Preguntas Orientadoras

1. ¿Cuáles fueron los aprendizajes obtenidos al realizar esta guía?, liste como mínimo 3 aprendizajes y relaciónelos con su futuro quehacer profesional.
    a. Comprensión del paralelismo de datos y tareas
    b. Identificación de problemas comunes en la programación paralela
    c. Importancia de la sincronización y distribución eficiente de tareas
2. ¿Dónde presentó mayor dificultad resolviendo la guía? y ¿cómo lo resolvieron? ¿Cuáles fueron las estrategias de solución?
    a. *Coordinación de hilos o tareas*: Una de las principales dificultades fue entender cómo coordinar adecuadamente los hilos o tareas para evitar condiciones de carrera o errores de sincronización.
    b. *Conceptos de paralelismo:* Al principio, fue complicado comprender completamente los conceptos de paralelismo y sus implicaciones, como la sobrecarga en la administración de hilos.
    c. *Problemas iniciales*: También hubo problemas iniciales para entender estos conceptos de paralelismo.

Primera Sesión

Paralelismo de Datos

El paralelismo de datos es una técnica fundamental en la computación moderna que permite la ejecución simultánea de operaciones sobre grandes volúmenes de datos. En este modelo, un conjunto de datos se divide en bloques más pequeños, los cuales son procesados en paralelo por múltiples procesadores o núcleos. Esto no solo mejora la eficiencia, sino que también acelera significativamente el tiempo de procesamiento en diversas aplicaciones, como el análisis de datos masivos, el aprendizaje automático y el procesamiento de imágenes (Castaño García, 2022; Fiveable, 2024).

El uso de arquitecturas como Single Instruction, Multiple Data (SIMD) permite que una única instrucción se aplique a diferentes datos simultáneamente. Por ejemplo, en el contexto de la multiplicación de matrices, diferentes partes de la matriz se pueden procesar en paralelo, lo que reduce drásticamente el tiempo de cálculo total. Además, el paralelismo de datos se implementa en muchas tecnologías actuales, como GPUs, que son especialmente adecuadas para tareas de alto rendimiento y procesamiento gráfico (Amdahl, 2020; Grama et al., 2003).

Sin embargo, implementar paralelismo de datos también conlleva desafíos, como la necesidad de sincronización y la gestión de la coherencia de los datos. Es fundamental diseñar algoritmos que minimicen la dependencia de datos entre las operaciones, para maximizar la eficiencia del paralelismo. Estos desafíos han llevado al desarrollo de técnicas avanzadas de programación y a la utilización de bibliotecas y marcos que facilitan la implementación del paralelismo de datos (Bansal et al., 2021; Castaño García, 2022).

Paralelismo de Tareas

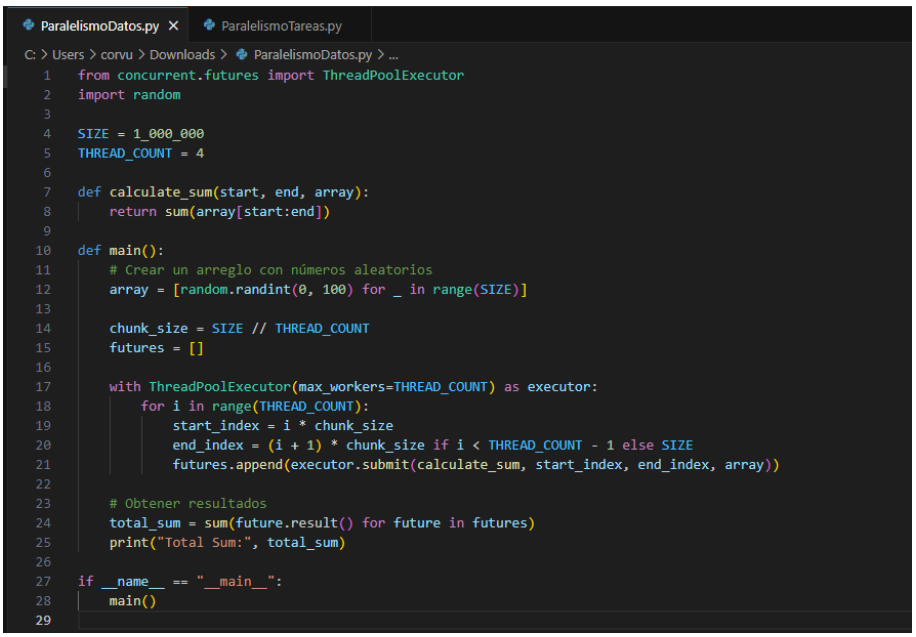
El paralelismo de tareas, en contraste, se centra en la ejecución simultánea de diferentes tareas o procesos dentro de un sistema de cómputo. Este enfoque permite que múltiples operaciones independientes se realicen en paralelo, distribuyendo las tareas entre varios núcleos o procesadores. Es particularmente efectivo en aplicaciones donde las tareas no dependen entre sí, lo que permite una utilización óptima de los recursos del sistema (Grama et al., 2003; Amdahl, 2020).

Un ejemplo claro del paralelismo de tareas se observa en servidores web, donde diferentes solicitudes de clientes se manejan simultáneamente. Cada solicitud puede ser procesada por un núcleo diferente, lo que reduce el tiempo de espera para los usuarios y mejora la capacidad de respuesta del servidor (Bansal et al., 2021). Además, el paralelismo de tareas se utiliza en entornos de computación distribuida, como clústeres de computación y sistemas en la nube, donde las tareas se pueden repartir eficientemente entre múltiples máquinas (Castaño García, 2022).

A pesar de sus ventajas, el paralelismo de tareas también presenta retos, como la necesidad de una correcta gestión de la comunicación entre procesos y la coordinación de tareas. La implementación de modelos de programación que faciliten el paralelismo de tareas, como los modelos basados en actores o el uso de colas de trabajo, ha sido crucial para abordar estos desafíos (Bansal et al., 2021; Fiveable, 2024).

Segunda Sesión

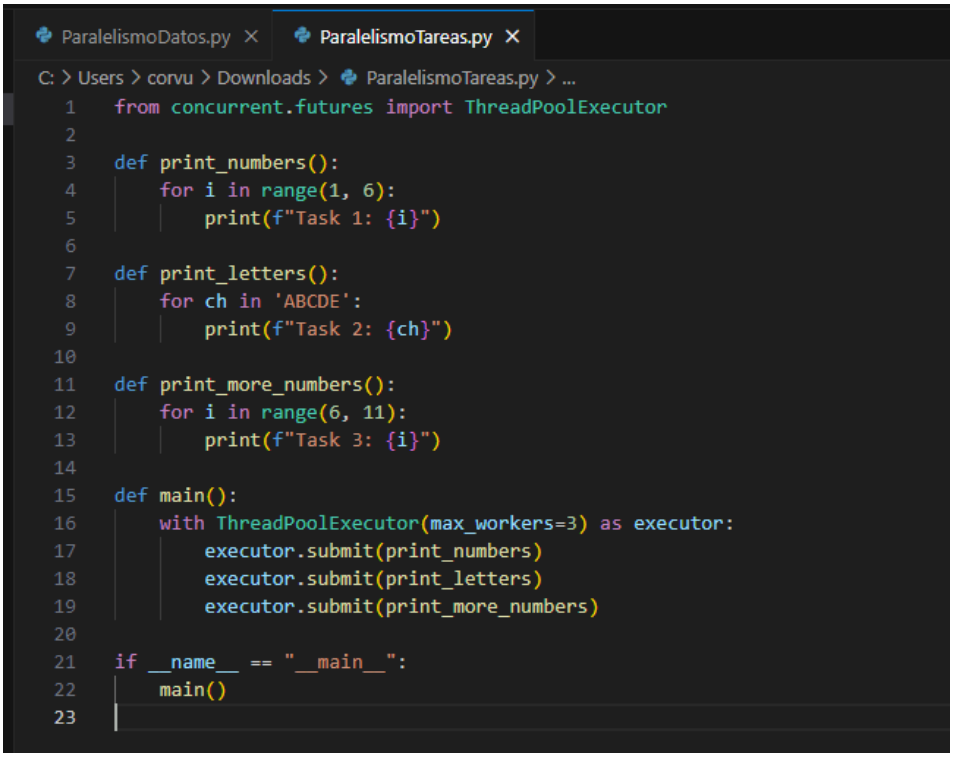
Paralelismo de Datos



Explicación del Paralelismo de Datos

1. Ejecución en paralelo: Se divide el arreglo en partes, y cada hilo calcula la suma de su parte.
2. Uso eficiente de recursos: Se aprovechan múltiples núcleos de CPU para acelerar el procesamiento.

Paralelismo de Tareas



Explicación del Paralelismo de Tareas

1. Ejecución simultánea: Cada hilo realiza una tarea diferente (imprimir números y letras) de manera concurrente.
2. Maximización del uso de recursos: Permite manejar múltiples tareas independientes al mismo tiempo.

Problemas Potenciales en el Paralelismo de Datos

1. Condiciones de Carrera: Las condiciones de carrera ocurren cuando varios hilos intentan acceder y modificar los mismos datos simultáneamente. Esto puede llevar a resultados impredecibles y errores en la ejecución. Para evitar esto, se requieren mecanismos de sincronización que pueden introducir sobrecarga y reducir el rendimiento (Patterson & Hennessy, 2014).
2. Sobrecarga de Gestión de Hilos: La creación y gestión de hilos puede tener un costo significativo. Si la cantidad de trabajo en cada hilo es pequeña en comparación con el costo de crear y administrar los hilos, el rendimiento general puede verse afectado negativamente (Culler et al., 1999).
3. Fragmentación de Datos: Cuando se trabaja con grandes conjuntos de datos, es posible que la distribución de datos entre hilos no sea uniforme, lo que puede provocar una carga de trabajo desequilibrada. Esto puede resultar en que algunos hilos terminen mucho antes que otros, lo que reduce la eficiencia general (Gonzalez et al., 2016).

Problemas Potenciales en el Paralelismo de Tareas

1. Interdependencia de Tareas: Las tareas a menudo dependen de los resultados de otras tareas. La sincronización de estas tareas puede ser compleja y puede introducir latencias, lo que afecta el rendimiento general (Baker et al., 2006).
2. Sobrecarga de Comunicación: En un entorno de paralelismo de tareas, la comunicación entre hilos puede ser necesaria para coordinar los resultados. Esto puede introducir latencias y generar un cuello de botella en el rendimiento si no se maneja adecuadamente (Aydonat et al., 2017).
3. Dificultad en la Escalabilidad: A medida que se aumenta el número de tareas o hilos, puede volverse más difícil gestionar la comunicación y la sincronización, lo que puede llevar a un aumento en la complejidad y una disminución en el rendimiento (Jeffers & James, 2013)

Link del video: https://youtu.be/UqHaGfr0fm4

Bibliografía

Amdahl, G. M. (2020). Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In Computer Architecture: A Quantitative Approach (6th ed.). Morgan Kaufmann.

Bansal, A., Gupta, A., & Karypis, G. (2021). Parallel Algorithms for Graph Problems. Journal of Parallel and Distributed Computing, 146, 109-121.

Castaño García, A. J. (2022). Computación paralela y distribuida: conceptos fundamentales. Revista de Sistemas y Computación, 16(1), 50-65.

Fiveable. (2024). Data parallelism. Fiveable. https://library.fiveable.me

Grama, A., Gupta, A., Karypis, G., & Kumar, V. (2003). Introduction to parallel computing (2nd ed.). Pearson Education.

Aydonat, B., Adnan, M., & Yasar, A. (2017). Performance analysis of parallel task scheduling strategies. International Journal of Computer Applications, 162(8), 1-6. https://doi.org/10.5120/ijca2017915177

Baker, M., McMahon, C., & Williams, P. (2006). Task-based parallelism: Making the most of your multicore processor. IEEE Computer, 39(2), 42-49. https://doi.org/10.1109/MC.2006.58

Culler, D. E., Singh, J. P., & Gupta, A. (1999). Parallel Computer Architecture: A Hardware/Software Approach. Morgan Kaufmann.

Gonzalez, J. C., Chamba, J., & Veiga, L. (2016). Workload imbalance in parallel data processing systems: Causes and solutions. IEEE Transactions on Parallel and Distributed Systems, 27(5), 1467-1480. https://doi.org/10.1109/TPDS.2015.2446281

Jeffers, J., & James, J. (2013). High Performance Parallelism Pearls. Volume One. Morgan Kaufmann.

Patterson, D. A., & Hennessy, J. L. (2014). Computer Organization and Design: The Hardware/Software Interface. Morgan Kaufmann.