1. Objetivos

El objetivo de este proyecto es aprender a utilizar el simulador SMPCache 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.

El simulador SMPCache tiene implementados tres protocolos de espionaje para la coherencia caché: 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.

2. Requisitos

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 de memoria multiprocesador para el
   simulador SMPCache.
   Este programa genera trazas simulando la copia de un vector a otro,
   sin tener en cuenta los accesos a instrucción (capturas de instrucciones).
   Es fácil cambiar esta operación (copia de un vector en otro) por otras
   operaciones con vectores.
   Este programa genera los ficheros de traza para configuraciones con 1, 2,
   4 y 8 procesadores.
*/

#include <fstream.h>
#include <stdio.h>

const int LVEC  = 1000; // Longitud del vector
const int NPROC =    8; // Máximo número de procesadores
const int LEC   =    2; // Indica Lectura
const int ESC   =    3; // Indica Escritura

// Esta función escribe un acceso de memoria con el formato del simulador.
// Necesita el fichero de traza, el tipo de acceso a memoria, y la dirección
void traza(ofstream &fich,int tipo,void *dir)
{
 fich << tipo << " ";
 fich.width(8);
 fich.fill('0');
 fich << hex << (int)dir << endl;
}
```

```
int main()
{
 int a[LVEC],b[LVEC];  // Vectores de test
 ofstream fich[NPROC]; // Ficheros para cada procesador
 int n,i,proc;
 char nomfich[33];

 for (n=1;n <= NPROC;n*=2)
 {
   for (proc=0;proc < n;proc++)
   {
     sprintf(nomfich,"c:\\temp\\traza%d_%d.prg",n,proc+1);
     fich[proc].open(nomfich);
   }
   for (i=0;i < LVEC;i++)
   {
     a[i]=b[i]; // Operación de ejemplo
       //proc=n*i/LVEC;    // Reparto en trozos consecutivos
       proc=i % n;      // Reparto entrelazado
       traza(fich[proc],LEC,&b[i]); // Lectura de b[i]
       traza(fich[proc],ESC,&a[i]); // Escritura en a[i]
   }
   for (proc=0;proc < n;proc++) fich[proc].close();
 }
 return 0;
}
```

El fichero fuente de este programa se puede encontrar en el sitio web del simulador SMPCache. 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 cargar 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 de la correspondencia directa.

3. Desarrollo

3.1. Falsa compartición

A partir del programa dado en la sección anterior, 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 accesos al 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 caché: 512
- Correspondencia: Asociativa por conjuntos
- Conjuntos: 256 (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 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. Asociatividad

Uno de los problemas de las cachés es la correspondencia (mapeado) de las líneas de memoria en la caché. La correspondencia directa 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 principal.

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 correspondencia directa, y compararla con la asociativa por conjuntos de 2 y también la de 4 vías.

Para comprobar asimismo que la correspondencia asociativa por conjuntos también puede fallar, introducir un tercer vector en el programa que también comparta las líneas de caché de los otros dos vectores. Simular el nuevo programa con correspondencia directa, asociativa de 2 vías y asociativa de 4 vías.

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

4. Informes

Las conclusiones de este proyecto sobre multiprocesadores deberán presentarse por escrito. Básicamente hay que contestar a las siguientes cuestiones:

- ¿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 de la correspondencia directa? Justifica la respuesta numéricamente.
- ¿Cómo se ha tenido que modificar el programa original para poner de manifiesto el problema de la correspondencia asociativa con dos vías? Justifica la respuesta numéricamente.

Todas las respuestas deben justificarse con la ayuda de gráficas.