UNIVERSIDAD TECNOLOGICA DE SANTIAGO

UTESA

ALGORITMOS PARALELOS – INF-025-001

Tarea Semana 2.

Presentado Por:

Benjamin Tavarez

1-19-2141

Presentado a:

Ing. Iván Mendoza

04 de febrero, 2024

Santiago de los caballeros,

República Dominicana

Procesamiento paralelo

Consiste en la división de una tarea informática en subtareas que se pueden realizar al mismo tiempo, pero por diferentes procesadores, obteniendo de este modo que esta se complete de más rápidamente. Es importante destacar que no todas las tareas son adecuadas para ser descompuestas en un procesamiento paralelo.

Este procesamiento posee 2 enfoques: Multiprocesamiento y Multihilo.

- **Multiprocesamiento**

  Es ejecutar varios procesos, al mismo tiempo, en diferentes procesadores. Este proceso utiliza mas recursos, pero su ventaja es que casi no necesita de cambios en la codificación existente y su implementación es mucho más fácil

- **Multihilo**

  Poseemos un solo proceso que es dividido en hilos programados para ejecutarse de forma simultanea en el contexto de ese proceso. Su implementación es más complicada y tienes un mayor potencial de problemas causados por las condiciones de carrera que ocurrirían si no es controlado el acceso a algún recurso. Ahora bien, debido a que los hilos comparten los mismos recursos, como RAM, su huella tecnológica es menor.

Modelos de cómputo paralelos

Hay diversas categorías de modelos de cómputo paralelo, clasificados según su método de comunicación, la gestión de acceso a la memoria, la ejecución de tareas y, más recientemente, la diversidad del hardware en el que se implementan.

- **Modelo de intercambio de mensajes:**

  En este enfoque, el algoritmo se divide en segmentos procesados en un conjunto (cluster) de nodos (CPUs) conectados por un bus de comunicación. La sincronización se logra mediante intercambio de mensajes entre los nodos, y cada CPU tiene su propia memoria compartida, con un núcleo ejecutando cada proceso. Un ejemplo representativo de este modelo es la biblioteca MPI (Message Passing Interface).

- **Modelo de programación por hilos:**

  Este modelo tiene como objetivo la ejecución simultánea de múltiples tareas y utiliza memoria compartida. Su estructura básica incluye hilos que contienen las instrucciones del programa y la memoria necesaria para ejecutarse de manera independiente. Algunas implementaciones de este modelo incluyen la biblioteca POSIX Threads (Portable Operating System Interface for Linux) y OpenMP (Open Multiprocessing), una interfaz de programación de aplicaciones (API).

- **Modelos heterogéneos:**

  En este enfoque, un procesador principal gestiona los cálculos y controla otros dispositivos computacionales (CPU/GPU). La programación paralela se realiza mediante programas kernel que implementan la funcionalidad en los dispositivos. Este modelo surge debido a la creciente prevalencia de computadoras que utilizan tanto CPU como GPU, buscando aprovechar al máximo los recursos de cálculo disponibles. Ejemplos notables de este modelo son OpenCL (Open Computing Language) y Nvidia CUDA (Compute Unified Device Architecture).

Desempeño computacional de algoritmos paralelos

Es una métrica utilizada para evaluar la eficiencia y velocidad de un algoritmo que se ejecuta en un sistema de procesamiento paralelo. Se toman en cuenta varios factores como:

- **Speedup (Aceleración)**, es la medida del desempeño en paralelo del algoritmo. Se representa con el tiempo en que se ejecuta el algoritmo de forma secuencial dividido entre el tiempo que toma en un sistema paralelo. Un resultado ideal, sería igual al número de procesadores.
- **Eficiencia**, mide el aprovechamiento real de la capacidad de computación paralela y es calculada con el speedup por el número de procesadores.
- **Escalabilidad**, es la capacidad que tiene el algoritmo paralelo de mantener o mejorar su desempeño al irle agregando mas procesadores. Los algoritmos escalables son aquellos que pueden manejar grandes volúmenes de datos, sin perder desempeño.
- **Balance de carga**, es la distribución equitativa de la carga de trabajo entre los procesadores. Los algoritmos paralelos deben ser diseñados para minimizar el desequilibrio de carga.
- **Overhead de comunicación**, la comunicación es costosa entre procesadores por lo que los algoritmos deben minimizar el overhead de comunicación para obtener un buen desempeño. Se logra reduciendo la cantidad de catos a transmitir entre procesadores y, de ese modo, optimizar los patrones de comunicación.
- **Granularidad**, es el tamaño de las tareas o subprocesos que se van a ejecutar en paralelo. Esta dividida en 2 categorías:
  - **Gruesa**, una grande cantidad de trabajo, independiente entre tareas y con poca sincronización.
  - **Fina**, cantidades de trabajo pequeñas, poco independientes y una alta demanda en sincronización.

Complejidad de la comunicación

La complejidad de la comunicación en sistemas paralelos o distribuidos se refiere a la cantidad de datos que deben ser transmitidos entre procesadores o nodos durante la ejecución de un algoritmo. Esta comunicación impacta significativamente en el rendimiento del sistema, ya que las operaciones de transmisión de datos son más costosas en tiempo y recursos que las operaciones de cálculo.

Para medir la complejidad de la comunicación, se utilizan varios indicadores:

- **Número de mensajes:** Se refiere al recuento total de mensajes enviados durante la ejecución del algoritmo. Una mayor cantidad de mensajes indica una complejidad más alta.
- **Cantidad de datos transmitidos:** Representa la cantidad total de datos transmitidos a través de la red durante la ejecución del algoritmo. Este indicador es especialmente importante cuando se manejan mensajes de diferentes tamaños.
- **Ancho de banda utilizado:** Se centra en la eficiencia del uso del ancho de banda de red para la transmisión de datos entre procesadores. Un uso eficiente del ancho de banda resulta en una complejidad de comunicación menor.
- **Latencia de comunicación:** Es el tiempo que tarda un mensaje desde su origen hasta su destino. Un algoritmo con una latencia alta puede generar retrasos significativos debido a la comunicación.

La evaluación de estos factores proporciona una comprensión completa de la complejidad de la comunicación en un sistema paralelo o distribuido, permitiendo optimizar el diseño y la implementación de algoritmos para lograr un mejor rendimiento.

Optimización

La optimización en el contexto de algoritmos paralelos busca mejorar tanto el rendimiento como la eficiencia al realizar múltiples tareas de forma simultánea en varios procesadores o núcleos.

Para lograr esto, se pueden seguir diversas técnicas o estrategias:

- **División de tareas eficiente**: Distribuir el trabajo de manera equitativa entre los procesadores para evitar desequilibrios de carga.
- **Minimización de la comunicación:** Dado que la comunicación es costosa, es crucial minimizar la cantidad de datos transmitidos entre procesadores. Estrategias incluyen comunicación asincrónica, agrupación de datos y reducción de sobrecarga de comunicación.
- **Algoritmos paralelos específicos:** Utilizar o diseñar algoritmos específicamente adaptados para trabajar de manera eficiente en paralelo.

- **Afinación de parámetros:** Realizar ajustes cuidadosos en los parámetros de los algoritmos paralelos, como el número de hilos o procesadores a utilizar.
- **Uso de memoria caché eficiente:** Optimizar la gestión de la memoria caché para evitar cuellos de botella y maximizar el acceso eficiente a los datos.
- **Parallel patterns (patrones de paralelización):** Emplear patrones como map-reduce, paralelismo de tareas o de datos para simplificar el diseño del algoritmo y permitir una optimización más efectiva.
- **Escalabilidad:** Considerar la capacidad del algoritmo para mantener su rendimiento a medida que se le agregan más recursos paralelos.
- **Herramientas de profiling y monitoreo:** Utilizar herramientas que identifiquen áreas de mejora en el rendimiento del algoritmo, como la identificación de cuellos de botella.