ARQUITECTURA DE COMPUTADORES II

AUTORES:
David Expósito Singh
Florin Isaila
Daniel Higuero Alonso-Mardones
Javier García Blas
Borja Bergua Guerra

Área de Arquitectura y Tecnología de Computadores
Departamento de Informática
Universidad Carlos III de Madrid

Julio de 2012

TEMA 2:
PROGRAMACIÓN PARALELA (Y III)

1. Métricas de Rendimiento
2. Aspectos a considerar en la fase de particionamiento
   - Balanceo de Carga
   - Sincronización Mínima
   - Comunicación Mínima
   - Trabajo Extra Mínimo
3. Acceso a Datos y Comunicación en Multiprocesadores
4. Aspectos a considerar cuando se fijan patrones de comunicación
   - Aprovechar la localidad espacial y temporal
   - Reducir costos de comunicación
5. Modelos de rendimiento

1. **Métricas de Rendimiento**
2. Aspectos a considerar en la fase de particionamiento
   - Balanceo de Carga
   - Sincronización Mínima
   - Comunicación Mínima
   - Trabajo Extra Mínimo
3. Acceso a Datos y Comunicación en Multiprocesadores
4. Aspectos a considerar cuando se fijan patrones de comunicación
   - Aprovechar la localidad espacial y temporal
   - Reducir costos de comunicación
5. Modelos de rendimiento

Métricas de Rendimiento para Sistemas Paralelos

- Tiempo de Ejecución:
  - T_s en un programa secuencial, tiempo que transcurre entre el comienzo y la finalización de la ejecución del programa.
  - T_p en un programa paralelo, el tiempo que transcurre entre el comienzo y la finalización del último proceso que termina la ejecución.

Métricas de rendimiento para Sistemas Paralelos

- Aceleración: S

Medida que captura el beneficio relativo de resolver un problema en paralelo.

$$S = \frac{T_s}{T_p}$$

Ejemplo

- Suma de *n* números sobre un hipercubo de *n* procesos:
  - El programa más rápido secuencial es de $\mathcal{O}(n)$.
  - En un hipercubo, $T_p = \mathcal{O}(\log n)$.
  - $S = \mathcal{O}\left(\frac{n}{\log n}\right)$

Ejemplo en detalle

- Suma de n números sobre un hipercubo de p procesos:
  - 1 unidad de tiempo sumar dos números.
  - 1 unidad de tiempo comunicación entre procesadores adyacentes.

$$T_s = (n-1)\big|_{\lim n\to\infty} = n$$

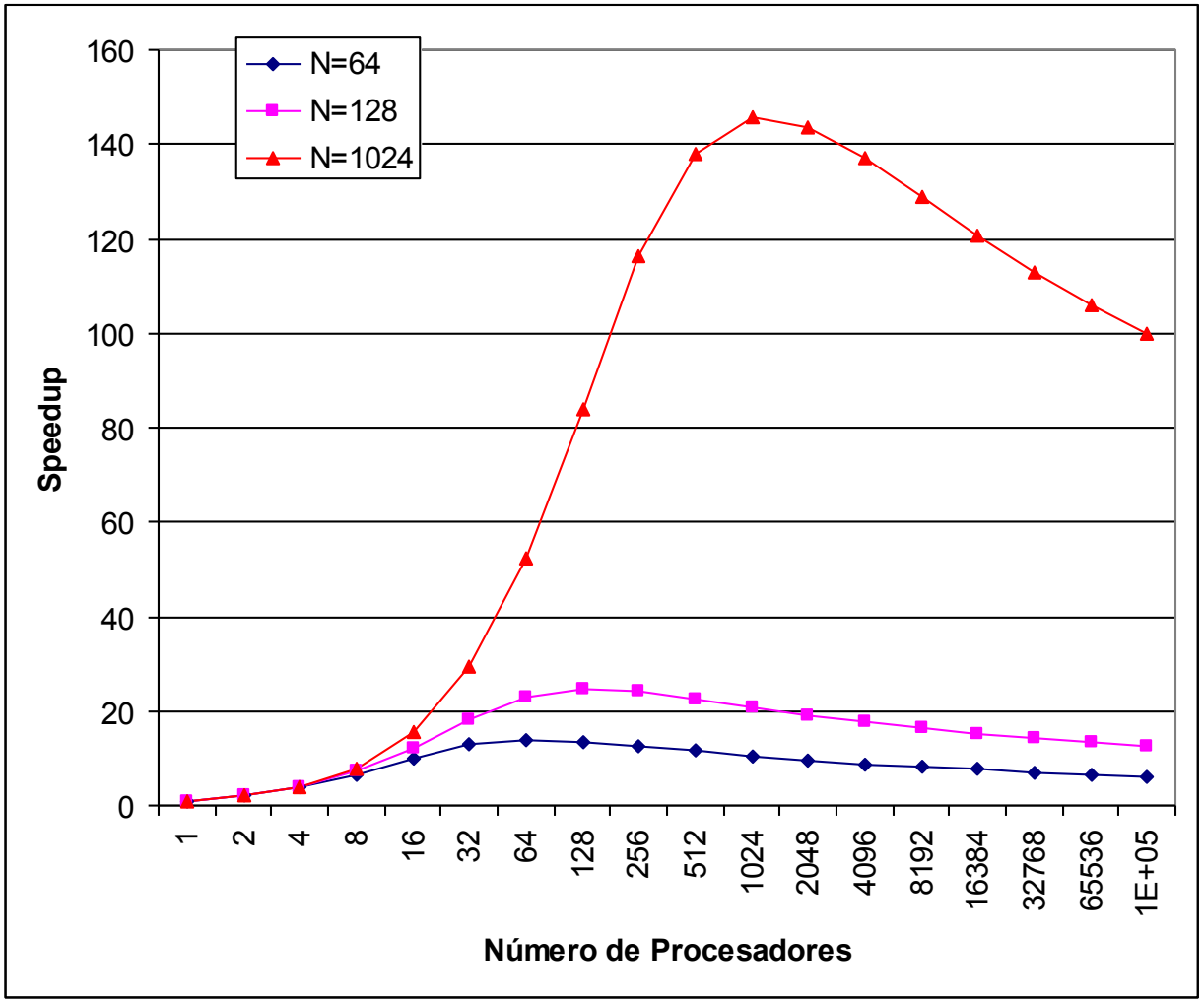
$$T_p = \left(\frac{n-p}{p} + \log p + \log p\right)\Big|_{\lim n\to\infty} = \frac{n}{p} + 2\times\log p$$

Ejemplo en detalle

$$S = \frac{np}{n + 2p\log p}$$

¡La aceleración no aumenta linealmente cuando aumenta el número de procesos!

Ejemplo en detalle



Métricas de Rendimiento para Sistemas Paralelos

- Eficiencia:
  - Parte del tiempo en el que los procesadores están realizando trabajo útil.
  - Solo un sistema ideal de p procesadores puede alcanzar una aceleración de p. Esto se debe a costes asociados a sincronización, comunicación etc.

$$E = \frac{S}{p}$$

  - En el ejemplo:

$$E = \frac{n}{n + 2p\log p}$$

Eficiencia para el ejemplo

n	p = 1	p = 4	p = 8	p = 16	p = 32
64	1.0	**.80**	.57	.33	.17
192	1.0	.92	**.80**	.60	.38
320	1.0	.95	.87	.71	.50
512	1.0	.97	.91	**.80**	.62

Escalabilidad

- Habilidad de mantener la eficiencia aumentando simultáneamente:
  - Tamaño del problema.
  - Número de procesadores.
  - Ejemplo:
    - n=64, p=4, E = 0.8
    - n= 192, p = 8, E = 0.8
    - n = 512, p = 16, E = 0.8
    - Por lo tanto, el *sistema es escalable*.

Objetivo: aceleración

- Tarea del arquitecto:

  Observar cómo los programas hacen uso de la máquina y mejorar el diseño para aumentar el rendimiento.

- Tarea del programador:

  Observar cómo el programa usa la máquina y mejorar la implementación para mejorar el rendimiento.

Índice

1. Métricas de Rendimiento
2. **Aspectos a considerar en la fase de particionamiento**
   - Balanceo de Carga
   - Sincronización Mínima
   - Comunicación Mínima
   - Trabajo Extra Mínimo
3. Acceso a Datos y Comunicación en Multiprocesadores
4. Aspectos a considerar cuando se fijan patrones de comunicación
   - Aprovechar la localidad espacial y temporal
   - Reducir costos de comunicación
5. Modelos de rendimiento

Aspectos a considerar en la fase de particionamiento

Meta: alcanzar máxima aceleración.

$$S \leq \frac{T_s}{\max(T_{trabajo} + T_{sincronizacion} + T_{comunicacion} + T_{trabajoextra})}$$

- Objetivos en conflicto:
  - Balanceo de carga.
  - Sincronización mínima.
  - Comunicación mínima.
  - Trabajo extra mínimo.

Sincronización Mínima

- Para lograr sincronización mínima es necesario:

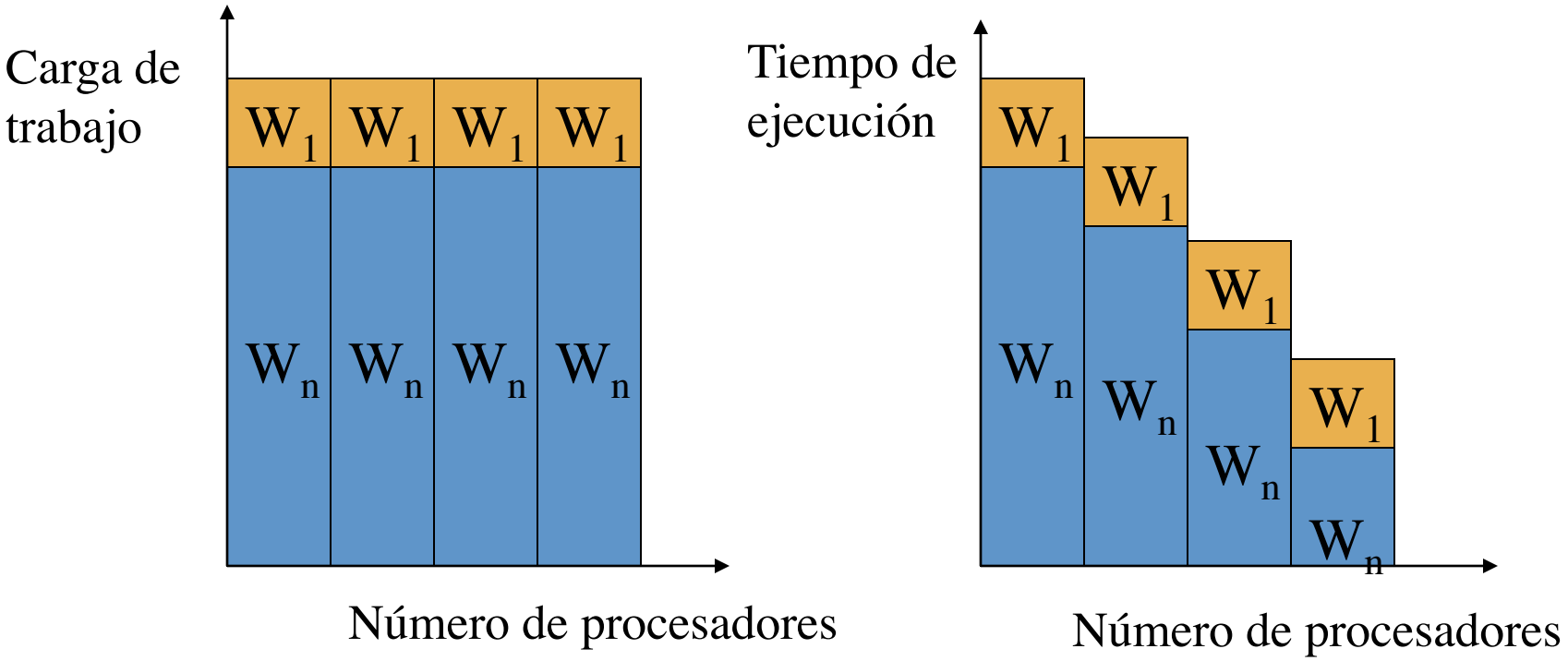
1. Identificar suficiente concurrencia en la descomposición del problema.
2. Decidir cómo manejarla: distribución estática o dinámica.
3. Determinar el grado de granularidad y cómo explotar la concurrencia.
4. Reducir serialización y costes de sincronización.

Sincronización Mínima
Identificar suficiente concurrencia

- Niveles de paralelismo:
  - Paralelismo Funcional:
    - Grandes tareas (procedimientos) pueden realizarse en paralelo.
    - No suele haber muchas tareas, no aumenta con el tamaño del problema.
    - Dificultad para aplicar balanceo de carga.
  - Paralelismo por Datos:
    - Más escalable, proporcional al tamaño del problema.
    - Es factible aplicar balanceo de carga.

Sincronización Mínima
Identificar suficiente concurrencia

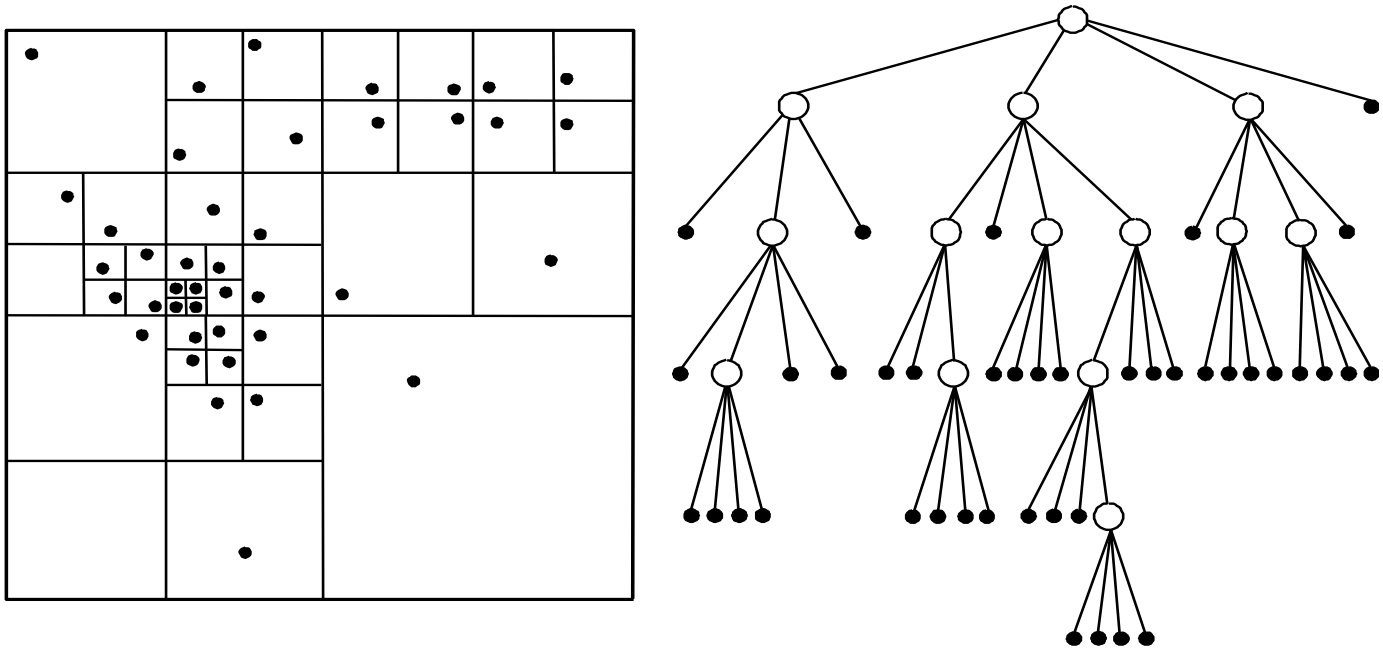
- Ley de Amdahl



Sincronización Mínima
Manejando Concurrencia

- Técnicas estáticas *versus* dinámicas:
  - Técnicas estáticas:
    - Asignación basada en la entrada.
    - Bajo *overhead*.
    - Siempre que sea posible, es preferible.
  - Técnicas dinámicas:
    - Adapta el balanceo en tiempo de ejecución.
    - Aumenta la comunicación y el *overhead*.

Sincronización Mínima
Manejo de Concurrencia
Ejemplo: Barnes-Hut



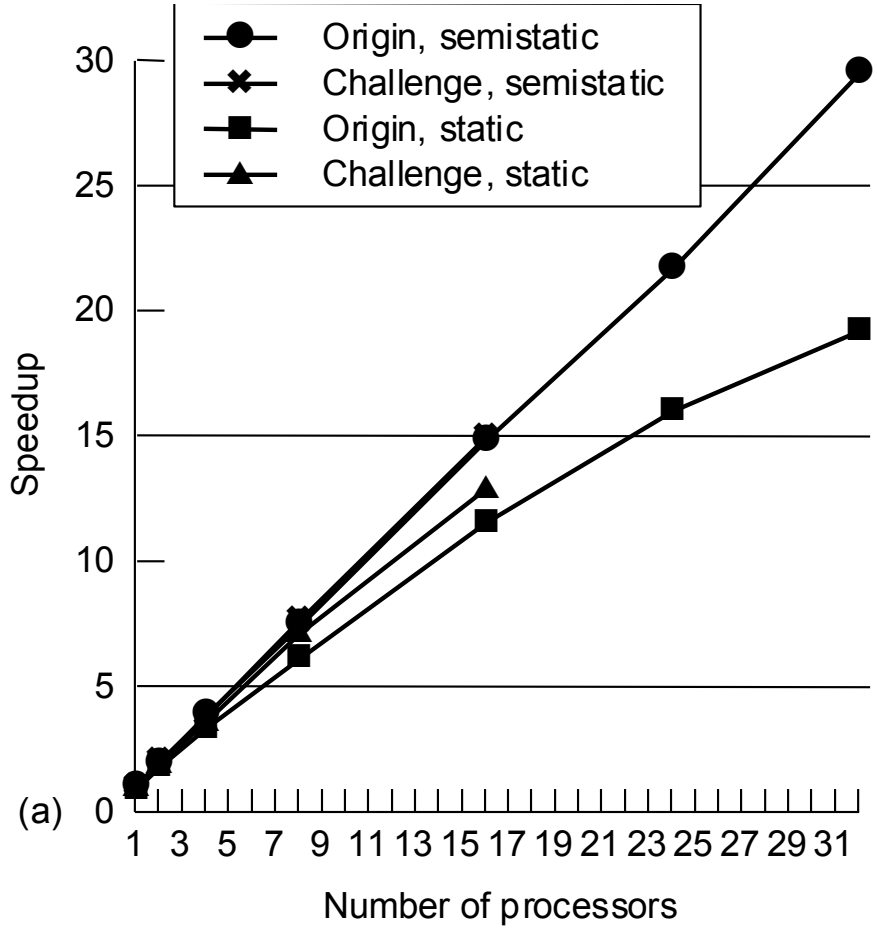
(a) The spatial domain

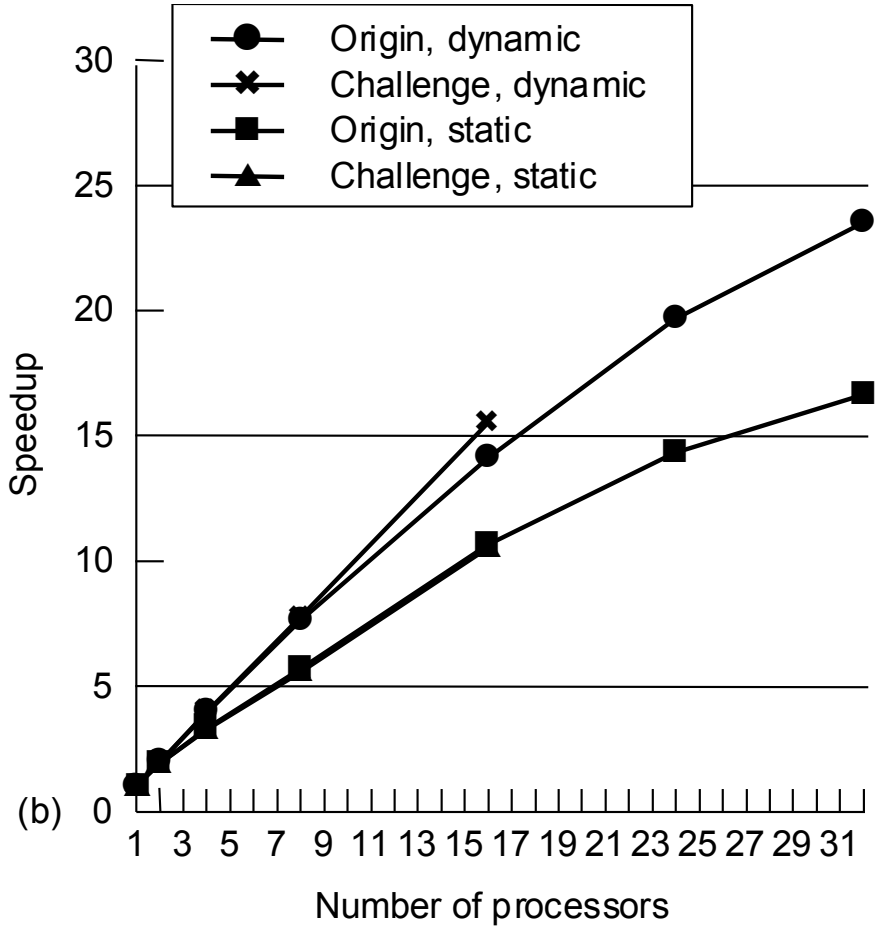
(b) Quadtree representation

Asignación dinámica y no uniforme

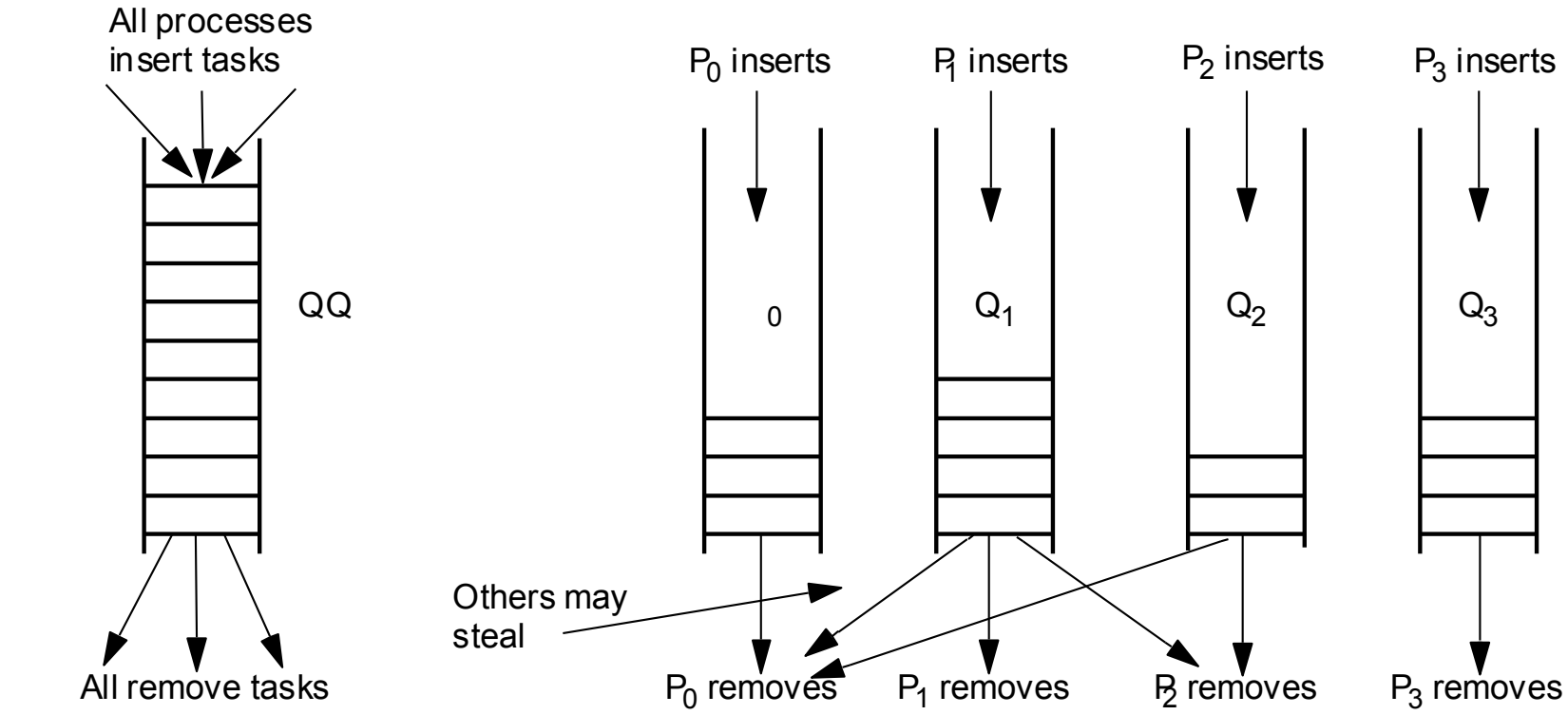
Impacto de Asignación Dinámica

Barnes-Hut + Raytrace on SGI Origin 2000





Manejo de Concurrencia
Técnicas dinámicas



(a) Centralized task queue

(b) Distributed task queues (one per process)

Sincronización Mínima Determinación de la granularidad de las tareas

- **Granularidad:**

  Cantidad de trabajo asociado a una tarea.

- Regla general:
  - *Grano grueso*: pocas oportunidades de balanceo de carga.
  - *Grano fino*: mayor *overhead*, mayor comunicación, más sincronización.

Sincronización Mínima
Reducir la Serialización

- Sincronización de eventos:
  - Global *versus* punto a punto.
  - Sincronización a bajo nivel produce mayor cantidad de sincronizaciones.

- Exclusión mutua:
  - Regiones críticas pequeñas.
  - Dispersar las regiones críticas en el tiempo.

Reducir Serialización
Ejemplos

Los procesos P_1, P_2, … P_N operan sobre una base de datos **B**, leyendo y actualizando sus registros.

Posibilidades de exclusión mutua:

- B en su totalidad.
- Individualmente los registros de B.
- Agrupar registros de B y definir regiones críticas sobre las agrupaciones.
- Cualquiera de las anteriores permitiendo una sola escritura pero varias lecturas.

Comunicación Mínima

$$S \leq \frac{T_s}{\max(T_{trabajo} + T_{sincronizacion} + T_{comunicacion} + T_{trabajoextra})}$$

- La comunicación es costosa.
- Medir: $T_{comumicación} \div T_{cómputo}$ da idea del ancho de banda requerido.
- Ejemplos:
  1. 1GB de comunicación, 1 segundo de cómputo.
  2. 1GB de comunicación, 1 hora de cómputo.

Reducir la Comunicación

Comunicación se debe a:

- Asignación de tareas a procesos.
- Esquema productor-consumidor.

Soluciones:

- Usar algoritmos que necesiten poca comunicación.
- Asignar al mismo proceso las tareas que usen los **mismos datos**.

Trabajo Extra Mínimo

¿De dónde proviene el trabajo extra?

- Cálculos extras para tener un particionamiento que permita buen balanceo de carga. Ej. Barnes-Hut.
- Cómputos redundantes para evitar comunicación.
- Manejo de tareas, datos y procesos realizadas por aplicaciones, lenguajes y sistemas de operación.
- Uso de estructuras especiales para permitir la comunicación.

Índice

1. Métricas de Rendimiento.
2. Aspectos a considerar en la fase de particionamiento.
   - Balanceo de Carga.
   - Sincronización Mínima.
   - Comunicación Mínima.
   - Trabajo Extra Mínimo.
3. **Acceso a Datos y Comunicación en Multiprocesadores.**
4. Aspectos a considerar cuando se fijan patrones de comunicación
   - Aprovechar la localidad espacial y temporal
   - Reducir costos de comunicación
5. Modelos de rendimiento

Acceso a Datos y Comunicación en Multiprocesadores

- ¿Qué es un Multiprocesador?
  - Una colección de procesos comunicantes:

    Objetivos para mejorar el funcionamiento: balancear carga, reducir comunicación, reducir trabajo extra.
  - Un sistema con varias memorias, varias caches:

    El papel de estos componentes es independiente del modelo de programación empleado.

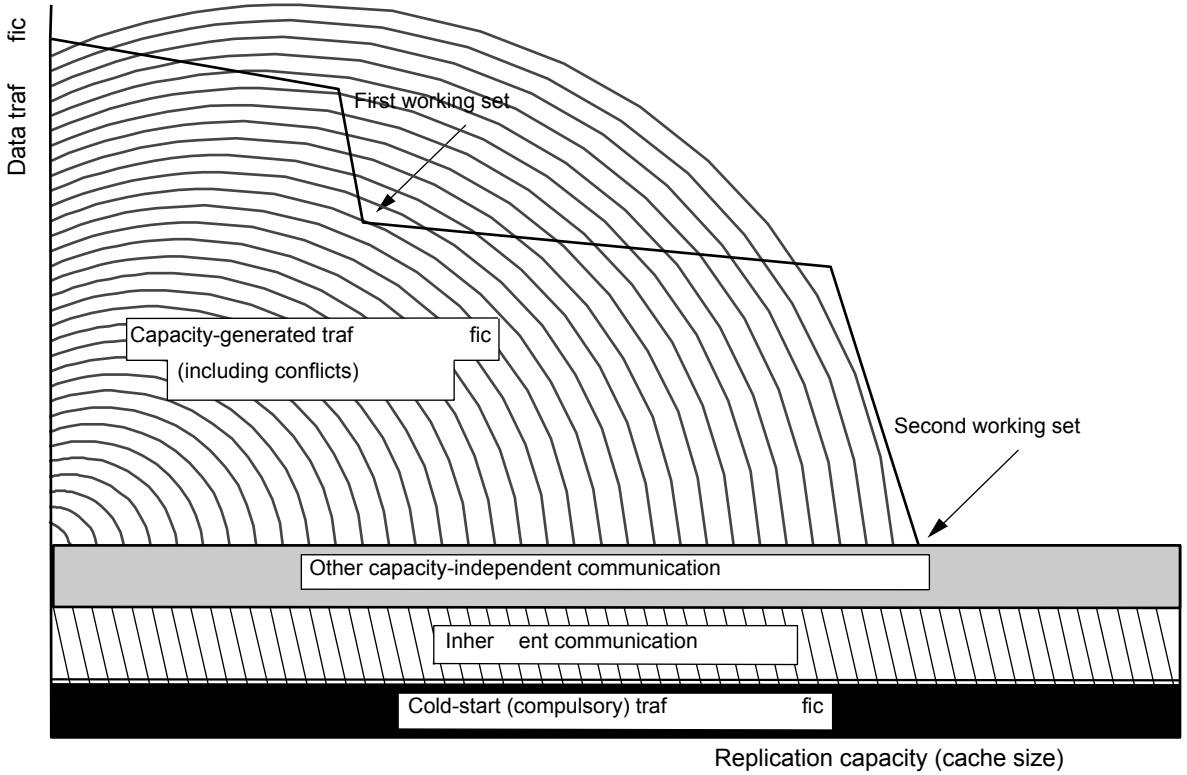
Visión orientada a Memoria

- El multiprocesador puede verse como un procesador con memoria extendida.

- Niveles en la jerarquía extendida:
  - Registros, caches, memoria local, memoria remota (topología).
  - Lo que une a estos componentes es la arquitectura de comunicación.
  - Los niveles se comunican con transferencias de datos de diferentes granularidades.

- Se necesita aprovechar la **localidad temporal y espacial** en la jerarquía.

Jerarquía de Working Sets



Modelo C-C-C + C-

El tráfico originado puede ser local o no local.

Índice

1. Métricas de Rendimiento
2. Aspectos a considerar en la fase de particionamiento
   - Balanceo de Carga
   - Sincronización Mínima
   - Comunicación Mínima
   - Trabajo Extra Mínimo
3. Acceso a Datos y Comunicación en Multiprocesadores
4. **Aspectos a considerar cuando se fijan patrones de comunicación**
   - Aprovechar la localidad espacial y temporal
   - Reducir costos de comunicación
5. Modelos de rendimiento

Aspectos a considerar cuando se fijan patrones de comunicación

- Aprovechar localidad espacial y temporal.
- Costes de comunicación.
- Estrategias para minimizar los costes de comunicación:
  - Reducir *overhead*.
  - Reducir retardo en la red.
  - Evitar la contención.
  - Solapar comunicación con cómputo.

Explotando la Localidad Temporal

Estructurar el algoritmo de manera que el *working set* se ajuste a las jerarquías de memoria.

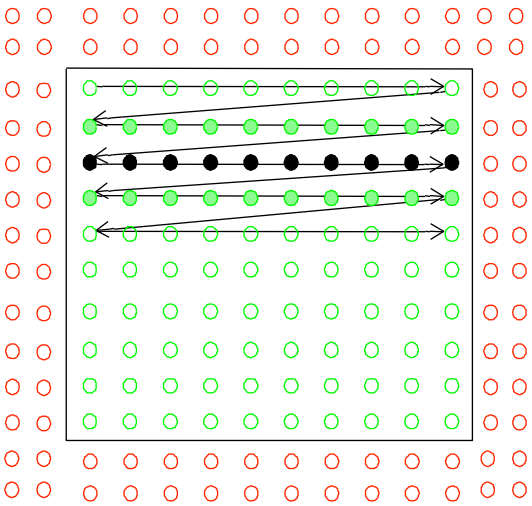
- A menudo las técnicas para reducir la comunicación funcionan bien en este contexto.
- Reutilizar los datos una vez que han sido asignados.

Explotando la Localidad Temporal

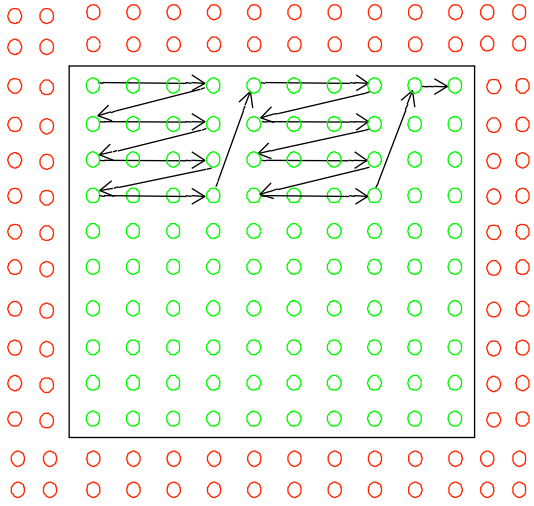
- Resolución de Ecuaciones

  Accesos en una fila: `A[i,j-1], A[i,j], A[i,j+1],…`

  Posteriormente: `A[i-1,j], A[i,j], A[i+1,j]`

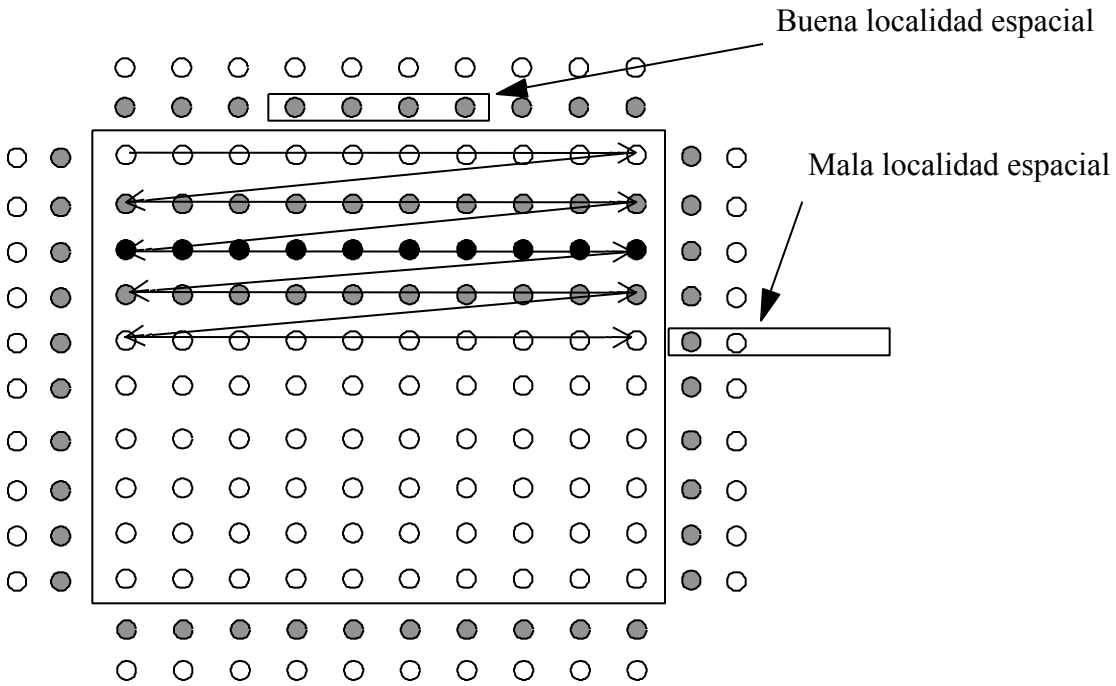


(a) Unblocked access pattern in a sweep



(b) Blocked access pattern with B = 4

Localidad Espacial



Coste de la comunicación

$$C = f * (o + l + \frac{n_c/m}{B} + t_c - overlap)$$

f = frecuencia de los mensajes.
o = *overhead* por mensaje.
l = retardo de la red por mensaje.
n_c = cantidad de datos comunicados.
m = número de mensajes.
B = ancho de banda por mensaje.
t_c = coste por contención de mensaje.
overlap = cantidad de latencia escondida gracias al solapamiento de cómputo y comunicación.

¿Qué hacer para mejorar el rendimiento?

- Reducir volumen de datos comunicados (n_c).
- Reducir *overhead* en la comunicación (f x o).
- Reducir demoras (f x l).
- Reducir contención (f x t_c).
- Solapar cómputo y comunicación.

Reducir el *Overhead*

Problema: *o* viene determinado por la arquitectura.

- El programa debería tratar de reducir *f* enviando la información en mensajes más grandes.
- Lo anterior está bajo el control del programador cuando la comunicación es explícita.
- Es más sencillo cuando la comunicación es regular, y/o es de grano grueso.

Reducir Retardo en la Red

- Retardo en la red = f x h x t_h
  - h = número de *hops* que se atraviesan en la red.
  - t_h = latencia de enlace + *switch* en la red por *hop*.

- Reducir f:
  - Comunicar menos.
  - Hacer mensajes más largos.

- Reducir h:
  - Hacer que los patrones de comunicación coincidan con la topología.

Tipos de Contención

- Contención de la red:

  Se produce en los enlaces o *switches* de la red.

  Se reduce cuando se ajusta el patrón de comunicaciones a la topología de la red.

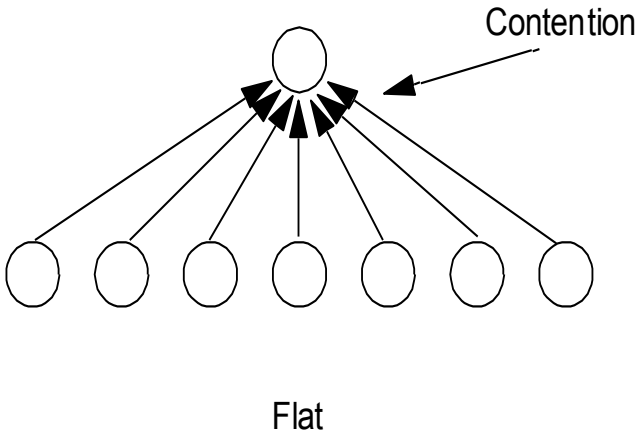
- Contención en los puntos finales:

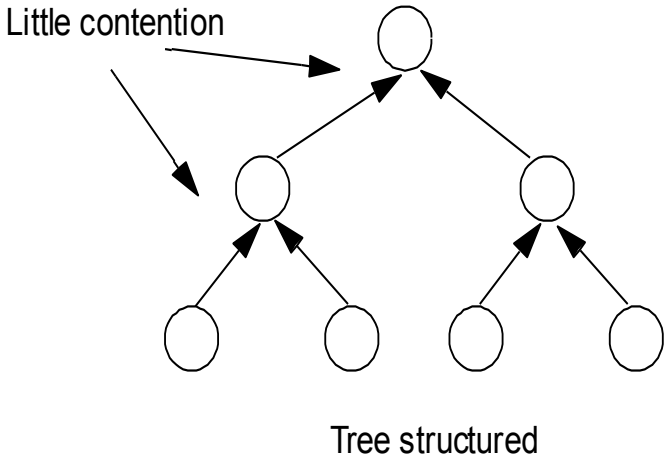
  Se produce en los nodos de procesamiento.

  El nodo en el que la contención es severa se denomina ***hot spot.***

  Ejemplo, acumulación de sumas parciales en suma global.

Comunicación Muchos a Uno





Solapar comunicación con cómputo

- Ocultar latencia.
- Requiere de concurrencia extra.
- Técnicas:
  - Pre-captura (búsqueda del dato antes de ser necesario).
  - Mientras la comunicación progresa, realizar otro trabajo. Usar técnica multi-hilos.

Costes de comunicación

- Costes de acceso a los datos:
  - Grandes transferencias:
    - Amortizan *overheads*.
    - Amortizan latencia.
  - Pequeñas transferencias:
    - Reducen contención.

Índice

Modelos de Rendimiento

Al diseñar un programa paralelo para una arquitectura se puede utilizar un modelo de rendimiento para predecir si la implementación de un programa será mejor que otra y guiar las decisiones acerca de comunicación, equilibrado de carga, trabajo extra etc.

Aspectos del Modelo de Rendimiento

- Modelar las características de la máquina.
- Modelar las características de la aplicación.

Desarrollo de modelo de funcionamiento analítico que tome como entrada ambos modelos y produzca como salida el tiempo de ejecución.

Modelar las características de la Máquina

Para conocer parámetros como overhead (**o**), latencia (**l**)

¿Cómo? Mediante pruebas patrón *(benchmarks)*, ejemplo *Parkbench*. Permite obtener una estimación del tiempo de transmisión de un mensaje.

Modelar las características de la aplicación

- Puede ser difícil cuando la aplicación es compleja e irregular.

- Es más difícil de realizar bajo un modelo compartido que bajo uno que use pase de mensajes pues la aplicación no tiene control sobre los eventos que se producen.

Bibliografía

D. Culler, J. Pal Singh, A. Gupta. *Parallel Computer Architecture. A Hardware/Software Approach*. Morgan Kaufmann. Capítulo 3.