Práctica 6

SMPCache: simulador de cachés para multiprocesadores simétricos

1. Objetivos

El objetivo de la presente práctica es aprender a utilizar el simulador de cachés en sistemas multiprocesadores simétricos (memoria central compartida) y realizar un estudio de los principales protocolos de coherencia de cachés basados en bus.

Gracias al simulador se pueden ver parámetros importantes de la ejecución como son las transacciones del bus, los fallos de caché, etc. A partir del número de transacciones en el bus, y haciendo algunas suposiciones sobre el coste temporal de las operaciones a realizar, es posible calcular el tiempo aproximado de ejecución de un determinado problema.

2. Desarrollo

2.1 El simulador SMPCache

Este simulador permite ver con detalle las transacciones que tienen lugar en un sistema multiprocesador con coherencia de caché. El simulador sólo sirve para sistemas con memoria de acceso uniforme (UMA), también conocidos como sistemas simétricos o con memoria centralizada. Esto a su vez implica que los sistemas a simular están basados en un único bus compartido por todos los procesadores. El tipo de protocolos de coherencia de cachés en estos sistemas es el de sondeo, habiéndose implementado tres de estos protocolos: MSI, MESI y DRAGON. El MSI y el MESI son de invalidación, mientras que el DRAGON es de actualización; esto va a permitir estimar cual de las dos estrategias de mantenimiento de la coherencia es más adecuada en según qué casos.

SMPCache funciona bajo Windows y es totalmente visual y amistoso. El simulador funciona a partir de ficheros de trazas donde se especifica el tipo de acceso a memoria (instrucción, lectura o escritura) y la dirección a la que se accede. Estos ficheros de trazas están formados por dos columnas, en la primera se pone la operación y en la segunda la dirección en hexadecimal. En esta sesión utilizaremos un programa para generar ficheros de trazas de forma sencilla. Como vamos a estar más interesados en los accesos de lectura y escritura, no se incluirán los accesos de búsqueda de instrucción.

En la página web de la asignatura (http://informatica.uv.es/iiguia/AC/) se puede encontrar el simulador, así como un ejemplo para generar las trazas, además de los manuales de uso y otro material adicional.

2.2 Generación de las trazas para el simulador

El simulador SMPCache toma como entradas las trazas de una ejecución de un programa. La traza de un programa es una lista con los accesos a la memoria que se van realizando. Una forma sencilla de obtener la traza de un programa consiste en incluir código para ir registrando los accesos que se puedan ir produciendo. En el programa siguiente se muestra el ejemplo de un programa que copia un vector en otro. Como no se han considerado los accesos a instrucción, la traza consistiría simplemente en un acceso de lectura y otro de escritura para cada uno de los elementos del vector:

```
// Programa de ejemplo para generar trazas para el simulador SMPCache.
// Este programa genera trazas simulando la copia de un vector a otro,
// sin tener en cuenta los accesos a instruccion.
// Se generan los ficheros para 1, 2, 4 y 8 procesadores.

using namespace std;

#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <string>
```

```
const int NUM  = 1000;  // Longitud del vector
const int N    =    8;  // Maximo numero de procesadores
const int LEC  =    2;  // Indica Lectura
const int ESC  =    3;  // Indica Escritura

// Esta funcion escribe una transaccion en el formato del simulador.
// Se le pasa el fichero, el tipo y la direccion
void traza(ofstream  &f,int caso, void *p)
{
 f << caso << " ";
 f.width(8);
 f.fill('0');
 f << hex << (int)p << endl;
}

int main()
{
int a[NUM],b[NUM];  // Vectores de test
ofstream f[N];      // Ficheros para cada procesador
int n,i,proc;
string nombre;
 for (n=1;n<=N;n*=2)
 {
   for (proc=0;proc<n;proc++)
   {
     nombre=string("c:\\temp\\traza") + char(n+'0') + "_" + char(proc+'1') + ".prg";
     f[proc].open(nombre.c_str());
   }
   for (i=0; i<NUM; i++)
   {
     a[i]=b[i]; // operacion de ejemplo
       //proc=n*i/NUM;    // reparto en trozos consecutivos
       proc=i%n;        // reparto entrelazado
       traza(f[proc],LEC,&b[i]);  // Lectura de b[i]
       traza(f[proc],ESC,&a[i]);  // Escritura en a[i]
   }
   for (proc=0;proc<n;proc++) f[proc].close();
 }
 system("PAUSE");
 return 0;
}
```

El fichero fuente de este programa se puede encontrar en la página web de la asignatura. El simulador SMPCache utiliza un fichero de trazas para cada uno de los procesadores que se utilicen, de manera que si se simula con un procesador hará falta un fichero (llamado *traza1_1.prg*) si se simula con dos procesadores harán falta dos ficheros (*traza2_1.prg* y *traza2_2.prg*) y así sucesivamente. Este programa genera todos los ficheros necesarios para poder simular con 1, 2, 4 y 8 procesadores. En el nombre de los ficheros de trazas aparecen dos números, el primero es el número de procesadores para los que se ha generado la traza y el segundo es el procesador específico al que se debe descargar el fichero.

En el programa anterior hay dos tipos de reparto entre los procesadores de las tareas a realizar: uno es un reparto en trozos consecutivos y el otro es un reparto entrelazado. En el reparto consecutivo cada procesador se encarga de un trozo completo de elementos consecutivos del vector; en el reparto entrelazado los elementos se van repartiendo entre los procesadores de manera que elementos del vector consecutivos están en procesadores distintos. Esto permite poner de manifiesto el problema de la **falsa compartición**.

A partir de este programa de ejemplo se pueden realizar programas más elaborados incluyendo trazas más complejas, como por ejemplo, el caso del acceso a semáforos y barreras. También se puede cambiar el tamaño del vector con el fin de poner de manifiesto el problema del mapeado directo.

El programa se puede compilar en Windows con el entorno de programación Dev-C++ que está disponible en la página web de la asignatura de fundamentos de programación.

3. Trabajo a realizar

3.1 Falsa compartición

A partir del programa dado anteriormente, se compila y ejecuta para generar los ficheros de traza. A continuación se lanza el simulador SMPCache, se cargan las trazas y se simula para 1, 2, 4 y 8 procesadores con los protocolos disponibles, es decir, MSI, MESI y DRAGON.

Los parámetros importantes a destacar en la simulación son el número de transferencias de bloque en el bus y la tasa de fallos y aciertos de la caché. Estos parámetros se pueden obtener fácilmente a partir de la vista de la evolución del multiprocesador.

Como se quiere poner de manifiesto la falsa compartición, habrá que hacer todo el estudio con las dos formas de reparto de los elementos del vector, es decir, reparto entrelazado y reparto consecutivo.

El sistema multiprocesador deberá configurarse con las siguientes características:

- Procesadores: 8
- Arbitraje del Bus: LRU
- Tamaño de palabra: 8 bits
- Palabras por bloque: 128
- Bloques en la memoria: 8192
- Bloques en la cache: 64
- Mapeado: Asociativa por conjuntos
- Conjuntos: 32 (2 vías)
- Reemplazo: LRU

No es necesario cambiar el número de procesadores en el sistema para simular con un número menor de 8 procesadores, así que basta configurar el sistema con 8 procesadores (es el máximo) y luego cargar la traza solamente en aquellos procesadores que se vayan a simular.

Hay que tomar nota de los resultados de cada simulación realizada para poder sacar conclusiones acerca del comportamiento y rendimiento del multiprocesador.

3.2 Mapeado directo

Uno de los problemas de las cachés es el mapeado (correspondencia) de las líneas de memoria en la caché. El mapeado directo puede causar problemas cuando se intenta reemplazar una y otra vez la misma línea en la caché, aunque ésta corresponde a diferentes posiciones en la memoria.

Para poner de manifiesto este problema se puede modificar el programa generador de trazas buscando un tamaño adecuado para los vectores que se están utilizando, de manera que tanto un vector como el otro ocupen la misma línea de caché a pesar de estar en zonas de memoria diferentes.

Una vez encontrado el tamaño crítico de estos vectores, realizar una simulación utilizando el mapeado directo y compararlo con el asociativo por conjuntos de 2 y también el de 4 vías.

Para comprobar asimismo que el mapeado asociativo por conjuntos también puede tener problemas, introducir un tercer vector en el programa que también comparta las líneas de los otros dos vectores. Simular el nuevo programa con mapeado directo, asociativo de 2 vías y asociativo de 4 vías.

Dado que este problema es independiente del número de procesadores, es preferible **realizar las simulaciones con un único procesador**.

3.3 Trabajo a presentar

Las conclusiones de esta primera sesión sobre multiprocesadores deberán anotarse por escrito para poderlas recordar en el examen. Básicamente hay que contestar a las siguientes cuestiones (entre otros aspectos que se quieran destacar):

¿Cuanto es peor el reparto entrelazado frente al reparto consecutivo de las tareas en un multiprocesador? ¿Por qué?

¿Cómo influye el número de procesadores en el problema de la falsa compartición? ¿Por qué?

¿Qué protocolo de coherencia se comporta mejor frente al problema de la falsa compartición? ¿Por qué?

¿Cómo se ha tenido que modificar el programa original para poner de manifiesto el problema del mapeado directo? Justifica la respuesta numéricamente.

¿Cómo se ha tenido que modificar el programa original para poner de manifiesto el problema del mapeado asociativo con dos vías? Justifica la respuesta numéricamente.

Todas las respuestas deben justificarse con la ayuda de gráficas.