*Programación Paralela*

Es el uso de **múltiples recursos**, en este caso, **procesadores**, para resolver un problema. Este tipo de programación tiene un problema, lo divide en una serie de pasos más pequeños, entrega instrucciones y los procesadores **ejecutan** las soluciones al **mismo** **tiempo**. También es una forma de programación que ofrece los mismos resultados que la programación concurrente, pero en **menos tiempo y con más eficiencia**. Muchas computadoras, como laptops y computadoras personales, usan esta programación en su hardware para asegurar que las tareas se completen rápidamente en segundo plano.

* Ventajas

Hay dos ventajas principales al usar esta programación sobre la programación concurrente. Una es que **todos los procesos se aceleran** cuando se usan estructuras paralelas, lo que **aumenta la eficiencia y los recursos utilizados para lograr resultados rápidos.** Otro beneficio es que la computación paralela es **más rentable** que la programación concurrente simplemente porque lleva menos tiempo obtener los mismos resultados. Esto es increíblemente importante, ya que los procesos paralelos son necesarios **para acumular cantidades masivas de datos** en conjuntos de datos que pueden ser fáciles de procesar o para resolver problemas complicados.

* Desventajas

Existen varias desventajas para el procesamiento en paralelo. La primera es que puede ser **difícil de aprender**; La programación que apunta a arquitecturas paralelas puede ser abrumadora al principio, por lo que lleva tiempo entenderla completamente. Además, el ajuste de código no es sencillo y debe **modificarse para diferentes arquitecturas de destino** para mejorar el rendimiento correctamente. También es **difícil estimar resultados** consistentes porque la comunicación de resultados puede ser problemática para ciertas arquitecturas. Finalmente, el **consumo de energía** es un problema para quienes instituyen una multitud de procesadores para diversas arquitecturas; Se requerirá una variedad de tecnologías de **enfriamiento para enfriar los grupos paralelos.**

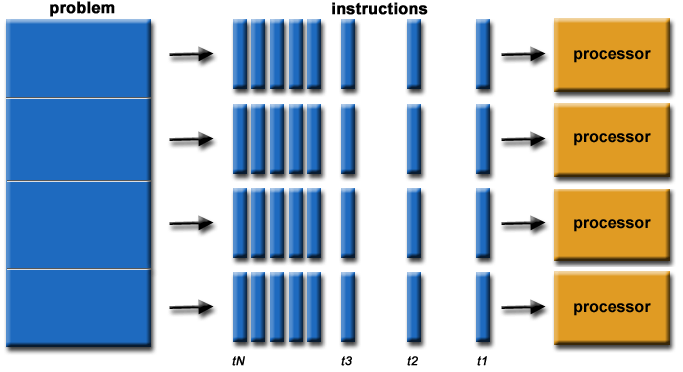
* ¿Dónde se usa?

Este tipo de programación se puede utilizar para todo, desde ciencia e ingeniería hasta venta minorista e investigación. Su uso más común desde una perspectiva social es en **motores de búsqueda web, aplicaciones y tecnologías multimedia.** Casi todos los campos en Estados Unidos usan esta programación en algún aspecto, ya sea para investigación y desarrollo o para vender sus productos en la web, lo que la convierte en una parte importante de la informática.

La computación paralela es el futuro de la programación y ya está allanando el camino para resolver los problemas con los que se encuentra constantemente la programación concurrente. Aunque tiene sus ventajas y desventajas, es uno de los procesos de programación más consistentes en uso hoy en día. Ahora que hay una respuesta a la pregunta de qué es la programación paralela, la siguiente pregunta debería ser, ¿cómo puede un profesional usar este tipo de programación en su campo?

En el sentido más simple, la computación paralela es el uso simultáneo de múltiples recursos informáticos para resolver un problema computacional:

* Un problema se divide en partes discretas que pueden resolverse simultáneamente
* Cada parte se desglosa en una serie de instrucciones.
* Las instrucciones de cada parte se ejecutan simultáneamente en diferentes procesadores
* Se emplea un mecanismo general de control / coordinación



*Límites y costos de la programación paralela*

* Ley de Amdahl:

La Ley de Amdahl establece que la aceleración potencial del programa se define por la fracción de código (P) que se puede paralelizar:

1

acelerar = --------

1 - P

Si ninguno de los códigos puede ser paralelo, P = 0 y la aceleración = 1 (sin aceleración).

Si todo el código está en paralelo, P = 1 y la aceleración es infinita (en teoría).

Si el 50% del código se puede paralelizar, la velocidad máxima = 2, lo que significa que el código se ejecutará el doble de rápido.

Al presentar el número de procesadores que realizan la fracción paralela de trabajo, la relación se puede modelar de la siguiente manera:

1

acelerar = ------------

P + S

---

norte

donde P = fracción paralela, N = número de procesadores y S = fracción en serie.

* Complejidad:

En general, las aplicaciones paralelas son mucho más complejas que las aplicaciones seriales correspondientes, tal vez un orden de magnitud. No solo tiene múltiples flujos de instrucciones ejecutándose al mismo tiempo, sino que también tiene datos que fluyen entre ellos.

Los costos de complejidad se miden en el tiempo del programador en prácticamente todos los aspectos del ciclo de desarrollo de software:

* + Diseño
  + Codificación
  + Depuración
  + Sintonización
  + Mantenimiento

Adherirse a las "buenas" prácticas de desarrollo de software es esencial cuando se trabaja con aplicaciones paralelas, especialmente si alguien además de usted tendrá que trabajar con el software.

* Escalabilidad:

Dos tipos de escalado en función del tiempo hasta la solución: escalado fuerte y escalado débil.

* + Fuerte escalamiento:
    - El tamaño total del problema permanece fijo a medida que se agregan más procesadores.
    - El objetivo es ejecutar el mismo tamaño de problema más rápido
    - La escala perfecta significa que el problema se resuelve en tiempo 1 / P (en comparación con el serial)
  + Escala débil:
    - El tamaño del problema por procesador permanece fijo a medida que se agregan más procesadores. El tamaño total del problema es proporcional al número de procesadores utilizados.
    - El objetivo es ejecutar un problema mayor en la misma cantidad de tiempo
    - El escalado perfecto significa que el problema Px se ejecuta al mismo tiempo que la ejecución de un solo procesador

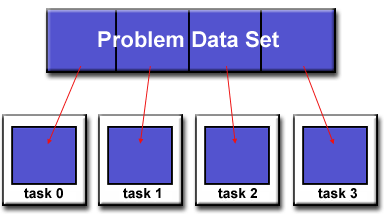
*Diseñando Programas Paralelos*

**Fraccionamiento**

* Uno de los primeros pasos para diseñar un programa paralelo es dividir el problema en "partes" de trabajo discretas que se puedan distribuir a múltiples tareas. Esto se conoce como descomposición o partición.
* Hay dos formas básicas de dividir el trabajo computacional entre tareas paralelas: descomposición de dominio y descomposición funcional.

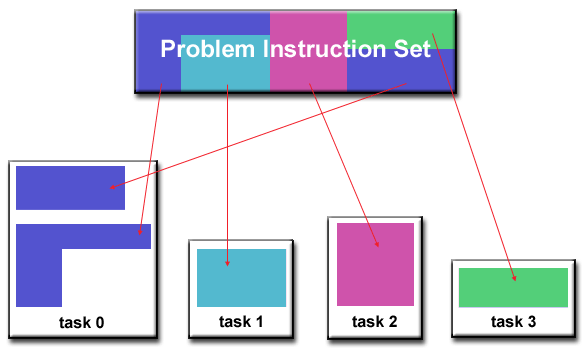
**Descomposición de dominio:**

En este tipo de particionamiento, los datos asociados con un problema se descomponen. Cada tarea paralela funciona en una parte de los datos.



**Descomposición funcional:**

En este enfoque, la atención se centra en el cálculo que debe realizarse en lugar de en los datos manipulados por el cálculo. El problema se descompone de acuerdo con el trabajo que debe hacerse. Cada tarea realiza una parte del trabajo general.



Diseñando Programas Paralelos

Factores para considerar:

Hay varios factores importantes a considerar al diseñar las comunicaciones entre tareas de su programa:

Sobrecarga de comunicación

La comunicación entre tareas prácticamente siempre implica una sobrecarga.

Los ciclos de la máquina y los recursos que podrían usarse para el cálculo se usan en su lugar para empaquetar y transmitir datos.

Las comunicaciones con frecuencia requieren algún tipo de sincronización entre tareas, lo que puede resultar en que las tareas pasen tiempo "esperando" en lugar de trabajar.

El tráfico de comunicación competitivo puede saturar el ancho de banda de red disponible, agravando aún más los problemas de rendimiento.

Latencia vs ancho de banda

latencia es el tiempo que se tarda en enviar un mensaje mínimo (0 bytes) desde el punto A al punto B. Comúnmente expresado en microsegundos.

el ancho de banda es la cantidad de datos que se pueden comunicar por unidad de tiempo. Comúnmente expresado como megabytes / seg o gigabytes / seg.

Enviar muchos mensajes pequeños puede hacer que la latencia domine los gastos generales de comunicación. A menudo es más eficiente empaquetar mensajes pequeños en un mensaje más grande, lo que aumenta el ancho de banda efectivo de las comunicaciones.

Visibilidad de las comunicaciones.

Con el modelo de paso de mensajes, las comunicaciones son explícitas y generalmente bastante visibles y están bajo el control del programador.

Con el modelo paralelo de datos, las comunicaciones a menudo ocurren de manera transparente para el programador, particularmente en arquitecturas de memoria distribuida. Es posible que el programador ni siquiera pueda saber exactamente cómo se realizan las comunicaciones entre tareas.

Comunicaciones sincrónicas versus asincrónicas

Las comunicaciones sincrónicas requieren algún tipo de "comunicación" entre tareas que comparten datos. El programador puede estructurarlo explícitamente en código o puede suceder en un nivel inferior desconocido para el programador.

Las comunicaciones síncronas a menudo se denominan comunicaciones de bloqueo, ya que otro trabajo debe esperar hasta que las comunicaciones se hayan completado.

Las comunicaciones asincrónicas permiten que las tareas transfieran datos de forma independiente entre sí. Por ejemplo, la tarea 1 puede preparar y enviar un mensaje a la tarea 2, y luego comenzar inmediatamente a hacer otro trabajo. Cuando la tarea 2 realmente recibe los datos no importa.

Las comunicaciones asincrónicas a menudo se denominan comunicaciones sin bloqueo, ya que se puede realizar otro trabajo mientras se realizan las comunicaciones.

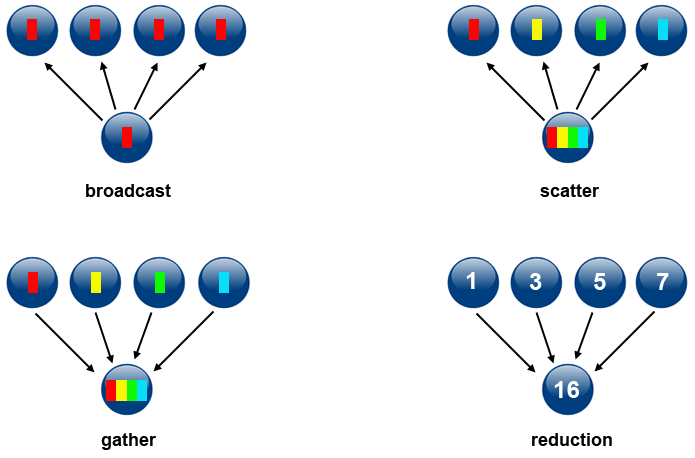
El cálculo entrelazado con la comunicación es el mayor beneficio para el uso de comunicaciones asincrónicas.

Alcance de las comunicaciones.

Saber qué tareas deben comunicarse entre sí es fundamental durante la etapa de diseño de un código paralelo. Los dos alcances descritos a continuación se pueden implementar de forma síncrona o asíncrona.

Punto a punto: implica dos tareas, una de las cuales actúa como emisor / productor de datos y la otra como receptor / consumidor.

Colectivo: implica el intercambio de datos entre más de dos tareas, que a menudo se especifican como miembros de un grupo común o colectivo. Algunas variaciones comunes (hay más):



Eficiencia de las comunicaciones.

A menudo, el programador tiene opciones que pueden afectar el rendimiento de las comunicaciones. Solo unos pocos se mencionan aquí.

¿Qué implementación para un modelo dado debería usarse? Utilizando el modelo de paso de mensajes como ejemplo, una implementación de MPI puede ser más rápida en una plataforma de hardware dada que otra.

¿Qué tipo de operaciones de comunicación se deben usar? Como se mencionó anteriormente, las operaciones de comunicación asíncrona pueden mejorar el rendimiento general del programa.

Tejido de red: diferentes plataformas utilizan diferentes redes. Algunas redes funcionan mejor que otras. Elegir una plataforma con una red más rápida puede ser una opción.

Gastos generales y complejidad

