Corto 5

COMPUTACIÓN PARALELA Y DISTRIBUIDA

Sección 20

Mónica Alejandra Salvatierra Chacón - 22249

Derek Fabian Arreaga Kelson - 22537

Paula Rebeca Barillas Alvarez – 22764

1. Diseño de la idea

En nuestra simulación, nos enfocados en carreras de F1.

Describir qué parte se ejecutará en paralelo y cómo se pueden aplicar:

- **parallel for**
  - Usaremos un parallel for para simular los tiempos por sector de cada piloto en cada vuelta.
  - Cada iteración corresponde a un piloto en una vuelta específica, y se calcula su tiempo en paralelo.
- **sections**
  Dividiremos tareas independientes en secciones:
  - Una sección calcula estadísticas de la carrera (mejor vuelta, promedio).
  - Otra registra el orden de posiciones.
  - Otra actualiza la información del clima.
- **firstprivate**
  - Se aplicará para variables que deben inicializarse con un valor común, pero que no deben ser compartidas (por ejemplo, el tiempo base inicial para cada piloto antes de aplicar variaciones aleatorias).
- **shared**
  - Variables globales como la matriz de tiempos, el número de pilotos, vueltas y sectores serán compartidas entre todos los hilos.
- **reduction**
  Para calcular:
  - El tiempo total de la carrera por piloto sumando sus sectores.
  - Encontrar el tiempo mínimo (mejor vuelta global).

Variables Compartidas y Privadas:

- **Privadas:**
  - Indices de bucles (i, j, k)
  - tiempo_base
  - tiempo_sector
- **Compartidas:**
  - tiempos[NUM_PILOTOS][NUM_VUELTAS][NUM_SECTORES]
  - mejor_vuelta_global
  - NUM_PILOTOS
  - NUM_VUELTAS
  - NUM_SECTORES

2. Pseudocódigo

```
// Declaración de constantes
NUM_PILOTOS ← 20
```

```
NUM_VUELTAS ← 40
NUM_SECTORES ← 4

// Declaración de variables compartidas
tiempos[NUM_PILOTOS][NUM_VUELTAS][NUM_SECTORES]
mejor_vuelta_global ← valor grande
ganador ← -1

// Variables privadas en los bucles paralelos
i, j, k, tiempo_base, tiempo_sector

// Inicializar matriz de tiempos a 0
para i desde 0 hasta NUM_PILOTOS-1:
        para j desde 0 hasta NUM_VUELTAS-1:
                para k desde 0 hasta NUM_SECTORES-1:
                        tiempos[i][j][k] ← 0

// Sección 1: Calcular tiempos por sector en paralelo
#pragma omp parallel for private(i, j, k, tiempo_sector) firstprivate(tiempo_base)
shared(tiempos)
para i desde 0 hasta NUM_PILOTOS-1:
        para j desde 0 hasta NUM_VUELTAS-1:
                para k desde 0 hasta NUM_SECTORES-1:
                        // tiempo_base es inicializado con un valor base común
                        tiempo_sector ← tiempo_base + variación aleatoria
                        tiempos[i][j][k] ← tiempo_sector

// Sección 2: Calcular tiempo total por piloto y mejor vuelta
#pragma omp parallel for reduction(min: mejor_vuelta_global)
para i desde 0 hasta NUM_PILOTOS-1:
        tiempo_total_piloto ← 0
        para j desde 0 hasta NUM_VUELTAS-1:
                tiempo_vuelta ← sumar(tiempos[i][j][:])
                si tiempo_vuelta < mejor_vuelta_global:
                        mejor_vuelta_global ← tiempo_vuelta
                tiempo_total_piloto ← tiempo_total_piloto + tiempo_vuelta
                // Guardar el tiempo total en una variable compartida
```

```
// Sección 3: Estadísticas y clima (parallel sections)
#pragma omp parallel sections
{
        #section 1: Calcular posiciones finales
        // Ordenar pilotos por tiempo total

        #section 2: Registrar estadísticas adicionales
        // Mejor vuelta por piloto, promedio de vueltas

        #section 3: Actualizar información del clima
        // Simular cambios de temperatura, humedad, lluvia
}

// Imprimir resultados finales
Imprimir ganador, mejor_vuelta_global, posiciones finales
```

3. **Implementación y Ejecución**
    - **Link de Repositorio:** https://github.com/paulabaal12/CORTO5-PARALELA