



UNIVERSIDAD CATÓLICA SAN PABLO

Pruebas sobre el comportamiento de la memoria caché: bucles anidados

Computer Science — Parallel and Distributed Computing

Harold Villanueva Borda

1. Primer Par de Bucles (i,j):

En el primer par de bucles, se recorre la matriz A por filas (i) y luego por columnas (j). Esto significa que los elementos de A se acceden secuencialmente en la memoria, lo cual puede aprovechar mejor la localidad espacial y mejorar el rendimiento del caché. Este tipo de acceso secuencial puede resultar en menos fallos de caché y una mejor utilización de la jerarquía de memoria, lo que generalmente conduce a un mejor rendimiento.

2. Segundo Par de Bucles (j,i):

En el segundo par de bucles, se recorre la matriz A por columnas (j) primero y luego por filas (i). Esto implica un acceso no contiguo a los elementos de A, lo que puede resultar en peor rendimiento debido a una menor eficiencia en el uso del caché. Al acceder a los elementos de A de esta manera, es más probable que se produzcan fallos de caché debido a la falta de localidad espacial. Conclusión:

3. Code

```
1 #include <iostream>
2 #include <chrono>
3
4 const int max = 10000;
5 double A[max][max], x[max], y[max];
6
7 void initialize() {
8     for (int i = 0; i < max; i++) {
9         x[i] = 1.0;
10        y[i] = 0.0;
11        for (int j = 0; j < max; j++)
12            A[i][j] = 1.0;
13    }
14 }
15
16 void loop1() {
17     for (int i = 0; i < max; i++)
18         for (int j = 0; j < max; j++)
19             y[i] += A[i][j] * x[j];
20 }
21
```

```
22 void loop2() {
23     for (int j = 0; j < max; j++)
24         for (int i = 0; i < max; i++)
25             y[i] += A[i][j] * x[j];
26 }
27
28 int main() {
29     initialize();
30
31     auto start1 = std::chrono::high_resolution_clock::now();
32     loop1();
33     auto end1 = std::chrono::high_resolution_clock::now();
34     std::chrono::duration<double> elapsed1 = end1 - start1;
35
36     initialize(); // Reset
37
38     auto start2 = std::chrono::high_resolution_clock::now();
39     loop2();
40     auto end2 = std::chrono::high_resolution_clock::now();
41     std::chrono::duration<double> elapsed2 = end2 - start2;
42
43     std::cout << "Time for loop 1: " << elapsed1.count() << " s\n";
44     std::cout << "Time for loop 2: " << elapsed2.count() << " s\n";
45
46     if (elapsed1.count() < elapsed2.count()) std::cout << "The loop 1 is the winner\n";
47     else if (elapsed1.count() > elapsed2.count()) std::cout << "The second loop is the winner\n";
48     else std::cout << "Same time\n";
49 }
```

main (code)

4. link del repositorio:

[GitHub](#).