UNIVERSIDAD TECNOLOGICA DE SANTIAGO
(UTESA)

PRESENTADO POR:

ERIK CRUZ
1-18-0759

PRESENTADO A:

IVAN MENDOZA

ASIGNATURA:

ALGORITMOS PARALELOS

ASIGNACION:

TAREA SEMANA 3

Santiago de los Caballeros
República Dominicana
Febrero, 2024

Investigar

Arboles balanceados

Los árboles balanceados son fundamentales en informática para mantener una estructura de datos eficiente y optimizada, especialmente cuando se trata de operaciones de búsqueda, inserción y eliminación. La idea clave detrás de un árbol balanceado es garantizar que la diferencia de alturas entre las ramas izquierda y derecha de cada nodo no sea excesiva, lo que ayuda a mantener la profundidad del árbol en un nivel aceptable.

Los árboles AVL, por ejemplo, son árboles binarios de búsqueda que siguen una regla estricta de balance: para cada nodo, la diferencia de alturas entre sus subárboles izquierdo y derecho no puede ser mayor que 1. Esto asegura que la altura del árbol se mantenga en O(log n), lo que resulta en tiempos de búsqueda y operaciones logarítmicas.

Por otro lado, los árboles rojo-negro son una variante de los árboles binarios de búsqueda que utilizan reglas de coloración para garantizar el balance. Cada nodo está etiquetado como rojo o negro, y se aplican reglas para garantizar que el árbol esté balanceado de manera aproximada.

Los árboles B y B+ son comúnmente utilizados en bases de datos y sistemas de archivos. Estos árboles permiten un mayor grado de flexibilidad en el número de hijos que puede tener un nodo, y se aseguran de que todas las hojas estén en el mismo nivel, lo que facilita la búsqueda eficiente.

Técnica Pointer Jumping

Es un enfoque utilizado en algoritmos de programación competitiva y estructuras de datos para mejorar la eficiencia en operaciones como búsqueda, consulta o actualización en estructuras de datos complejas, como árboles, grafos u otras estructuras.

La idea básica detrás del Pointer Jumping es utilizar punteros para realizar saltos exponenciales en la estructura de datos en lugar de recorrerla paso a paso. Esto permite reducir significativamente el tiempo de ejecución de ciertas operaciones, ya que en lugar de realizar un número lineal de comparaciones o consultas, el algoritmo puede alcanzar rápidamente una posición relevante en la estructura de datos mediante saltos exponenciales.

Un ejemplo común de aplicación de la técnica Pointer Jumping es en algoritmos para encontrar el ancestro común más bajo (LCA, Lowest Common Ancestor) en árboles. En lugar de realizar una búsqueda desde el nodo actual hacia arriba hasta encontrar el ancestro común más bajo, el algoritmo puede hacer saltos exponenciales desde el nodo actual hacia arriba utilizando punteros precalculados, lo que reduce

significativamente el tiempo de ejecución.

Divide y venceras

Es un paradigma de diseño de algoritmos que consiste en resolver un problema dividiéndolo en subproblemas más simples, resolviendo cada subproblema de forma independiente y luego combinando las soluciones para obtener la solución final al problema original. Esta estrategia se utiliza para resolver una amplia gama de problemas en informática y matemáticas.

El enfoque general de "Divide y Vencerás" sigue tres pasos principales:

- **Dividir:** El problema original se divide en varios subproblemas más simples y manejables. Esta división puede realizarse de manera recursiva o iterativa.

- **Conquistar:** Cada subproblema se resuelve de forma independiente. Idealmente, los subproblemas deben ser lo suficientemente pequeños como para ser resueltos de manera eficiente utilizando métodos directos, como la fuerza bruta o algoritmos más simples.

- **Combinar:** Las soluciones de los subproblemas se combinan para obtener la solución al problema original.

"Divide y Vencerás" se utiliza en algoritmos como el quicksort y la búsqueda binaria. En quicksort, la lista se divide en sub-listas, se ordenan recursivamente y se combinan. En la búsqueda binaria, el conjunto se divide repetidamente hasta encontrar el elemento deseado. Ambos ejemplos muestran cómo esta estrategia divide eficientemente problemas en subproblemas más simples para encontrar la solución final.

Particionamiento

El particionamiento se refiere a la división de un conjunto de datos en varias partes o particiones que pueden procesarse simultáneamente por múltiples procesadores o hilos de ejecución. Este enfoque es fundamental para aprovechar al máximo el potencial de paralelismo en sistemas computacionales modernos, donde se utilizan múltiples núcleos de CPU, GPU u otros recursos de procesamiento.

El particionamiento en algoritmos paralelos busca minimizar la comunicación y la sincronización entre los diferentes procesadores o hilos de ejecución, ya que estas operaciones pueden introducir sobrecarga y reducir el rendimiento global del sistema. En su lugar, se busca maximizar la cantidad de trabajo que se puede realizar simultáneamente de manera independiente. La eficacia del particionamiento depende en gran medida de la naturaleza del problema, la arquitectura del sistema y la habilidad del programador para diseñar algoritmos eficientes y paralelizables.

El particionamiento puede adoptar varias formas dependiendo del problema y del entorno de ejecución:

- **Particionamiento de datos:** Consiste en dividir el conjunto de datos en partes más pequeñas, donde cada parte se asigna a un procesador o hilo de ejecución para su procesamiento. Este enfoque es común en algoritmos paralelos como el procesamiento de imágenes, el análisis de datos distribuidos y la computación de alto rendimiento.

- **Particionamiento de trabajo:** En lugar de dividir los datos, este enfoque divide las tareas o trabajos a realizar entre los procesadores o hilos de ejecución disponibles. Cada procesador se encarga de una porción del trabajo total, lo que puede ser útil en algoritmos donde las operaciones no son igualmente costosas en términos de tiempo de CPU.

- **Particionamiento de dominio:** En este enfoque, el espacio de búsqueda o el dominio del problema se divide en regiones más pequeñas, que luego son procesadas independientemente. Esto es común en algoritmos de búsqueda, simulaciones físicas y problemas de optimización.

- **Particionamiento híbrido:** A menudo, se utilizan enfoques combinados que aprovechan tanto el particionamiento de datos como el de trabajo. Esto puede ser útil cuando se enfrentan problemas complejos que requieren un procesamiento paralelo eficiente.

Tecnica de Pipelining

Es un método utilizado en arquitectura de computadoras y diseño de microprocesadores para mejorar el rendimiento y la eficiencia de ejecución de instrucciones. En esencia, el pipelining divide la ejecución de una instrucción en varias etapas o pasos, permitiendo que múltiples instrucciones se ejecuten en paralelo, de manera superpuesta.

El proceso de pipelining implica dividir la ejecución de una instrucción en varias etapas secuenciales, como búsqueda de instrucción, decodificación, ejecución, acceso a memoria y escritura de resultados. Cada etapa del proceso se realiza de forma independiente y secuencial, pero múltiples instrucciones pueden estar en diferentes etapas del pipeline simultáneamente.

El funcionamiento del pipelining se puede visualizar como un flujo de trabajo en una línea de ensamblaje, donde cada etapa realiza una tarea específica en una instrucción antes de pasarla a la siguiente etapa. Mientras una instrucción está en una etapa, la siguiente instrucción puede entrar en la etapa anterior, lo que permite un solapamiento de ejecución.

Las principales ventajas del pipelining incluyen:

- **Mejora del rendimiento:** Al permitir que múltiples instrucciones se ejecuten en paralelo, el pipelining puede mejorar significativamente el rendimiento del procesador y reducir el tiempo total de ejecución de un programa.

- **Aprovechamiento del paralelismo a nivel de instrucción:** Al dividir la ejecución de instrucciones en etapas, el pipelining aprovecha el paralelismo a nivel de instrucción, lo que significa que diferentes partes de diferentes instrucciones pueden ejecutarse simultáneamente.

- **Utilización eficiente de los recursos del procesador:** El pipelining permite que los recursos del procesador, como la unidad de ejecución y la unidad de memoria, se utilicen de manera más eficiente al mantenerlos ocupados en diferentes etapas del pipeline.

Sin embargo, el pipelining también puede presentar desafíos, como:

- **Dependencias entre instrucciones:** Las dependencias entre instrucciones, como las dependencias de datos y las dependencias de control, pueden causar burbujas de pipeline, donde algunas etapas del pipeline no pueden avanzar debido a instrucciones anteriores que aún no se han completado.

- **Overhead de pipeline:** El pipelining introduce cierto overhead debido al registro de estado de las instrucciones en cada etapa del pipeline y al control del flujo de instrucciones a través del pipeline, lo que puede afectar ligeramente el rendimiento.

Aceleramiento en cascada

El aceleramiento en cascada implica aplicar primero una técnica de paralelismo para optimizar una parte del algoritmo y luego aplicar otra técnica de paralelismo para optimizar otra parte del algoritmo, y así sucesivamente. Cada técnica de paralelismo puede ser utilizada para abordar un aspecto específico del problema o para aprovechar diferentes tipos de recursos de hardware.

Por ejemplo, en un sistema heterogéneo que incluye tanto CPU como GPU, el aceleramiento en cascada podría implicar utilizar la CPU para ciertas tareas de procesamiento y luego utilizar la GPU para acelerar otras tareas que se ejecutan en paralelo. De manera similar, se pueden combinar técnicas de paralelismo a nivel de datos (por ejemplo, paralelismo SIMD) con técnicas de paralelismo a nivel de tarea (por ejemplo, paralelismo de hilos) para optimizar diferentes aspectos del algoritmo.

El aceleramiento en cascada es particularmente útil cuando no existe una única técnica de paralelismo que sea adecuada para optimizar todo el algoritmo. Al combinar múltiples técnicas de paralelismo de manera secuencial, se puede aprovechar mejor el potencial de paralelismo del sistema y maximizar el rendimiento global del algoritmo.

Tecnica Symmetry Breaking

La técnica de "Symmetry Breaking" (ruptura de simetría) se refiere a una estrategia utilizada en algoritmos y problemas de optimización para eliminar o reducir la simetría entre posibles soluciones. La simetría en este contexto se refiere a situaciones en las que múltiples soluciones son esencialmente equivalentes en términos de su estructura o configuración, lo que puede llevar a una duplicación innecesaria de esfuerzos de búsqueda o a soluciones subóptimas.

La ruptura de simetría busca identificar y explotar características únicas de una solución para evitar la exploración redundante de soluciones simétricas. Esto puede conducir a una reducción significativa en el espacio de búsqueda y una mejora en la eficiencia del algoritmo. Se pueden utilizar diferentes enfoques, como ordenamiento de variables, selección de representante, restricciones adicionales, inicialización aleatoria y heurísticas específicas del problema. La ruptura de simetría es útil en problemas donde la simetría puede conducir a una explosión combinatoria en el espacio de búsqueda, mejorando la eficiencia y el rendimiento del algoritmo al enfocarse en áreas más prometedoras del espacio de búsqueda.

Cluster

Se refiere a un grupo o conjunto de computadoras interconectadas que trabajan juntas para realizar tareas o procesamiento de datos de manera colaborativa. Los clusters son utilizados en una variedad de aplicaciones para mejorar el rendimiento, la disponibilidad y la capacidad de procesamiento de sistemas informáticos.

Hay varios tipos, como los de alta disponibilidad, balanceo de carga, alto rendimiento (HPC) y almacenamiento. Estos clusters se utilizan para mejorar el rendimiento, la disponibilidad y la capacidad de procesamiento de sistemas informáticos en una variedad de aplicaciones, desde servidores web hasta simulaciones científicas.

Balanceo de Cargas

Es una técnica utilizada en redes informáticas y sistemas distribuidos para distribuir el tráfico de manera equitativa entre múltiples recursos, como servidores, nodos de procesamiento o enlaces de red. El objetivo principal del balanceo de cargas es evitar que un solo recurso se sobrecargue mientras otros permanecen subutilizados, lo que mejora el rendimiento, la disponibilidad y la escalabilidad del sistema en su conjunto.

Algunas de las estrategias de balanceo de cargas son:

- **Round Robin:** Las solicitudes se asignan secuencialmente a los recursos en un ciclo repetitivo. Es simple y equitativa, pero no tiene en cuenta la carga de los recursos.

- **Balanceo de carga basado en ponderaciones:** Se asignan ponderaciones a los recursos según su capacidad o rendimiento. Las solicitudes se distribuyen de acuerdo con estas ponderaciones, lo que permite asignar más solicitudes a recursos más potentes.

- **Balanceo de carga basado en la carga actual:** Se monitorea la carga de cada recurso y las solicitudes se asignan al recurso menos ocupado en ese momento. Ayuda a evitar la sobrecarga y optimiza la utilización del sistema.

- **Algoritmos predictivos:** Utilizan modelos para anticipar la carga futura y distribuir las solicitudes en consecuencia, lo que ayuda a evitar problemas de congestión anticipadamente.

- **Balanceo de carga basado en contenido:** Las decisiones se toman en función del contenido de la solicitud, como la URL o el tipo de archivo. Esto puede optimizar el rendimiento al dirigir las solicitudes al recurso más adecuado para manejarlas.

Sistema Operativo

Es un software que actúa como intermediario entre los usuarios y el hardware de una computadora. Su principal función es gestionar los recursos del sistema, como la memoria, el procesador, el almacenamiento y los dispositivos de entrada/salida, para permitir la ejecución de programas de aplicación.

Los sistemas operativos pueden clasificarse en diferentes tipos, como sistemas de tiempo compartido, sistemas de tiempo real, sistemas distribuidos y sistemas embebidos, cada uno diseñado para satisfacer necesidades específicas en diferentes entornos informáticos.

Las funciones clave de un sistema operativo son:

- **Gestión de recursos:** Administra y asigna eficientemente los recursos del sistema, como memoria, CPU, almacenamiento y dispositivos de entrada/salida.

- **Gestión de procesos:** Controla la ejecución de los procesos o programas, incluyendo la creación, terminación y suspensión de procesos, así como la programación de la CPU.

- **Gestión de memoria:** Administra la memoria del sistema, asignando espacio de memoria a los programas en ejecución y gestionando la memoria virtual para optimizar el uso de la memoria física.

- **Gestión de archivos:** Proporciona interfaces para crear, leer, escribir y eliminar archivos en el sistema de almacenamiento, y organiza la estructura de directorios.

- **Gestión de dispositivos:** Administra los dispositivos de hardware, permitiendo a los programas acceder y utilizar dispositivos como discos, impresoras, teclados y ratones de manera eficiente.

- **Interfaz de usuario:** Proporciona una interfaz gráfica o de línea de comandos para que los usuarios interactúen con la computadora y ejecuten programas de manera

Procesador de Tareas

El procesador de tareas (también conocido como planificador de procesos) es un componente clave que se encarga de asignar tiempo de CPU a los diferentes procesos en ejecución. Su función principal es decidir qué proceso se ejecutará a continuación y durante cuánto tiempo, siguiendo ciertas políticas de planificación para optimizar el rendimiento del sistema.

El procesador de tareas puede implementar diversas políticas de planificación, como el planificador de prioridades, el planificador de tiempo compartido o el planificador en tiempo real, entre otros. Estas políticas determinan cómo se asigna el tiempo de CPU entre los diferentes procesos, según criterios como la prioridad, la equidad, la utilización de CPU, los plazos de ejecución y la respuesta del sistema.