Pontificia Universidad Católica Madre y Maestra

Objetivo:

Se realizará un informe de lectura del libro: “Parallel and High Performance Computing”.

Asignatura:

Programación Paralela y Concurrente.

Nombre:

1014-3611 Starlin Frías.

Profesor:

Freddy A. Peña P.

Santiago de los Caballeros, República Dominicana

Junio 16 del 2024

Introducción:

El propósito de este informe es profundizar en la lectura “Parallel and High Performance Computing”. Dicha lectura consiste en los fundamentos claves y las prácticas avanzadas que van de la mano con los aspectos de una programación paralela y de alto rendimiento.

La tecnología lleva un avance bastante acelerado, y no es sorpresa para nadie que los datos crecen de manera exponencial hoy en día, por ende, por ser mayor la cantidad de datos, se requiere de una mayor eficiencia y rendimiento en la gestión de los mismos para mantener la misma velocidad o mejorarla. Es importante recalcar que la demanda de los usuarios finales es que los software, sistemas y servicios, ofrezcan la mejor velocidad posible, pero con el aumento inmensurable de información y datos, el mantener esto es complicado, ahí es donde entra en juego la programación paralela y concurrente.

El uso de técnicas avanzadas para mejorar los sistemas y permitir una mejor optimización se ha vuelto un papel crucial en el contexto de la computación [1]. Para ello se emplea diferentes paradigmas de programación, la que estamos acostumbrado a observar y usar en muchos software y servicios es la programación secuencial, la cual simplemente se ejecuta un paso tras otro, por lo que si una instrucción anterior no es ejecutada, automáticamente la ejecución se detiene, básicamente cada proceso depende del anterior.

Luego tenemos la programación paralela, mientras que la secuencial utiliza un solo proceso para mantener en ejecución la tarea, la paralela lo que hace es lo siguiente, esta permite la división de las tareas en múltiples procesos los cuales se ejecutan de manera simultánea [2]. Todo esto puede resultar en una exponencial aceleración de manera significativa en los tiempos de procesamientos.

Por otro lado también está la computación de alto rendimiento (HPC), está básicamente se enfoca en la utilización de las supercomputadoras y clusters de computadoras [1], todo para realizar cálculos que exijan mucha capacidades

computacionales y que sean complejos, además, de imposible o muy lentos de ejecutar en sistemas convencionales.

El libro "Parallel and High Performance Computing" de Robert Robey y Yuliana Zamora, es una obra fundamental debido a que esta aborda los temas expuesto anteriormente, con un enfoque práctico y también accesible. Permite a los lectores crear una perspectiva profunda de estos conceptos y técnicas esenciales para permitir un uso eficiente y de alto rendimientos en los sistemas computacionales. Además, no es solo relevante para las personas inmersas dentro de este entorno de programación paralela o esta disciplina, sino para cualquier tipo de profesionales que buscan de alguna manera sacar el mejor rendimiento de su equipo utilizando los núcleos del mismo para resolver problemas complejos en sus respectivas áreas.

En fin , todas estas técnicas y enfoques avanzados de programación, las cuales nos ofrecen el mayor rendimiento del equipo, son vitales para cualquier profesional que requiera de cálculos intensivos, o complejos dentro de cualquier rama o área. Por lo tanto, la lectura de Robey es una obra invaluable para profundizar en estos aspectos de relevante importancia.

Desarrollo:

Introducción al Cómputo Paralelo y de Alto Rendimiento:

- **Definición y Relevancia:** Como se ha mencionado anteriormente este paradigma de computación paralela permite la ejecución simultánea de los procesos en múltiples núcleos del equipo [1]. Esto es un contraste de lo que es la computación secuencial, donde las tareas se ejecutan una tras otra. El libro en esta parte subraya de manera reiterada la importancia de estas técnicas de paralelismo debido al auge creciente de las aplicaciones, y los datos que requieren gran capacidad de procesamiento, como lo es la inteligencia artificial, el aprendizaje automático y aspectos como la simulación de fenómenos naturales.

Es importante el hecho de conocer de estas herramientas y técnicas de programaciones paralelas y concurrentes, ya que brindan un rendimiento y eficiencia incuestionable en las aplicaciones que se permiten utilizarlas. El conocimiento de estas es de relevancia debido a que hoy en día se utilizan muchas aplicaciones de manera secuencial por la razón de que no muchos se preparan para comprender lo crucial de los conceptos y paradigma de la programación paralela y concurrente. El conocer estos conceptos permite sacar el máximo provecho a nuestro equipo, ya que no solo se enfoca el desarrollo de aplicaciones en aspectos de software, sino que va más allá, saca provecho del hardware en general, ya que toma en cuenta detalles como los procesadores, la GPUs, etc.

Recursos en Computadoras Paralelas:

Arquitectura de Hardware: En la lectura se analizan los diferentes componentes esenciales de cómo está conformado un sistema de cómputo paralelo, donde se incluyen los procesadores multinúcleo (sin esto no existe la programación paralela), la memoria compartida y la distribuida. De forma muy puntual y relevante se debaten las diferencias entre estos componentes y como cada uno puede ser utilizado para permitir un rendimiento máximo en las diversas aplicaciones existentes y las por venir.

Modelo de Programación: El libro por supuesto que da a conocer los diferentes modelos de programación, como lo es el modelo de la memoria compartida y también el modelo de paso de mensajes (bastante útil), da a conocer las ventajas y desventajas de cada uno en los posibles escenarios de aplicaciones.

En el modelo de la memoria compartida básicamente los múltiples procesos comparten el mismo espacio de memoria, esto facilita la interacción o comunicación entre ellos mismos, además, permite la escritura y lectura directas entre ellos. Todo

esto es utilizado con un único propósito de simplificar la comunicación entre procesos, y crear una estructura más sencilla, debido a que no existe la necesidad de que los mensajes se envíen de forma explícita. Esto tiene un problema, y es que como sabemos ambos procesos se encuentran en el mismo espacio de memoria, lo que puede causar ciertos bloqueos o problemas de concurrencia, por ende, a pesar de ser un modelo que facilita la comunicación debe ser gestionado con bastante cuidado para evitar los bloqueos.

Podríamos decir que las ventajas de la memoria compartida es la facilidad que brinda para la comunicación entre los diferentes procesos, además, es bastante eficiente siempre y cuando se haga una implementación correcta del mismo, y por último, es más fácil de entender a diferencia de otros modelos como lo son los de paso de mensajes.

Cuando hablamos de las desventajas del mismo, entra en juego todo lo mencionado anteriormente con respecto a los bloqueos y errores generado con problemas de concurrencia, también se debe ser muy selectivo de donde se utilizara, ya que no es conveniente su uso para enfoques de sistemas distribuidor, donde los procesos trabajan en las diferentes máquinas, y por último, tiene una curva un poco difícil a la hora de escalar, todo debido a los problemas en la sincronización.

Por otro lado tenemos el modelo de pasos de mensajes, en este modelo los procesos no se encuentran en el mismo espacio de memoria, por lo que no pueden comunicarse directamente, por lo que necesitan de alguna manera para lograr pasar los mensajes entre ellos. En este modelo cada proceso por sí solo tiene su espacio único de memoria, impidiendo que los procesos puedan compartir directamente entre ellos, pero esto evita los problemas de concurrencia de los cuales mencionamos en el modelo de memoria compartida. Como se ha mencionado, debido a que cada proceso se gestiona en una espacio de memoria diferente es un poco más complejo la elaboración de la misma estructura para permitir el funcionamiento correcto del paso de mensajes.

Las ventajas de los pasos de mensajes son la baja probabilidad de que se ocasione problemas de concurrencia, debido a que todos los procesos se encuentran en espacio diferentes, además, permite una mejor seguridad y encapsulamiento para cada proceso en particular, por la misma razón de que no comparten detalles de ellos con los demás, por ende, es como si fueran privados.

Cuando hablamos de las desventajas que se encuentran en este modelo, se podría mencionar la necesidad de mantener una gestión de cada proceso en particular, lo que necesita un mayor desarrollo y complejidad para controlar los mensajes de cada proceso.

En base a estos dos modelos, tanto el de memoria compartida como el paso de mensajes, el uso de los mismos dependerá de cual sea el problema a resolver, y cuál modelo se adapte mejor a este, se debe tener en cuenta que el de memoria compartida brinda una implementación más simple, pero aumenta problemas de concurrencia, mientras que el paso de mensaje es más complejo depurar y entender, pero garantiza una probabilidad de problemas de concurrencia menor.

Evaluación del Rendimiento:

- **Medición y Medida del rendimiento:** Se mostraron técnicas con el objetivo y el propósito de medir la eficiencia y el rendimiento de las aplicaciones que trabajan en paralelo. Los enfoques que se mostraron fueron el speedup y la eficiencia como tal, estas misma se muestran con pequeños ejemplos de optimización, que muestran de manera muy notable lo conveniente que son.

- **Amdahl's Law y Gustafson's Law:** También se hablaron de estas leyes, las cuales básicamente tratan de explicar cuales son los límites de lo que es la aceleración que se puede lograr con el paralelismo, y cómo

estos límites pueden ser errados, o en casos aprovechados, todo dependiendo la naturaleza del problema en cuestión.

Buenas Prácticas en el Desarrollo:

Es importante tener presente en las diferentes situaciones computacionales que no todo se resuelve con una misma solución, con esto me refiero a que en muchas personas aplican las misma soluciones a todos los escenarios o entornos con los que se encuentran. No es recomendable aplicar la misma solución a todo, debido a que no hay ninguna herramienta o práctica que se adapte a todos los escenarios de la forma más óptima, por ello la buena práctica, como se menciona en muchos puntos en el libro, es crucial.

La buena práctica en el desarrollo lo que permite es identificar los diferentes problemas, y la mejor manera de resolverlos, ya que se basa en los fundamentos de las cosas, no solo permanece con lo superficial, y cuando se conoce lo fundamentos respecto a algo, se puede trabajar en eso con una buena estructura de datos y las mejores soluciones.

En la lectura del libro se mencionan diferentes herramientas junto con los diferentes entornos de desarrollo para lograr una buena aplicación paralela. Dentro de estas herramientas se encuentran lo que son compiladores, depuradoras y ciertas bibliotecas específicas para el paralelismo. Entre los compiladores tenemos: Intel Parallel Studio XE, GNU Compiler Collection (GCC). Los depuradores: GNU Debugger (GDB), TotalView. Las bibliotecas: MPI (Message Passing Interface), CUDA (Compute Unified Device Architecture)

Estructuras de Datos y Algoritmos:

Este tema de las estructuras de datos siempre está presente en cualquier libre que busque la mejor eficiencia y rendimiento de una aplicación. Una de las estructuras que se tratan en el libro son las matrices dispersas, estas matrices básicamente permiten un mejor manejo para la representación de las matrices en las cuales muchos de los valores son 0 o en muchos casos no se encuentran presentes. Debido a lo mencionado anteriormente con respecto a estas estructuras de datos, en muchas ocasiones se utilizan para obtener un mejor rendimiento en cuanto al almacenamiento y el procesamiento de datos. La forma en que trabajan es que en lugar de almacenar los elementos de la matriz, incluyendo los ceros, estas mismas solo permiten guardar los elementos que no son nulos y las ubicaciones.

También se hace mención de la estructura conocida como las colas concurrentes, las cuales básicamente nos permiten una manera óptima y segura de gestionar los múltiples accesos de forma simultánea en los programas concurrentes. Son muy fundamentales su uso en los sistemas, debido a la facilidad para ejecutar varios procesos que necesitan añadir y extraer elementos de la cola, y todo evitando generar inconsistencias.

El objetivo o propósito general de la estructura de datos de colas concurrentes es permitir el acceso simultáneo sin generar ningún conflicto entre los datos de un proceso. Esta estructura incorpora dentro de sí lo que son mecanismos de sincronización como lo son bloqueos, los famosos wait-free, todo con la misión de garantizar una integridad genuina en los datos.

Diseño de Algoritmos Paralelos:

Para la creación de una buena aplicación en paralelo, se necesita de un buen diseño de algoritmos para garantizar que sea óptimo. La lectura trata las diferentes formas que existen para lograr este propósito de alcanzar un algoritmo que pueda ser corrido en paralelo y con un buen rendimiento, entre estos se mencionan técnicas como son la descomposición de tareas, la paralelización de bucles y por último, el manejo de las dependencias de datos.

La descomposición de tareas como su nombre lo indica su propósito se basa en ir dividiendo un proceso general en trozos más pequeños de procesos, para facilitar las soluciones de los mismo, para al final reunir cada solución y obtener la solución general, pero todo estas divisiones se van ejecutando de manera separada y de forma concurrente.

Las maneras en las cuales se pueden descomponer estos procesos o tareas es de diferentes formas, una de ellas es la descomposición funcional esta busca separar los procesos en funciones o algo similar a un componente, pero que se puedan ejecutar de forma paralela. Por otro lado está la descomposición por dominio de datos, en el área donde más se utiliza es la científica, debido a que cada simulación se puede dividir para ejecutarse de forma simultánea y que sean procesadas de manera independiente.

Luego tenemos lo que es la paralelización de bucles como su mismo nombre lo dice es una técnica que se utiliza para conseguir ejecutar iteraciones de bucles de forma paralela. Estas son convenientes utilizar en situaciones particulares, donde las iteraciones no dependan de las demás, sino que sean independientes entre sí. Una de las herramientas que son mayormente utilizadas para lograr la paralelización de bucles es OpenMP.

Uno de los aspectos de alta importancia que trata la lectura es la forma en la cual se hace un buen manejo de las dependencias de los datos, ya que en ocasiones se tienen operaciones donde los resultados de estas pues dependen de otros, lo que ocasiona que un proceso no pueda ejecutarse hasta que termine el otro. Por ende, la gestión correcta

de estos escenarios es de suma importancia para garantizar una ejecución correcta y el rendimiento para una aplicación donde se usen algoritmos paralelos.

Existen los que son las dependencias inmediatas, y las dependencias transitivas, la primera sucede cuando de forma directa una iteración depende de la otra, estas dependencias se recomienda solucionarlas pues reorganizando el orden o las operaciones dentro del bucle, o con el uso de técnicas avanzadas de paralelismo. Por otra lado, las transitivas es cuando se cuenta con dependencias entre múltiples operaciones, lo que la convierten más difícil de manejar debido a su alto acoplamiento, igual se recomienda el uso de técnicas avanzadas, una de ellas es los gráficos de tareas.

Lenguajes de Programación para GPUs:

CUDA, OpenCL y HIP: El libro presenta estos diferentes lenguajes de programación para permitir que las aplicaciones se ejecuten en la GPUs en vez de la CPU. La lectura muestra la introducción a cada lenguaje de los mencionados anteriormente. Además, el libro enfatiza en las diferencias de cada lenguaje, realizando comparaciones para cada ciertos problemas que podrían presentarse, y en cuales entornos uno encaja mejor que otro.

En la lectura se presenta cada lenguaje por separado para dar una breve introducción, CUDA (Compute Unified Device Architecture) es una de las plataformas las cual fue creada por NVIDIA, todo fue con el objetivo de que los programadores o desarrolladores puedan utilizar la GPUs para sus aplicaciones o las tareas de computaciones en general. Lo que brinda es menos complejidad a la hora de escribir una aplicación que necesite trabajar de forma explícita en la GPUs.

Las ventajas que presenta este lenguaje de CUDA es que como se ha mencionado brinda bastante facilidad para trabajar en la GPU, y como ya sabemos esta ofrece un mayor rendimiento y optimización en las aplicaciones. También cuenta con bastante apoyo, y es bastante utilizada por la comunidad inclinada por la programación paralela

y concurrente, por ende, ofrece una amplia documentación, lo que significa que no permite muchas ambigüedades entre sus líneas de código.

Entre las desventajas que podemos encontrar de CUDA es que no es compatible con todo tipos de hardware, lo que puede ser una limitante bastante crucial para muchos desarrolladores, sucede que CUDA se encuentra arraigado con las GPU que pertenecen a NVIDIA, por ende, cualquier otro hardware que no cuente con esta GPU no podrá ser compatible con el mismo.

OpenCL (Open Computing Language) como su nombre lo indica este es una librería de open source, la cual ofrece un código abierto a la comunidad, bastante conveniente porque cualquiera con las capacidades necesarias podría aportar a las mejoras del mismo. Lo que brinda este lenguaje que lo vuelve muy competente y útil es su versatilidad para manejar los diferentes hardware, como se menciona anteriormente con CUDA que solo soporta los GPU de NVIDIA, bueno, pues OpenCL resuelve este asunto, ya que ofrece múltiples plataformas, incluyendo CPUs, GPUs, y otros aceleradores. Los creadores de este fueron los de Apple, actualmente se encuentra mantenido por la empresa o grupo de Khronos Group.

Entre las características y ventajas que ofrece OpenCL es su compatibilidad con la diversidad de hardware existente, lo cual es muy útil, y brinda una gran portabilidad (algo importante en el mundo tecnológico), esto es crucial debido a lo cambiante que es la tecnología de manera rápida. También debido a ser de código abierto está no está sujeta a ninguna empresa en particular o fabricante, lo que promueve un desarrollo más rápido de la misma por la competencia de la comunidad.

Cuando hablamos de las desventajas entra en juego su complejidad, debido a que es de mucho menor nivel a diferencia de CUDA, lo que requiere un mayor entendimiento de los fundamentos de la programación, por ende, su curva de aprendizaje es bastante compleja, y requiere de esfuerzo para comprenderla y dominarla en un punto. Otros aspecto que se podría mencionar entre sus desventajas es que debido a su

compatibilidad con los diferentes hardware este no es muy óptimo, lo que hace que en ciertos escenarios brinde un rendimiento menor al esperado.

HIP (Heterogeneous-computing Interface for Portability) fue desarrollado por AMD, esta brinda la facilidad de escribir códigos de formas paralelas tanto para GPUs y Nvidia pero que permanezca a AMD. Este es bastante óptimo y ofrece un buen rendimiento, ya que es similar a CUDA lo único que fue diseñado para ofrecer la portabilidad que no brinda CUDA.

Entre las ventajas se menciona su misma portabilidad, ya que permite flexibilidad al momento de la necesidad de traspasar códigos de CUDA a GPUs de AMD todo esto sin requerir de muchos esfuerzos. Además, también es de código abierto, donde cuenta con bibliotecas que brindan altos rendimientos de velocidad para realizar alguna tarea en particular.

Ejemplos Relevantes y Aplicaciones Prácticas:

Algo que a través de los años se ha querido lograr hacer es la predicción de eventos naturales, esto es algo muy complejo de realizar, y dado a todas las técnicas que se han utilizado para tratar de hacerlo posible, aún no hay una que funcione con certeza o precisión. En el libro se mencionan como algunas técnicas relacionadas a la computación son bastante útiles y sirven para simulaciones de los mismos eventos, permitiendo entrenar a los modelos de detección, de esta manera se busca agudizar o mejorar la precisión de los mismos.

El ejemplo mencionado anteriormente es algo bastante relevante, debido a que podría mitigar los grandes números de pérdidas que generan los eventos naturales, ya que si se logra obtener un buen modelo de clasificación para detección de estos, el tiempo de reacción o de preparación para emergencias sería mayor, lo que logra una mejor anticipación por partes de los ciudadanos de una población.

Una aplicación muy relevante y práctico es en el campo de la Inteligencia Artificial, como se sabe, los datos cada día son mayor a tiempos pasados debido a su aumento exponencial, y los algoritmos de aprendizaje automático requieren de la necesidad de procesar inmensas cantidades de datos para conseguir un buen modelo de clasificación , pero para procesar todos esos datos de manera rápida y más eficiente se utiliza el uso de GPUs y técnicas de paralelismo.

En los cálculos intensivos, se observan y exploran ciertos estudios de cálculos intensivos, donde se utilizan los diferentes algoritmos computacionales de forma paralela para realizar la simulación de interacciones moleculares, y el análisis de estructuras complejas en la ingeniería.

En los asuntos financieros es bastante útil la eficiencia y el mejor rendimiento, sucede que las transacciones financieras 1 segundo de retraso como tal es algo crítico y crucial, por ende es necesario el uso de las aplicaciones paralelas y concurrentes. También pueden ser muy útiles para las predicciones del mercado en tiempo real y los análisis de riesgos en sectores financieros.

Conclusiones:

El libro "Parallel and High Performance Computing" de Robert Robey y Yuliana Zamora nos brinda una perspectiva muy acertada para el campo de la programación paralela y concurrente, y además, bastante detallada de las técnicas y herramientas necesarias para lograr la mayor optimización y alto rendimiento en cuestión de la computación paralela.

Como se ha ido observando a través de todo el informe el cual está basado en la lectura, las técnicas de la programación paralelas son meramente cruciales para lograr la resolución de problemas que exijan una alta demanda de recursos computacionales o cálculos intensivos. Todo el propósito de la programación paralela y concurrente es permitir a los desarrolladores lograr el mejor rendimiento con los equipos disponibles, sacando provecho tanto del hardware como del software, logrando así reducir los

tiempos de ejecución de las aplicaciones, y otros aspectos que requieran de altas demandas computacionales.

En conclusión, lograr comprender las técnicas que componen la programación paralela puede llegar a ser una curva de aprendizaje un tanto compleja, pero los beneficios que ofrece en cuanto a rendimiento y eficiencia son indudablemente lo suficiente para comprender la importancia de estas. La lectura del libro brinda una perspectiva lo suficientemente amplia para reconocer su vitalidad y el porqué es tan crucial. En fin, la programación paralela y concurrente nos permite lograr aprovechar el máximo rendimiento de nuestro equipo, por ello es importante para cualquier profesional sin importar a la área que permanezca el mismo.

Referencias Bibliográficas:

- [1] R. Robey and Y. Zamora, *Parallel and High Performance Computing*. Shelter Island, NY, USA: Manning Publications, 2021. [Online]. Available: https://learning.oreilly.com/library/view/parallel-and-high/9781617296468/

- [2] Difference Between Sequential and Parallel Computing," GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/difference-between-sequential-and-parallel-computing/. [Accessed: May 18, 2024].