Examen Multiprocesadores

Primavera 2006

Preguntas (sin apuntes 60')

Responder razonadamente las siguientes cuestiones:

1. A que se refiere el término "source based routing"
2. Explicar la siguiente versión de la ley de Amdahl. Indicar que representa y en que se mide el factor "o".

$$S(p) = \frac{1}{(1-f) + \frac{f}{p} + o*(p-1)}$$

3. Explicar cómo se implementa una comunicación MPI punto a punto en un sistema con memoria compartida.
4. Diferencia entre los conceptos de coherencia y consistencia.
5. Explicar las transiciones de salida del estado E en un protocolo MESI en un sistema basado en bus.

Problema 1

Tenemos un sistema con 64 nodos cada uno de ellos SMP de cuatro procesadores con cache privada L2 de 4MB y 8 GB de memoria física compartida. La línea de caches es de 128 bytes. Existe un único adaptador de red (NIC) por cada nodo capaz de inyectar y drenar (full duplex) 250 MB/s de la red. La implementación de MPI es por memoria compartida dentro del nodo. El procesador es de 1 GHz. El ancho de banda soportado por el bus de 1GB/s. El tiempo de fallo en L2 sin contención es de 200 ciclos.

Tenemos dos aplicaciones con las siguientes características:

Aplicación A: cada iteración tiene una fase de cálculo de 10^9 instrucciones y una de comunicación en la que cada proceso intercambia con el anterior y posterior 25 MB. En la fase de cálculo el IPC que podría obtener sin fallos de cache es de 1. La proporción de fallos de cache es de 4 por cada 1000 instrucciones si el tamaño del problema no cabe en la cache. El tamaño total de las estructuras de datos es de 64 GB.

Aplicación B: Es moldable. La fase de cálculo tiene un total (perfectamente divisible entre el numero de procesadores que se asigne) de $960*10^8$ instrucciones con un IPC ideal de 1 y una proporción insignificante de fallos de L2 por cada 1000 instrucciones. La comunicación es de tipo Alltoall con tamaño para cada destino de 125/#procs MB

Preguntas sobre la aplicación A

1. Cuanto tiempo tarda la fase de comunicación si se ejecuta el programa A con 64 procesos mapeados uno en cada nodo? Suponer que no hay conflictos en la red.
2. Cuantos MIPS obtiene el programa A en ese caso? Suponer que no se pueden solapar varios fallos.
3. Cuanto tarda la fase de comunicación si se ejecuta el programa A en 16 nodos mapeado de forma contigua?
4. Cuantos MIPS obtiene el programa A en este caso?
5. Describir que pasaría si se entrelazaran los procesos en nodos?
6. Es de esperar alguna superlinealidad en el speedup para este sistema? Por que?

Preguntas sobre la aplicación B

7. Hacer una estimación del tiempo que tarda la fase de comunicación del programa B cuando ejecuta en 64 nodos.
8. Y se se mapean los procesos consecutivos en 16 nodos?
9. Que MIPS obtiene el programa B con 64 procesos en los dos casos anteriores?
10. Y si se ejecuta con 16 procesos en 16 nodos?

Preguntas sobre planificación:

11. Se envían al sistema de colas un trabajo A de 64 procesos y 1000 iteraciones y dos trabajos B moldables de 1000 y 2000 iteraciones respectivamente. Justificar como los planificarias.

Problema 2

Diseñamos un procesador con un par de instrucciones nuevas que llamamos BT (begin transaction) ET (end transaction). Todo lo que se ponga entre las dos se va ejecutando, pero si el ET detecta que el efecto de la ejecución no ha sido atómico se encarga de volver el contador de programa a la instrucción BT y garantizar que el código intermedio no ha tenido ningún efecto (todos los registros están como al empezar la transacción y no se ha modificado ninguna posición de memoria).

Queremos usar este procesador reaprovechando el subsistema de bus y memoria de un diseño snoop estandar.

Por otro lado tenemos el siguiente código:

```
void f (int index, double limit, double incr) {
double sum=0;
node *p=root;
```

```
node *pprev;

while  ((p!=NULL) && (sum<limit))  {
  sum += p->val;
  pprev=p;
  p=((p->index<index)? p->left; p->righ);
  }
pprev->val +=incr;
}
```

Que es llamado

```
main() {
double limit,incr;
int i,index;
…
#pragma  …
for (i=0; i<N; i++) {
  compute(&index,&limit,&incr);
  f(limit, incr);
  }
…
}
```

1. Comparar las instrucciones propuestas con LL-SC.
2. Implementar la instrucción CAS (compare and swap) basado en BT-ET.
3. Completar la directiva OpenMP para que cada procesador llame a la rutina exactamente N veces (no preocuparse de la problemática de atomicidad)
4. Para este programa queremos garantizar atomicidad en el sentido de que el resultado sea siempre el que corresponde a un entrelazado de iteraciones completas. Como se puede conseguir en OpenMP? Que grado de paralelismo tendría el programa? Depende de la duración promedia relativa de compute_a_while y f? Explicar. Suponiendo que *compute* es muy rápida, que grado de paralelismo tendría?
5. Modificar el programa usando las instrucciones propuestas para intentar mejorar el rendimiento en el caso de *compute* muy rápida (instantánea). Que posible grado de paralelismo tendríamos? Explicar de qué dependería.
6. Cómo debería modificarse la arquitectura del procesador original para implementar las instrucciones BT y LT sin modificar el sistema de memoria.
7. Podría la solución anterior dar lugar a algún problema de rendimiento? De que tipo? Sería de utilidad usar padding y alineamiento en la definición de la estructura node? Como?
8. (opcional) Describir una solución al problema anterior basada en etiquetar líneas de cache con números de versión (se incrementa cada vez que se escribe).