

Instrumentación y visualización

Práctica 2

1 Objetivo

El objetivo de esta práctica es que el alumno practique los conceptos de los temas de **medición** y **visualización**. Para ello, el alumno aprenderá a instrumentar programas en Windows 2000, añadiendo instrumentación al inyector desarrollado en la práctica 1 y coordinándolo con el monitor de prestaciones de Windows 2000. A partir de los datos obtenidos se realizará un análisis gráfico.

Además, la práctica servirá para comprender mejor los conceptos de tiempo de respuesta, productividad, régimen transitorio y régimen permanente que aparecen a lo largo de toda la asignatura. El trabajo práctico mostrará las dificultades que pueden surgir a la hora de medir un sistema: esfuerzo empleado por el analista en la instrumentación, sobrecarga inyectada en el sistema por las sondas de instrumentación, manejo de relojes, etc.

El trabajo desarrollado servirá en futuras prácticas para realizar un **análisis** del estado del sistema que permita determinar cuáles son las máximas prestaciones que puede ofrecer el sistema y qué elementos las limitan.

2 Descripción

2.1 Instrumentación del cargador

Dos de los valores fundamentales para analizar las prestaciones de cualquier sistema son el **tiempo de respuesta** y la **productividad**. Para medir estos valores en nuestro sistema se va a instrumentar el inyector, es decir, se van añadir unas instrucciones cuyo objetivo es obtener información del sistema, en concreto, cuánto tarda cada petición (tiempo de respuesta) y cuántas peticiones por unidad de tiempo sirve el sistema (productividad).

Para medir el tiempo de respuesta hay que tomar el tiempo antes de comenzar una petición y después de recibir la respuesta. A continuación se muestra el pseudocódigo de un usuario (ver página 2 de la práctica 1) y los puntos donde habría que tomar estos tiempos:

```
Para i = 0 hasta NumPeticiones hacer
    Tomar tiempo de inicio;
    Conectarse al servidor;
    Enviar una cadena de petición (250 bytes);
    Esperar por la respuesta del servidor (250 bytes);
    Cerrar la conexión;
    Tomar tiempo de finalización;
    Dormirse para simular el tiempo de reflexión;
fPara
```

La diferencia entre el tiempo de inicio y el de finalización de una petición sería el tiempo de respuesta para esa petición. Es fundamental observar que el tiempo de reflexión no entra dentro del tiempo de respuesta.

En el pseudocódigo anterior, la línea “Conectarse al servidor” englobaba dos funciones: la llamada a `socket()` para crear el socket y la llamada a `connect()` para realizar la conexión efectiva al servidor. Como hasta que no ocurra la segunda el servidor en realidad no empezará a recibir la petición, el tiempo de inicio debe de estar justo antes del `connect()` y no debe incluir la llamada a `socket()`.

El API de Windows ofrece muchas funciones para tomar tiempos. Las de mayor precisión utilizan el denominado *Contador de rendimiento* (*Performance Counter*). La precisión de este contador es dependiente del sistema. Para obtenerla se debe llamar a la siguiente función:

```
BOOL QueryPerformanceFrequency (LARGE_INTEGER* lpFrequency);
```



Práctica 2

Esta función recibe un puntero a un entero largo (64 bits) donde devuelve el número de ticks del contador de rendimiento por segundo. Devuelve un valor booleano que indica si el contador de rendimiento está disponible, ya que en algunas arquitecturas puede no estarlo.

El tipo `LARGE_INTEGER` está definido así:

```
typedef union _LARGE_INTEGER {
    struct {
        DWORD LowPart;
        LONG HighPart;
    };
    LONGLONG QuadPart;
} LARGE_INTEGER;
```

Es decir, es una unión de una estructura y un tipo `LONGLONG`. La estructura permite dividir el valor de 64 bits en dos segmentos de 32 bits, uno que referencia la parte alta y otro la parte baja. El miembro `QuadPart` permite acceder a los 64 bits a la vez.

Un aspecto importante que hay que tener siempre en cuenta al medir el rendimiento son las magnitudes con las que se va a trabajar. Suele ser conveniente utilizar sólo una magnitud en todo el programa, para no encontrarse con problemas debido a la mezcla de unidades. Además, hay que tener en cuenta que si utilizamos una unidad demasiado grande (por ejemplo, segundos) cuando nuestros tiempos de respuesta sean del orden de unos pocos segundos, estaremos perdiendo demasiada información; en cambio, si utilizamos una magnitud demasiado pequeña (por ejemplo, nanosegundos) tendremos valores muy grandes que pueden no caber en el formato en el que los almacenemos. En nuestro sistema los tiempos de respuesta serán del orden de segundos y las duraciones de las pruebas del orden de minutos. Parece por lo tanto conveniente utilizar como unidad básica el **milisegundo** y almacenarlo en un tipo `float`, lo que nos permitirá tener un rango suficientemente grande. Para ello, el valor devuelto por la función `QueryPerformanceFrequency` se debe pasar de un `LARGE_INTEGER` que indica los ticks por segundo a un `float` que indique los ticks por milisegundo.

Para acceder al valor del contador de rendimiento se utiliza la siguiente función:

```
BOOL QueryPerformanceCounter (LARGE_INTEGER* lpPerformanceCount)
```

Esta función devuelve el número de ticks que hay en un momento dado en el contador de rendimiento.

Para medir la productividad, simplemente habría que tomar el tiempo de inicio de la primera petición y el tiempo de finalización de la última. Esto nos daría el intervalo de duración de la medición. Si contamos el número de peticiones realizadas en ese intervalo y lo dividimos por su longitud, obtendremos la productividad. Pero antes de hacer esto hay que tener en cuenta la influencia de los períodos de régimen transitorio. En el siguiente punto se explica con detalle.

2.2 Control de la longitud de la prueba

Cuando se realiza una prueba de inyección de carga, se puede observar un comportamiento como el siguiente:

Práctica 2

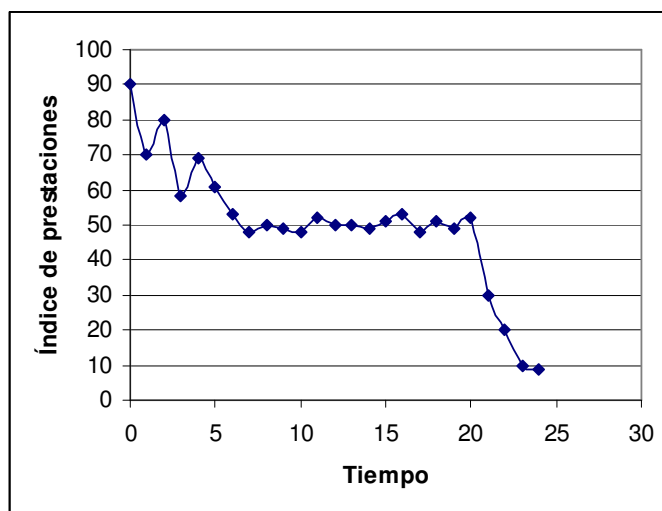


Fig. 1 Transitorio y permanente

Es decir, para algunos índices de prestaciones se ve que hay un período central durante el cual se mantienen estables rodeados de dos períodos, denominados transitorio de entrada y transitorio de salida, durante los cuales los valores son muy distintos del valor medio (no tienen necesariamente que ser como en la figura, más altos al principio y más bajos al final). Estos períodos transitorios pueden ser debidos a diversas causas:

- El transitorio de entrada: Una razón habitual para el transitorio de entrada es que las cachés inicialmente están vacías y según se van recibiendo peticiones se van llenando, haciendo que se reduzcan los tiempos de respuesta.
- El transitorio de salida: Una razón habitual del transitorio de salida es que si hay varios usuarios realizando peticiones, no todos acabarán a la vez. Según vayan acabando usuarios, el sistema será sometido a una carga inferior, lo que variará sus índices de prestaciones.

En general, interesa estudiar el comportamiento del sistema en el **régimen permanente**. Por ello, hay que descartar las mediciones de los regímenes transitorios. Esto requerirá modificaciones en el inyector. A continuación se indica una forma de hacerlo.

En lugar de controlar el número de peticiones que hace un usuario mediante un valor constante, se emplearán unos tiempos. En concreto se definirán dos intervalos que el analista debe poder controlar: el **intervalo de arranque** y el **intervalo de medición**. La idea global será (ver Fig. 2): el inyector lanzará todos los hilos en un momento dado, que llamaremos *Instante base*, y empezarán a hacer peticiones hasta que haya transcurrido el intervalo de arranque. El tiempo de respuesta de estas mediciones no será guardado, ya que no interesa. Tampoco se contará su número para calcular la productividad. Cuando haya finalizado el intervalo de arranque, comenzará el intervalo de medición. Las peticiones que empiecen y acaben antes de que finalice este intervalo de medición sí serán medidas. En cuanto finalice este intervalo de medición, se dejarán de lanzar peticiones.

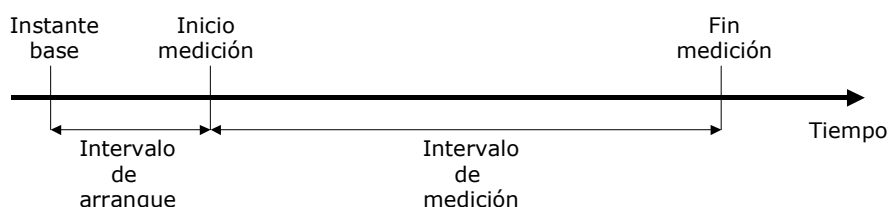


Fig. 2 Intervalo de arranque e intervalo de medición

Práctica 2

En la práctica anterior se había creado un vector que tenía una posición para cada hilo. Este vector debe almacenar para cada hilo:

- Un contador que cuente el número de peticiones que empezaron y acabaron dentro del intervalo de medición.
- Un vector que para cada petición de las que empezaron en el intervalo contendrá:
 - El tiempo de inicio.
 - El tiempo de fin.

El `Instante base` se puede calcular llamando a `QueryPerformanceCounter` justo antes de comenzar a lanzar los hilos. Sumándole a ese valor el número de ticks correspondiente al intervalo de arranque se obtendrá el instante `Inicio Medición` en el que los hilos deben empezar a guardar los datos de las peticiones. Sumándole a este segundo tiempo el intervalo de medición se obtendrá el instante `Fin medición` que indicará cuándo los hilos deben terminar de hacer peticiones. Todos estos valores los calculará el hilo principal antes de lanzar los hilos cliente y éstos los leerán a través de una variable global.

Tras la finalización de todos los hilos, el hilo principal puede recorrer el vector con la información de todos los hilos para obtener el tiempo de respuesta medio. Hay que tener en cuenta que el número de peticiones en cada hilo será distinto. Para calcular la productividad, simplemente se suman todas las peticiones realizadas (dentro del intervalo de medición) por todos los hilos y se divide entre la longitud del intervalo de medición.

Además de volcar a disco la media de la productividad y el tiempo de respuesta, como en esta práctica se trata de analizar los transitorios, **se volcarán los datos de cada petición**, es decir, cuándo empezó y cuándo acabó. Este volcado también lo hará el hilo principal recorriendo el vector con los datos de cada hilo.

2.3 Coordinación con el monitor de Windows 2000

Cuando se lance una prueba, además de lanzar el inyector habrá que lanzar el monitor de rendimiento de Windows 2000 para poder observar el uso de recursos en el servidor. El monitor debe ser lanzado antes que el inyector y debe ser detenido después de que este acabe. Se habrá configurado el monitor para que deje en un archivo .TSV los datos. Cada muestra tendrá su correspondiente marca de tiempo obtenida del reloj del sistema.

Surge un problema: el monitor ha estado lanzado más tiempo del que se corresponde con el tiempo de medición. Como sólo debemos considerar los datos correspondientes al intervalo de medición y no a los transitorios de entrada y salida, debemos ser capaces de determinar qué muestras del intervalo están dentro del tiempo de medición. Para ello debemos saber el tiempo de sistema en el que ocurrió el `Inicio medición` y el `Fin medición`.

Una forma de obtener la hora del sistema es utilizar la función de ANSI C `time`, que devuelve en un valor `time_t` el número de segundos desde las 00:00:00 del 1 de enero de 1970 (UTC).

Lo que se debe hacer es, al mismo tiempo que se obtiene el tick del `Instante base`, se debe llamar a esta función, con lo cual se tendrá tanto en ticks como en la hora del sistema el instante que se utiliza para el resto de los cálculos. Al final de la prueba, entre los datos que se deben volcar a fichero, además de la productividad y el tiempo de respuesta, estarán estos dos valores:

- La hora del sistema a la que comenzó la medición: Se calculará sumándole la duración del intervalo de arranque a la hora que se obtuvo en el `Instante base`.
- La hora del sistema a la que finalizó la medición: Se calculará sumándole al valor anterior la duración del intervalo de medición.

Para escribir estos valores se puede utilizar la función `ctime`. Es conveniente copiar la cadena que devuelve esta función a otra cadena, ya que la que devuelve la función es un buffer estático y si se imprimen directamente con dos llamadas a `ctime` dos valores de tiempo distintos, es posible que el resultado sea el del último tiempo.

2.4 Pruebas a realizar

Se van a hacer tres réplicas de una prueba con el objetivo de comprobar el correcto funcionamiento de la instrumentación y para que el alumno aprenda a distinguir el régimen transitorio del régimen estable. El proceso de prueba será el siguiente:

- Utilizar dos máquinas, una para el servidor y otra para el cliente.
- Arrancar el servidor con los siguientes parámetros: 10000 50 5 200 4 10 c:\trabajo\datosSIF\.
- Preparar un registro de contador del monitor de rendimiento de Windows 2000. Este registro se preparará en el cliente (para reducir la perturbación en el servidor al volcar los datos a disco), aunque **los contadores serán del servidor**. El tiempo de toma de muestras debe de ser de un segundo. Los contadores a incluir son los siguientes:
 - Procesador: % tiempo de procesador, % tiempo de interrupción.
 - Memoria: Bytes disponibles, Bytes de cache, Bytes comprometidos, Pág./s, Fallos de s.
 - Archivo de paginación: % uso.
 - Disco Físico: % idle, Long media de la cola de disco.
 - Sistema: Cambios de s. y Llamadas s.
 - Interfaz de red: Total de s.
- Lanzar el cliente tres veces. Cada ejecución del cliente tendrá los siguientes parámetros: 10 usuarios con tiempo de reflexión medio de 1 segundo, 0 de intervalo de arranque y 7 minutos de intervalo de medición.
- Detener el monitor de rendimiento. Introducir en un Libro de Excel tanto los resultados del monitor de rendimiento como de la ejecución de los clientes para realizar un análisis del transitorio.

3 Material a entregar

Se entregará una memoria en la que se justifique:

- Cuál debería ser la duración del intervalo de arranque para que no influya en el experimento.
- Si hay diferencias significativas entre las tres réplicas del experimento.

Para ello el alumno debe escoger el número y formato de gráficas convenientes e introducir las explicaciones textuales que considere adecuadas. Se valorará tanto la capacidad de análisis del alumno como la de síntesis, es decir, el alumno debe mostrar todas las gráficas que sean convenientes pero no más de las necesarias.

La memoria debe incluir además la siguiente tabla rellena para comprobar el correcto funcionamiento del inyector:

Concepto	Réplica 1	Réplica 2	Réplica 3
Media del tiempo de reflexión			
Media del tiempo de respuesta			
Productividad			
Media del uso del archivo de paginación			
Media del tiempo inactivo del disco			
Media de la longitud de la cola del disco			
Media de los bytes comprometidos			
Media de los bytes de caché			
Media de los bytes disponibles			
Media del contador Fallos de s.			
Media del contador Pag./s			
Media del % de tiempo de int.			
Media del % de procesador			
Media de los cambios de contexto			
Media del número de llamadas al sistema			