Semana 1 | Workshop 1

Entrega por: Sergio A. Sánchez - T00055303

Fecha: 06/02/2026

Descripción del problema.

Semestralmente se construyen horarios académicos para docentes y estudiantes por igual. La meta principal de este plan académico es definir los cursos, grupos, profesores y aulas dentro del campus que serán usadas para impartir clases y satisfacer el uso eficiente de la infraestructura de la universidad. Todos los grupos deben coexistir sin conflicto en sesiones que no se crucen entre sí, elevando así la complejidad y las restricciones que tienen por las combinaciones de cada sesión en curso.

En este escenario descrito, el paralelismo juega un papel vital, dado que su función nos permitiría verificar las restricciones de muchas sesiones simultáneamente, por ejemplo, nos permitiría medir los espacios vacíos ("huecos") entre sesiones, o validar requerimientos del usuario tales como que las "clases terminen antes de 20:00 horas". Es decir, el uso iría más enfocado en el manejo de las restricciones, que en la verificación de cupos para matricula porque requiere un estado compartido.

Descripción de la solución propuesta

Una vez se cuenta con un horario que sirve de prospecto, se validan las restricciones aplicables para cada caso:

Caso 1: Cruce de sesiones, por grupo o aula.

Caso 2: Espacios entre jornadas ("Huecos entre clases").

Caso 3: Restricciones de jornada (Diurna - Nocturna) o que finalice "antes de" o empiece "después de".

Caso 4: Distribución de sesiones. (lunes a viernes) | (lunes a sábado, etc.)

En el diseño de la solución se plantea el uso del paralelismo de datos, dado que nos permite inspeccionar miles de sesiones con las mismas reglas y muchos datos distintos.

Detalles de la implementación

Para la implementación serial se procesa el horario que se tiene como prospecto en una ejecución secuencial. Los datos esperados al final del proceso deben permitirnos calcular el speedup y la eficiencia.
El primer paso será la preparación del horario que se generará de manera aleatoria para efectos del ejercicio, donde cada sesión va a incluir grupo académico, aula, día, hora de inicio y hora de fin. Esta agrupación permite que el algoritmo compare únicamente sesiones que realmente compiten por el mismo recurso.

Especificación del Entorno de Ejecución

Procesador: Apple M4 (ARM)
- **Arquitectura:** ARM64
- **Núcleos físicos totales:** 10 cores
- **Hilos lógicos:** 10 (1 hilo por núcleo físico)
- **GPU:** 10 cores.

Memoria: 16 GB de RAM unificada
Sistema Operativo: macOS 26 Tahoe.

Resultados más relevantes

Métrica	Valor inicial	Valor final
Tiempo serial	0.003 s	~2.0 s
Tiempo paralelo	0.161 s	~0.5-0.7 s
Speedup	0.02x	~3.0-3.5x
Eficiencia	45%	~75-85%
Procesos	8	4

Ventajas de la solución paralela:

- **Independencia de tareas**: cada horario se valida de forma independiente, lo que lo convierte en un problema *donde* no hay dependencias entre tareas ni necesidad de sincronización.
- Al aumentar la carga (más horarios o repeticiones), el speedup se acerca al ideal teórico ($S = p$).

Limitaciones

- Con cargas muy pequeñas (ej. 50 horarios x 200 sesiones sin repeticiones = 0.003s serial), el overhead (~0.14s) domina completamente y el paralelo es **MÁS LENTO** que el serial.
- Con tareas muy pequeñas, el costo de enviar/recibir datos entre procesos (IPC) supera el beneficio del paralelismo.

Cuellos de botella identificados

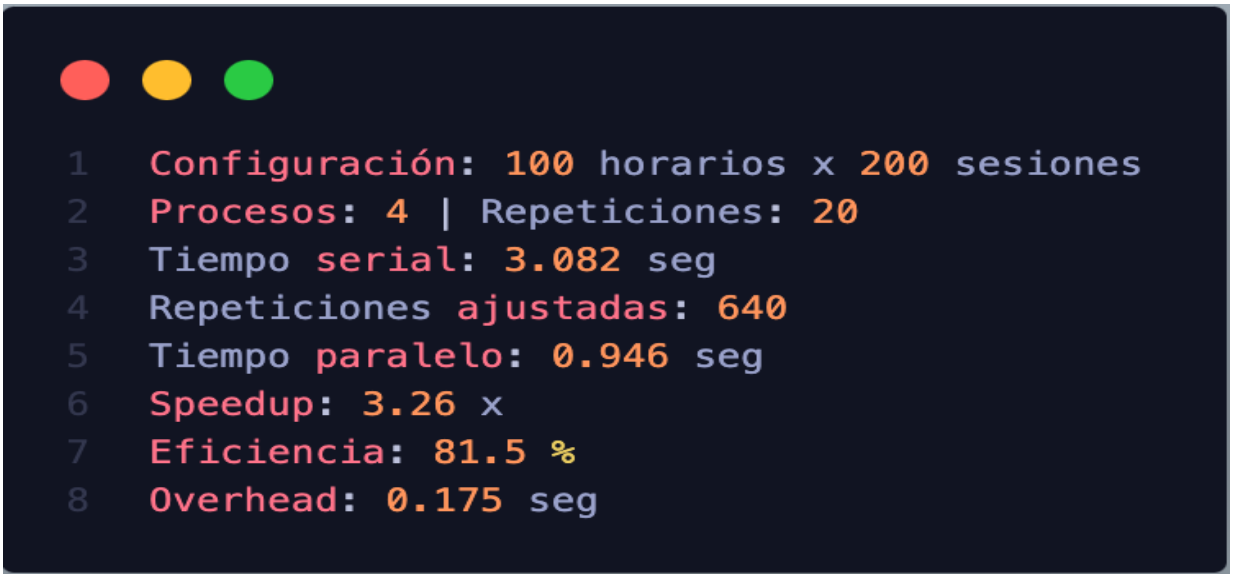
1. El cuello de botella principal fue que la carga por proceso era mínima. La validación de un horario toma microsegundos, pero crear un proceso con spawn cuesta ~50ms.
2. Con 8 procesos y poca carga, cada uno recibía muy poco trabajo. Reducir a 4 mejoro la eficiencia porque, menos procesos con el mismo speedup da mejor eficiencia.

Oportunidades de mejora

- **Mas horarios o sesiones**: aumentar la carga real sin repeticiones creo que daría una prueba más representativa.

Conclusiones

- El paralelismo **no siempre acelera** los tiempos de la ejecución, solo es beneficioso cuando supera significativamente el overhead.
- Reducir el número de procesos (de 8 a 4) **aumento** la eficiencia, demostrando que más procesos ≠ mejor rendimiento.



GitHub para referencia del proyecto: https://github.com/SASD01/Threads-and-Process.git

Ubicarse en la SEMANA 1