Teoria 4- diseño de algoritmos paralelos

miércoles, 9 de abril de 2025 10:57

Pasos generales a realizar:
- Identificar porciones de trabajo (tareas) que puedan resolverse en paralelo
- Asignar tareas a procesos que se ejecutan en diferentes procesadores
- Distribuir datos de entrada, de salida e intermedios asociados con el programa
- Administrar accesos a datos compartidos
- Sincronizar procesos en diferentes etapas del programa

Descomposición en taresas:
Dividir el cómputo en partes más chicas-> a más tareas pequeñas más flexibilidad
Una primera aproximación consiste en pensar tareas de igual código (paralelismo de datos o de dominio) o tareas de código diferente (paralelismo funcional)

Descomposición de datos:
descomponer los datos asociados a un problema en pequeñas porciones (usualmente del mismo tamaño) y luego asociarle el cómputo relacionado a las mismas para generar las tareas.

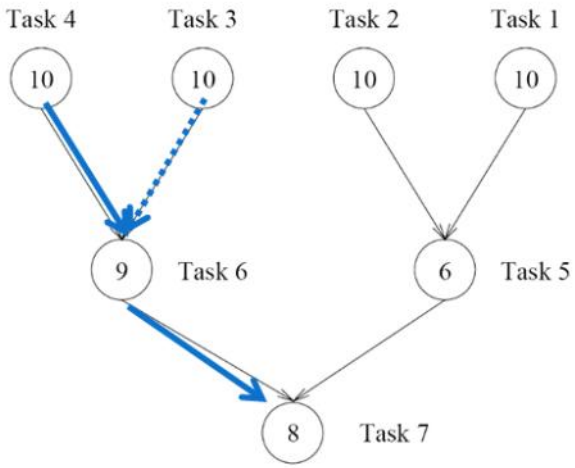
Descomposición funcional:
Se enfoca en el cómputo a realizar:divide el cómputo en tareas disjuntas y después ve los datos.
Lo ideal es que no necesiten los mismos datos, pero si si pasa hay que tener cuidado y comunicarlos para no cagarla
Enfocarse en el cómputo a realizar facilita la estructuración del programa y el descubrimiento de oportunidades de optimización (situación no tan obvia cuando uno se enfoca en los datos)

Descomposición de tareas:
Caso ideal: tareas full independientes, todas se pueden computar a la vez.
Se púede usar grafo de dependiencias para saber el orden

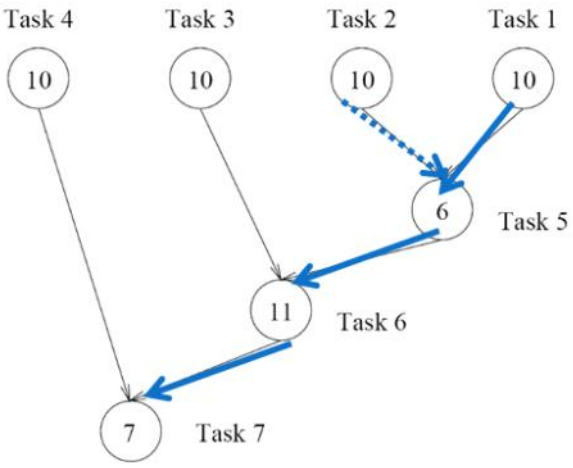
Gredo de concurrencia:
Cantidad de tareas que se ejecutan en paralelo
El nro varía durante la ejecución, lo que npos interesa es el promedio o el máximo
El camino criticop determina el promedio. Es el camino dirigido más largo entre un nodo inicial y uno final.
La suma de los pesos de los nodos es la lonmgitud del camin crítico

$$Grado\ de\ concurrencia\ promedio = Peso\ total / Longitud\ del\ camino\ crítico$$

Camino crítico corto--> mayor grado de concurrencia



(a)



(b)

$$GCP = \frac{63}{27} = 2.33$$

$$GCP = \frac{64}{34} = 1.88$$

Del análisis anterior, podría parecer que el tiempo de ejecución paralela se puede reducir indefinidamente al lograr una descomposición que sea cada vez más fina (más tareas de menor tamaño) ✂ esto no es lo usual · En general, hay un límite inherente al problema sobre qué tan fina puede ser una descomposición. Por ejemplo, el producto matriz-vector no admite más de n2 tareas concurrentes. · Además se debe tener en cuenta que las tareas deben comunicarse y sincronizar ✂ esto significa overhead que limita el speedup alcanzable Un adecuado balance entre cómputo y comunicación definirá el rendimiento alcanzable

No me queda claro

Aglomeración de tareas:

Ver si conviene combinar tareas para tener menos tareas pero más grande. El nro de tareas debería ser igual a la cantidad de nucleos de proces a usar

- Aumentar la granularidad: elimina la necesidad de comunicar datos entre ellas al combinarlas
- Mantenimiento de la flexibilidad: al combinar tareas se limita la escalabilidad del algoritmo. Si puede crear un nro variables de tareas, entonces tiene más portabilidad y escalabilidad.
- Menos costo de desarrollo

Cómo descompongo en tareas?

Técnica recursiva:

Bueno para los problmeas divide y vencerás.
Dividis en subproblemas independeintes. Despues seguis dividiendo recursivamente hasya conseguir cierta granularidad.
A veces se pueden combinar resultados parciales.
A veces hay que reprdenar el cómputo para que se pueda usar

Técnica basada en los datos:

Bueno para cuando se opera sobre grandes estructuras de datos,
Se particionar los datos a procesar
Se usa la partición para inducir una descomposición del cómputo en tareas
Varias formas:

- Basada en los datos de salida:
  - Se usa cuando cada elemento de salida se puede calcular independientemente como función de los datos de entrada. Ej multip. De matrices

    - A cada tarea se le asigna el cómputo relacionado a una porción de la entrada
- Basada en datos de entrada
    - No siempre se puede por salida. Ej cuando se buscan mínimos o máximos en una lista o Cuando se ordfena un vector
    - A cada tarea se le asigna una porción de los datos de entrada y hace todo los cómputos relacionados a la misma. A veces se hace reducción de salidas parciales
    - Ej: contar ocurrencias en un vector, buscar elementos, etc.
- Basada en datos ibntermedios
    - Hay veces que la salida de una etapa es la entrada de la siguiente
    - Entonces se descomponen los datps de una etapa intermedia
    - (no entendí :c)
- Exploratoria:
    - desarrollar algunos niveles desde la configuración inicial en forma secuencial. Luego, cada nodo es asignado a una tarea para realizar la búsqueda en forma concurrente. Cuando una la encuentra, le avisa al resto.
- Especulativaa:
    - Se usa cuando hay varios caminos distintos que puede tomar el programa.
    - Es como un case que se evaluan todas las opciones al mismo tiempo. CUando la enrada del case está disponible, se descartan las que no y sigue.
      En especulativa, la entrada de una bifurcación que lleva a múltiples nuevas tareas es desconocida.
      En cambio con exploratoria, la salida de las múltiples tareas que salen de una bifuración son desconocidas.
- Híbrida:
    - combinacion de las anteriores

Etapa de mapeo de tareas a procesos: