Documentación - Práctica 3 SCP

Implementación Concurrente con Pthreads

Daranuta Zob Cristian

55261830Y

Esquinas Fernández Guillermo

49750576N

16/12/2025

1. Arquitectura de la Solución Concurrente

1.1 Modelo de Paralelización

La solución implementa un modelo de paralelización por descomposición del dominio con las siguientes características:

- **Estrategia**: Reparto intercalado de combinaciones
- **Granularidad**: Cada hilo evalúa combinaciones de forma independiente
- **Comunicación**: Mínima, ya que solo se comunican para actualizar el óptimo global

1.2 Reparto de Trabajo: Intercalado vs Bloques

Decisión de Diseño: Reparto Intercalado

- Hilo 0 procesa: 1, 1+N, 1+2N, 1+3N, ...
- Hilo 1 procesa: 2, 2+N, 2+2N, 2+3N, ...
- Hilo 2 procesa: 3, 3+N, 3+2N, 3+3N, ...
- ...

Ventajas del reparto intercalado:

1. **Balance de carga superior**: Las combinaciones complejas se distribuyen uniformemente
2. **Sin desbalance al final**: Todos los hilos terminan aproximadamente al mismo tiempo
3. **Mejor uso de caché**: Acceso más distribuido a la memoria

2. Mecanismos de Sincronización

2.1 Mutex (`pthread_mutex_t`)

Se utilizan **3 mutex** diferentes para proteger secciones críticas específicas:

Mutex 1: `mutex_optimo`

Propósito: Proteger el acceso al óptimo global compartido

Análisis de Condiciones de Carrera:

- **Sin mutex**: Dos hilos podrían leer el mismo `OptimoGlobal.Coste` simultáneamente y ambos intentar actualizarlo, causando pérdida de actualizaciones o datos inconsistentes

- **Con mutex**: Solo un hilo puede actualizar el óptimo a la vez, garantizando atomicidad

Optimización aplicada:

- Cada hilo trabaja por su cuenta
    - Calcula combinaciones
    - Guarda su mejor resultado local (`CosteMejorCombinacionLocal`)
    - Comprueba si realmente mejora el mejor resultado global
        - Si no lo mejora → sale rápido
        - Si lo mejora → actualiza

Mutex 2: `mutex_estadisticas`

Propósito: Proteger las estadísticas globales compartidas

Problema Resuelto: Sin este mutex, múltiples hilos podrían sumar sus operaciones, a las operaciones simultáneas, causando resultados incorrectos

Decisión de Diseño: Actualizar estadísticas solo una vez al final de cada hilo para minimizar contención.

Mutex 3: `mutex_progreso`

Propósito: Proteger el contador de hilos inicializados, usado con una variable de condición

Problema que resuelve:

- Varios hilos se inicializan en paralelo
- Cada hilo debe notificar cuando está listo
- El hilo principal debe esperar a que todos los hilos estén preparados antes de continuar

Sin protección:

- Dos hilos podrían modificar `hilos_inicializados` al mismo tiempo
- El hilo principal podría despertarse demasiado pronto o no despertarse nunca.

2.2 Variable de Condición

Variable: `cond_todos_listos`

Propósito: Sincronizar el inicio del procesamiento entre el hilo principal y los hilos trabajadores.

Funcionamiento:

- Cada hilo:
  - Incrementa `hilos_inicializados` bajo protección del mutex.
  - Si es el último hilo, emite un `pthread_cond_broadcast`.
- El hilo principal:
  - Espera con `pthread_cond_wait` hasta que todos los hilos estén inicializados.

Esta sincronización garantiza que el cálculo comienza únicamente cuando todos los hilos están preparados.

2.3 Semáforo

Semáforo: `sem_print`

Propósito: Evitar la intercalación de salidas por pantalla producidas por múltiples hilos.

Problema que resuelve:

- Sin semáforo:
  - Las llamadas a `printf` de distintos hilos se mezclarían.
  - La salida sería ilegible e inconsistente.

Solución: Se utiliza un semáforo binario para garantizar que solo un hilo imprime información de progreso a la vez

3. Problemas Detectados y Soluciones Aplicadas

3.1 Contención excesiva por uso de mutex

- **Problema inicial:** El mutex del óptimo global se bloqueaba en cada iteración
- **Solución aplicada:** Uso de óptimos locales y sincronización sólo cuando existe posibilidad real de mejora

3.2 Condiciones de carrera en estadísticas

- **Problema inicial:** Actualización directa de estadísticas globales desde cada hilo
- **Solución aplicada:** Estadísticas locales por hilo y actualización global única al finalizar

3.3 Desbalance de carga entre hilos

- **Problema inicial:** Reparto por bloques causaba que algunos hilos terminaran antes
- **Solución aplicada:** Reparto intercalado de combinaciones.

3.4 Salida por pantalla desordenada

- **Problema inicial:** Mensajes de progreso mezclados entre hilos
- **Solución aplicada:** Uso de semáforo binario para serializar la salida estándar

3.5 Recomputación innecesaria del óptimo final

- **Problema inicial:** El óptimo final se recalculaba recorriendo nuevamente todas las combinaciones
- **Solución aplicada:** Almacenamiento directo de la combinación óptima global durante la ejecución