Estudio del efecto de localidad de los accesos a memoria en las prestaciones de programas en microprocesadores

Cristian Novoa Gonzalez, Ivan Quintáns González

Arquitectura de Computadoras
Grupo 04
{cristian.novoa,ivan.quintans}@rai.usc.es

20 de marzo de 2023

Resumen

Realización de un estudio sobre el efecto de localidad en las prestaciones de programas en microprocesadores. Análisis y caracterización del coste temporal por lectura de datos, mediante la variación de diferentes parámetros de acceso y del tamaño del conjunto de datos, aplicando estos parámetros sobre la jerarquía de memorias cache del microprocesador, valorando la importancia de los principios de localidad espacial*,* localidad temporal *y* prefetching *en el coste por acceso a memoria del programa.*

Palabras clave: Cache, Localidad , Prefetching, Ciclos

I. Introducción

Este estudio se centra en el análisis del *efecto de localidad* y del *prefetching* en los sistemas de memoria caché.

Se realizará este estudio mediante la medición de los ciclos que tarda el microprocesador en realizar una operación de reducción de suma en punto flotante.

Esta operación se realizará sobre los R elementos de un vector A, mediante acceso indirecto, teniendo en cuenta los siguientes parámetros de entrada: La variable D, que indica el tamaño del salto en las posiciones de memoria; y la variable L, que nos indica el número de líneas caché diferentes que se desean leer.

Para la realización de este experimento se va a utilizar un nodo del CESGA con un procesador Intel(R) Xeon(R) Platinum 8352Y CPU @ 2.20G con 64 cores y 256GB de memoria ya que es un experimento muy costoso computacionalmente y en una máquina normal sería una ejecución muy llevadera y costosa.

En primer lugar se describirá la metodología empleada, incluyendo en ella la explicación de los conceptos de *prefectching* y del *efecto de localidad*, tratando también los parámetros de entrada empleados, su significado y sus valores.

En segundo lugar se explicarán los resultados obtenidos del experimento y se analizará su relación con los *principios de localidad* y *prefetching*.

A continuación se realizarán dos modificaciones sobre el experimento base: uso de tipos de datos enteros en vez de doubles y acceso directo al vector A. Se compararán los resultados obtenidos de estos nuevos experimentos con respecto al experimento base y se analizará el porqué de esta variación.

Finalmente, se describirán las conclusiones obtenidas de este estudio.

II. Entorno del experimento

A la hora de la realización del experimento es fundamental comprender el funcionamiento de la jerarquía de la cache de nuestro microprocesador y la influencia que tienen sobre este el *principio de localidad* y el *prefetching*.

A. Jerarquía de la Caché

En el nodo del CESGA proporcionado fue necesaria la búsqueda de las especificaciones de la jerarquía de la caché. Esta caché está formada por 3 niveles, de las que son más significativas para este experimento las de nivel 1 y 2 ya que pertenecen de forma exclusiva a cada uno de los microprocesadores.

En la cache de nivel 1 está divida en dos espacios: el espacio para datos, formado por 48KiB por core y 3MiB entre todos los cores, con un tamaño de línea cache de 64 bytes; y el espacio para instrucciones formado por 32 KiB por core y 2 MiB entre todos los cores y con un tamaño de línea de 64 bytes.

Por otro lugar en la cache de nivel 2 podemos encontrarnos con un espacio de datos de 1280 KiB por core y de 80 MiB entre todos los cores, con un tamaño de línea de 64 bytes.

Como el experimento está realizado para un único core y únicamente son relevantes los espacios para datos, se utilizarán en el experimento los siguientes valores:

Para nivel 1, 48Kib y para nivel 2, 1280 Kib. Ambas cachés tienen un tamaño de línea de 64 bytes. Estos tamaños si los dividimos entre el tamaño de línea, obtenemos que $S1$=768 líneas y $S2$=20480 líneas respectivamente siendo $S1$ el numero de líneas de la caché nivel 1 y siendo $S2$ el de la cache nivel 2 .

B. Principio de Localidad y Prefetching

Estos dos fenómenos afectan directamente a la funcionalidad y al comportamiento de la jerarquía de la caché los cuales afectan de una manera bastante significativa en el experimento por lo que es conveniente una breve explicación de estos factores.

En primer lugar, el *principio de localidad* se basa en que los programas durante su ejecución no acceden con la misma probabilidad a los datos o instrucciones.

Existen dos tipos de localidad: la espacial y la temporal. La localidad espacial se basa en que cuando un programa accede a una instrucción o a un dato existe una elevada probabilidad de que instrucciones o datos cercanos sean accedidos pronto. La localidad temporal se basa en que, cuando un programa accede a una instrucción o un dato, existe una elevada probabilidad de que esa misma instrucción o dato vuelva a ser accedido pronto.

Por otro lado cabe destacar el efecto del *prefetching* o precarga, que consiste en llevar a las caches superiores antes de que estos sean accedidos para ahorrar el acceso a memoria principal una vez requeridos, haciendo de esta manera que la ejecución sea más rápida.

Este fenómeno utiliza el principio de localidad para saber que datos debe de traer a caché. En el experimento es un factor importante que nos permite observar la diferencia en ciclos de reloj cuando se logra un *prefetching* efectivo (los datos están próximos en memoria) y cuando no es posible.

III. Metodología

Para estudiar la influencia que tienen estos factores sobre el número de ciclos que tarda el microprocesador en acceder a memoria, se realizo un script en lenguaje c que calcula los ciclos medios por acceso a memoria. Además tomará diversos parámetros (el salto que se realiza entre las posiciones de memoria y el número de líneas cache diferentes que se quieren leer) que alteran la efectividad del *prefetching* basándose en el principio de localidad.

Este script en C calcula la media de ciclos de reloj por acceso a memoria caché de un vector de doubles A de R por D elementos realizando operaciones de reducción de suma.

Los ciclos de reloj son calculados mediante las funciones *start_counter()* y *get_counter()* colocadas al principio y final del bucle que ejecuta 10 veces la operación de reducción de suma. En cada una de estas 10 iteraciones se guarda cada uno de los resultados en un vector de soluciones que se debe imprimir al final del programa para evitar optimizaciones del compilador.

La repetición de este conjunto de operaciones para cada valor de D y de L constituye un único experimento. Este experimento se realiza un total de 10 veces para cada valor de D y cada valor de L mediante un script en bash de forma secuencial y guardamos los datos obtenidos de cada uno de estos experimentos

en un archivo csv.

Para la correcta obtención de datos es imprescindible comprender los distintos *parámetros* que emplea el script así como realizar una fase de *calentamiento* de la caché.

A. Parámetros

Los parámetros del script son un factor que varía de una manera considerable el número de ciclos de reloj de acceso a memoria.

Estos 3 parámetros son D, L y R.

D determina el salto en posiciones de memoria que se realiza en cada acceso al vector A para la operación de reducción de suma. Para sus valores, se escogieron las 5 potencias de 2 que mostraran los datos más representativos. Estas son: el 2, como un valor muy pequeño, que resultaría en un total de 4 iteraciones por línea (véase parámetro R); el 8, que equivaldría a una iteración por línea para valores de tipo *double*; el 16, que supondría una iteración por línea para valores de tipo *int*; el 32 por ser un valor de D elevado; y 256 por ser un valor de D extremadamente elevado donde el principio de localidad espacial no tiene ningún efecto en la ejecución del proceso.

L indica el número de líneas caché diferentes que se pretenden leer en ese experimento. En este estudio se emplearon valores de L calculados en función de S1 y S2 los cuales son los siguientes:

$$\frac{S1}{2}; 3 \cdot \frac{S1}{2}; \frac{S2}{2}; 3 \cdot \frac{S2}{4}; 2 \cdot S2; 4 \cdot S2; 8 \cdot S2$$

Estos valores permiten observar la diferencia en ciclos de reloj por accesos a memoria cache de nivel 1 y de nivel 2, ya que L supone el número de líneas caché que se desean leer en una ejecución del script. Sus valores ocupan desde la mitad del espacio de la cache L1 (mitad de líneas) hasta 8 veces el tamaño de la cache L2 (8 veces el número de líneas). Aún para valores de L pequeños, puede ocurrir que se accedan a memorias caché de L2, L3 o incluso memoria principal dependiendo del tamaño del salto entre valores (D).

El parámetro R determina el número de iteraciones que se deben realizar para acceder a L líneas caché distintas con un salto en posiciones de memoria de D unidades. Por esta razón R vendrá dado por el número de iteraciones necesarias para leer una línea caché multiplicado por el número de líneas totales. Para valores de D mayores o iguales a la cantidad de datos que caben en una línea cache, $R = L$. En caso contrario, R viene dada por la siguiente ecuación:

$$\frac{NumDatosporLineaCache \cdot (L-1)}{D} + 1$$

B. Calentamiento

La fase de calentamiento comprende la fase de inicialización de todos los R por D valores del vector A.

Estos valores son calculados de forma aleatoria mediante la función *rand* y acotados para comprender el rango $[-2,-1] \cup [1,2]$.

Esta fase de calentamiento es imprescindible ya que inicializa el vector A y guarda todos sus datos en memoria caché debido al principio de localidad temporal.

C. Interpretación y Presentación de los datos

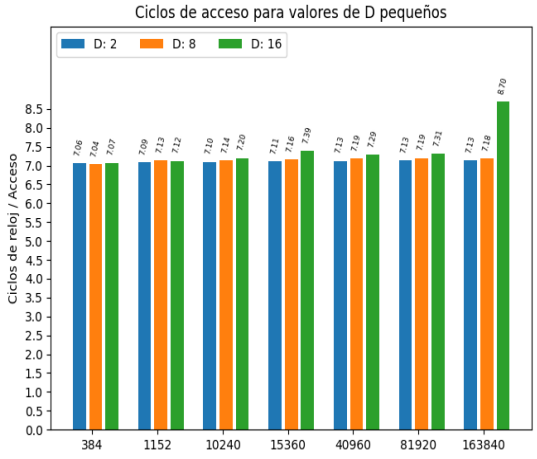
Una vez comprobado el correcto funcionamiento del script fue necesario el procesamiento de los datos que en este caso eran los ciclos de reloj promedio de las tres mejores (más rápidas) ejecuciones. Estos datos fueron redireccionados a un archivo para una más cómoda manipulación de estos.

Una vez obtenidos los resultados utilizamos matplotlib [1] para poder representarlos en gráficas y de esta manera poder conseguir una mejor visualización de los resultados obtenidos.

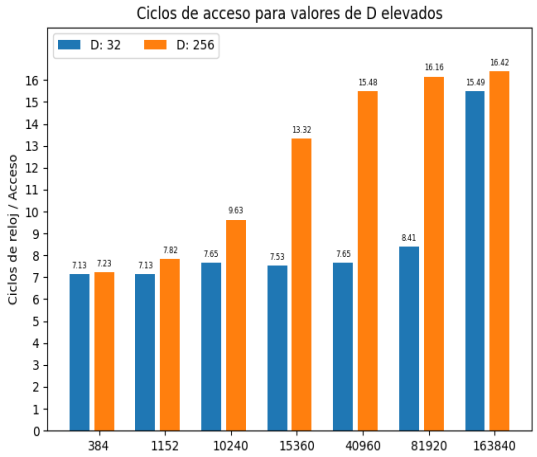
IV. Resultados

Mediante el uso de las técnicas utilizadas en la sección C, los resultados obtenidos para el experimento principal son observables en la figura 1.

En esta figura se observan mediante dos gráficas el número medio de ciclos por acceso a memoria en función de los parámetros D escogidos debido a las condiciones recogidas en la sección A.



(a) *Valores de D pequeños*



(b) *Valores de D grandes*

Figura 1: *Ciclos de Acceso a Memoria para Doubles*

En la figura 1a se observa el número de ciclos de acceso a memoria medios para valores de D pequeños y es notable que para los distintos valores de L se obtiene una media de ciclos similar y próxima a 7.

También se puede observar un cambio en la tendencia para los valores más grandes de D y de L. Esta subida en los ciclos medios se debe a un cambio en la ubicación de los datos en la jerarquía de memoria A descendiendo algún nivel en esta o incluso llegando a memoria principal.

En la figura 1b se observa una magnitud superior en los ciclos medios de acceso a memoria, en parte dado por la cantidad de datos con las que se trata (tener en cuenta el tamaño del vector A explicado en la sección A) obliga a acceder a los niveles de cache inferiores y a memoria principal.

Otro factor fundamental de la variación en los resultados se debe a los principios explicados en la sección B ya que para valores pequeños de D (menores que el número de elementos máximos por línea) los elementos del vector están próximos entre sí (en la misma línea o en una línea próxima). Debido al *prefetching* y al efecto de localidad espacial, muchos de estos valores ya se encuentran en la cache de nivel 1 listos para ser accedidos.

Este fenómeno se aprecia para valores de D grandes donde los datos a los que accede el vector A están muy separados entre sí y aunque se realice *prefetching* los datos próximos a aquellos a los que se acaban de acceder, el siguiente dato a extraer no se encuentra entre ellos provocando accesos a niveles de memoria inferiores.

V. Experimento con Ints

Para poder observar la variación de realizar el experimento con un tipo de dato diferente, int en nuestro caso, se realizó una variación en el script para adecuar este al tipo de dato deseado.

Una vez obtenidos los resultados se pueden comparar los datos de int con los de double. Para ello se disponen estos datos en la figura 2 en la que elegimos dos valores de D. En primer lugar se escoge el 8 por ser el punto para doubles en el que si se quiere leer un dato nuevo del array hay que leer una línea diferente y a continuación en el caso de ints 16 por la misma razón.

Como se puede apreciar en la figura 2 el número de ciclos medio por accesos es menor en el caso de int, esto es debido a que por línea caben más elementos lo que provoca una mayor probabilidad de que los elementos se encuentren más próximos favoreciendo así la *localidad espacial* y el *prefetching*.

Por lo que como se aprecia en la figura el número de ciclos medio en este caso es menor siempre para ints que para doubles independientemente del valor de L y del valor de D empleado.

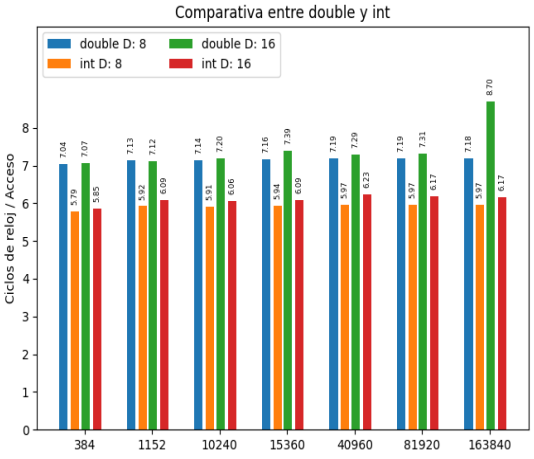


Figura 2: *Double vs Int*

VI. Experimento con Acceso Directo

Finalmente se realizó un experimento con accesos al vector A de forma directa.

En los experimentos previos, se accedía al vector A mediante un array dinámico *index* de tamaño R inicializado con las posiciones del vector A que se deben extraer.

$$index_i = i \cdot D \rightarrow \forall i = 0..R$$

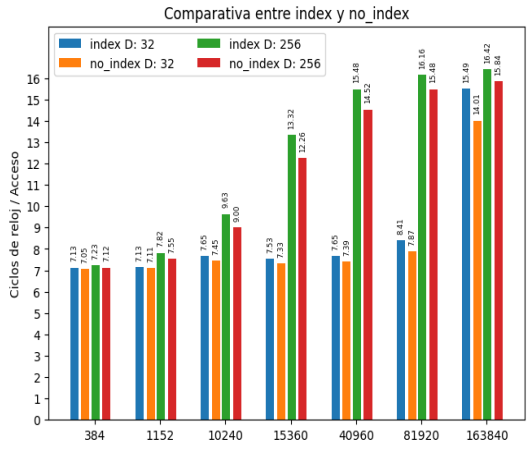


Figura 3: *Index vs NoIndex*

En la figura 3 se realiza la comparación entre la ejecución con acceso indirecto (IV) y la de acceso directo. Se puede comprobar que existe una mejoría en ciclos de reloj realizando el acceso directo ($A[i \cdot D]$) en lugar del indirecto ($A[index[i]]$). Esto se debe a que en el acceso indirecto es necesario acceder a la memoria en dos ocasiones: la primera para recuperar el valor almacenado en *index* en la posición *i*; y la segunda en acceder al valor de *A*.

Al no tener que realizar este acceso a memoria extra, los resultados de acceso directo no cargan con el coste de acceso a memoria adicional. De esta manera muestran pequeñas diferencias para valores de D y L pequeños, pero a medida que estos incrementan, los accesos extra que tiene que realizar el método de acceso indirecto conllevan un coste computacional significante debido a lo explicado en las secciones B y IV.

VII. Conclusiones

En este estudio se ha analizado la influencia de el *principio de localidad* y del efecto del *prefetching* mediante el número medio de ciclos de reloj de acceso a memoria que realiza un programa de reducción de suma en punto flotante de un vector dinámico. Para lograr dicho estudio, se han realizado diversos experimentos alterando la distancia en memoria entre los datos accedidos y el número de líneas cachés diferentes a las que se desea acceder. De este modo se observa el impacto en la funcionalidad de la jerarquía de memoria y el *prefetching*.

En base a los resultados obtenidos se puede llegar a la conclusión de que es fundamental tener en cuenta el efecto de localidad a la hora de crear los programas para aprovechar en su totalidad el beneficio del *prefectching* y optimizar el coste computacional de los accesos a memoria.

Como posibles estudios futuros a realizar serían interesantes el análisis y cálculo de las velocidades de acceso a los distintos niveles de la jerarquía caché y memoria principal y por otra parte las posibles optimizaciones que realiza el compilador.

Referencias

[1] Matplotlib 3.7.1 documentation., https://matplotlib.org/stable/index.html, [online] última visita 20 de marzo de 2023.