Legendario_115795
www.wuolah.com/student/Legendario_115795

Test EC practica6.pdf

Tests de EC

2º Estructura de Computadores

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
UGR - Universidad de Granada

Practicas tema 6 EC

1. En la práctica de la cache, el código de "size.cc" accede al vector saltando de 64 en 64. ¿Por qué?

a) Porque cada elemento del vector ocupa 64 bytes

b) Para recorrer el vector más rápidamente

c) Porque el tamaño de cache L1 de todos los procesadores actuales es de 64KB

-> d) Para anular los aciertos por localidad espacial, esto es, que sólo pueda haber aciertos por localidad temporal

2. ¿Cuál de las siguientes afirmaciones sobre las caches es *FALSA*?

a) La cache de nivel 3 no contiene toda la memoria que maneja el programa

b) Las direcciones a las que accede un programa no son completamente aleatorias, sino que se rigen por ciertos patrones de localidad

c) Un procesador actual tiene varias caches de nivel 1

-> d) Casi ningún procesador actual tiene memoria cache L2

3. ¿Cuál de las siguientes afirmaciones sobre el programa size.cc de la práctica 5 es cierta?

a) La diferencia de velocidades entre L2 y L3 es mayor que la diferencia de velocidades entre L1 y L2.

puede que haya sistemas con esa característica, pero desde luego los que usamos nosotros en prácticas no.

b) Si continuáramos multiplicando por 2 el tamaño del vector en el eje X obteniendo más puntos de la gráfica, esta continuaría horizontal para cualquier valor más allá de 64 MB.

-> c) La gráfica tiene escalones hacia arriba porque en cada punto del eje X accedemos al mismo número de elementos del vector y el número de aciertos por localidad temporal disminuye bruscamente en ciertos puntos al aumentar el tamaño del vector.

si cabe el vector en cache, los millones de accesos (siempre el mismo número de accesos) son todo aciertos por localidad temporal

d) La gráfica tiene tramos horizontales porque el hecho de realizar la mitad de accesos al vector en cada punto de un tramo horizontal respecto al anterior punto de ese mismo tramo horizontal es compensado por el número de fallos creciente en ese mismo tramo horizontal.

4. El servidor de SWAD tiene dos procesadores Xeon E5540 con 4 núcleos cada uno. Cada procesador tiene 4 caches L1 de instrucciones de 32 KB, 4 caches L1 de datos de 32 KB, 4 caches unificadas L2 de 256 KB y una cache unificada L3 de 8MB. Suponga que un proceso swad, que se ejecuta en un núcleo, tiene que ordenar un vector de estudiantes accediendo repetidamente a sus elementos. Cada elemento es una estructura de datos de un estudiante y tiene un tamaño de 4KB. Si representamos en una gráfica las prestaciones en función del número de estudiantes a ordenar, ¿para qué límites teóricos en el número de estudiantes se observarán saltos en las prestaciones debidos a accesos a la jerarquía de memoria?

a) 32 / 256 / 8192 estudiantes

b) 16 / 32 / 64 estudiantes

-> c) 8 / 64 / 2048 estudiantes

d) 4 / 32 / 512 estudiantes

5. En el programa "size" de la práctica de la cache, si el primer escalón pasa de tiempo = 1 para todos los tamaños de vector menores o iguales que 32 KB a tiempo = 3 para los tamaños 64 KB y 128 KB, podemos asegurar que:

(Nota: en las opciones de respuesta, "rapidez" se refiere a Ti, el tiempo de acceso efectivo al nivel i, no a ti, el tiempo de acceso propio al nivel i. Si dice "La cache L1 es 3x más rápida que L2", se refiere a que T1=T2/3, no a que t1=t2/3. Ver T6 tr.26)

_____

a) la cache L2 es como mucho el doble de rápida que la memoria principal

b) la cache L2 es al menos el doble de rápida que la memoria principal

c) la cache L1 es como mucho tres veces más rápida que la cache L2

-> d) la cache L1 es al menos tres veces más rápida que la cache L2

sólo sería exactamente 3 si el tiempo de cómputo adicional a los accesos de memoria fuera despreciable. Como en general no lo es, los accesos deben ser aún más rápidos para que el incremento sea 3x.

Matemáticamente, el enunciado dice que c+m1=1, c+m2=3, entonces (c+m2)/(c+m1)=3, y despejando sale m1=(m2)/3-2c/3, es decir, sólo si c=0 entonces m1=(m2)/3, y si c>0 entonces m1<(m2)/3

6. Suponer una memoria cache con las siguientes propiedades: Tamaño: 512 bytes. Política de reemplazo: LRU. Estado inicial: vacía (todas las líneas inválidas). Suponer que para la siguiente secuencia de direcciones enviadas a la cache: 0, 2, 4, 8, 16, 32, la tasa de acierto es 0.33. ¿Cuál es el tamaño de bloque de la cache?

a) 4 bytes

-> b) 8 bytes

c) 16 bytes

d) Ninguno de los anteriores

7. En la práctica de la cache, el código de line.cc incluye la sentencia

```
for (unsigned long long line=1;
    line<=LINE; line<<=1) { ... }
```

¿Qué objetivo tiene la expresión line<<=1?

a) sacar un uno (1) por el stream line

b) volver al principio del vector cuando el índice exceda la longitud del vector

-> c) duplicar el tamaño del salto en los accesos al vector respecto a la iteración anterior

d) salir del bucle si el tamaño de línea se volviera menor o igual que 1 para algún elemento del vector

8. En la práctica de cache hemos hecho una gráfica con el código size.cc ¿Qué forma tiene la gráfica que se debe obtener?

a) Forma de U (o V) con un tramo descendente y otro ascendente

b) Forma de U (o V) invertida, con un tramo ascendente y otro descendente

c) Forma de /, una gráfica siempre creciente y sin escalones

-> d) Una escalera con varios tramos horizontales

9. Abajo se ofrece el listado de una función para multiplicar matrices C = A x B.

```
void mult_matr(float A[N][N], float B[N][N], float C[N][N]){
/* Se asume valor inicial C = {0,0...} */
int i,j,k;
for (i=0; i<N; i++)
 for (j=0; j<N; j++)
  for (k=0; k<N; k++)
   C[i][j] += A[i][k] * B[k][j];
}
```

Suponer que:

- **El computador tiene una cache de datos de 8 MB, 16-vías, líneas de 64 bytes.**
- **N es grande, una fila o columna no cabe completa en cache.**
- **El tamaño de los tipos de datos es como en IA32.**
- **El compilador optimiza el acceso a C[i][j] en un registro.**

Imaginar que se modifica la última sentencia (el cuerpo anidado) por esta otra

```
  C[i][j] += A[i][k] * B[j][k];
```

de manera que se calcule C = A x B' (A por traspuesta de B). Aproximadamente, ¿qué tasa de fallos se podría esperar de esta nueva función para valores grandes de N?

a) 1/8

b) 1/4

c) 1/2

-> d) 1/16

uno de cada 16 A[i][k] y otro de cada 16 B[i][k]

10. En el programa "size" de la práctica de la cache, si el primer escalón pasa de tiempo=2 para un tamaño de vector menor que 32KB a tiempo=8 para un tamaño mayor que 32KB, podemos asegurar que:

_____

(Nota: en las opciones de respuesta, "rapidez" se refiere a Ti, el tiempo de acceso efectivo al nivel i, no a ti, el tiempo de acceso propio al nivel i. Si dice "La cache L1 es 4x más rápida que L2", se refiere a que T1=T2/4, no a que t1=t2/4. Ver T6 tr.26)

_____

a) La cache L1 es como mucho cuatro veces más rápida que la cache L2

b) La cache L1 es seis veces más rápida que la cache L2

-> c) La cache L1 es al menos cuatro veces más de rápida que la cache L2

sólo sería exactamente 4 si el tiempo de cómputo adicional a los accesos de memoria fuera despreciable. Como en general no lo es, los accesos deben ser aún más rápidos para que el incremento sea 4x.

Matemáticamente, el enunciado dice que c+m1=2, c+m2=8, entonces (c+m2)/(c+m1)=4, y despejando sale m1=(m2)/4-3c/4, es decir, sólo si c=0 entonces m1=(m2)/4, y si c>0 entonces m1<(m2)/4

d) La cache L1 es cuatro veces más rápida que la cache L2

11. En la práctica de la cache, el código de size.cc accede al vector saltando de 64 en 64. ¿Por qué?

a) Porque cada elemento del vector ocupa 64 bytes

b) Para recorrer el vector más rápidamente

c) Porque el tamaño de cache L1 de todos los procesadores actuales es de 64KB

-> d) Para anular los aciertos por localidad espacial, esto es, que sólo pueda haber aciertos por localidad temporal

12. Sea un computador de 32 bits con una memoria cache L1 para datos de 32 KB y líneas de 64 bytes asociativa por conjuntos de 2 vías. Dado el siguiente fragmento de código:

```
int v[262144];
for (i = 0; i < 262144; i += 8)
  v[i] = 9;
```

¿Cuál será la tasa de fallos aproximada que se obtiene en la ejecución del bucle anterior?

a) 0 (ningún fallo)

-> b) 1/2 (mitad aciertos, mitad fallos)

es un array de ints (4B) y se salta i+=8, los accesos están separados 32B y las líneas son de 64B, los accesos con índice i par son fallos, con índice impar son aciertos.

c) 1/8 (un fallo por cada 8 accesos)

d) 1 (todo son fallos)

13. En la práctica de la cache, el código de "line.cc" incluye la sentencia

```
for (unsigned long long line=1; line<=LINE; line<<=1) { … }
```

¿Qué objetivo tiene la expresión line<<=1?

a) sacar un uno (1) por el stream line

b) volver al principio del vector cuando el índice exceda la longitud del vector

c) salir del bucle si el tamaño de línea se volviera menor o igual que 1 para algún elemento del vector

-> d) duplicar el tamaño del salto en los accesos al vector respecto a la iteración anterior

15. Abajo se ofrece el listado de una función para multiplicar matrices C = A x B.

```
void mult_matr(float A[N][N], float B[N][N], float C[N][N]){
/* Se asume valor inicial C = {0,0...} */
int i,j,k;
for (i=0; i<N; i++)
 for (j=0; j<N; j++)
  for (k=0; k<N; k++)
   C[i][j] += A[i][k] * B[k][j];
}
```

Suponer que:

- **El computador tiene una cache de datos de 8 MB, 16-vías, líneas de 64 bytes.**
- **N es grande, una fila o columna no cabe completa en cache.**

- **El tamaño de los tipos de datos es como en IA32.**
- **El compilador optimiza el acceso a C[i][j] en un registro.**

Aproximadamente, ¿qué tasa de fallos se podría esperar de esta función para valores grandes de N?

-> a) 1/2

1/2 por cada B[k][j] + 1/32 por cada 16-ésimo A[i][k]

b) 1/4

c) 1/16

d) 1/8

16. El código del programa "size" de la práctica de la cache accede al vector saltando...

a) de byte en byte

-> b) de 64 en 64 bytes

es el tamaño de línea deducido en la práctica "line"

c) de 1 KB en 1 KB

d) de 64 KB en 64 KB

18. En el programa line.cc de la práctica 5, si para cada tamaño de línea (line) recorremos una única vez el vector, la gráfica resultante es decreciente porque:

a) Cada vez que line aumenta al doble, el número de aciertos por localidad temporal aumenta, porque ya habíamos accedido a cada posición i del vector cuando lo recorrimos en el punto anterior del eje X.

b) Cada vez que line aumenta al doble, el número de aciertos por localidad espacial aumenta, porque ya habíamos accedido a cada posición i-1 del vector cuando lo recorrimos en el punto anterior del eje X.

c) Cada vez que line aumenta al doble, se accede con éxito a más posiciones del vector en niveles de la jerarquía de memoria más rápidos.

-> d) Cada vez que line aumenta al doble, realizamos la mitad de accesos al vector que para el valor anterior.

19. Sea un computador de 32 bits con una memoria cache L1 para datos de 32 KB y líneas de 64 bytes asociativa por conjuntos de 2 vías. Dado el siguiente fragmento de código:

```
int v[262144];

for (i = 0; i < 262144; i += 2)

  v[i] = 9;
```

¿Cuál será la tasa de fallos aproximada que se obtiene en la primera ejecución del bucle anterior?

a) 0 (ningún fallo)

b) 1/2 (mitad aciertos, mitad fallos)

-> c) 1/8 (un fallo por cada 8 accesos)

d) 1 (todo son fallos)