

UNIVERSIDAD TECNOLÓGICA DE SANTIAGO (UTESA)



ASIGNATURA:

Algoritmos Paralelos

TEMA:

Tarea Semana 2

NOMBRE Y MATRICULA

YEYLEN ACOSTA DOMINGUEZ

1-19-1201

PRESENTADO A:

IVAN MENDOZA

SANTIAGO, REP. DOM.

06-06-2025

Procesamiento paralelo

El procesamiento paralelo es una técnica fundamental en la computación moderna que permite ejecutar múltiples tareas o procesos de manera simultánea, dividiendo un problema en subproblemas más pequeños que pueden resolverse al mismo tiempo. Esta metodología no solo incrementa significativamente la velocidad de ejecución, sino que también optimiza el uso de los recursos disponibles, como los múltiples núcleos de un procesador o un conjunto de máquinas distribuidas. Al permitir que varias operaciones se ejecuten en paralelo, se reduce el tiempo total requerido para completar tareas complejas, lo cual es crucial en contextos donde el rendimiento y la eficiencia son esenciales, como en el análisis de grandes volúmenes de datos o la simulación científica.

Modelos De Computos Paralelos

Los modelos de cómputo paralelos representan las diversas formas en que se puede organizar y ejecutar la computación paralela. Entre los más conocidos se encuentran el modelo de memoria compartida y el modelo de memoria distribuida. En el primero, varios procesadores acceden a una memoria común, facilitando la comunicación entre tareas, mientras que en el segundo cada procesador tiene su propia memoria local y se comunica con otros mediante mensajes. También existen enfoques híbridos y modelos específicos como SIMD (Single Instruction, Multiple Data) y MIMD (Multiple Instruction, Multiple Data), que permiten aprovechar distintos niveles de paralelismo según la arquitectura subyacente. La elección del modelo adecuado depende de la naturaleza del problema a resolver y de los recursos computacionales disponibles.

Desempeño Computacional De Algoritmos Paralelos

El desempeño computacional de algoritmos paralelos es una métrica clave para evaluar la eficiencia de estas técnicas. Se analiza en función de la aceleración (speedup), la eficiencia y la escalabilidad. La aceleración mide cuánto más rápido se ejecuta un algoritmo en paralelo comparado con su versión secuencial. La eficiencia evalúa el uso proporcional de los recursos paralelos, y la escalabilidad determina cómo se comporta el algoritmo al aumentar el número de procesadores. Un algoritmo paralelo bien diseñado debe lograr un equilibrio entre dividir el trabajo de manera equitativa y minimizar la sobrecarga de coordinación y comunicación entre procesos, lo cual nos lleva al siguiente concepto crucial: la complejidad de la comunicación.

Complejidad De La Comunicacion

La complejidad de la comunicación se refiere al costo computacional asociado con el intercambio de datos entre tareas paralelas. En arquitecturas distribuidas, este aspecto puede convertirse en un cuello de botella si no se gestiona correctamente, ya que el tiempo dedicado a la sincronización o transferencia de datos puede superar los beneficios del paralelismo. Una buena estrategia de diseño debe reducir la dependencia entre tareas, usar estructuras de datos adecuadas y minimizar los puntos de sincronización, de forma que la comunicación no degrade el rendimiento global del sistema.

Optimizacion

La optimización, en este contexto, implica refinar tanto el algoritmo como su implementación para maximizar la eficiencia del sistema paralelo. Esto puede lograrse mediante la eliminación de redundancias, la distribución balanceada de carga, el uso eficiente de memoria y la minimización de la latencia en la comunicación. Optimizar un sistema paralelo requiere no solo conocimientos algorítmicos, sino también un entendimiento profundo de la arquitectura del hardware subyacente, ya que los cuellos de botella pueden surgir a nivel de CPU, GPU, red o disco.

Notificacion Push

En el ámbito de las aplicaciones modernas, la *notification push* o notificación push juega un papel clave en la interacción usuario-aplicación. Este mecanismo permite que los servidores envíen mensajes o alertas de manera proactiva a dispositivos móviles o navegadores sin que el cliente tenga que solicitar información constantemente. Es ampliamente utilizada en aplicaciones móviles, sistemas de mensajería y servicios web en tiempo real, ofreciendo una experiencia más dinámica y reactiva. Las notificaciones push requieren una arquitectura que soporte conexiones persistentes, lo que a menudo implica el uso de sockets, colas de mensajería y servicios en la nube especializados.

Docker

Para facilitar la implementación y despliegue de sistemas complejos como estos, herramientas como Docker se han vuelto indispensables. Docker es una plataforma de contenedorización que permite empaquetar aplicaciones junto con todas sus dependencias en contenedores aislados y portables. Esto garantiza que el software se ejecute de manera consistente en cualquier entorno, ya sea local, de prueba o producción. Con Docker, los desarrolladores pueden crear entornos reproducibles, reducir errores causados por discrepancias en configuraciones y acelerar el ciclo de desarrollo.

Docker-Compose

Docker-Compose es una extensión de Docker que permite definir y administrar múltiples contenedores como un solo servicio. A través de un archivo YAML, se pueden orquestar contenedores que colaboran entre sí, como una base de datos, una API, una aplicación web y un servidor de notificaciones, asegurando su correcta conexión, volúmenes compartidos y redes personalizadas. Esta herramienta es vital para proyectos que utilizan arquitecturas de microservicios, ya que facilita la administración y el escalado de sistemas modulares.

Nginx

Complementando estas herramientas, NGINX se posiciona como uno de los servidores web y proxies inversos más poderosos y eficientes del mercado. NGINX se utiliza para balanceo de carga, manejo de conexiones concurrentes y redirección de tráfico a distintos servicios. En arquitecturas modernas, es común ver a NGINX actuando como punto de entrada a sistemas de microservicios, gestionando las peticiones entrantes y distribuyéndolas inteligentemente entre diferentes contenedores o instancias. Su alto rendimiento, bajo consumo de recursos y extensibilidad lo convierten en una solución robusta tanto para aplicaciones pequeñas como para sistemas empresariales de gran escala.

Amazon Web Services Principales Servicios

Finalmente, Amazon Web Services (AWS) ofrece un ecosistema completo de servicios en la nube que permiten implementar, escalar y mantener aplicaciones modernas con alta disponibilidad y bajo costo. Entre los servicios más destacados se encuentran EC2 para instancias de servidores virtuales, S3 para almacenamiento de objetos, RDS para bases de datos relacionales, Lambda para ejecución de funciones sin servidor (serverless), y API Gateway para exponer y gestionar interfaces RESTful de forma segura. AWS también proporciona herramientas avanzadas de monitoreo, autenticación, redes privadas virtuales (VPC), inteligencia artificial y contenedores (como ECS y EKS), facilitando la creación de soluciones escalables, resilientes y globales. Gracias a su modelo de pago por uso y su infraestructura distribuida mundialmente, AWS ha transformado la forma en que las empresas desarrollan y despliegan software en el siglo XXI.