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 3

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

- *Recordemos algunas generalidades estudiadas en la Unidad temática anterior…*

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

- **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

En la Unidad anterior estudiamos los primeros dos pasos. En esta Unidad nos centraremos en los pasos 3° y 4°…

Modelos de Algoritmos Paralelos

- Hay muchos algoritmos en la literatura que resuelven un amplio abanico de problemas
- La mayoría de algoritmos han sido diseñados con un sistema en mente
- Sin embargo, hay algunos paradigmas básicos de programación a los que responden muchos algoritmos

Modelos de Algoritmos Paralelos

- Cuestión crucial:
  - Estructurar un algoritmo para descomponer y asignar tareas

¿Cómo minimizar las interacciones y balancear la carga?

  - Criterio simple:
    - ¿El problema puede descomponerse según el dominio o la funcionalidad?
    - ¿Cuál es la arquitectura subyacente?

Modelos de Algoritmos Paralelos

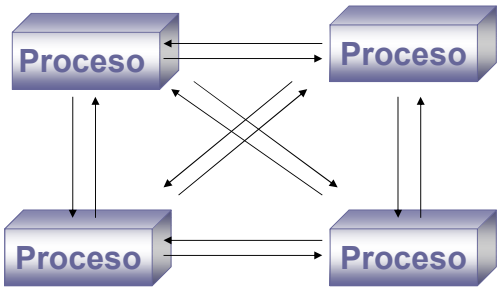
- Cuando se utiliza descomposición de dominio...
  - Modelo de Paralelismo de Datos
  - Modelo Master/Worker
- Cuando se utiliza descomposición funcional...
  - Modelo de Grafo de Tareas
  - Modelo Pipeline

Modelos de Algoritmos Paralelos

- Modelo de **Paralelismo de Datos**
  - También denominado SPMD (Simple Program Multiple Data)
  - Este tipo de paralelismo aplica concurrentemente operaciones idénticas en diferentes items de datos
    - El trabajo puede realizarse en fases
    - En cada fase pueden tratarse datos diferentes
    - Requiere sincronía o comunicaciones para proveer nuevos datos o sincronizar las tareas
    - Presenta condiciones naturales de escalabilidad
  - Fácilmente se consigue balancear la carga
    - Partición uniforme de tareas
    - Asignación de tareas estática o semi-estática
    - Cada nodo ejecuta las mismas operaciones sobre datos diferentes

Modelos de Algoritmos Paralelos

- Modelo de Paralelismo de Datos

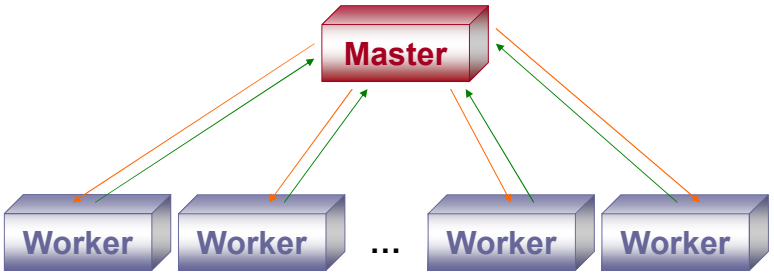


Modelos de Algoritmos Paralelos

- **Modelo Master/Worker**
  - Un caso particular de paralelismo de datos, en que existe un proceso Master que coordina el cómputo que realiza el resto de procesos Workers
  - **Proceso Master**
    - Genera tareas y las asigna a los workers
  - **Procesos Worker**
    - Procesan tareas iterativamente

Modelos de Algoritmos Paralelos

- Modelo Master/Worker



Modelos de Algoritmos Paralelos

- Modelo Master/Worker
  - El **balanceo de carga** depende de la homogeneidad y la granularidad
    - Cada worker ejecuta las mismas operaciones sobre datos diferentes
    - Asignación de tareas dinámica
    - Costo de procesamiento >> Costo de Comunicación
    - Aspecto clave: el master
      - Cuello de botella cuando
        - Tareas muy pequeñas
        - Workers muy veloces
  - **Sincronismo – Asincronismo**
    - En algunos casos, el trabajo debe realizarse en varias fases; el trabajo de una etapa debe concluirse antes de que el trabajo de la siguiente fase sea generado. En este caso el master fuerza la sincronización de todos los workers al final de cada fase
    - Interacciones asíncronas pueden ayudar a solapar la interacción y los cálculos asociados con la generación de trabajo del master

Modelos de Algoritmos Paralelos

- Dependencia entre iteraciones
  - *Dependencia entre iteraciones*: El master necesita los resultados de todos los workers para poder generar un nuevo conjunto de datos
  - *Entradas de datos independientes*: Los datos van llegando al master, y éste no requiere los resultados de los datos anteriores para poder generar nuevos conjuntos de datos

Modelos de Algoritmos Paralelos

- Método de distribución de datos
  - *Distribuir todos los datos disponibles*: Cuando el master tiene listo un conjunto de datos para mandárselo a los workers, los distribuye todos siguiendo alguna política determinada
  - *Distribuir bajo petición*: El master distribuye un subconjunto de datos a los workers y a medida que van terminando, le van pidiendo más datos

Modelos de Algoritmos Paralelos

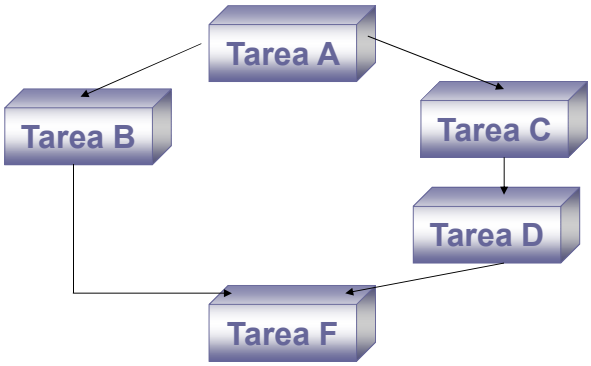
- Modelo de **Grafo de Tareas**
  - Grafo de dependencias de tareas
    - Abstracción que se utiliza para representar las dependencias entre tareas y su orden relativo de ejecución
    - Grafo dirigido acíclico
    - Los nodos representan tareas
    - Las aristas indican dependencias entre las tareas
    - Un grafo de dependencia de tareas puede ser inconexo y el conjunto de aristas podría ser vacío.

Modelos de Algoritmos Paralelos

- Modelo de **Grafo de Tareas**
  - Algunas tareas pueden necesitar datos producidos por otras tareas
    - Deben esperar la terminación de dichas tareas para terminar o continuar su ejecución
    - La tarea correspondiente a un nodo puede ejecutarse cuando todas las tareas conectadas a este nodo por aristas entrantes han sido completadas.
    - En el modelo de grafo de tareas las interrelaciones entre tareas se utiliza para promover localidad o reducir los costos de interacción
    - Este modelo se utiliza para resolver problemas en los cuales la cantidad de datos asociados a las tareas es grande en relación a la cantidad de cálculo asociado con él.
    - Usualmente las tareas son mapeadas estáticamente para ayudar a optimizar el costo del movimiento de datos entre tareas

Modelos de Algoritmos Paralelos

- Modelo de **Grafo de Tareas**

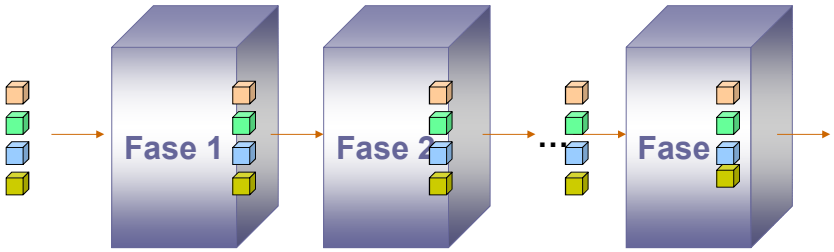


Modelos de Algoritmos Paralelos

- Modelo **Pipeline**
  - El flujo de datos pasa a través de una sucesión de procesos, cada uno de los cuales ejecuta alguna tarea sobre él
  - La ejecución simultánea de diferentes programas sobre un flujo de datos se denomina paralelismo de flujo.
  - Un pipeline es una cadena de productores y consumidores
    - Cada proceso en el pipeline puede considerarse un consumidor de una secuencia de items de datos de los procesos que lo preceden en el pipeline, y también puede considerarse como productor de datos para los procesos posteriores a él en el pipeline.
    - El pipeline no necesariamente debe ser lineal, puede ser un grafo dirigido. El modelo pipeline normalmente involucra una asignación estática de tareas a procesos

Modelos de Algoritmos Paralelos

- Modelo Pipeline



Modelos de Algoritmos Paralelos

- Modelo Pipeline
  - El balanceo de carga depende de la granularidad
    - Grano grueso → toma mucho tiempo llenar el pipeline
    - Grano fino → incrementa la interacción para obtener nuevos datos
      - La técnica de reducción de interacción más común aplicable a este modelo es el solapamiento de interacción y computación

Modelos de Comunicación

- Cuestión crucial:
  - Elegir el lenguaje de programación y librerías para implementar la aplicación

¿Qué características debe tener el programa?
¿Cuál es la arquitectura subyacente?

  - Criterio simple:
    - ¿Cómo deben intercomunicarse las tareas?

Modelos de Comunicación

- **Memoria Compartida**
  - Todos los datos accedidos por la aplicación ocupan la memoria global accesible desde todos los nodos
  - Esto significa que cada procesador puede manejar independientemente datos de cualquier región de la memoria
  - El modelo de memoria compartida se caracteriza por la necesidad de sincronización para preservar la integridad de las estructuras de datos compartidas

Modelos de Comunicación

- **Paso de Mensajes**
  - Los datos se asocian a un nodo particular.
  - De este modo para acceder a datos remotos es necesario establecer comunicaciones.
  - En general para obtener datos remotos, el nodo propietario de los datos debe enviarlos y el nodo cliente debe recibirlos.
  - En este modelo, las primitivas de envío y recepción encarnan el papel de la sincronización

Modelos de Comunicación

- Aunque estos Modelos de Comunicación están inspirados en las correspondientes arquitecturas paralelas, su uso no es restrictivo. Es posible combinarlos, aunque pueden conllevar pérdidas en el rendimiento.

Resumen

- Para obtener una aplicación eficiente, es fundamental una adecuada elección de:
  - estrategia de descomposición
  - modelo de algoritmo paralelo
  - modelo de comunicación
- ¿Cómo combinarlas?
  - Recordemos…

Resumen

- Las estrategias de descomposición de *dominio* y *explorativa*, presentan gran afinidad con:
  - Algoritmos **SPMD** y **Master/Worker**
- Las estrategias de descomposición *funcional, recursiva y especulativa* poseen mayor afinidad con:
  - Algoritmos como **Grafo de Tareas** y **Pipeline**

Resumen

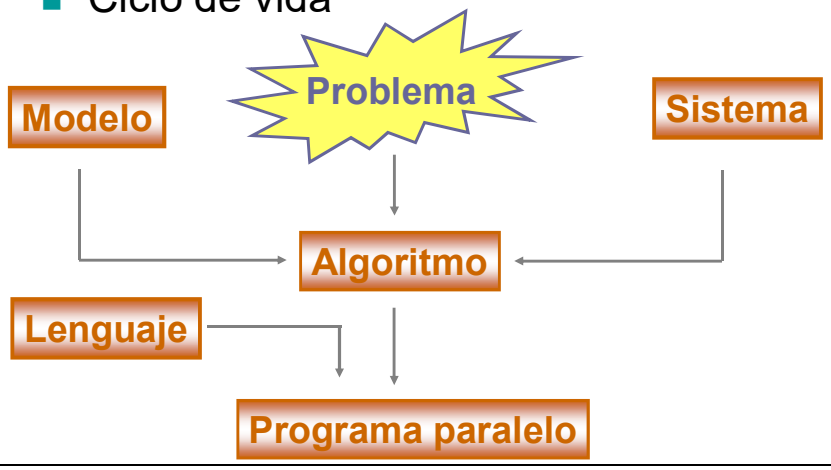
- Los diferentes modelos de comunicación pueden ser aplicados tanto si existe paralelismo de datos o de tareas
- Sin embargo, el modelo de comunicación a utilizar suele estar determinado por la arquitectura o sistema subyacente

Resumen

- Aplicación, sistema y modelo de programación están fuertemente relacionados.
- No se puede concebir una aplicación paralela sin tener una idea bastante aproximada del sistema y modelo que se van a utilizar.

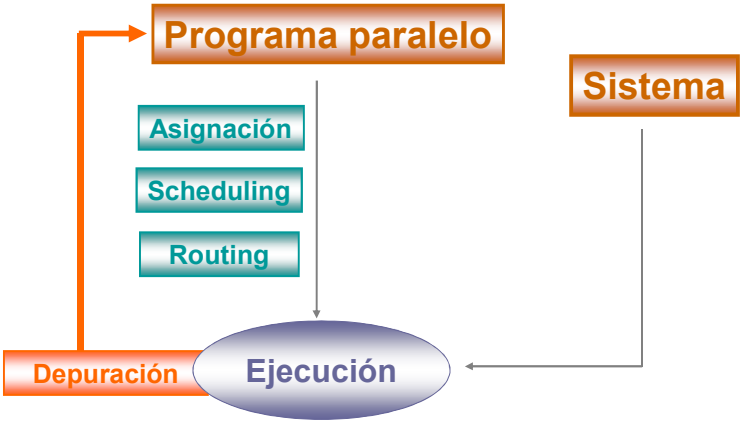
Resumen

- Ciclo de vida



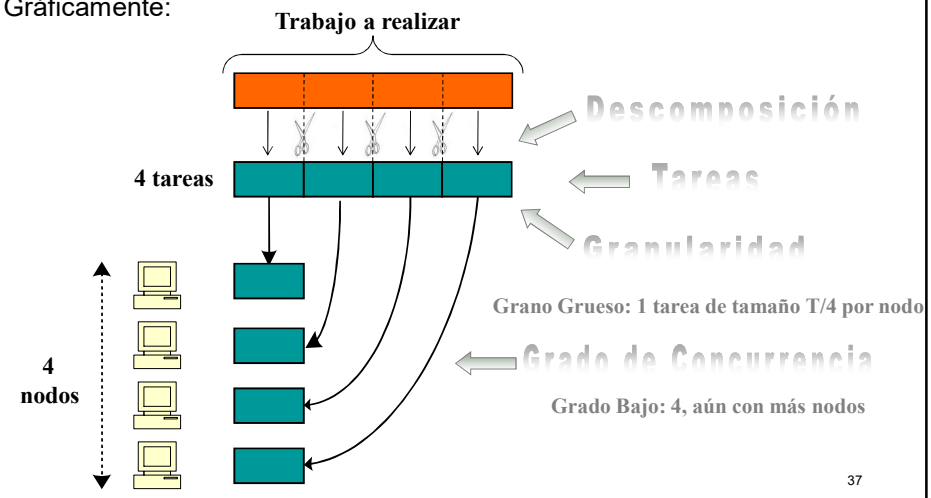
Resumen

- Ciclo de vida



Resumen

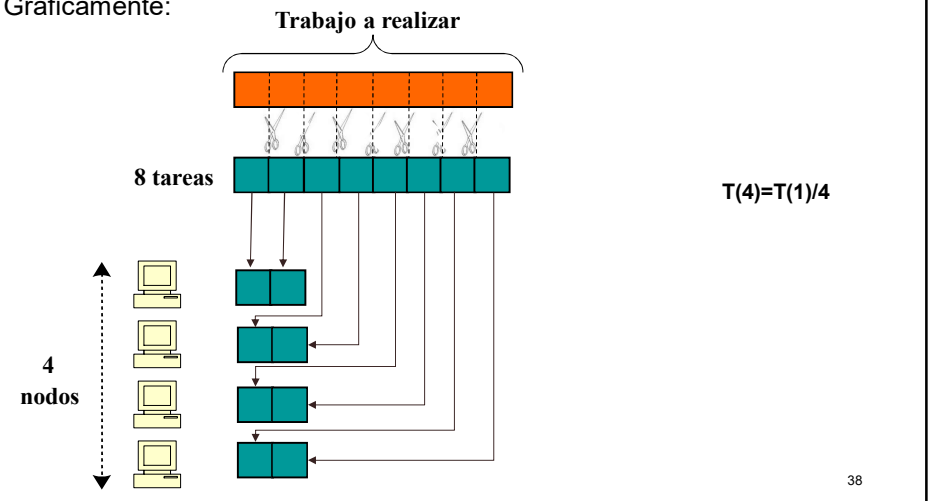
Gráficamente:



37

Resumen

Gráficamente:

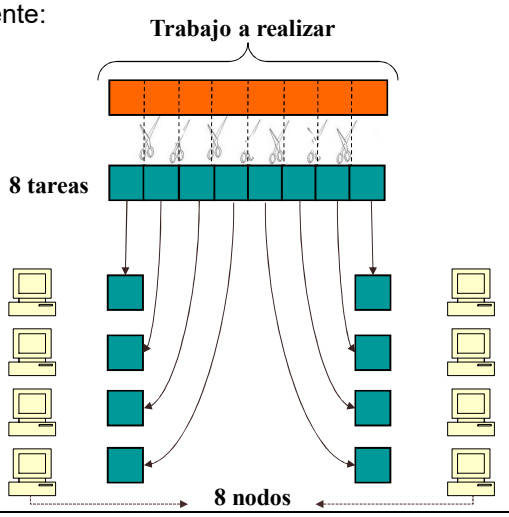


T(4)=T(1)/4

38

Resumen

Gráficamente:

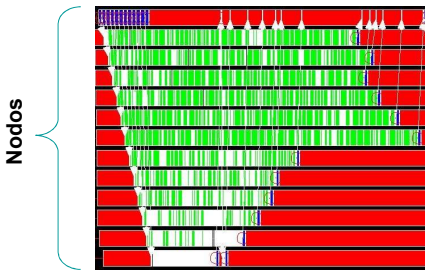


T(8)=T(1)/8

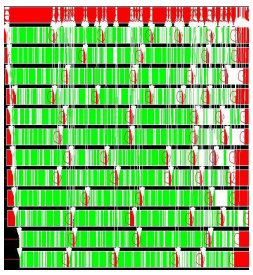
39

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
- 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**
- **Continuar con la resolución de la Práctica**

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.