Proyecto — OpenMP

Lenin Pavón Alvarez [1]

[1]Facultad de Ciencias, UNAM

Agosto 2024

1. Código fuente

Se pueden consultar las implementaciones paralelas y secuenciales en los apéndices A.1 y A.2 y en el repositorio de la práctica en GitHub (link).

2. Tiempos de ejecución

Observemos los tiempos de ejecución de la versión paralela en comparación de los tiempos de la ejecución de la versión secuencial que podemos apreciar en la cuadro 1

Hilos	P1	P2	P3	P4	P5
2	630598	600538	595621	587692	586463
3	600506	584753	647737	593303	580700
4	599294	599398	576687	575249	601074

Tabla 1: Tabla de tiempos de ejecución (en milisegundos) de cada implementación paralela, las columnas indican la cantidad de hilos usados y la fila el número de prueba.

N. Prueba	Tiempo $[s]$
Prueba 01	613530
Prueba 02	610846
Prueba 03	604268
Prueba 04	607981
Prueba 05	610959
Prueba 06	636690
Prueba 07	636825
Prueba 08	628045
Prueba 09	635612
Prueba 10	628125

Tabla 2: Tabla de tiempos de ejecución (en segundos) de la implementación secuencial, las columnas indican la cantidad de hilos usados y la fila el número de prueba.

A. Códigos fuente

A.1. Suma secuencial

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include <omp.h>
#include <stdbool.h>

#define MAX_ROWS 3959
#define MAX_COLS 9
#define RANK 56


// Calcula frecuencia
int count_ones(int* mascara, int n){
        int suma = 0;
        for ( int i = 0 ; i < n ; i++ ) {
                suma += mascara[i];
        }
        return suma;
}
// Visualizar vector
void plista(int* lista, int n)
{
        for ( int i = 0 ; i < n ; i++ ) {
                printf("%d,", lista[i]);
        }
        printf("\n");
}
// Read CSV
void leer_csv(char *filename, int *numeros, int n_args){
        FILE *file;
        char line[256];
        int nproducto, concurso, R[7];
        float bolsa;
        char fecha[11];
        int i, j,k, fila_actual = -1;
        int filas_validas[MAX_ROWS];
        int counter = 0, frecuencia[7] = {0};
        int total[7][MAX_ROWS] = {0}; // Replica de historico.csv
        int mascara[7][MAX_ROWS] = {0};
        int single[RANK] = {0}; // Frecuencia de aparici n n meros  nicos
        int pair[RANK][RANK] = {0}; // Frecuencia pares
        int adicionales[RANK][RANK] = {0}; // Frecuencia pares

        // Abrir el archivo CSV por primera vez
```

```
file = fopen(filename, "r");
if (file == NULL) {
        printf("Error: No se pudo abrir el archivo %s\n", filename);
        exit(1);
}
// Primera lectura l nea por l nea
while( fgets( line, sizeof(line), file ) )
{
        sscanf(line, "%d,%d,%d,%d,%d,%d,%d,%d,%f,%s", &nproducto, &concurso, &R[0], &R
        // B squeda del primer n mero
        if (fila_actual > -1)
        {
                for ( j = 0 ; j < 7 ; j++ ) {
                        total[j][fila_actual] = R[j];
                        single[R[j] - 1] += 1;
                }
        }
        fila_actual++;
}
fclose(file);
printf("Frecuencia n meros  nicos \n");
for ( i = 0 ; i < RANK ; i++ ) {
        printf("%2d,%3d\n", i+1, single[i]);
}
/*
 * C lculo natural con natural
 */
// Primer n mero
#pragma omp parallel for private(j, fila_actual) collapse(3)
for ( i = 1 ; i <= RANK ; i++ ) {
        // Segundo n mero
        for ( j = 1 ; j <= RANK ; j++ ) {
                // Iteramos sobre el historico
                for ( fila_actual = 0 ; fila_actual < MAX_ROWS ; fila_actual++ ) {
                        int min,max;
                        if (i < j){
                                min = i;
                                max = j;
                        } else {
                                min = j;
                                max = i;
                        }
                        bool primerQ = false;
                        bool segundoQ = false;
                        int pos = 0;
                        for ( k = 0 ; k < 6 ; k++ ) {
                                if ( total[k][fila_actual] == min )
                                {
                                        primerQ = true;
                                        pos = k;
```

```
                                        break;
                                }
                        }
                        for ( k = pos + 1 ; k < 6 ; k++ ) {
                                if ( total[k][fila_actual] == max )
                                        segundoQ = true;
                        }
                        if ( primerQ && segundoQ )
                        {
                                #pragma omp critical
                                pair[i][j] += 1;
                        }
                }
        }
}
// Impresi n de resultados
for ( i = 0 ; i <= RANK ; i++ ) {
        printf("%2d,",i);
}
for ( i = 0 ; i <= RANK ; i++ ) {
        printf("%2d,",i);
        for ( j = 1 ; j <= RANK ; j++ ) {
                printf("%2d,",pair[i][j]);
        }
        printf("\n");
}
printf("\n");
printf("-------------------------\n");
printf("Adicionales\n");
printf("-------------------------\n");
/*
 * C lculo natural con adicional
 */
for ( i = 0 ; i <= RANK ; i++ ) {
        printf("%2d,",i);
}
printf("\n");
#pragma omp parallel for private(j, fila_actual) collapse(3)
for ( i = 1 ; i <= RANK ; i++ ) {
        // Segundo n mero
        for ( j = 1 ; j <= RANK ; j++ ) {
                // Iteramos sobre el historico
                for ( fila_actual = 0 ; fila_actual < MAX_ROWS ; fila_actual++ ) {
                        bool primerQ = false;
                        bool segundoQ = false;
                        for ( k = 0 ; k < 6 ; k++ ) {
                                if ( total[k][fila_actual] == i )
                                        primerQ = true;
                        }
                        if ( total[6][fila_actual] == j )
```

```
                                segundoQ = true;
                        if ( primerQ && segundoQ )
                        {
                                #pragma omp critical
                                adicionales[i][j] += 1;
                        }
                }
        }
        }
        // Impresi n de resultados
        for ( i = 0 ; i <= RANK ; i++ ) {
                printf("%2d,",i);
        }
        for ( i = 0 ; i <= RANK ; i++ ) {
                printf("%2d,",i);
                for ( j = 1 ; j <= RANK ; j++ ) {
                        printf("%2d,",adicionales[i][j]);
                }
                printf("\n");
        }
}

int main(int argc, char *argv[]) {
        struct timeval stop, start;
        int numeros[7] = {0};
        int i;
        if ( argc > 2 )
        {
                printf("Error: El programa no acepta argumentos.\n");
                return 1;
        }
        // Leer CSV
        gettimeofday(&start, NULL);
        leer_csv("historico.csv", numeros, argc - 1);
        gettimeofday(&stop, NULL);
        printf("Time: %lu\n", (stop.tv_sec - start.tv_sec) * 1000000 + stop.tv_usec - start.tv_us
        return 0;
}
```

A.2. Suma paralela

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include <stdbool.h>

#define MAX_ROWS 3959
#define MAX_COLS 9
#define RANK 56


// Calcula frecuencia
int count_ones(int* mascara, int n){
        int suma = 0;
        for ( int i = 0 ; i < n ; i++ ) {
                suma += mascara[i];
        }
        return suma;
}
// Visualizar vector
void plista(int* lista, int n)
{
        for ( int i = 0 ; i < n ; i++ ) {
                printf("%d,", lista[i]);
        }
        printf("\n");
}
// Read CSV
void leer_csv(char *filename, int *numeros, int n_args){
        FILE *file;
        char line[256];
        int nproducto, concurso, R[7];
        float bolsa;
        char fecha[11];
        int i, j,k, fila_actual = -1;
        int filas_validas[MAX_ROWS];
        int counter = 0, frecuencia[7] = {0};
        int total[7][MAX_ROWS] = {0}; // Replica de historico.csv
        int mascara[7][MAX_ROWS] = {0};
        int single[RANK] = {0}; // Frecuencia de aparici n n meros  nicos
        int pair[RANK][RANK] = {0}; // Frecuencia pares
        int adicionales[RANK][RANK] = {0}; // Frecuencia pares

        // Abrir el archivo CSV por primera vez
        file = fopen(filename, "r");
        if (file == NULL) {
                printf("Error: No se pudo abrir el archivo %s\n", filename);
                exit(1);
```

```
}
// Primera lectura l nea por l nea
while( fgets( line, sizeof(line), file ) )
{
        sscanf(line, "%d,%d,%d,%d,%d,%d,%d,%d,%f,%s", &nproducto, &concurso, &R[0], &R
        // B squeda del primer n mero
        if (fila_actual > -1)
        {
                for ( j = 0 ; j < 7 ; j++ ) {
                        total[j][fila_actual] = R[j];
                        single[R[j] - 1] += 1;
                }
        }
        fila_actual++;
}
fclose(file);
printf("Frecuencia␣n meros␣ nicos \n");
for ( i = 0 ; i < RANK ; i++ ) {
        printf("%2d,%3d\n", i+1, single[i]);
}
/*
 * C lculo natural con natural
 */
// Primer n mero

for ( i = 0 ; i <= RANK ; i++ ) {
        printf("%2d,",i);
}
printf("\n");
for ( i = 1 ; i <= RANK ; i++ ) {
        printf("%2d,",i);
        // Segundo n mero
        for ( j = 1 ; j <= RANK ; j++ ) {
                // Iteramos sobre el historico
                for ( fila_actual = 0 ; fila_actual < MAX_ROWS ; fila_actual++ ) {
                        int min,max;
                        if (i < j){
                                min = i;
                                max = j;
                        } else {
                                min = j;
                                max = i;
                        }
                        bool primerQ = false;
                        bool segundoQ = false;
                        int pos = 0;
                        for ( k = 0 ; k < 6 ; k++ ) {
                                if ( total[k][fila_actual] == min )
                                {
                                        primerQ = true;
```

```
                                        pos = k;
                                        break;
                                }
                        }
                        for ( k = pos + 1 ; k < 6 ; k++ ) {
                                if ( total[k][fila_actual] == max )
                                        segundoQ = true;
                        }
                        if ( primerQ && segundoQ )
                                pair[i][j] += 1;
                }
                printf("%2d,",pair[i][j]);
        }
        printf("\n");
}
printf("-------------------------\n");
printf("Adicionales\n");
printf("-------------------------\n");
/*
 * C lculo natural con adicional
 */
for ( i = 0 ; i <= RANK ; i++ ) {
        printf("%2d,",i);
}
printf("\n");
for ( i = 1 ; i <= RANK ; i++ ) {
        printf("%2d,",i);
        // Segundo n mero
        for ( j = 1 ; j <= RANK ; j++ ) {
                // Iteramos sobre el historico
                for ( fila_actual = 0 ; fila_actual < MAX_ROWS ; fila_actual++ ) {
                        bool primerQ = false;
                        bool segundoQ = false;
                        for ( k = 0 ; k < 6 ; k++ ) {
                                if ( total[k][fila_actual] == i )
                                        primerQ = true;
                        }
                        if ( total[6][fila_actual] == j )
                                segundoQ = true;
                        if ( primerQ && segundoQ )
                                adicionales[i][j] += 1;
                }
                printf("%2d,",adicionales[i][j]);
        }
        printf("\n");
}
}

int main(int argc, char *argv[]) {
        struct timeval stop, start;
```

```
        int numeros[7] = {0};
        int i;
        if ( argc > 2 )
        {
                printf("Error: El programa no acepta argumentos.\n");
                return 1;
        }
        // Leer argumentos
        numeros[0] = 1;
        numeros[1] = 2;
        // Leer CSV
        gettimeofday(&start, NULL);
        leer_csv("historico.csv", numeros, argc - 1);
        gettimeofday(&stop, NULL);
        printf("Time: %lu\n", (stop.tv_sec - start.tv_sec) * 1000000 + stop.tv_usec - start.tv_us
        return 0;
}
```