de documentos*

- Registrar el día y la hora de llegada de los documentos.
- Registrar el número y procedencia de los documentos.
- Avisar a planificación de la llegada y volumen.

2. *Planificación*

- Establecer las previsiones de producción en función de los recursos. Se incluyen todos los centros productivos:
 - conversión de datos
 - preparación
 - ejecución
 - acabados
 - control
- Asegurar la disponibilidad de las informaciones que han de tratarse.
- Dar las órdenes de lanzamiento a los distintos centros productivos.
- Realizar el seguimiento de los trabajos y las incidencias.
- Especificar las medidas que es preciso tomar en casos especiales: avances, retrasos, trabajos aleatorios.

De manera global, el objetivo básico debe incluir la posibilidad de responder a peticiones imprevistas, reservando un mínimo de recursos para tal eventualidad.

3. *Control*

- Verificar aleatoriamente de la corrección de los resultados obtenidos.
- Verificar la calidad de impresión y acabado de los documentos.
- Verificar el número de documentos producidos en relación con los que presumiblemente se esperaban.
- Analizar los listados de error y detectar, si procede, la necesidad de relanzamiento.

4. *Distribución de documentos*

- Registrar día y hora de la salida de la producción.
- Registrar número y destinatario de los documentos.

- Advertir a planificación de la entrega efectiva de la producción.

Todo ello configura el área funcional de control-planificación como el verdadero centro director de todo el departamento de explotación, encargado de la función general de dirigir el flujo de trabajos y la documentación pertinente.

b) *Área de producción*

1. *Conversión de datos*

Su tarea básica es la transcripción de información a los soportes adecuados para permitir su entrada en el sistema.

2. *Preparación de trabajos*

La preparación de trabajos contempla dos aspectos fundamentalmente: la preparación (mecánica) clásica y la preparación técnica que requiere personal especializado en sistemas operativos y en lo que se ha dado en llamar sistemas.

Preparación clásica

Ha de proporcionar, muchas veces en interrelación con lo que se ha llamado biblioteca-almacén, todos los elementos necesarios para que los distintos centros productivos puedan proceder a la realización de los trabajos:

- Archivos que hay que tratar.
- Soportes.
- Programas.
- Parámetros específicos de cada lanzamiento.
- Instrucciones de procedimientos y manipulación a los distintos centros de producción: conversión de datos, operación de consola, operación de periféricos, realización de acabados, etc.

Preparación técnica

- Establecimiento de normas técnicas de explotación.
- Confección de las corrientes de control para lanzamientos normales y procedimientos de relanzamiento.
- Realización de un primer análisis de incidencias especiales.
- Intervención y decisión técnicas en los relanzamientos especiales (no previstos)

3. *Ejecución*

Se cubren tareas como:

- Ejecución de las instrucciones establecidas por el preparador técnico y que la preparación clásica ha puesto al alcance de los operadores.
- Registro de tiempos e incidencias (Diario de explotación).

- Resolución directa de los incidentes que tengan procedimiento de relanzamiento definido.
- Advertencia a la preparación técnica en caso de relanzamientos imposibles o no previstos.

4. *Acabados*

Se encarga de efectuar las tareas prescritas en las distintas máquinas terminales: cortado, separación de copias, encuadernado de impresos, etc.

c) *Área de biblioteca-almacén*

1. *Biblioteca*

Gestiona la biblioteca de cintas, programas, órdenes de control, etc., es decir, todo lo referente a elementos técnicos de la explotación. En la práctica, las ayudas software permiten disminuir la envergadura de esta función, quedando relegada a una supervisión externa de tales recursos, que puede confundirse con la gestión del propio preparador de trabajos.

Lo mismo sucede con el control del espacio en disco, que suele dejarse también a cargo del preparador técnico.

2. *Almacén*

Gestiona el stock de suministros y en general todo tipo de material utilizado directamente por el departamento de explotación (papel de ordenador, situación de volúmenes de cintas y discos, etc.).

Paralelamente a estas tres grandes áreas, hay que incluir un área de gestión para la supervisión y dirección de todo el proceso. Obviamente, esta cuarta área está en íntima interrelación con la dirección del centro de proceso de datos en aspectos tales como la gestión de costos y presupuestos y la supervisión directa y aprobación final de las normas y procedimientos de trabajo dentro del departamento de explotación (tareas habituales de dirección).

Los elementos específicos que deben normalizarse son:

- Carpetas técnicas de explotación.
- Documentación interna dentro del departamento.
- Sistemas de gestión de bibliotecas y almacén de suministros.
- Procedimientos específicos para las fases de puesta a punto de aplicaciones (por la constante intervención del departamento de sistemas en esta fase) separados explícitamente de los procedimientos para la explotación normal.
- Procedimientos para recepción, expedición y control de los documentos.
- Procedimientos habituales de planificación y preparación de trabajos.

8.4. Técnicas empleadas en la explotación

El departamento de explotación, como última etapa de la acumulación de aciertos y errores informáticos de todas las áreas funcionales, es, bajo el punto

de vista del usuario, y por tanto de la empresa, el que da la medida de la efectividad real de todo el centro de proceso de datos.

Dicha efectividad se mide en relación a lo que se ha venido llamando nivel de servicio, entendiéndose éste como un conjunto de aspectos susceptibles de negociación con los usuarios, tales como plazos de entrega, tiempo de respuesta, calidad, volúmenes, coste, etc. Por tanto, el objetivo que se planteará todo departamento de explotación será el de alcanzar y mantener el nivel de servicio comprometido.

Se debe considerar que, en ocasiones, los problemas detectados en la explotación no serán internos, sino que se habrán originado en otros departamentos, como el de análisis y programación, sistemas e incluso los propios departamentos usuarios. En estos casos no será posible aportar soluciones sin involucrar a dichos departamentos.

Teniendo en cuenta todo lo anterior, en este apartado se van a considerar algunas técnicas tendentes a conseguir el objetivo anteriormente citado.

8.4.1. Planteamiento general

Como cualquier actividad que esté caracterizada por el logro de unos objetivos, la actividad del departamento de explotación puede ajustarse al típico esquema planificación-ejecución-medida-realimentación, el cual servirá de base para el desarrollo.

En la figura 8.3 se puede ver un esquema general de la gestión de un centro de proceso de datos. El objetivo es, como ya se ha mencionado, un determinado compromiso de nivel de servicio a los usuarios, el cual se concreta en unas necesidades de aplicaciones y de recursos.

La combinación de ambos permitirá realizar una planificación a largo plazo de los trabajos y una planificación de los recursos que éstos van a consumir. La realización en el tiempo de ambos planes se concreta en una planificación diaria de trabajos, que debe satisfacer las exigencias de nivel de servicio que se pretende.

Lo que resta, a partir de la ejecución del plan diario, es puramente información para la gestión del centro de proceso de datos (las líneas gruesas representan esta información o realimentación). Este control debe realizarse a dos niveles fundamentalmente:

- para tratar de salvar el plan diario o corregir anomalías esporádicas;
- como resultado de un análisis *a posteriori* de desviaciones y tendencias, para negociar con los usuarios un nuevo plan a largo plazo, o para rediseñar los recursos en función del rendimiento y nivel de utilización medidos.

No debe entenderse este segundo nivel como una simple reacción frente a los resultados obtenidos, puesto que aquí también deben incidir las previsiones de implantación de nuevas aplicaciones y el crecimiento vegetativo, tanto para la confección del plan a largo plazo como para la planificación de recursos, por la sencilla razón de que, en circunstancias normales, son factores mucho más relevantes.

vantes que lo que puedan representar las desviaciones respecto del objetivo marcado.

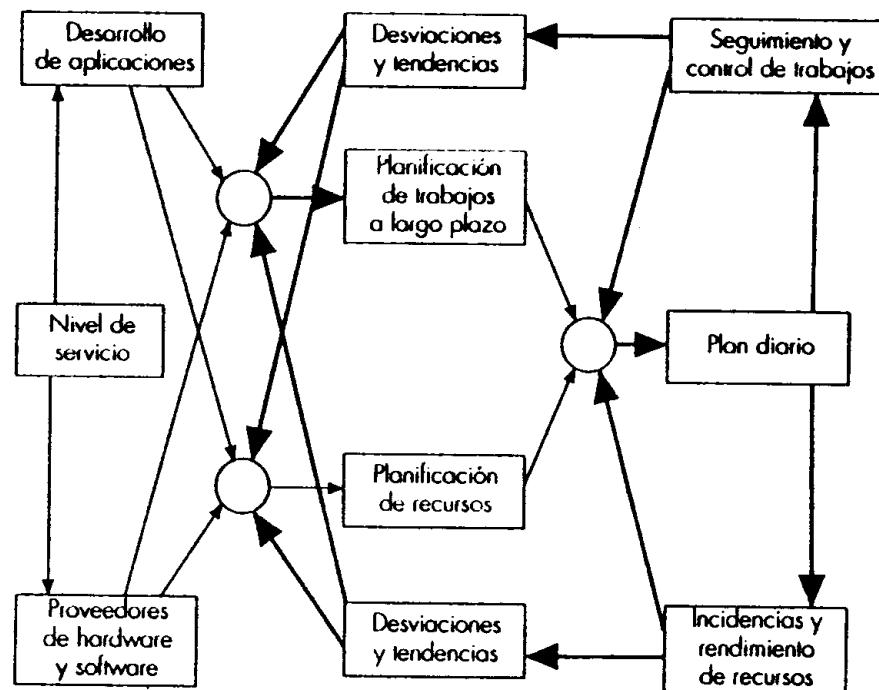


FIGURA 8.3.

Como consecuencia, surgen dos criterios básicos que hay que tener en cuenta para la consecución de los objetivos de la explotación:

- La necesidad de ser más activos que reactivos, es decir, planificación.
- La necesidad de un sistema de información para la gestión y control del centro de proceso de datos.

8.4.2. Planificación

El término planificación tiene características muy distintas según el nivel donde se aplique:

1. *Planificación a largo plazo de los trabajos y recursos que se van a utilizar.*

Se trata cronológicamente del primer nivel. Sus características principales consisten en:

- Negociar con los usuarios las fechas, plazos de entrega, tiempos de respuesta, volúmenes, coste del servicio.
- Negociar con proveedores la cantidad y calidad de recursos que se van a utilizar.

- Considerar conjuntamente todo el sistema informático de la empresa.
- Revisar y actualizar en conjunción con los usuarios.

2. Planificación a corto plazo o diaria.

- Se basa en unos compromisos adquiridos y recursos disponibles: no existe negociación previa.
- Se contemplan exclusivamente un número de trabajos limitado.
- Se realiza internamente en el departamento de explotación.
- Tiene por objetivo el cumplimiento del plazo de entrega al usuario.
- Requiere un seguimiento de cada trabajo a nivel de detalle de las distintas operaciones de que consta.
- Se efectúa diariamente y se actualiza constantemente en función de desviaciones, trabajos imprevistos, incidencias, etc.

3. Planificación a nivel de cada estación de trabajo dentro del centro de proceso de datos o microplanificación.

Consiste en lo siguiente:

- Una lista cronológica de operaciones que hay que realizar en cada estación de trabajo, con hora de entrada, duración y hora de salida.
- La utilización de recursos críticos (limitados) por cada operación.
- La información de operaciones precedentes (obligatorias) y prioridades que es necesario tener en cuenta.
- Sus objetivos son el cumplimiento del horario en cada estación de trabajo y la optimización en el uso de recursos.

Una vez fijadas las características de la planificación a los distintos niveles, se planteará cómo conseguir un sistema de planificación, es decir, qué etapas se deben seguir para su consecución.

a) Primera etapa: planificación manual

Su objetivo es la recogida y validación de todos los datos necesarios. Para ello se considerará el departamento de explotación como una caja cerrada para el usuario, pero internamente compuesto por un conjunto de estaciones de trabajo relacionadas entre sí, en función de la propia lógica de las operaciones que componen un trabajo.

Esta doble concepción permite hacer dos consideraciones:

1. Al usuario, en general, no le interesa saber si su trabajo ha entrado o salido ya del ordenador, sino si tiene o no los resultados en su poder, y, en caso de no tenerlos, si los tendrá a tiempo o con qué retraso.
2. Al planificador le interesa poder hacer una planificación y un seguimiento a nivel de cada estación de trabajo por las que ha de pasar la petición del usuario, para poder anticiparse a los posibles retrasos e informar de los mismos a los usuarios afectados, si no puede corregirlos.

Lo que realmente interesa al planificador es la duración y secuencia de las operaciones. Parece claro que la figura del planificador debe estar, dentro del esquema funcional del departamento de explotación, en una posición tal que le permita actuar sobre el flujo de trabajos en cualquiera de las estaciones de trabajo y recibir información sobre el estado del trabajo.

En una primera etapa, se trata pues de conseguir de cada trabajo la siguiente información:

- Nombre del trabajo, unificado según unos estándares apropiados de nomenclatura, tanto para el usuario como para explotación.
- Personas responsables del usuario (nombre, dirección y teléfono).
- Ciclo básico de ejecución (mensual, semanal, etc.) y día y hora de compromiso de entrega dentro del ciclo (por ejemplo, el primer día hábil del mes).
- Regla de días festivos, en caso de coincidencia (por ejemplo, no procesar, o procesar al día siguiente).
- Secuencia y descripción de todas las operaciones que componen el trabajo, especificando:
 - Duración en cada estación de trabajo.
 - Recursos críticos (escasos) que consume.
 - Dependencias internas y externas.
 - Tiempo de transporte entre estaciones de trabajo (si procede).

Para llegar a conseguir estos datos de manera que sean fiables (especialmente los tiempos), además de necesitar un buen sistema de documentación, requiere del planificador un seguimiento constante, recibiendo y anotando las horas de comienzo y terminación de las operaciones, de forma que un análisis posterior le permita determinar su duración con unos márgenes de error aceptables (dentro del conjunto total).

Una simple hoja de seguimiento de los trabajos en curso puede ser la mejor herramienta para utilizar en esta etapa, la cual, a su vez, va a permitir responder rápidamente sobre la situación de un trabajo, o poder prever posibles retrasos y actuar en consecuencia.

En la medida en que el porcentaje de trabajos aperiódicos sea elevado, aparece otra dificultad para el planificador. Ante una situación de este tipo es necesario negociar con los usuarios sobre la propia periodicidad de sus trabajos, causa principal, por negligencia o excesiva tolerancia, de la mayoría de solicitudes tenidas por irremisiblemente aperiódicas. Es evidente que esto redunda en un nada despreciable beneficio para ambas partes y, en definitiva, para la empresa, pues de no ser posible concretar una periodicidad, el compromiso de servicio que se adquiera lleva implícita una renuncia a la optimización de recursos, si es que se quieren unas ciertas garantías de efectividad, lo cual implica un mayor coste.

Disponer de esta información va a permitir dar los primeros pasos en la creación y mantenimiento (negociación) del plan a largo plazo de los trabajos, cuyos objetivos básicos para explotación son:

- La determinación anticipada de las puntas de carga en cada una de las estaciones de trabajo.
- La negociación con los usuarios afectados respecto a posibles desplazamientos, tanto en la entrega de resultados como en la petición de los trabajos.
- El estudio de las desviaciones y tendencias que sufren los trabajos en la entrada, salida y duración, así como de la distribución y evolución de la carga en las estaciones de trabajo.

Los dos primeros objetivos van a permitir establecer compromisos bilaterales concretos para el próximo periodo de planificación (normalmente un mes), mientras que el tercero va a ser una de las premisas fundamentales que, junto a las previsiones de crecimiento, van a posibilitar una correcta planificación de recursos para el futuro inmediato.

b) Segunda etapa: planificación mecanizada

Para llegar a esta etapa se requiere ineludiblemente haber pasado por la anterior puesto que se partirá de los datos recogidos en la primera etapa. El objetivo claro de mecanizar esta información va a ser doble:

- Por una parte la confección de los planes diarios y a largo plazo sin el concurso de la experiencia personal.
- Por otra, la obtención de planes parciales diarios a nivel de estación de trabajo, lo cual parece bastante difícil con una planificación puramente manual.

Estos planes parciales consistirán fundamentalmente en una lista de operaciones que hay que realizar, clasificadas por orden de llegada prevista, e indicando el código de operación, su duración, los recursos críticos que utiliza, la hora tope de salida y las operaciones precedentes que es necesario completar antes de ejecutar una concreta.

Disponer de esta detallada información va a proporcionar una serie de ventajas importantes:

- Las estaciones de trabajo podrán funcionar con más autonomía al conocer anticipadamente su carga concreta y la holgura en cada operación.
- Esta misma holgura les va a permitir, siempre dentro de los límites marcados, anticipar o retrasar trabajos buscando una mejor optimización de recursos y distribución de carga.
- El planificador podrá hacer un seguimiento de los trabajos por excepción, en lugar de continuo como en la etapa anterior. En otras palabras, al planificador se le va a informar solamente de las operaciones que no han cumplido o no podrán cumplir el horario establecido, lo cual implica una información mucho más reducida y con más carga de significado, sin que ello implique perder el control de la situación del resto de trabajos.

Por lo que se refiere al plan a largo plazo, ahora representará una simple ejecución en ordenador de pocos minutos, lo que quiere decir una gran ayuda en la negociación con los usuarios al poder confeccionar tantos planes como sea necesario con un fiel reflejo de las implicaciones que un cambio pueda producir en el resto de trabajos.

Como resultado de esta etapa, la figura del planificador se habrá visto liberada de gran parte del trabajo rutinario y mecánico de la confección de planes y seguimiento exhaustivo, con lo cual podrá dedicar más su potencial creador a una auténtica gestión de la planificación diaria y una relación con usuarios más eficaz y documentada.

Por último, algunos inconvenientes de esta etapa, que introducirán la necesidad de la tercera, son las siguientes:

- No existe realimentación que automáticamente replanifique las estaciones de trabajo ante situaciones de retraso o de indisponibilidad en los recursos utilizados.
- Aun cuando se ha dicho que el seguimiento por excepción es una ventaja, por otro lado priva de información vital para la planificación futura, que consiste en la evolución que sufre la duración de las operaciones, tanto por el crecimiento normal del volumen como por el uso de nuevos recursos en las estaciones de trabajo. Esto obliga a realizar esporádicamente medidas de tiempos para actualizar la costosa base de información.

c) *Tercera etapa: planificación automática*

Los objetivos básicos de esta etapa son los siguientes:

- Explotar la documentación obtenida en la etapa anterior, de manera que produzca planes automáticos y continuamente actualizados en función de la evolución real de los trabajos.
- Tratar de conseguir una realimentación continua en todas las operaciones de manera que los datos sobre su duración sean el más fiel reflejo de la realidad.

Como ventajas adicionales de un sistema de este tipo se pueden citar:

- Comunicación automática (por consulta) de aquellos trabajos que no van a alcanzar su plazo de entrega.
- Obtención *a posteriori* de información estadística o detallada del nivel de servicio proporcionado datos, tales como:
 - trabajos terminados
 - resúmenes por aplicación y usuario
 - operaciones terminadas erróneamente
 - estadísticas de error en las operaciones
 - etc.

En este entorno, el planificador estará totalmente dedicado a velar por el servicio al usuario.

8.4.3. Planificación de recursos

Se va a abordar de forma superficial una parte fundamental para la gestión del departamento de explotación y que se caracteriza por el estudio directo del uso que se hace de los recursos existentes, prescindiendo en cierto modo de las aplicaciones.

La información obtenida del sistema de planificación, sobre todo en la tercera etapa, podría ser suficiente para facilitar la negociación del plan de carga a largo plazo, pero es poco significativa para abordar la planificación de recursos.

El planteamiento del tema se basa simplemente en una elemental regla de tres:

Si mis aplicaciones actuales ocupan unos recursos X , mis aplicaciones futuras ocuparán unos recursos Y . Esto quiere decir:

- Un análisis exhaustivo de qué aplicaciones y con qué distribución en el tiempo se están utilizando.
- Una correlación aplicación-recurso, describiendo el nivel de utilización en cada caso.
- Un perfil de utilización por horas de cada uno de los recursos.
- Una proyección en el tiempo de los tres puntos anteriores, en función de las perspectivas de crecimiento.
- La consideración de los sistemas apropiados de respaldo en función de los compromisos de disponibilidad frente a los usuarios y de las garantías de los suministradores.

De todo esto, lo que debe aportar la explotación son todos los datos relativos a la utilización actual de los recursos (nivel de carga) y el uso que de ellos hacen las distintas aplicaciones en marcha, así como las tendencias que se observan en el tiempo según el crecimiento.

8.4.4. Coste y calidad: incidencias en explotación

Si los componentes fundamentales de un buen nivel de servicio son tiempo, coste y calidad, se ha tratado únicamente del primero de ellos. Se van a abordar los otros dos conjuntamente considerando un hecho real y cotidiano en la explotación: las incidencias.

Cualquier incidencia, ya sea producida por mal funcionamiento de un recurso o por fallo humano, se traducirá en un incremento de los costes, a la vez que puede dar al traste con un compromiso de plazo de entrega. Este último aspecto puede ser salvado con un buen seguimiento a tiempo y un sistema de planificación ágil y dinámico, de manera que, aún repitiendo ciertas operaciones, se pueda entregar el trabajo a tiempo.

Asimismo, se debe considerar un nuevo elemento que aparece directamente relacionado con las incidencias, puesto que suele ser la etapa final de las mis-

mas: todo problema acaba solucionándose con un cambio. A su vez existen cambios constantes en todas las áreas de explotación, que pueden a su vez provocar problemas, de aquí la fuerte interrelación entre ambos elementos.

De todo ello se deduce la necesidad de introducir nuevas funciones que den soporte a las necesidades (figura 8.4).

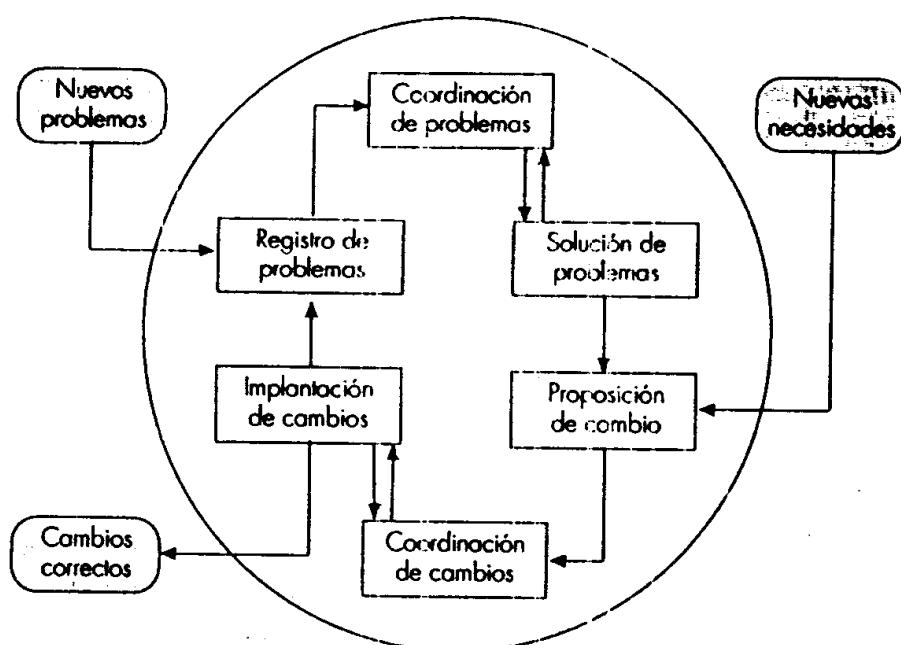


FIGURA 8.4.

La coordinación de problemas va a ser una función cuyos objetivos básicos son:

- Ante la aparición de un problema:
 - Valorar la documentación que se aporta.
 - Clasificarlo según un esquema de prioridades.
 - Ejecutar un *problem determination* que le permita actuar de filtro ante problemas que no son tales.
 - Asignar la función de resolución que debe hacerse cargo del problema.
- Durante la vida de un problema:
 - Contactar con la función de resolución en orden a registrar todas las posibles situaciones y acciones que se vayan tomando.
- Una vez cerrada la incidencia:
 - Confeccionar y estudiar los resúmenes estadísticos.
 - Sacar conclusiones sobre las causas más comunes de problemas.
 - Proponer medidas correctivas que permitan eliminarlos en el futuro.

En cuanto a la coordinación de cambios, sus objetivos son:

- **Ante una solicitud de cambio:**
 - Valorar el riesgo existente en el mismo.
 - Determinar posibles implicaciones.
 - Aceptarlo o denegarlo.
- **Ante la aceptación de un cambio:**
 - Planificarlo en el tiempo.
 - Informar a todas las funciones operativas afectadas.
 - Mantener la documentación pertinente a los cambios.
- **Ante la realización de un cambio:**
 - Contactar con la función de implantación, indicándole los métodos de respaldo que se van a utilizar en orden a una posible marcha atrás en caso de tener que deshacerlo.

Con un sistema de gestión de problemas y cambios basado en este esquema se puede conseguir una mejora en el servicio al usuario. Uno de los aspectos más importantes en el trabajo del coordinador de problemas es sin duda la posibilidad de anticiparse a problemas futuros mediante un análisis de las causas más frecuentes de los mismos y la propuesta de medidas correctoras. Ahora bien, para llevar a la práctica dicho análisis será preciso determinar los posibles distintos aspectos bajo los que puede contemplarse un problema, para así poder conseguir estadísticas resumiendo todos los problemas que tienen en común alguno de estos aspectos. A modo de ejemplo, se podrían sugerir los siguientes:

- Área impactada de recursos.
- Grado de impacto en el recurso.
- Grado de impacto en el usuario.
- Causa del problema.
- etc.

A nadie se le escapa la importancia que, para una buena gestión de la explotación, tiene el poder conocer con exactitud qué problemas pendientes hay, sobre qué áreas, quién los está solucionando, cuántas veces se han producido, con qué frecuencia, en qué afectan a los compromisos contraídos, con qué holgura se debe planificar en función de la experiencia, cuáles son los recursos más débiles, etc. Si a todo ello se añade la gran cantidad de información que hay que recoger para cada problema y la cantidad de problemas que se pueden originar en la explotación, se puede llegar a pensar en la necesidad de un enorme aparato burocrático de dudosa eficacia. Por todo ello será necesario introducir la gestión de incidencias por etapas que permitan ir constituyendo el sistema de gestión sin producir desbordamientos.

En una primera etapa se podría montar y probar un circuito de información, definiendo las responsabilidades de las coordinaciones de problemas y de cam-

bios, registro de incidencias e implantación de cambios, y eligiendo solamente algunos de los aspectos que hay que considerar en cada problema. Lo único que se deberá hacer es diseñar una hoja de registro de incidencias, la cual seguirá el circuito fijado para la resolución de cualquier incidencia. Dicho circuito se cerrará una vez se implante el cambio necesario para la solución del problema.

A medida que se vaya ganando experiencia se podrán ir introduciendo nuevos aspectos y/o ampliar los existentes.

En el momento en que el volumen de impresos archivados en poder del coordinador de problemas sea grande y, por tanto, dificultoso su tratamiento para la obtención de resúmenes, relaciones parciales, consultas, etc., es muy posible que se tengan un esquema y una experiencia de funcionamiento suficientes como para poder abordar la segunda etapa que va a posibilitar el control prácticamente total de los problemas, y que consiste en introducir la información en un archivo en disco con los siguientes objetivos:

- Altas y modificaciones en tiempo real, a través de terminal de pantalla.
- Consultas individualizadas a un problema concreto o búsqueda por características comunes, también a través de terminal.
- Listados en detalle, resumidos por distintos conceptos y períodos de tiempo.
- Mantenimiento de una cinta histórica de problemas cerrados para resúmenes de períodos más extensos o posibles consultas de incidentes atrasados.
- Listados de los cambios realizados en orden cronológico y por cada recurso afectado.

