Universidad Católica
San Pablo

CIENCIA DE LA COMPUTACIÓN

INFORME CACHE AND PROGRAMS

COMPUTACIÓN PARALELA Y DISTRIBUIDA

ANDRES CUSIRRAMOS MARQUEZ MARES

8vo SEMESTRE
2022

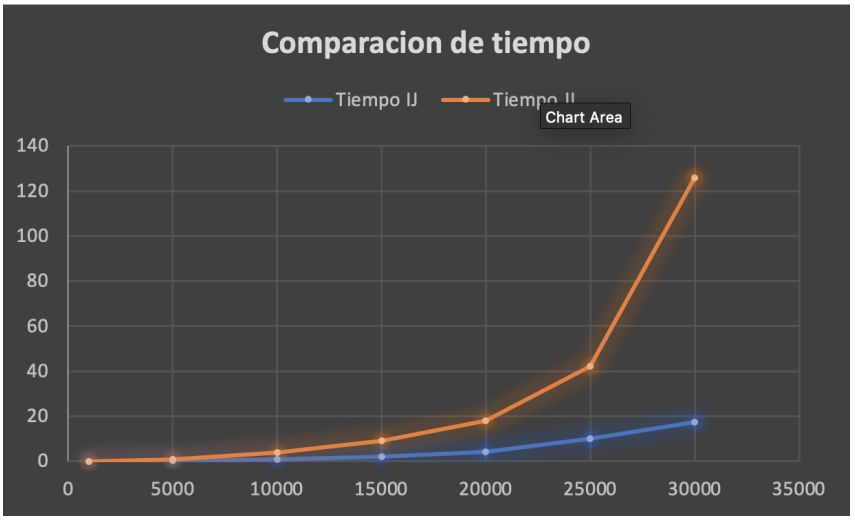
Se nos presentó un ejemplo de cómo se comportaba el caché y los programas luego de haber observado el comportamiento de este con el principio de localidad espacial y temporal el cual nos permite tener un control indirecto sobre cómo se almacena los datos en el caché. Se nos explica que en el lenguaje C las listas bidimensionales se almacenan en orden de fila, es decir la memoria es una enorme matriz unidimensional.En el ejemplo se nos presentan 3 listas, una bidimensional y 2 unidimensionales, los cuales van a ser recorridos por dos pares de for anidados.
La única diferencia de estos bucles es la forma como recorren los índices de las listas, dado que en la primera forma presentada se recorre con el iterador i y luego en el for anidado j, en la segunda forma lo contrario, primero el iterador j y luego i. Considerando que la forma en la que se almacena la lista es como se presenta en el ejemplo A[i][j].
Podríamos determinar que la primera forma de recorrer las listas sería más efectiva que la segunda por lo que accede a los datos de la matriz de dos dimensiones de forma secuencial pero para comprobar esto, realizamos pruebas con los dos bucles.

Se realizó una prueba con los pares de bucles anidados en la cual se consideró como número máximo a 1000, 5000, 10000, 15000, 20000,25000, 30000. Se consideró al primer for anidados como “IJ”y al segundo como “JI”.

Cantidad de Numeros	Tiempo IJ	Tiempo JI
1000	0.008345	0.015244
5000	0.212172	0.872546
10000	0.814804	3.81938
15000	2.10421	9.2381
20000	4.23497	17.9281
25000	10.0909	42.1449
30000	17.4999	125.956

Tabla 1. Tiempos de los pares de bucles anidados

Estos nos dieron unos tiempos, los cuales clasificamos en una tabla y en un gráfico para poder entender el comportamiento de estos.



Como se observa en la tabla anterior y el gráfico derivado de esta misma, el bucle anidado “IJ” es considerablemente más rápido sobre todo cuando se trata de números más grande se puede ver la diferencia en tiempo respecto a “JI”.

Por qué ocurre la diferencia entre los pares de bucles anidados, esto tiene que ver la cantidad de cache misses y también con la localidad espacial y temporal. Al momento que cualquiera de los dos pares de for anidados intenten acceder a A[0][0] tendrán un cache miss pero luego se cargará en fila las los datos de la lista dado que en el ejemplo se nos plantea que se pueden almacenar 4 datos por fila en el caché en este caso sería A[0][0], A[0][1], A[0][2], A[0][3]. El primer par de bucles nos daría un total de 4 cache misses, los cuales serían al terminar cada una de las filas.
En el segundo par de bucles intentará acceder a A[1][0], A[2][0], A[3][0], ninguno estara en cache por que lo que nos daría cache miss y al seguir recorriendo dado que el caché es pequeño va desalojar algunos datos para poder seguir con el recorrido y es por eso que podemos ver en la tabla y en el gráfico la diferencia entre los dos pares de bucles anidados.

Es importante como programadores considerar esto al momento de realizar todo tipo de algoritmos, dado que aunque los dos pares de bucles, tienen la misma complejidad la diferencia de rendimiento entre uno y otro es muy grande conforme los números y el recorrido se haga más grande.

Codigo Fuente :
https://github.com/Andrescmm/Computacion-Paralela/tree/main/Caches%20and%20programs