Arquitectura de Computadores

Práctica 1

Jerarquía de memoria y comportamiento de memoria caché: Estudio del efecto de la localidad de los accesos a memoria en las prestaciones de programas en microprocesadores

Objetivos: mediante la realización de un programa en lenguaje C, caracterizar el coste temporal por lectura (acceso) de datos, variando diferentes parámetros en el patrón de acceso a los datos y el tamaño del conjunto de datos. Observar e interpretar el efecto del sistema de memorias caché del microprocesador teniendo en cuenta los conceptos de localidad espacial, localidad temporal y precarga (*prefetching*).

Equipos de prácticas: grupos de dos personas.

Sistema para la toma de datos: FinisTerrae III del CESGA.

Tiempo de trabajo presencial: 4 sesiones

Plazo de entrega: Para cada equipo la fecha límite de entrega corresponde con la cuarta sesión de prácticas. Se recomienda la entrega anticipada de los diferentes apartados que forman parte de la práctica.

Forma de entrega y formato: presentar en el aula al profesor los experimentos realizados. En esta entrega se debe incluir: i) un esquema en el que se detalle la metodología seguida y el trabajo realizado, ii) el código fuente incluyendo comentarios y, iii) las figuras o tablas obtenidas y su correspondiente interpretación.

Valoración: 40% de la nota de prácticas (hasta 4 puntos sobre 10 de la nota de prácticas).

Hacer un programa en C que reserve memoria dinámica para un vector (*array*) A[] de números representados en punto flotante tipo *Double*. Llevar a cabo una operación de reducción de suma de punto flotante sobre los R elementos del vector siguientes: A[0], A[D], A[2D], A[3D],....,A[(R-1)*D] (es decir, sumar estos R elementos almacenando el valor en una variable tipo *Double*). D es un parámetro que vamos a variar para estudiar el efecto de la localidad. La referencia a los

R; parámetro que depende de caché

D; parámetro arbitrario. Indica a "separación"

elementos de A[] debe hacerse indirectamente a través de un vector de enteros ind[], es decir, hacer referencias del tipo A[ind[i]], con ind[i]=0, D, 2D, 3D, etc.

Repetir 10 veces la ejecución de la reducción para luego calcular el valor medio. Para cada repetición almacenar el resultado en un vector S[] de 10 elementos tipo *Double*. Imprimir todos los elementos de S[] (los resultados) al terminar. Mediante este proceso evitamos que el compilador realice optimizaciones que en este caso no deseamos.

Tomar la medida de ciclos (utilizando las rutinas que aparecen en el programa C adjunto) totales de las 10 repeticiones (no incluir la impresión de los resultados en esta medida), y obtener el **número de ciclos medio por acceso a memoria**.

Repetir el experimento para diferentes valores de R y D.

Valores de R: para un valor específico de D, es necesario realizar un total de 7 medidas. En cada medida seleccionamos el valor de R de modo que las lecturas del vector A[] correspondan a un total de L líneas caché diferentes. Tomar medidas de ciclos para los siguientes valores de L: 0.5*S1, 1.5*S1, 0.5*S2, 0.75*S2, 2*S2, 4*S2, 8*S2, siendo S1 el número de líneas caché que caben en la caché L1 de datos y S2 el número de líneas caché que caben en la caché L2.

Será necesario investigar cuales son los valores de S1 y S2. Es necesario estar seguro de que estos valores son los correctos, ya que determinarán la calidad de los experimentos.

Valores de D: 5 valores potencias de 2. Los valores de D deben seleccionarse de modo que los experimentos realizados permitan estudiar de forma representativa los diferentes efectos de la localidad de acceso a los datos. La selección de los valores de D adecuados es una parte muy importante del experimento.

Número de elementos do vector A[]: determinado por los valores seleccionados de R y D. Para la reserva de memoria resulta conveniente alinear el comienzo del vector con el comienzo de una línea caché. Una alternativa para hacer esto es utilizar la función *_mm_malloc(TC,CLS), donde TC es el número de bytes a reservar y CLS es el tamaño de línea caché en bytes. Esta función devuelve un puntero alineado a CLS bytes. Para liberar la memoria se recomienda utilizar la correspondiente función _mm_free(P), donde P es el puntero que apunta al comienzo del vector de memoria reservado. Para utilizar estas funciones es necesario poner esta línea en el archivo: #include <pmmintrin.h>. En la compilación con gcc es necesario utilizar la opción -msse3.

Se proporciona con este enunciado un programa C con las rutinas de medida de tiempo necesarias que incluye también la rutina mhz(). Esta rutina imprime la

frecuencia de reloj del procesador.

EXPERIMENTOS ADICIONALES: Para la obtención de la máxima calificación en la práctica será necesario realizar experimentos adicionales sobre lo propuesto anteriormente. Los experimentos adicionales que se proponen son dos:

- Comparar los resultados con los obtenidos al utilizar datos de tipo *Int* en lugar de *Double* para el array.
- Comparar los resultados con los obtenidos si se utilizan referencias directas a los elementos de A[] en lugar de hacerlo a través del vector de índices.

Presentación de resultados: Hacer una representación gráfica del coste en ciclos de cada acceso a memoria (eje Y) vs. el número total de líneas cache diferentes del vector A[] que se referencian (eje X) para los diferentes valores de D. Se pueden hacer representaciones gráficas adicionales si se considera necesario para explicar los resultados e incorporar tablas de datos.

Interpretación de resultados: Se deben interpretar los resultados obtenidos en relación con los conceptos de localidad temporal y espacial en el acceso a los datos. Además, en los microprocesadores se suele realizar precarga (*prefetching*) de datos. Busca información sobre cómo se realiza la precarga en el procesador que utilizas.

Variabilidad de los resultados: es recomendable tomar varias medidas para un mismo experimento para observar la variabilidad. Una estrategia posible es tomar 10 medidas y quedarse con las 3 mejores (menor coste en ciclos por iteración) y hacer la media geométrica de estos tres valores.

Calentamiento: para evitar efectos de inicialización de las cachés, de las TLBs, etc, es interesante realizar una fase de calentamiento antes de tomar las medidas. Para ello inicializar los datos del vector A[] que se van a leer en el bucle de computación. Para la inicialización de los datos es interesante utilizar valores aleatorios, pero intentando que no se produzcan desbordamientos en la representación *Double*. Una forma de evitarlo es que se usen valores iniciales con valor absoluto acotado en el intervalo [1,2) con signo positivo o negativo de forma aleatoria.

Procedimiento de ejecución: en general es mejor ejecutar un experimento individual cada vez. Así reducimos la probabilidad de resultados distorsionados por el efecto del planificador del sistema operativo.

Comentarios generales: Para hacer estos experimentos es imprescindible conocer la jerarquía de memoria caché del procesador en detalle. Hay diferentes posibles formas de averiguar esta información.

Plataforma de ejecución: Realizar las ejecuciones de todos los experimentos en un nodo del FinisTerrae III siguiendo las indicaciones proporcionadas en el campus virtual. Utilizar el sistema operativo Linux y el compilador gcc con la opción -O0 (no optimización).

CRITERIOS DE EVALUACIÓN: cumplimiento de las especificaciones del enunciado de la práctica, calidad de los resultados presentados y de las explicaciones proporcionadas al profesor. También se tendrá en cuenta la autonomía en el desarrollo del trabajo y la actitud frente al trabajo. La nota será individual para cada miembro del grupo. **Obviamente la copia de programas, gráficas u otros resultados se considerará motivo de suspenso.**

Se ejecutamos o comando 'lscpu' na terminal, podemos extraer a seguinte info

- L1 (datos): 32 · 48KB
- L2: 32 · 1'25 MB
- L3: 48 MB

Á súa vez, o sistema ten certas vars; ejecutando `getconf LEVEL1_DCACHE_LINESIZE` `2_CACHE` debemos o tamaño de línea da caché de datos de nivel 1 e tmb da caché de nivel 2

O nº de líneas; $nº\ líneas = \frac{tam.\ caché}{tam.\ línea \cdot nº vías}$

$S1 = nº\ líneas\ L1D = \frac{49\,152}{64}$;

S1 = 768

lscpu -B C

Tam L1D: 49 152 bytes

Tam L2: 1 310 720 bytes

$S2 = \frac{1310\,720}{64}$; S2 = 20480

$L = \{384, 1152, 10240, 15360, 40960, 81920, 163\,840\}$

DATOS

- Valor de D; 5 potencias de 2 que permitan ver los efectos de la localidad.
(Un double son 8 bytes; $\frac{64}{8}$ = 8 doubles por línea)

$$D = \{2^0, 2^2, 2^5, 2^6, 2^7\}$$

- Valores de R;

En una línea caché (64 B) se reparten los doubles; como sizeof(double) = 8 B ; $64/8$ = 8 doubles en 1 línea.

D=1; con un offset de 1 necesitas 8 lecturas para leer 1 línea caché nueva; como las 7 lecturas "inútiles", debemos tener un R = 8L //

D=4; Con un offset de 4, cada 2 lecturas tenemos 1 línea nueva; R = 2L //

Podemos razonar a fórmula $R = \frac{8}{D} \cdot L$, siempre e cuando que D > 8; R = L. Xa que, con un D > 8, obteríamos un R < L, esto non nos permitiría acceder as L liñas diferentes.

Valores de R;

D = 1 { 3072, 9216, 81920, 122880, 327680, 655360, 1310720 }

D = 4 { 768, 2304, 20480, 30720, 81920, 163840, 327680 }

D = 32, 64, 128 { 384, 1152, 10240, 15360, 40960, 81920, 163840 }

Aligned_alloc (size_t alignment, size_t size) tiene que; alignment ser potencia de 2 ($2^6 = 64$) e size ser múltiplo de alignment (cumplese en todo valor de R)

Tipo de dato; INT

$L = \{384, 1152, 10240, 15360, 40960, 81920, 163840\}$

Valor de D:

$D = \{ 2^0, 2^2, 2^5, 2^6, 2^7 \}$

Los valores de D serán los mismos para poder realizar la comparación

Valor de R

Ahora, debemos saber que un int ocupa 4 bytes.

$64/4 = 16$ ints en una misma línea.

D = 1; con D=1, leeremos 1 línea diferente cada 16 accesos; $R = \frac{16}{D} \cdot L$; $\boxed{R = 16L}$

D = 4; leeremos una línea diferente cada 4 accesos; $R = \frac{16}{4} \cdot L$; $\boxed{R = 4L}$

Entonces; con D ≥ 16; tendremos que $R = \frac{16}{16} \cdot L$; $\boxed{R = L}$

D = 1 { 6144, 18432, 163840, 245760, 655360, 1310720, 2621440 }

D = 4 { 1536, 4608, 40960, 61440, 163840, 327680, 655360 }

D = 32, 64, 128 { 384, 1152, 10240, 15360, 40960, 81920, 163840 }

$R = \frac{16}{D} \cdot L$