Laura Llamas López – alumno09

Informe Laboratorio 5. Explotando el paralelismo con Python en CPUs.

Este laboratorio ha resultado fundamental para comprender las técnicas de paralelización en Python aplicadas a cálculo científico. La posibilidad de experimentar directamente con `multiprocessing` y `Numba` sobre problemas reales ha permitido observar de forma cuantificable el impacto de distribuir el trabajo computacional entre múltiples núcleos.

La práctica demuestra de manera efectiva que Python puro, aunque versátil y legible, presenta limitaciones severas de rendimiento para tareas intensivas en cómputo. La paralelización emerge como una solución práctica y necesaria para reducir drásticamente los tiempos de ejecución en aplicaciones científicas reales. Los ejercicios propuestos ilustran claramente cómo problemas que tardarían minutos en ejecutarse secuencialmente pueden resolverse en segundos mediante un uso adecuado de los recursos hardware disponibles.

La comparación directa entre `multiprocessing` con `pool` y `Numba` con `prange` resulta particularmente instructiva. Los resultados obtenidos muestran que, aunque ambas técnicas logran mejoras significativas, `Numba` con paralelización automática consigue tiempos absolutos superiores debido a su menor *overhead*. Esta diferencia de comportamiento se hace especialmente evidente al ejecutar los notebooks en el clúster mediante la cola `mendel`. La especificación explícita de usar la cola `mendel` para las ejecuciones es un acierto que merece destacarse. Esta decisión elimina la variabilidad en los resultados que podría surgir si cada estudiante eligiera una cola diferente, facilitando la comprensión de las tendencias observadas.

En cuanto a aspectos mejorables, habría sido valioso incluir una explicación más detallada sobre el funcionamiento interno de `multiprocessing` y `Numba`. Aunque el material teórico cubre estos conceptos, una síntesis orientada específicamente a la práctica ayudaría a comprender mejor de dónde nacen las diferencias observadas en tiempos de ejecución y comportamientos, lo que enriquecería significativamente la experiencia de aprendizaje.

Durante la realización de la actividad extra surgió un aspecto que merece mención: la importancia del tamaño del problema para observar correctamente los beneficios de la paralelización. Inicialmente, con 10^6 puntos en el cálculo de π, `multiprocessing` mostraba resultados contraintuitivos donde usar más núcleos no mejoraba (o incluso empeoraba) el rendimiento. Solo al aumentar el tamaño a 10^7 puntos se hizo evidente la mejora esperada. Este fenómeno, relacionado con el *overhead* de creación y comunicación entre procesos, es un concepto fundamental en paralelización que merecería mayor énfasis en el guion de la práctica. Orientaciones sobre cómo elegir tamaños de problema adecuados para cada técnica de paralelización ayudarían a evitar confusión.

En conclusión, a pesar de los desafíos técnicos mencionados, el laboratorio cumple de forma efectiva con su objetivo de proporcionar competencias prácticas en paralelización de código Python para HPC. Las habilidades adquiridas sobre cuándo y cómo aplicar cada técnica de paralelización son directamente transferibles a proyectos científicos reales donde el rendimiento computacional es crítico.