Laboratorio Paralelismo

Karen Daniela Medina Naranjo

Pontificia Universidad Javeriana

Programa de ingeniería de Sistemas

Introducción a Sistemas Distribuidos

Ingeniero John J. Corredor F.

Bogotá, Colombia

15 de Agosto de 2025

1. Documentación del Código

1.1 MmClasicaOpenMp.c

Este programa implementa la multiplicación de matrices cuadradas haciendo uso de paralelismo con OpenMP. El usuario indica el tamaño de la matriz y el número de hilos que se usarían. Las matrices A y B se inicializan con valores aleatorios, y el resultado de la multiplicación se almacena en la matriz C. Para medir el rendimiento, el programa toma el tiempo de ejecución en microsegundos antes y después de realizar la multiplicación. Cabe resaltar que, si la matriz posee una dimensión menor a 9, el programa la imprime en pantalla, lo que puede facilitar la validación del cálculo.

```c
/*##############################################################################
 #* Fecha: 15-Agosto-2025
 #* Autor: Daniela Medina
 #* Tema: Laboratorio de Paralelismo
 #*     - Programa Multiplicación de Matrices algoritmo clásico
 #*     - Paralelismo con OpenMP
##############################################################################*/

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

// Variables globales para medir el tiempo de ejecucion
struct timeval inicio, fin;

// Funcion que marca el inicio de la meedicion del tiempo

void InicioMuestra(){
    // Captura el tiempo actual en inicio
    gettimeofday(&inicio, (void *)0);
}

// Funcion que marca el final de la medicion y calcula el tiempo en microsegundos y lo imprime

void FinMuestra(){
    // Captura el tiempo actual en fin
    gettimeofday(&fin, (void *)0);
    fin.tv_usec -= inicio.tv_usec;
    fin.tv_sec  -= inicio.tv_sec;
    // Convierte tiempo a microsegundos
    double tiempo = (double)(fin.tv_sec * 1000000 + fin.tv_usec);
    printf("%9.0f \n", tiempo);
}
```

Imagen 1. Código del programa MmClasicaOpenMP 1

```c
// Funcion que recibe un puntero a la matriz a imprimir y la dimension de la misma
// Imprime la matriz si su dimension es menor a 9

void impMatrix(double *matrix, int D){
    printf("\n");
    if(D < 9){
        //Recorre los elementos de la matriz
        for(int i = 0; i < D*D; i++){
            if(i % D == 0) printf("\n");
            printf("%.2f ", matrix[i]);
        }
        printf("\n**----------------------------**\n");
    }
}

// Funcion que recibe un puntero a la 1era matriz, un puntero a la 2da matrz y la dimension de las mismas
// Inicializa las dos matrices con valores entre 0-99

void iniMatrix(double *m1, double *m2, int D){
    for(int i = 0; i < D*D; i++, m1++, m2++){
        // Asigna valores aleatorios a cada matriz
        *m1 = (double)(rand() % 100);
        *m2 = (double)(rand() % 100);
    }
}

// Funcion que recibe una matriz A, una matriz B y la dimension de las mismas
// Realiza la multiplicacion de matrices, empleando paralelismo con OpenMP

void multiMatrix(double *mA, double *mB, double *mC, int D){
    double suma, *pA, *pB;
    // Inicia una region paralela (donde se crean los hilos)
    #pragma omp parallel
    {
        // Divide las iteraciones del siguiente for, entre todos los hilos disponibles, cada hilo tiene variables privadas
        #pragma omp for private(pA, pB, suma)
```

Imagen 2. Código del programa MmClasicaOpenMP 2

```c
        // Itera filas de A y C
        for(int i = 0; i < D; i++){
            // Itera columnas de B
            for(int j = 0; j < D; j++){
                // Apunta al inicio de la fila i de A
                pA = mA + i*D;
                // Apunta al inicio de la columna j de B
                pB = mB + j;
                // Reinicia acumulador para C
                suma = 0.0;
                // Recorre elementos de fila i y columna j
                for(int k = 0; k < D; k++, pA++, pB += D){
                    // Acumula producto A[i,k] * B[k,j]
                    suma += (*pA) * (*pB);
                }
                // Escribe resultado en C
                mC[i*D + j] = suma;
            }
        }
    }
}

// Funcion principal
int main(int argc, char *argv[]){
    if(argc < 3){
        printf("\n Use: $./clasicaOpenMP SIZE Hilos \n\n");
        exit(0);
    }

    // Dimension de las matrices
    int N = atoi(argv[1]);
    // Numero de hilos a usar
    int TH = atoi(argv[2]);

    // Reservar matrices con tipo double
    double *matrixA = (double *)calloc(N*N, sizeof(double));
```

Imagen 3. Código del programa MmClasicaOpenMP 3

```c
    double *matrixB = (double *)calloc(N*N, sizeof(double));
    double *matrixC = (double *)calloc(N*N, sizeof(double));

    // Verifica asignacion correcta
    if(!matrixA || !matrixB || !matrixC){
        fprintf(stderr, "Error al asignar memoria.\n");
        exit(1);
    }

    //Inicializacion de entorno paralelo
    srand(time(NULL));
    // Configura cantidad de hilos que usara OpenMP
    omp_set_num_threads(TH);

    // Inicializar matrices A y B con números aleatorios de 0-99
    iniMatrix(matrixA, matrixB, N);

    // Mostrar matrices si tienen dimensiones <9
    impMatrix(matrixA, N);
    impMatrix(matrixB, N);

    // Inicio de medicion de tiempo
    InicioMuestra();
    // Ejecuta la multiplicación
    multiMatrix(matrixA, matrixB, matrixC, N);
    // Fin de la medicion de tiempo
    FinMuestra();

    // Mostrar resultado si tiene dimension <9
    impMatrix(matrixC, N);

    // Liberación de Memoria
    free(matrixA);
    free(matrixB);
    free(matrixC);

    return 0;
```

Imagen 4. Código del programa MmClasicaOpenMP 4

1.2 Lanzador.pl

Este codigo automatiza la ejecución por conjuntos de MmClasicaOpenMP. Su propósito es probar diferentes tamaños de matrices y número de hilos para evaluar el rendimiento del programa. Para cada combinación, el script ejecuta el programa 30 veces y guarda los resultados en un archivo .dat nombrado de una forma específica. De esta manera, se obtiene un conjunto de archivos con datos que pueden analizar el comportamiento del algoritmo con diferentes cargas computacionales.

```perl
#!/usr/bin/perl
#*************************************************************
#            Pontificia Universidad Javeriana
#     Autor: Daniela Medina
#     Fecha: 15 AGosto 2025
#     Materia: Sistemas Distribuidos
#     Tema: Laboratorio de Paralelismo
#     Fichero: script automatización ejecución por lotes
#*************************************************************/

$Path = `pwd`;
chomp($Path);

# Nombre del programa a ejecutar
$Nombre_Ejecutable = "mmClasicaOpenMP";
# Lista de tamaños de matrices a probar
@Size_Matriz = ("240","880","1520","3040","4960","7200","9280","12800","14400","16080","18400","19920");
# Lista de cantidades de hilos
@Num_Hilos = (1,2,4,8,16,20);
$Repeticiones = 30;

# Bucle que recorre cada tamaño de matriz
foreach $size (@Size_Matriz){
  # Dentro de cada tamaño de matriz, recorre los distintos números de hilos
  foreach $hilo (@Num_Hilos) {
    # Nombre de archivo donde se guardan los resultados
    $file = "$Path/$Nombre_Ejecutable-".$size."-Hilos-".$hilo.".dat";
    # Ejecuta el programa 30 veces
    for ($i=0; $i<$Repeticiones; $i++) {
      # Llama al programa ./mmClasicaOpenMP con argumentos: tamaño de matriz y número de hilos
      system("$Path/$Nombre_Ejecutable $size $hilo  >> $file");
      #printf("$Path/$Nombre_Ejecutable $size $hilo \n");
    }
    close($file);
  $p=$p+1;
  }
}
```

Imagen 5. Script de lanzador.pl

1.3 MakeFile

Automatiza la compilación de un programa en C llamado MmClasicaOpenMP. Facilita la compilación repetitiva del proyecto en un solo comando.

```makefile
# Define el compilador a usar, gcc
GCC = gcc
# Define la bandera de compilacion, libreria matematica
CFLAGS = -lm
# Define las banderas para usar OpenMP
# -fopenmp habilita OpenMP, -O3 activa optimizaciones de alto nivel
FOPENMP = -fopenmp -O3
# Define la librería POSIX para manejo de hilos, pthread
POSIX = -lpthread

# Nombre del archivo fuente que se va a compilar
modulo = inter.c
# Define el programas que se van a generar con el Makefile, mmClasicaOpen MP
PROGRAMAS = mmClasicaOpenMP

# Compila todos los programas definidos en PROGRAMAS
ALL: $(PROGRAMAS)

# Compila el archivo mmClasicaOpenMP.c, lo enlaza con OpenMP y genera el ejecutable
mmClasicaOpenMP:
	$(GCC) $(CLFAGS) $@.c -o $@ $(FOPENMP)

# Limpia el directorio
# Elimina los ejecutables listados en PROGRAMAS
clean:
	$(RM) $(PROGRAMAS)
```

Imagen 6. Código de MakeFile

2. Diseño de Experimento

Se seleccionaron seis cantidades de hilos**:** 1**,** 4, 8, 16 y 20. Del mismo modo, se definieron doce tamaños de matrices diferentes**:** 240, 880, 1520, 3040, 4960, 7200, 9280, 12800, 14400, 16080, 18400 y 19920, garantizando que cada uno de ellos fuera múltiplo de 80. Esta elección respondió a una conveniencia matemática, ya que trabajar con múltiplos de una base común facilita la distribución de hilos y mejora la estructura interna de los cálculos.

Cabe señalar que, en las condiciones iniciales del taller, se establecía el uso de matrices de hasta 20.000 elementos, límite que posteriormente se redujo a 14.000. Sin embargo, la ejecución del programa se realizó con el tamaño original de 20.000. Además, algunas matrices no se procesaron debido al tiempo prolongado requerido en su ejecución.