Roberto Benavides Torres

Codigo:

const unsigned int MAXI = 20000;

using namespace std;

string d1, d2;

void test() {

srand(time(NULL));

double\*\* A = new double\*[MAXI];

for (long int i = 0; i < MAXI; i++) {

A[i] = new double[MAXI];

}

double\* x= new double[MAXI];

double\* y= new double[MAXI];

for (long int j = 0; j < MAXI; j++) {

y[j] = 0;

}

for (long int j = 0; j < MAXI; j++) {

x[j] = double(rand()) \* MAXI;

}

for (double\*\* i = A; i <A+MAXI; i++) {

double\* aux = \*i;

for (double\* j = aux; j < aux+MAXI; j++) {

\*j = double(rand()) \* MAXI;

}

}

std::chrono::time\_point<std::chrono::high\_resolution\_clock> start, end, start1, end1;

start = std::chrono::high\_resolution\_clock::now();

end = std::chrono::high\_resolution\_clock::now();

int64\_t duration =

std::chrono::duration\_cast<std::chrono::nanoseconds>(end - start).count();

for (long int i = 0; i < MAXI; i++) {

for (long int j = 0; j < MAXI; j++) {

y[i] += A[i][j] \* x[j];

}

}

for (long int j = 0; j < MAXI; j++) {

y[j] = 0;

}

start1 = std::chrono::high\_resolution\_clock::now();

for (int j = 0; j < MAXI; j++) {

for (int i = 0; i < MAXI; i++) {

y[i] += A[i][j]\* x[j];

}

}

end1 = std::chrono::high\_resolution\_clock::now();

int64\_t duration1 = std::chrono::duration\_cast<std::chrono::nanoseconds>(end1 - start1).count();

d1 = to\_string(duration);

d2 = to\_string(duration1);

}

Datos adquiridos

|  |  |  |
| --- | --- | --- |
| **size** | **1° bucle(ns)** | **2° bucle(ns)** |
| 100 | 400 | 50400 |
| 1000 | 500 | 12916000 |
| 5000 | 1000 | 400848900 |
| 10000 | 900 | 2307693700 |
| 15000 | 300 | 4323561900 |

El primer bucle es el más rápido dado que al obtener el primer dato en la cache se guarda todo el primer array y así sucesivamente teniendo todo el array en la caché, algo que no pasa en el segundo bucle ya que este recorre columnas primero haciendo que la cantidad de memory lost sea mucho mayor al primer bucle