Cognoms: . . . . . Nom: . . . . . .

3er Control Arquitectura de Computadors **Curs 2011-2012 Q2**

Problema 1. (3 puntos)

Disponemos de 8 discos físicos de 1 Tbyte de capacidad por disco, que ofrecen un ancho de banda efectivo de 200 Mbytes/s por disco. Con estos discos deseamos montar un sistema de almacenamiento en RAID 5.

a) **Calcula** la cantidad de información útil (datos) que puede almacenar el sistema RAID5

(8-1)* 1 Tbyte = 7 Tbytes

El controlador del RAID, es lo suficientemente eficiente como para distribuir las peticiones aleatorias entre todos los discos del sistema.

b) **Calcula** el ancho de banda efectivo cuando realizamos lecturas aleatorias y cuando realizamos escrituras aleatorias.

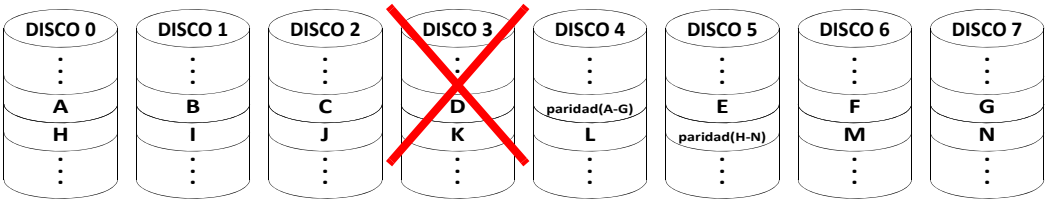
Lecturas aleatorias: AB = 200 · 8 = 1,6 GB/s

Escrituras aleatorias: AB = 200 · 8 / 4 = 400 MB/s

c) **Calcula** el ancho de banda efectivo cuando tenemos una mezcla de lecturas y escrituras aleatorias del 70% y 30% respectivamente.

Mezcla aleatorias: AB = 1,6 GB/s · 0,7 + 400MB/s · 0,3 = 1,24 GB/s

En la siguiente figura mostramos de forma parcial el RAID5, con el contenido de los bloques A, B, ... G i H, I, ... M, N, así como su paridad, que en este caso estaría almacenada en los discos 4 y 5.



d) Suponiendo que el DISCO 3, ha dejado de funcionar, qué hay que hacer para que el sistema RAID pueda obtener el bloque D.

Ha que leer A, B, C, paridad, E, F y G

D = A xor B xor C xor paridad xor E xor F xor G

En el resto del problema supondremos que siempre accedemos a bloques de 10 MB (10^7 bytes) y que un bloque está almacenado en 1 disco.

e) **Calcula** el tiempo que costaría leer 800 bloques en el RAID 5 con un disco inutilizado. Suponed que los 800 bloques están uniformemente distribuidos entre todos los discos. Para el calcular el coste de leer 1 bloque tened en cuenta exclusivamente el coste de la transferencia de información.

700 bloques se distribuyen uniformemente entre los discos operativos: equivale a 100 accesos

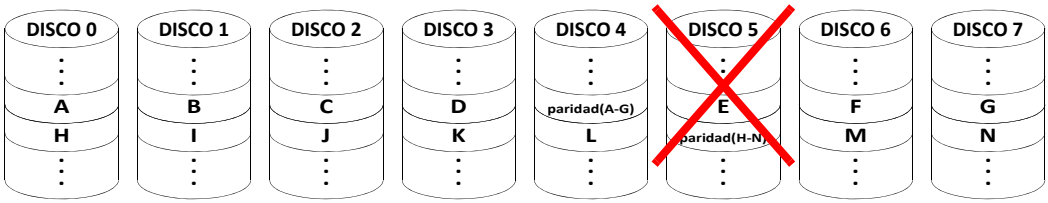
100 bloques son del disco defectuoso y obligan a leer 100 bloques de todos los discos

Tiempo = leer 200 bloques de 10MB / 200 MB/s = 10 s

f) **Calcula** el ancho de banda de leer los 800 bloques del apartado anterior.

AB = 800 · 10MB / 10 s = 800 MB/s

En la siguiente figura mostramos de forma parcial el RAID5, con el contenido de los bloques A, B, ... G i H, I, ... M, N, así como su paridad, que en este caso estaría almacenada en los discos 4 y 5.



g) Suponiendo que el DISCO 5, ha dejado de funcionar, qué hay que hacer para escribir el bloque D.

leer bloque D, paridad

paridad' = nuevo K xor K xor paridad

escribir paridad' y nuevo K

h) Suponiendo que el DISCO 5, ha dejado de funcionar, qué hay que hacer para escribir el bloque E.

leer A, B, C, D, F y G

paridad' = A xor B xor C xor D xor E' xor F xor G

escribir paridad'

i) Suponiendo que el DISCO 5, ha dejado de funcionar, qué hay que hacer para escribir el bloque M.

escribir M

Cuando en un RAID 5 se estropea un disco, se ha de sustituir por uno nuevo y rehacer su contenido.

j) **Explica** brevemente cómo sería el algoritmo para rehacer el contenido del disco sustituido en un RAID 5.

```
for (i=0; i<NumBloques; i++) {
  leer bloque[i] de todos los discos (- nuevo)
  calcular xor de los bloques leidos
  escribir nuevo bloque en el nuevo disco
}
```

k) Si tenemos en cuenta exclusivamente el tiempo de transferencia, cuánto tiempo cuesta reconstruir un disco defectuoso en nuestro RIAD 5.

Podemos suponer que las lecturas y escrituras se acaban solapando.

Sólo hay que contar el tiempo de escritura: 1 TB / 200 MB/s = 5000 s

Cognoms: . . . . . Nom: . . . . . .

3er Control Arquitectura de Computadors **Curs 2011-2012 Q2**

Problema 2. (4 puntos)

Tenemos una aplicación formada por dos fases. En una versión secuencial de la aplicación la Fase 1 representa el 20% del tiempo mientras que la Fase 2 representa el 80% del tiempo. Hemos programado una versión paralela de la misma aplicación para un multiprocesador con 4 cores integrados en el chip. La Fase 1 no ha podido ser paralelizada, por lo que se ejecutara en un solo core, sin embargo la Fase 2 ha podido ser paralelizada en 4 partes iguales de forma que el trabajo se distribuye equitativamente entre los 4 cores (el overhead debido a sincronización es despreciable).

a) **Calcula** el speed-up de la versión paralela sobre la versión secuencial.

S = 1/(0,2+0,8/4) = 2,5

Cada uno de los 4 cores tiene una intensidad de fugas de 2,5 A, una carga capacitiva equivalente de 5 nF y funciona a una frecuencia de 2 GHz y a una tensión de alimentación de 1 V. Todos los cores tienen el mismo consumo tanto si realizan trabajo útil (durante la Fase 2 los 4 cores ejecutan su parte del trabajo) como si no (durante la Fase 1 tres de los cores no realizan ningún trabajo útil pero siguen funcionando a la misma frecuencia y voltaje).

b) **Calcula** la potencia disipada por el chip durante la ejecución de la versión paralela.

P fugas = I*V = 2,5 A * 1 V = 2,5 W por core
P conmutación = $C*V^2*F = 5x10^{-9}$ F * $(1\ V)^2$ * $2x10^9$ Hz = 10 W por core
P chip = 4 * (2,5 W + 10 W) = 50 W

Los procesadores actuales pueden pasar un core a modo bajo consumo cuando no está realizando ningún trabajo útil y devolverlo a modo activo cuando el core vuelve a realizar trabajo útil. En nuestro caso, en modo bajo consumo, el voltaje del core se reduce a 0,8 V y la frecuencia a 1 GHz.

c) **Calcula** la potencia disipada durante la Fase 1 y la potencia media disipada durante la ejecución de la versión paralela con el modo bajo consumo activado.

Fase 1:
P fugas = 2,5 A * 0,8 V = 2 W por core
P conmutación = $5x10^{-9}$ F * $(0,8\ V)^2$ * $1x10^9$ Hz = 3,2 W por core
P chip = 12,5 W + 3 * (2 W + 3,2 W) = 28,1 W

(nota: en la versión paralela cada fase representa el 50% del tiempo)
P media aplicacion= 0,5 * 28,1 W + 0,5 * 50 W = 39,05 W

El TDP (Thermal Design Point) de un chip es la potencia máxima que el fabricante garantiza puede disipar sin que su funcionamiento se vea comprometido. En nuestro caso el TDP del chip se corresponde a la potencia disipada con los 4 cores activos (Fase 2). Sin embargo, con el modo bajo consumo activado, durante la Fase 1 se está disipando menos potencia ya que solo uno de los cores realiza trabajo útil. Los últimos procesadores aparecidos en el mercado son capaces de aumentar la frecuencia de funcionamiento, de modo que durante las fases en que algunos cores están en modo bajo consumo (no hacen trabajo útil), el resto puedan realizar su trabajo más rápido, siempre que no se supere el TDP. Este modo lo denominaremos modo turbo (AMD lo denomina Turbo Core e Intel Turbo Boost). En nuestro caso, en modo turbo un core aumenta el voltaje a 1,2 V y la frecuencia a 3 GHz. Obsérvese que durante la Fase 1 el procesador activo estaría en modo turbo, mientras el resto estarían en modo bajo consumo.

d) **Calcula** el speed-up del multiprocesador con modo turbo respecto al que solo tenia modo bajo consumo.

Aumentar la frecuencia a 3 GHz ---> speedup fase 1 = 3 GHz / 2 GHz = 1,5

S = 1/(0,5+0,5/1,5) = 1,2

e) **Calcula** la potencia disipada por la Fase 1 con modo turbo.

P fugas = 2,5 A * 1,2 V= 3 W por core turbo
P conmutación = $5x10^{-9}$ F * $(1,2\ V)^2$ * $3x10^9$ Hz = 21,6 W por core turbo
P chip = 3W + 21,6 W + 3 * 5,2 W = 40,2 W

f) **Calcula** la potencia media disipada con modo turbo.

La Fase 2 representa (0,5*t0)/(t0/1,2) = 60% del tiempo
La Fase 1 representa (0,5/1,5*t0)/(t0/1,2) = 40% del tiempo

P media = 0,4 * 40,2 W + 0,6 * 50 W = 46,08 W

g) **Calcula** la ganancia en energía del multiprocesador con modo turbo respecto al que solo tenia modo bajo consumo.

E = P*t ---> Ebc = Pbc*tbc ; Et = Pt*tbc/1,2

G = Pbc*tbc/(Pt*tbc/1,2) = Pbc/Pt*1,2 = 39,05 W / 46,08 W * 1,2 = 1,017

En este multiprocesador, cada core tiene una cache de datos (D1) y una de instrucciones (I1) local. Además, los 4 cores comparten una cache unificada de segundo nivel (L2). Las caches de datos de los 4 cores se comunican entre ellas y con L2 por medio de un bus compartido.

Con un juego de datos determinado, la versión secuencial de la aplicación ha tardado $15x10^9$ ciclos y se han realizado $5x10^9$ accesos a datos (el impacto de los accesos a instrucciones es despreciable). En la versión secuencial, la tasa de fallos en el primer nivel de cache de datos (D1) ha sido del 10% y se ha medido un tiempo de penalización medio por fallo de 20 ciclos (incluye los accesos a L2 y los fallos de L2 que se han servido desde memoria principal). La tasa de fallos y el tiempo de penalización es el mismo en la Fase 1 y la Fase 2, y el numero de accesos se distribuye proporcionalmente al tiempo de cada fase (20% y 80% respectivamente)

h) **Calcula** cuantos ciclos tardaría la Fase 2 en la versión secuencial si no hubiese fallos en la cache de datos.

Fase 2 sequencial:
Ciclos secuencial = $15x10^9$ ciclos *0,80 = $12x10^9$ ciclos
Accesos sequencial = $5x10^9$ ciclos *0,8 = $4x10^9$ acccesos
Ciclos secuencial ideal = $12x10^9$ciclos - $4x10^9$ accesos * 0,10 fallos/acceso * 20 ciclos/fallo = $4x10^9$ciclos

Para estudiar el impacto de la jerarquía de memoria en la versión paralela supondremos (por simplicidad) que todos los cores funcionan siempre a la misma frecuencia. En la versión paralela la Fase 1 se comporta (tasa de fallos y tiempo de penalización) exactamente igual que en la versión secuencial, dado que se ejecuta sobre un solo core. Durante la Fase 2, la tasa de fallos de las caches de datos (D1) ha aumentado al 15% por las invalidaciones debidas al mecanismo de coherencia entre los distintos cores. Respecto al tiempo de penalización por fallo, en la Fase 2, pueden darse 2 situaciones: 1) el bloque de memoria puede obtenerse de la cache D1 de otro core, en cuyo caso el tiempo de penalización es de 6 ciclos; 2) debe obtenerse de L2, en cuyo caso el tiempo de penalización medio es de 21 ciclos (ligeramente mayor que en la Fase 1 debido a la mayor saturación del bus de datos). Se ha observado que el 60% de los fallos son servidos desde otro core y el resto desde L2.

i) **Calcula** cuantos ciclos tardaría la versión paralela teniendo en cuenta la jerarquía de memoria.

Fase 1: Ciclos = $15x10^9$ ciclos *0,20 = $3x10^9$ ciclos

Fase 2 paralela:
Ciclos ideal por core = $4x10^9$ciclos/4 = $1x10^9$ciclos
Accesos por core = $4x10^9$ accesos/4 = $1x10^9$accesos
Ciclos por core = $1x10^9$ciclos + $1x10^9$accesos * 0,15 f/a * (0,6 * 6 c/f + 0,4 * 21 c/f) = $2,8x10^9$ ciclos

Ciclos Paralela = $3x10^9$ ciclos + $2,8x10^9$ ciclos = $5,8x10^9$ ciclos

En teoría, al paralelizar una aplicación, como máximo se puede obtener un speed-up igual al número de procesadores. Sin embargo hay ocasiones en que se obtiene un speed-up superlineal (mayor que el número de procesadores).

j) **Calcula** el Speed-up de la Fase 2 en la versión paralela respecto la secuencial y en caso de que sea superlineal explica a que es debido.

Speed-up = $12x10^9$ ciclos / $2,8x10^9$ ciclos = 4,29

A pesar de que hay más fallos, una parte importante de ellos es servida desde las caches de los otros cores, con lo que el tiempo de penalización medio por fallo es menor y por tanto se reduce tiempo de ejecución más de lo que cabria esperar para 4 procesadores.

Cognoms: . . . . . Nom: . . . . . .

3er Control Arquitectura de Computadors **Curs 2011-2012 Q2**

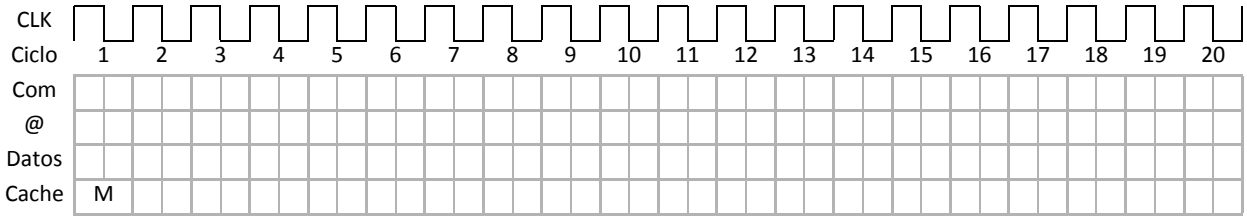
Problema 3. (3 puntos)

Un sistema de bajo consumo está formado por un procesador de 32 bits a 1GHz con una memoria cache de datos con líneas de 64 bytes y una memoria de instrucciones SDRAM formada por un DIMM de 8 bytes de ancho con una latencia de fila de 5 ciclos, una latencia de columna de 4 ciclos y un tiempo para el comando PRECHARGE de 2 ciclos que transfiere ráfagas de 64 bytes. En este problema solo tendremos en cuenta los accesos a datos.

En el sistema de memoria descrito realizamos un acceso a datos en lectura con fallo de cache. Para indicar la ocupación de los distintos recursos utilizaremos la siguiente nomenclatura:

- ACT: comando ACTIVE
- RD: comando READ
- PRE: comando PRECHARGE
- @F: ciclo en que se envía la dirección de fila
- @C: ciclo en que se envía la dirección de columna
- Di: ciclo en que se transmite el paquete de datos i (D0, D1, D2, ?)
- M: miss de cache
- D: dato servido desde la cache

a) **Rellena** el siguiente cronograma indicando la ocupación de los distintos recursos.



b) **Calcula** el ancho de banda de pico y el ancho de banda efectivo de la memoria principal suponiendo que en cada acceso a memoria principal leemos una línea útil de 64 bytes y realizamos un nuevo acceso a datos tan pronto hemos completado el acceso anterior.

Peak BW=8 bytes/1x10^{-9}s=8Gbytes/s

Ef BW=64 bytes/19x10^{-9}s=3,37Gbytes/s

El programa que se está testeando en el sistema anterior ejecuta un bucle de 10^9 de iteraciones. Si probamos este programa con una memoria principal ideal (que tiene un tpf de 0 ciclos) vemos que dicho programa genera un fallo de cache de datos en cada iteración, exactamente cada 10 ciclos.

c) **Calcula** el tiempo de ejecución del programa y el ancho de banda real obtenido con la memoria principal real si en cada fallo de cache leemos un dato útil de 4 bytes que la cache sirve al ciclo siguiente de obtener la línea desde la memoria principal.

T=10^9 iter x (10 c/iter +18 c/iter) x 10^{-9} s/c=28s

BW=4bytesx10^9/28=143Mbytes/s

Con los datos anteriores se observa que la memoria es el gran cuello de botella del sistema. También se ha medido en el sistema con memoria principal ideal que el dato que provoca el fallo de cache no se necesita hasta 32 ciclos después de su correspondiente load. Así pues, y dado que el código estudiado no permite accesos en paralelo a la memoria principal, se decide usar una memoria cache no bloqueante con un solo MSHR para mejorar el sistema.

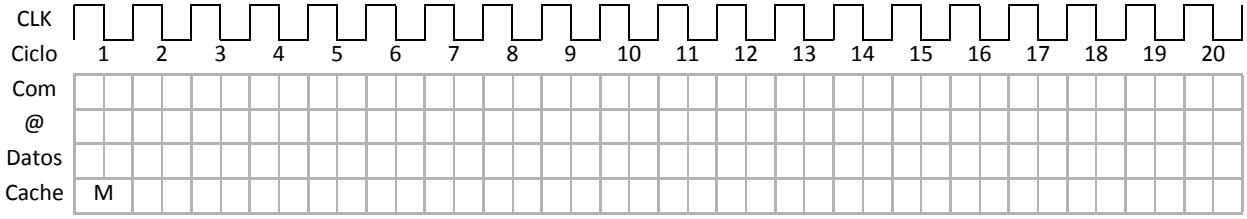
d) **Calcula** el nuevo tiempo de ejecución del programa con esta mejora..

T= 10^9 accesos x 19 c/acceso x 10^{-9} s/c = 19s

Limitado por la máxima velocidad de acceso a la memoria

Dado que se ve que a pesar de la mejora anterior, la memoria principal sigue siendo el cuello de botella del sistema se decide cambiar la SDRAM por una DDR SDRAM con Transferencia en Desorden y Continuación Anticipada.

e) **Rellena** el siguiente cronograma indicando la ocupación de los distintos recursos en este último caso.



f) **Calcula** el nuevo tiempo de ejecución del programa con todas las mejoras introducidas..

T= 10^9 accesos x 15 c/acceso x 10^{-9} s/c =15s

El límite sigue siendo el tiempo entre accesos consecutivos a la memoria principal. En realidad la Transferencia en Desorden y la Continuación Anticipada no sirven de nada en este caso.

Finalmente se decide reprogramar el código de forma que dos accesos consecutivos a datos que provocan fallo de cache vayan a parar a bancos distintos de la memoria principal ya que se sabe que la DDR SDRAM empleada es capaz de gestionar dichos accesos en paralelo.

g) **Explica** razonadamente cuantos MSHRs son necesarios como mínimo en el sistema descrito para aprovechar al máximo las capacidades de la cache no bloqueante con el código de prueba..

2, así se pueden realizar accesos en paralelo a la memoria y se esconde totalmente su latencia.

Al realizar esta mejora vemos que el cuello de botella del sistema deja de ser la memoria y nos planteamos que un sistema de memoria más sencillo podría ser suficiente para nuestras necesidades y gastar menos energía.

h) De las mejoras anteriores marcad cuales de ellas podrían eliminarse sin penalizar el tiempo de ejecución del programa con el código reprogramado.

Caché no bloqueante	
Memoria principal DDR	X
Transferencia en desorden	X
Continuación anticipada	X