Forma

Descripción generada automáticamente con confianza media

Instituto Tecnológico de Estudios Superiores Monterrey

CAMPUS QUERÉTARO

TC2037 Implementación de métodos computacionales

Pedro Oscar Pérez Murueta

Grupo 602

**Actividad 5.2 Programación paralela y concurrente**

**Alumnos**:

Erick Alfredo García Huerta - A01708119

Alan Fernando Razo Peña - A01703350

Fecha:

4 de Junio de 2022

Actividad 5.2 Programación paralela y concurrente

**Resultados obtenidos:**

En la elaboración de esta actividad se tuvieron que realizar dos versiones de un programa que calculara la suma de todos los números primos menores a 5,000,000 (cinco millones). Se utilizó el lenguaje de programación C++ ya que tiene la particularidad de manejar hilos a través de pthreads. Con esto se lograron obtener los siguientes resultados.

* La primera implementación fue de manera secuencial (concurrente).
  + **Resultado**:

EXITO SECUENCIAL!!!

Suma final: 8.38597e+011

avg time = 2420.9ms

* La segunda versión se realizó de manera paralela utilizando multi-hilo para su ejecución.
  + **Resultado**:

EXITO MULTI-THREAD!!!

Suma final: 8.386e+011

avg time = 1880.6ms

Como se puede apreciar, ambas versiones del programa arrojaron el mismo resultado: 838,596,693,108. Además de que la programación paralela presenta un resultado más rápido que la programación secuencial.

Código fuente:

#include <iostream>

#include <iomanip>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#include "utils.h"

using namespace std;

const int size = 5000000;

const int threads = 8;

typedef struct {

int start, end;

} Block;

bool isPrimo(int x){

if(x < 2){

return false;

}

for(int i = 2; i <= sqrt(x); i++){

if(x % i == 0){

return false;

}

}

return true;

}

void\* primo\_multihilo(void\* param) {

double \*acum;

Block \*block;

int i;

block = (Block \*) param;

acum = new double;

(\*acum) = 0;

for(i = block->start; i < block->end; i++){

if(isPrimo(i) == true){

(\*acum) += i;

}

}

return ((void\*\*) acum);

}

int main(int argc, char\* argv[]) {

double correct = 838596693108;

double ms, ms2;

/\*----------------------------------Secuencial--------------------------------------\*/

double count= 0;

ms2 = 0;

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

start\_timer();

if(isPrimo(i) == true){

count += i;

}

ms2 += stop\_timer();

}

if(count == correct){

cout << "EXITO SECUENCIAL!!!" << endl;

}else{

cout << "FALLO!!! " << count << " != " << correct << endl;

}

cout << "Suma final: " << count << endl;

cout << "avg time = " << setprecision(5) << (ms2 / N) << "ms\n";

/\*----------------------------------Multi-hilo--------------------------------------\*/

int blocksize, i, j;

double result, \*acum;

Block blocks[threads];

pthread\_t tids[threads];

blocksize = size / threads;

for(i = 0; i < threads; i++){

blocks[i].start = i \* blocksize;

if(i != (threads - 1)){

blocks[i].end = (i + 1) \* blocksize;

} else{

blocks[i].end = size;

}

}

ms = 0;

for(j = 0; j < N; j++){

start\_timer();

result = 0;

for(i = 0; i < threads; i++){

pthread\_create(&tids[i], NULL, primo\_multihilo, (void\*) &blocks[i]);

}

for(i = 0; i < threads; i++){

pthread\_join(tids[i], (void\*\*) &acum);

result += (\*acum);

delete acum;

}

ms += stop\_timer();

}

if(result == correct){

cout << "EXITO MULTI-THREAD!!! " << result << endl;

}else{

cout << "FALLO!!! "<< result << " != " << correct << endl;

}

cout << "avg time = " << setprecision(5) << (ms / N) << "ms\n";

return 0;

}