Pogramación Paralela y Distribuida

Licenciatura en
Ciencias de la Computación
UNCuyo – Facultad de Ingeniería

Ministerio de Educación, Ciencia y Tecnología
Universidad Tecnológica Nacional
Facultad Regional Mendoza

LICPaD
Laboratorio de Investigación en Cómputo Paralelo/Distribuido

UNCUYO UNIVERSIDAD NACIONAL DE CUYO | FACULTAD DE INGENIERÍA

Early Adopters Awarded
Fall 2011 (NSF/IEEE)

Pogramación Paralela y Distribuida

Aspectos de la Programación Paralela

Unidad 2

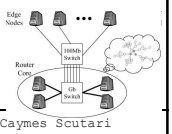
Ministerio de Educación, Ciencia y Tecnología
Universidad Tecnológica Nacional
Facultad Regional Mendoza

LICPaD
Laboratorio de Investigación en Cómputo Paralelo/Distribuido

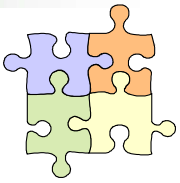
UNCUYO UNIVERSIDAD NACIONAL DE CUYO | FACULTAD DE INGENIERÍA

Introducción

- Repasemos un poquito…
  - Cálculo Paralelo
    - Es el método mediante el cual se divide un gran problema en componentes, tareas o cálculos más pequeños que pueden resolverse en paralelo, es decir "*al mismo tiempo*".
  - Su auge se ha debido a…
    - Demanda de rendimiento computacional
    - Complejidad creciente de los problemas a resolver
  - Su uso se ha facilitado gracias a…
    - MPP
    - Sistemas Distribuidos



Introducción



- Resumiendo…
  - Necesidad de recursos proporcionales a:
    - el volumen de datos que debe procesarse
    - la complejidad de los cálculos que deben efectuarse sobre los datos

Deben combinarse sofisticadas aplicaciones con potentes sistemas para resolver el problema en el menor tiempo posible, realizando un buen aprovechamiento de los recursos utilizados.

Introducción

- Sin embargo…
  - ¿Es directo el desarrollo de programas paralelos? ¿Basta con el paradigma secuencial y/o de orientación a objetos?
  - ¿Qué aspectos deben considerarse?
- En los ejemplos (lavado de ropa y jardinería) hemos visto que resulta crucial la coordinación entre las actividades que realiza cada actor.

Introducción

- **Diseño Paralelo**
  - Las aplicaciones deben diseñarse de acuerdo al paradigma paralelo
    - Modelos de algoritmos paralelos
    - Librerías de comunicación
      - Dado que por lo general trabajaremos sobre sistemas distribuidos
- **Alto rendimiento**
  - Las aplicaciones deben optimizarse para proveer un comportamiento eficiente
  - El comportamiento de la aplicación puede depender del conjunto de datos de entrada

Introducción

- **Paralelismo**
  - Variedad de acciones simultáneas que ocurren en un sistema, especialmente en un sistema paralelo
- **Concurrencia**
  - Propiedad de un algoritmo para ejecutar múltiples operaciones en un momento dado

La concurrencia constituye un requerimiento fundamental para la construcción de programas paralelos

Introducción

- Existen tres modelos de computadoras para implementar concurrencia:
  - Multiprogramación en un único procesador
  - Multiprocesador
  - Multicomputadora
  - Concurrencia Aparente
    - El número de procesos es mayor que el número de procesadores disponibles. Cada proceso avanza su ejecución intercalando con el resto.
  - Concurrencia Real
    - Cada proceso es ejecutado en un procesador

Introducción

- El desarrollo de aplicaciones paralelas debe realizarse de una forma específica que permita su ejecución en un sistema paralelo
  - Debe definirse el conjunto de acciones que puede ejecutarse simultáneamente
  - La ***concurrencia*** resulta esencial para obtener beneficios de rendimiento del uso de sistemas paralelos.

Introducción

- Para explotar la concurrencia ha sido necesario el diseño de:
  - Arquitecturas
  - Compiladores
  - Sistemas operativos

En esta asignatura nos centraremos en concurrencia a niveles más abstractos

que permitiesen implementar, extraer y utilizar el paralelismo con el fin de alcanzar mejoras en el rendimiento computacional

Introducción

- Como diseñadores y usuarios en general del paralelismo:

  ¿qué debe preocuparnos?

1. Discernir cuándo un problema puede resolverse utilizando un algoritmo paralelo en lugar de uno secuencial
   - *Algoritmo secuencial*
     - Secuencia de un solo flujo de sentencias para resolver un problema mediante la utilización de un unicomputador.
   - *Algoritmo paralelo*
     - Secuencia que indica cómo resolver el problema mediante la utilización de un sistema paralelo
2. Conocer los conceptos y herramientas necesarios para poder implementar el algoritmo correspondiente
   - Deben contemplarse aspectos adicionales al meramente funcional…

Introducción

- Un problema puede resolverse paralelamente cuando…
  - El algoritmo consta de varias subtareas independientes
    - Cada subtarea es asumida por un procesador
      - Ejemplo de los estudiantes que lavan la ropa

  - Maneja una cantidad considerable de datos
    - Se divide el conjunto de datos en subconjuntos
    - Cada subconjunto es tratado por un procesador
      - Ejemplo de los jardineros

Introducción

- El desarrollo de aplicaciones paralelas involucra un conjunto de aspectos adicionales a los funcionales
  - Ejemplos
    - Sistema distribuido
      - Retardos inherentes a la transmisión de la información
    - Sistema heterogéneo o tiempo compartido
      - Rendimiento variable de cada nodo

Introducción

- Resulta fundamental considerar:
  - Características del problema
  - Plataforma subyacente
- Aspectos clave en el diseño paralelo
  - **División** del trabajo en porciones más pequeñas
  - **Asignación** del trabajo a los nodos paralelos

Introducción

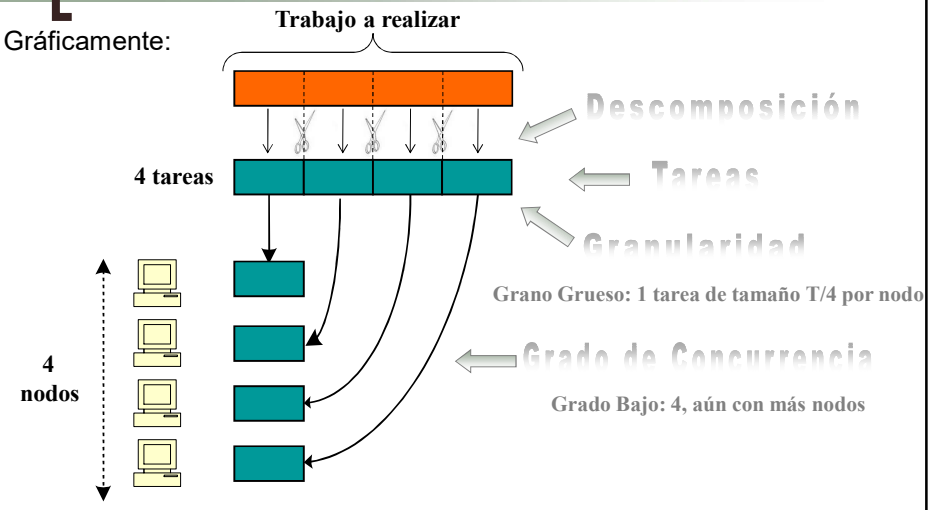
- **Descomposición**
  - Proceso de dividir el trabajo en porciones más pequeñas, algunas o todas pudiéndose ejecutar en paralelo
- **Tarea**
  - Cada una de las porciones o unidades computacionales en que se descompone el trabajo
- **Granularidad**
  - Cantidad y tamaño de tareas en las que se divide el problema
- **Grado de Concurrencia**
  - Máximo número de tareas que pueden ejecutarse simultáneamente en un programa paralelo. Normalmente cuanto más fina es la granularidad, más alto es el grado de concurrencia

Introducción

- **Balanceo de carga**
  - Equilibrio en el trabajo de todos los nodos.
    - Sistemas homogéneos
      - División equitativa de cómputo y comunicaciones
    - Sistemas heterogéneos
      - Asignación más compleja, deben considerarse las capacidades de cada nodo
- **Escalabilidad**
  - Configuraciones paralelas mayores pueden resolver problemas proporcionalmente mayores en tiempos de ejecución equivalentes, o problemas más pequeños en menor tiempo

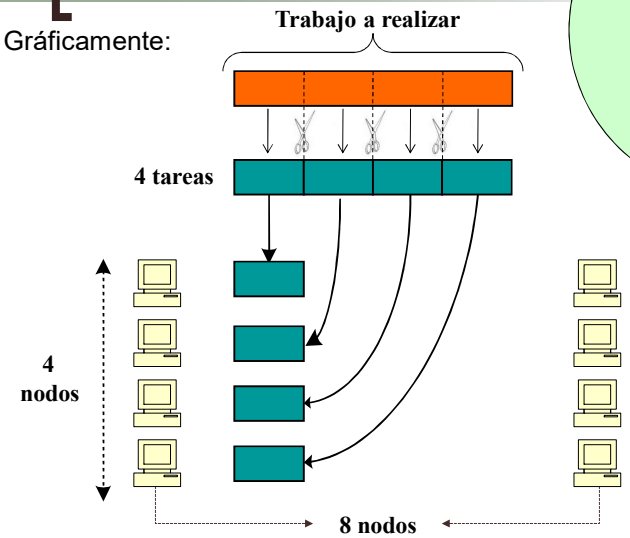
Introducción

Gráficamente:



Introducción

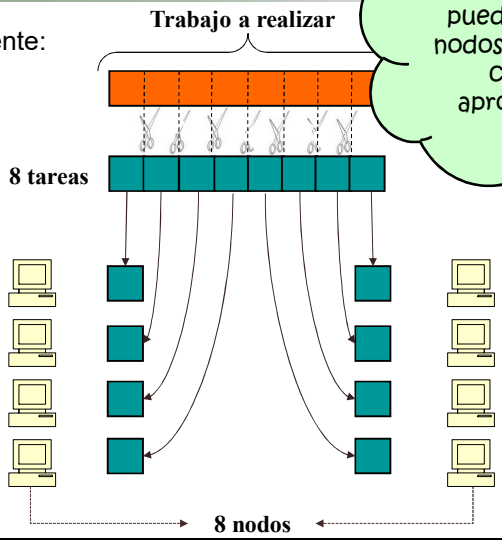
Gráficamente:



Aunque haya más nodos, el nivel de concurrencia está limitado por la granularidad. En este caso, no pueden trabajar más de 4 nodos a la vez.

Introducción

Gráficamente:



Con una granularidad más fina, es posible aumentar el grado de concurrencia, ya que pueden trabajar más nodos a la vez. En este caso, pueden aprovecharse los 8 nodos.

Pero ¡cuidado! Este es un ejemplo ilustrativo para comprender la relación entre granularidad y grado de concurrencia. Pero reflexionemos: si aumenta el número de nodos nuevamente, estaremos en el mismo problema de antes: no podrán trabajar más de 8 nodos a la vez…

Introducción

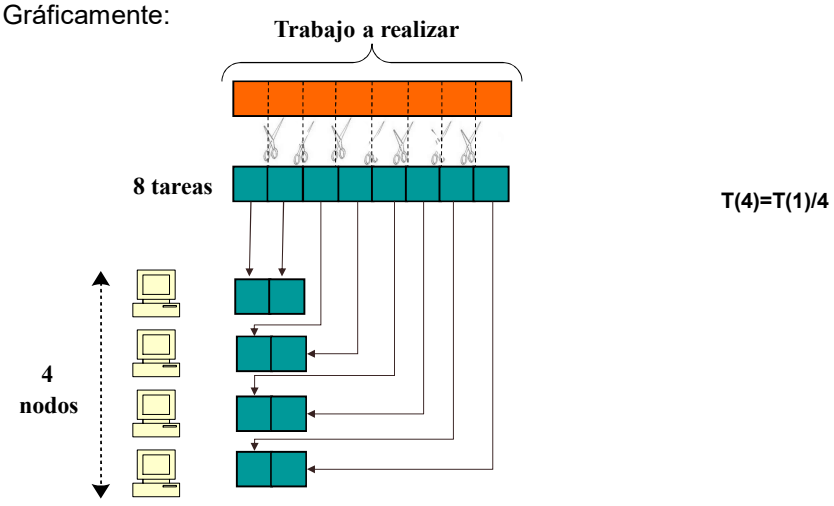
- Si bien cada una de las nociones presentadas son simples de entender, hallar una combinación eficiente de todas ellas es una tarea no trivial.
- La mala elección de alguna de ellas puede incidir drásticamente en la calidad y utilidad de la aplicación final.

Introducción

- Aquí entra en juego la ***escalabilidad***
  - La utilización de una política de descomposición inapropiada puede anular la posibilidad de ejecutar un programa paralelo en un entorno con más nodos con el fin de obtener los resultados en un tiempo más corto
  - Volvamos al ejemplo…

Resumen

Gráficamente:

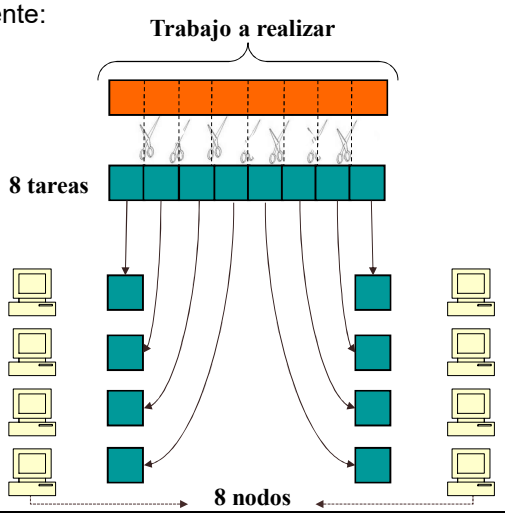


T(4)=T(1)/4

22

Resumen

Gráficamente:

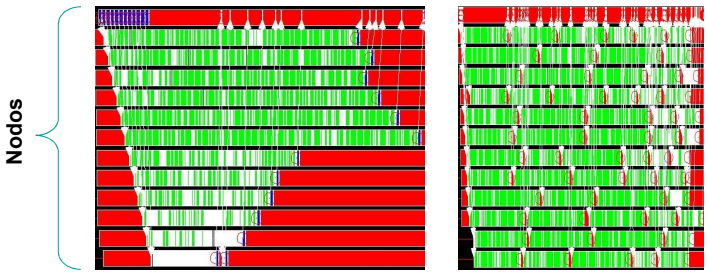


T(8)=T(1)/8

23

Introducción

- En cuanto al balanceo de carga, veamos los siguientes gráficos:



Ejecución desbalanceada Ejecución balanceada

Introducción

- **En resumen**:
  - El desarrollo de aplicaciones paralelas involucra un conjunto de aspectos adicionales a los funcionales
    - Características del problema
    - Plataforma subyacente
  - Aspectos clave en el diseño paralelo
    - **División** del trabajo en porciones más pequeñas
    - **Asignación** del trabajo a los nodos paralelos
      - Descomposición
      - Tarea
      - Granularidad
      - Grado de Concurrencia
      - Balanceo de Carga
      - Escalabilidad

Introducción

- Especificación de un algoritmo paralelo

1. Identificación de Paralelismo
2. Elección de la Estrategia de descomposición
3. Elección del modelo de algoritmo paralelo
4. Elección del modelo de comunicación

Identificación de Paralelismo

- Cuestión crucial:
  - Identificar las porciones de código donde puede explotarse el paralelismo

¿Cuándo puede ejecutarse en paralelo un conjunto de sentencias de un algoritmo?

  - Criterio simple:
    - ¿Las sentencias comparten datos o no?
    - ¿De qué manera lo hacen?
    - ¿Hay intersección entre entradas y salidas de sentencias?

Identificación de Paralelismo

- Estudiaremos las Condiciones de Bernstein que permiten identificar formalmente el paralelismo

  - Sea el siguiente fragmento algorítmico:

```
1:      funcion Dependencia(x,y)
2:          z := x*y
3:          w := 3*z
4:      finfuncion
```

Identificación de Paralelismo

```
1:      funcion Dependencia(x,y)
2:          z := x*y
3:          w := 3*z
4:      finfuncion
```

- Condiciones de Bernstein
  Sean C_1 y C_2 dos tareas.

C_1 `z := x*y` C_2 `w := 3*z`

C_1 y C_2 pueden ejecutarse en paralelo sin sincronización

si y sólo si

las siguientes condiciones se cumplen:

1. Salida(C_1) ∩ Entrada(C_2) = Ø
2. Entrada(C_1) ∩ Salida(C_2) = Ø
3. Salida(C_1) ∩ Salida(C_2) = Ø

Identificación de Paralelismo

1. ***Primera Condición*** *(que debe cumplirse)*
   Salida(C_1) ∩ Entrada(C_2) = Ø

"C_1 escribe datos que posteriormente C_2 NO lee"

- ¿Qué sucedería si NO se cumpliera esta condición?

C_1 escribe datos que posteriormente C_2 lee

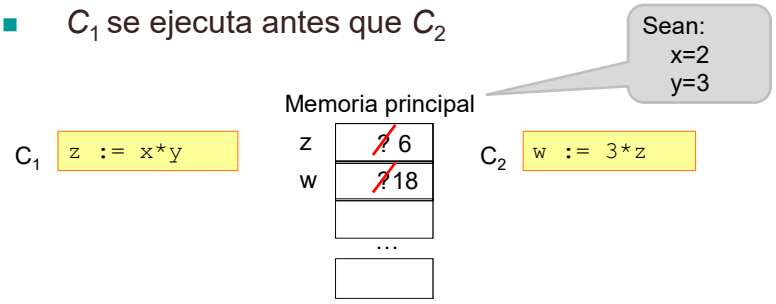
- Esta situación es comúnmente conocida como **Read-After-Write** o **RAW**

Identificación de Paralelismo

1. *Primera Condición: no* **RAW**

¿Qué sucedería si NO se cumpliera esta condición?
Veamos las diferentes situaciones…

- C_1 se ejecuta antes que C_2

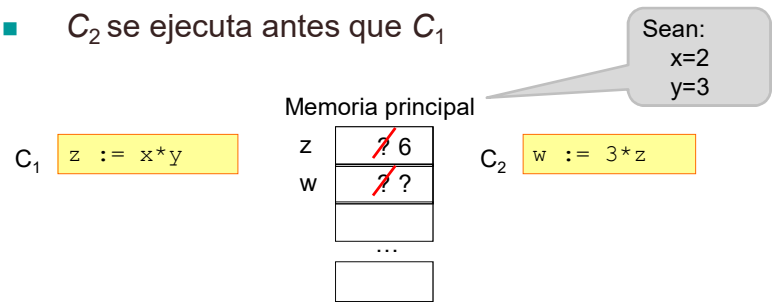


Identificación de Paralelismo

1. *Primera Condición: no* **RAW**

¿Qué sucedería si NO se cumpliera esta condición?
Veamos las diferentes situaciones…

- C_2 se ejecuta antes que C_1



Identificación de Paralelismo

1. *Primera Condición: no* **RAW**

¿Qué sucedería si NO se cumpliera esta condición?

- Vemos que el incumplimiento de la primera condición introduce **dependencias en el flujo de ejecución**
  - Según el orden en el que se ejecutan las tareas se obtiene un resultado diferente.
  - Esta es una **situación no deseada**

Identificación de Paralelismo

2. ***Segunda Condición*** *(que debe cumplirse)*
Entrada(C1) ∩ Salida(C2) = Ø

"C_1 lee datos que posteriormente C_2 NO escribe "

¿Qué sucedería si NO se cumpliera esta condición?

C_1 lee datos que posteriormente C_2 escribe

- Esta situación es comúnmente conocida como **Write-After-Read** o **WAR**
  - → Se introducirían las denominadas **antidependencias**

Identificación de Paralelismo

3. ***Tercera Condición*** *(que debe cumplirse)*
   Salida(C1) ∩ Salida(C2) = Ø

"C_1 escribe datos que posteriormente C_2 NO sobrescribe"

¿Qué sucedería si NO se cumpliera esta condición?

C_1 escribe datos que posteriormente C_2 sobrescribe

- Esta situación es comúnmente conocida como **Write-After-Write** o **WAW**
  - → Se introducirían dependencias de **salida**

Identificación de Paralelismo

- En resumen…
  - Condiciones de Bernstein

C_1 y C_2 pueden ejecutarse en paralelo
sin sincronización (es decir que son independientes)

si y sólo si

1. *No* **RAW** -
2. *No* **WAR** -
3. *No* **WAW** -

¡Atención!
Las condiciones de Bernstein NO limitan el paralelismo, pero SÍ limitan el asincronismo…

Identificación de Paralelismo

- Las dependencias existentes entre las diferentes sentencias y/o tareas, frecuentemente requieren la utilización de diferentes métodos de sincronización
  - Más adelante los repasaremos
  - Trabajaremos sobre ellos a lo largo de la Práctica Nº 2.

Identificación de Paralelismo

- **Sincronismo**
  - es indispensable cuando existen dependencias entre las tareas: todos los procesos deben frenar su ejecución en un cierto punto para asegurar congruencia de datos y continuar la ejecución
- **Asincronismo**
  - los procesos pueden ejecutarse libremente siempre y cuando no presenten dependencias
- **Sincronismo vs Asincronismo**
  - Debe elegirse de acuerdo a la naturaleza del problema

Identificación de Paralelismo

- **Para pensar…**
  - ¿Cómo se solucionarían los problemas antes vistos por medio de la utilización de sincronismo?

Estrategias de Descomposición

- Cuestión crucial:
  - Elegir la estrategia para dividir el trabajo en piezas que puedan ejecutarse en paralelo.

¿De qué manera las tareas son paralelas?
¿Qué constituye una tarea?

  - Criterio simple:
    - ¿El conjunto de datos a tratar es muy grande?
    - ¿El tratamiento de los datos tiene claras subtareas?

Estrategias de Descomposición

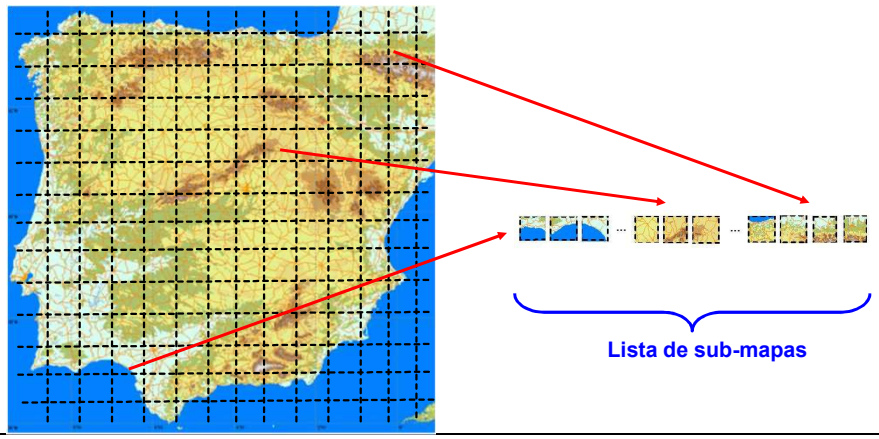
- **Descomposición de Dominio**
  - Se utiliza cuando es posible resolver un problema aplicando la misma operación sobre partes diferentes de su dominio de datos
    - En otras palabras, se utiliza en la siguiente situación: debe tratarse un gran volumen de datos y realizar sobre ellos el mismo procesamiento

Estrategias de Descomposición

- Descomposición de Dominio
  - El dominio de datos del problema se divide en múltiples regiones o particiones
  - Cada región se asigna a un nodo diferente
  - Cada nodo realiza tareas similares sobre el subconjunto de datos
  - Se usa más comúnmente en problemas científicos
    - Utiliza varios nodos concurrentemente
    - Exhibe condiciones naturales de escalabilidad

Estrategias de Descomposición

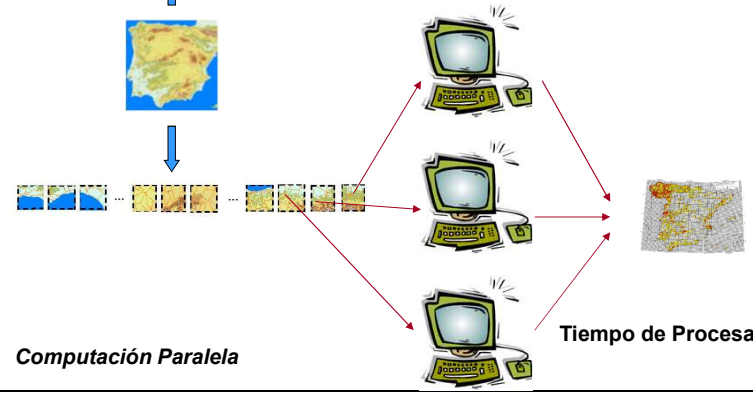
- **Descomposición de Dominio**
  - Ejemplo: Mapas de Riesgo de Incendio



Estrategias de Descomposición

Computación Secuencial

Tiempo de Procesamiento= N * t



Computación Paralela

Tiempo de Procesamiento= $\frac{N * t}{3}$

Estrategias de Descomposición

- **Descomposición Funcional**
  - La descomposición se ajusta a la propia arquitectura de la aplicación a paralelizar
    - Pueden identificarse las partes funcionales del cálculo del programa que constituyen las principales fases de ejecución
    - Se identifican sus interdependencias:
      - Superposición de datos mínima y volumen de flujo de datos pequeño resulta en un grafo de tareas con cierta topología
      - Excesivo solapamiento de datos resulta en demasiadas comunicaciones. En este caso, posiblemente deba utilizarse otra técnica de descomposición

Estrategias de Descomposición

- **Descomposición Funcional**
  - Luego de definir las fases funcionales e identificar sus interdependencias…
  - Se planifica la ejecución paralela de las tareas que son independientes
    - Cada tarea se asigna a un nodo
  - Suele estar limitado a bajos grados de paralelismo
    - Conjunto limitado de subtareas
    - No presenta propiedades naturales de escalabilidad

Estrategias de Descomposición

- **Descomposición Funcional**
  - Ejemplo: Armado de automóviles
    - Hace varias décadas, una sola persona podía ser responsable de armar un vehículo completo
      - De esta manera, se requería un grado bastante alto de conocimientos de cada persona empleada para armar autos
    - Con los años, la especialización del trabajo, hizo del armado de coches una tarea sucesiva y cooperativa entre varias personas (también pueden ser máquinas o robots)
      - Cada persona es responsable de una tarea específica dentro del montado del auto
        - Colocación de ruedas
        - Pintura
        - Motor
        - Interior
        - …

Estrategias de Descomposición

- Otras estrategias:
  - *Descomposición recursiva*
  - *Descomposición exploratoria*
  - *Descomposición especulativa*

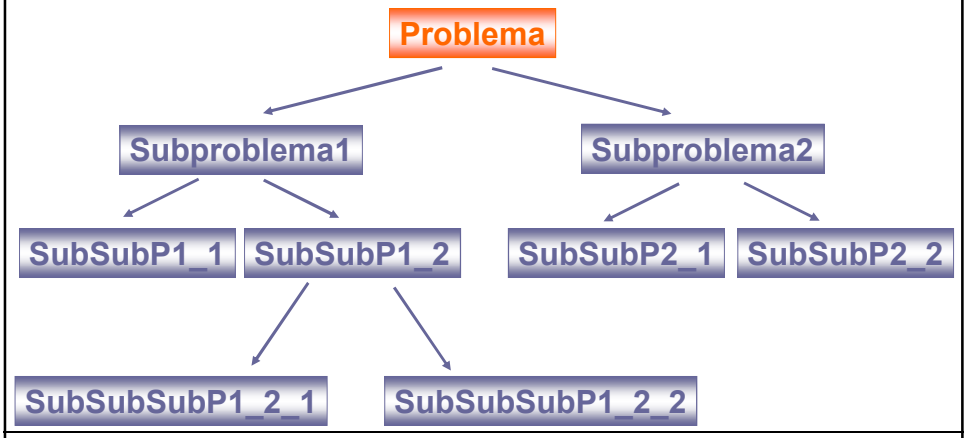
- Las comentaremos a continuación…

Estrategias de Descomposición

- **Descomposición recursiva**
  - Se utiliza la estrategia ***divide y vencerás*** *(divide & conquer)*
    - El problema se divide en un conjunto de subproblemas independientes
    - Cada subproblema se resuelve recursivamente subdividiéndolo sucesivamente
    - Finalmente, se combinan los resultados

Estrategias de Descomposición

- Descomposición Recursiva (D&C)

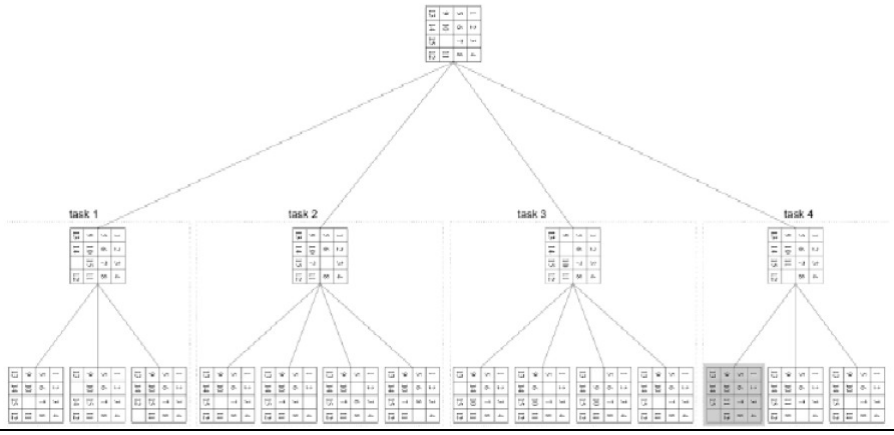


Estrategias de Descomposición

- **Descomposición exploratoria**
  - Se utiliza en aquellos problemas en los que el objetivo perseguido es la búsqueda de una solución entre las muchas posibilidades disponibles
  - Se divide el espacio de búsqueda en zonas más pequeñas que se asignan a diferentes nodos
  - Cada nodo busca la solución en su subespacio, deteniéndose cuando alguno de ellos encuentra la solución
  - En esta estrategia, no todos los nodos involucrados contribuyen a la solución del problema

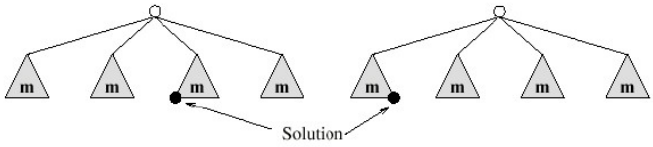
Estrategias de Descomposición

- Descomposición Exploratoria



Estrategias de Descomposición

- Dependiendo de dónde se encuentre la solución dentro del espacio, la formulación paralela puede requerir más trabajo que la formulación serial

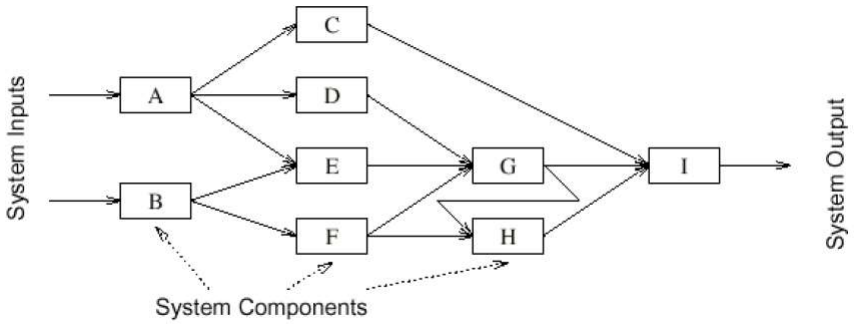


Estrategias de Descomposición

- **Descomposición especulativa**
  - Se utiliza en aquellos casos en los que el flujo de ejecución depende del resultado de alguna tarea
  - Se configuran todos los flujos posibles de ejecución (asumiendo los diferentes resultados de la tarea actual), y se asigna cada uno de ellos a un nodo para ejecutar los pasos posteriores
  - En esta estrategia, se desconocen los datos de entrada, mientras que en la anterior se desconocen los datos de salida

Estrategias de Descomposición

- Descomposición especulativa

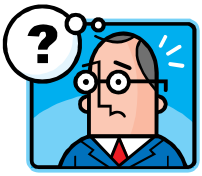


Estrategias de Descomposición

- Descomposición especulativa
  - Si la predicción es errónea…
    - El trabajo realizado se desperdicia
    - Puede ser necesario deshacer los resultados de la tarea y restaurar el contexto

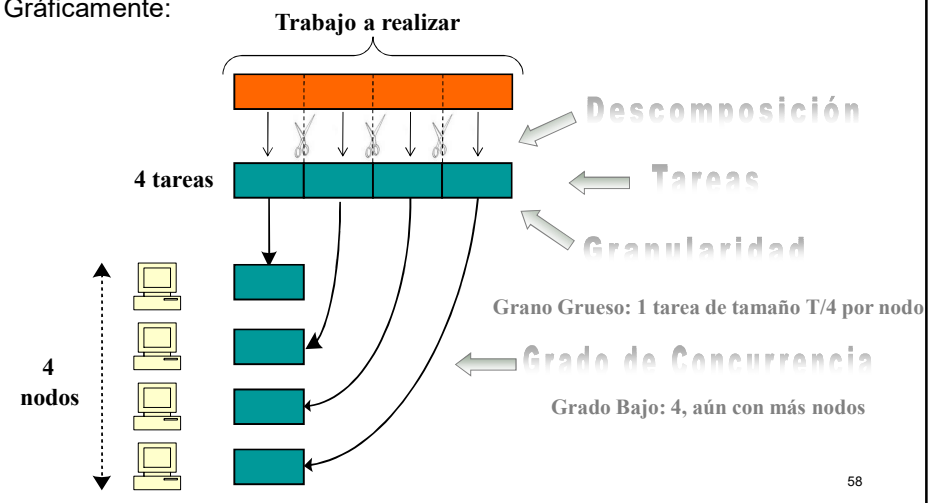
Estrategias de Descomposición

- Para pensar…
  - ¿En qué tipo de problemas son aplicables las condiciones de Bernstein?



Resumen

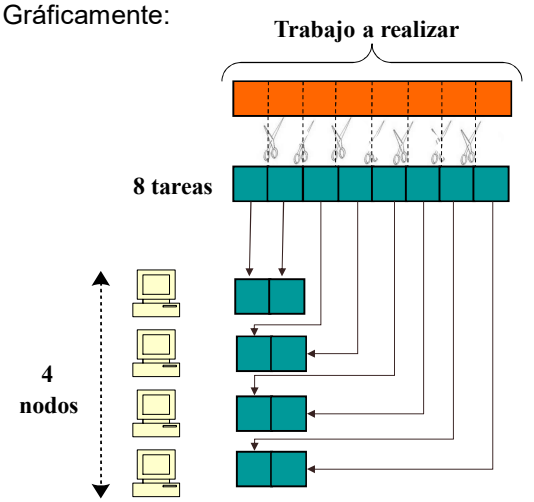
Gráficamente:



58

Resumen

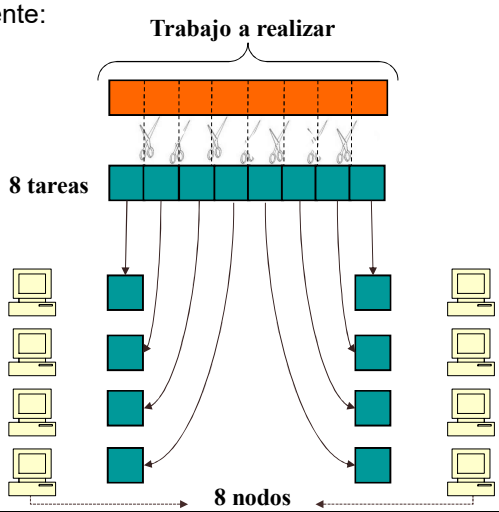
Gráficamente:



T(4)=T(1)/4

Resumen

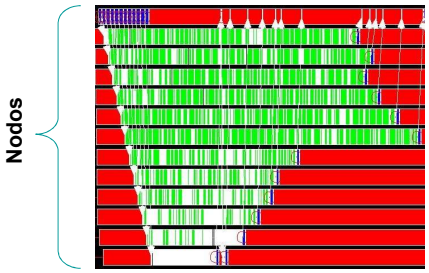
Gráficamente:



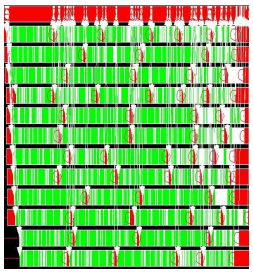
T(8)=T(1)/8

Resumen

- En cuanto al balanceo de carga, recordemos los siguientes gráficos:



Ejecución desbalanceada



Ejecución balanceada

Aquí vemos que si cada nodo procesa una cantidad de tareas proporcional a su velocidad o disponibilidad, el sistema global se aprovecha mejor, ya que no hay nodos ociosos ni sobrecargados.

Resumen

- Cómo identificar regiones paralelas
  - Condiciones de Bernstein
    - Sincronismo vs Asincronismo
- Estrategias de Descomposición
  - Dominio y Funcional
  - Recursiva, exploratoria y especulativa
- En la siguiente unidad estudiaremos los conceptos necesarios para los pasos 3 y 4 en la especificación de programas paralelos:
- Modelos de Algoritmos Paralelos
  - SPMD y Master/Worker
  - Grafo de Tareas y Pipeline
- Modelos de Comunicación
  - Memoria Compartida
  - Paso de Mensajes

Tarea para la casa…

- **Repasar los conceptos presentados**
- **Ampliar la información presentada en clases**
- **Comenzar con la resolución de la Práctica Nº 2**

Bibliografía

- **Carretero Pérez J., De Miguel Anasagasti P., García Caballeira F., Pérez Costoya F.,** *"Sistemas Operativos-una visión aplicada". McGraw-Hill, 2001.*
- **Foster Ian** "*Designing and Building Parallel Programs*", Addison Wesley. 1995 (también disponible on-line en: http://www-unix.mcs.anl.gov/dbpp/)
- **Grama A., Gupta A., Karypis G., Kumar V.,** "*Introduction to Parallel Computing*". Pearson Addison Wesley. Second Edition. 2003.
- **Morrison R. S. –** "*Cluster Computing –Architectures, Operating Systems, Parallel Processing and Programming Languages*" – GNU General Public License 2002. http://static.schoolrack.com/20736/2_Cluster_Computing_-_Architectures,_Operating_Systems,_Parallel_Processing_&_Programming_Languages_(v2.4).pdf
- **Silva L., Buyya R.,** ***Parallel Programming Models and Paradigms, High Performance Cluster Computing, Programming and Applications,*** **Rajkumar Buyya (editor), Prentice Hall PTR, NJ, USA, 1999.** http://www.buyya.com/cluster/v2chap1.pdf
- **Silberschatz A., Galvin P.**, *"Operating Systems Concepts".* Addison-Wesley, 1998.
- **Wilkinson B., Allen M.,** "*Parallel Programming - Techniques and Applications Using Networked Workstations and Parallel Computers*". Pearson Prentice Hall. Second Edition. 2005.