UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES

Rendimiento Computacional
Computación Heterogénea

Profesor: Dr. Joel Fuentes - jfuentes@ubiobio.cl

Ayudantes:
- Daniel López - daniel.lopez1701@alumnos.ubiobio.cl
- Sebastián González - sebastian.gonzalez1801@alumnos.ubiobio.cl

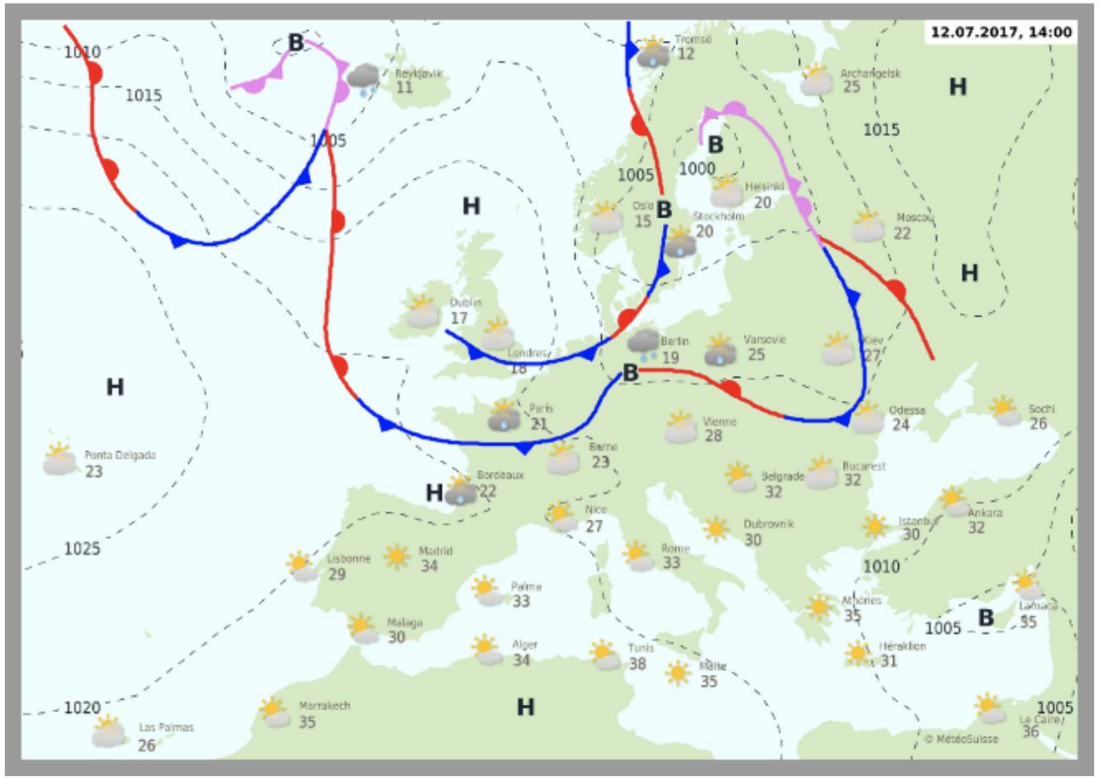
Página web del curso: http://www.face.ubiobio.cl/~jfuentes/classes/ch

Contenido

- Conceptos generales
- Rendimiento y escalamiento
- Métricas de desempeño
  - Ley de Amdahl
  - Ley de Gustafson-Barsis
- Modelos de ejecución
  - DAG

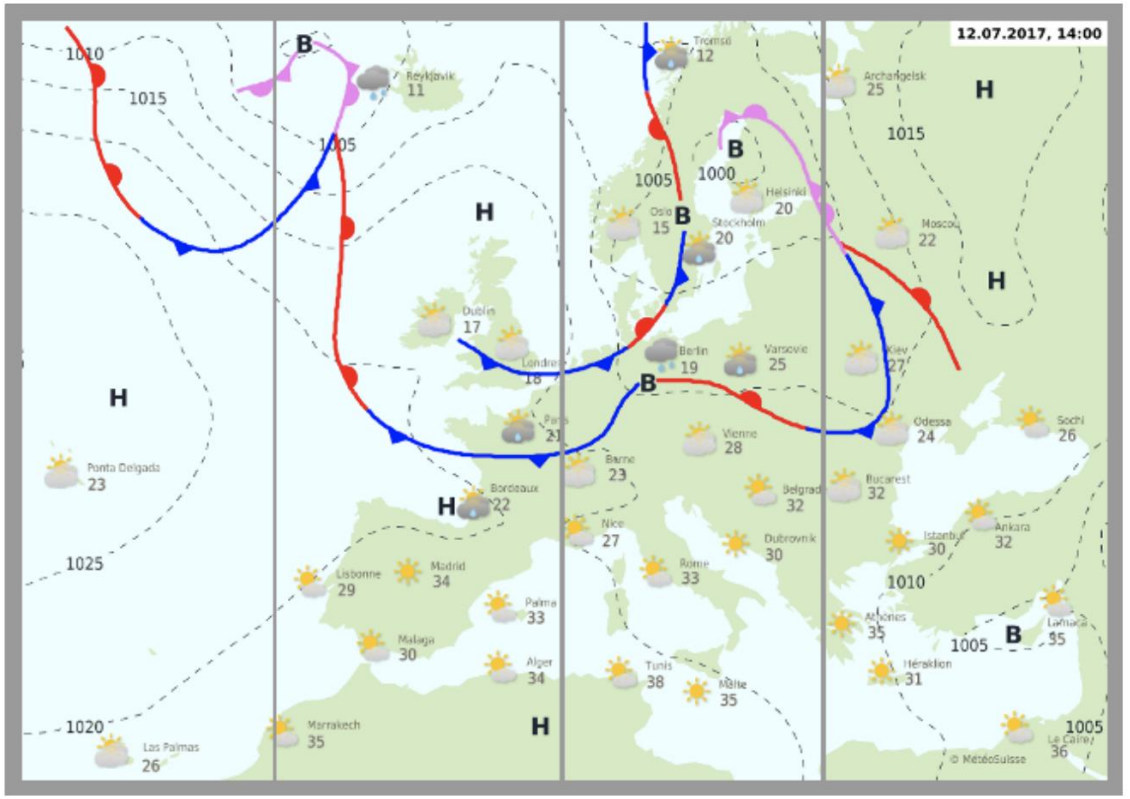
Conceptos generales

- Procesamiento secuencial



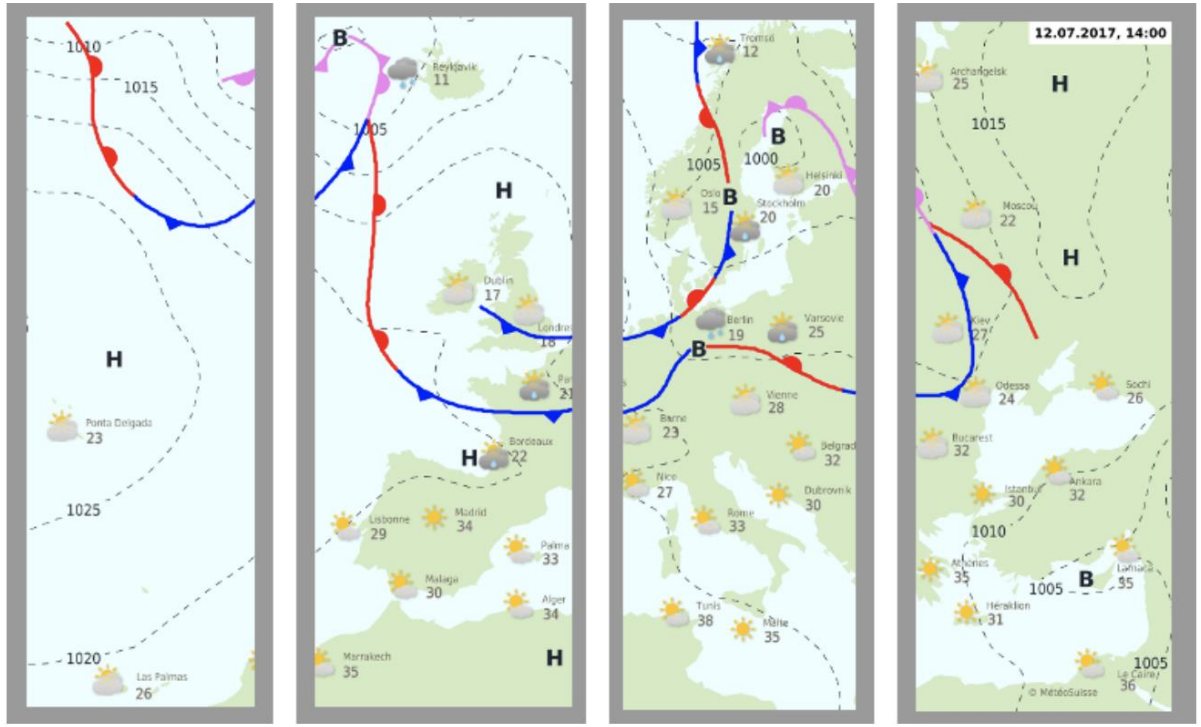
Conceptos generales

- Procesamiento paralelo en memoria compartida



Conceptos generales

- Procesamiento paralelo en memoria distribuída

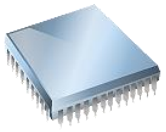


Rendimiento y Escalamiento

¿Qué es el Rendimiento Computacional?

- En computación, rendimiento (o performance) es definido por 2 factores:
  - Requerimientos computacionales (que debe ser realizado)
  - Recursos computacionales (cuál es el costo de hacerlo)
- Problemas computacionales se traducen en requerimientos.
- Recursos computacionales actúan como tradeoff.

$$Performance \sim \frac{1}{Recursos\ de\ la\ solución}$$



Hardware

Tiempo

Energía

Dinero

¿Qué es el Rendimiento/Desempeño Paralelo?

- Nos interesa conocer problemas en rendimiento cuando se utiliza ambientes de computación paralela.
- Rendimiento es la razón del paralelismo:
  - Si el rendimiento no es mejor, paralelismo no es necesario.
- Procesamiento paralelo incluye muchas técnicas y tecnologías:
  - Hardware, redes, sistemas operativos, bibliotecas, lenguajes de programación, compiladores, algoritmos, herramientas, etc.
- Paralelismo debe entregar mejor desempeño
  - Cómo? Cuánto mejor?

Desempeño esperado

- Si cada procesador trabaja a k MFLOPS y hay p procesadores, ¿tendremos entonces $k * p$ MFLOPS de desempeño?
- Si una tarea toma 100 segundos en 1 procesador, ¿no debería tomar 10 segundos en 10 procesadores?
- Muchas causas afectan el desempeño de un algoritmo paralelo.
  - Es necesario entender todas estas causas.
  - Solución a un problema podría crear otro.
- Escalamiento es una característica deseada en paralelismo.

Cómputo paralelo "embarazoso"

- Cómputo paralelo embarazoso es el que puede ser dividido de forma obvia en partes independientes que se ejecuta de forma simultánea.
  - En muchos no es necesario para interacción entre procesadores.
  - En otros puede ser necesario la distribución de resultados entre procesadores.
- Algoritmos con este tipo de cómputo paralelo tienen el potencial de lograr aceleración máxima en plataformas paralelas.
  - Si resolver un problema secuencial toma tiempo T, potencialmente se podría lograr tiempo T/P con P procesadores.

Escalamiento

- Un algoritmo puede escalar a utilizar muchos procesadores.
- ¿Cómo evaluar el escalamiento?
- Evaluación comparativa:
  - Si aumentamos al doble el número de procesadores, ¿Es el escalamiento lineal?
- La clave es aplicar métricas de desempeño

Métricas de Desempeño

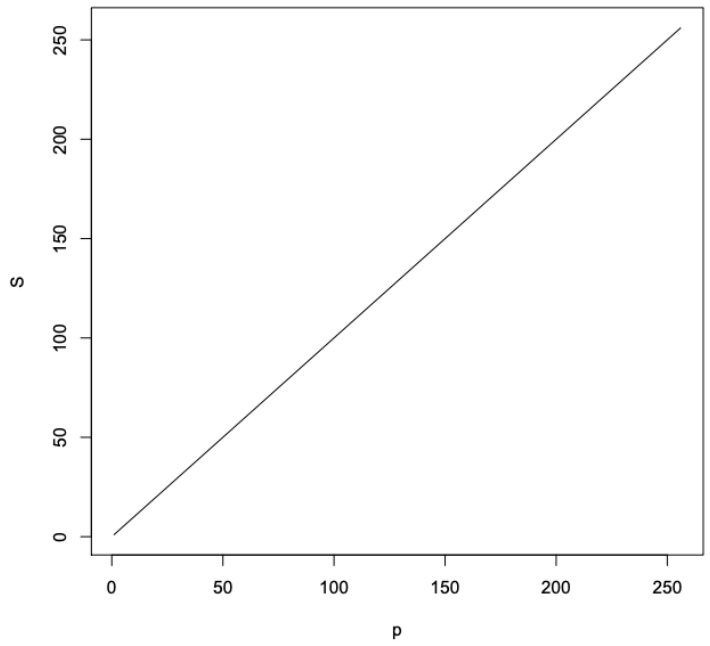
Métricas de desempeño

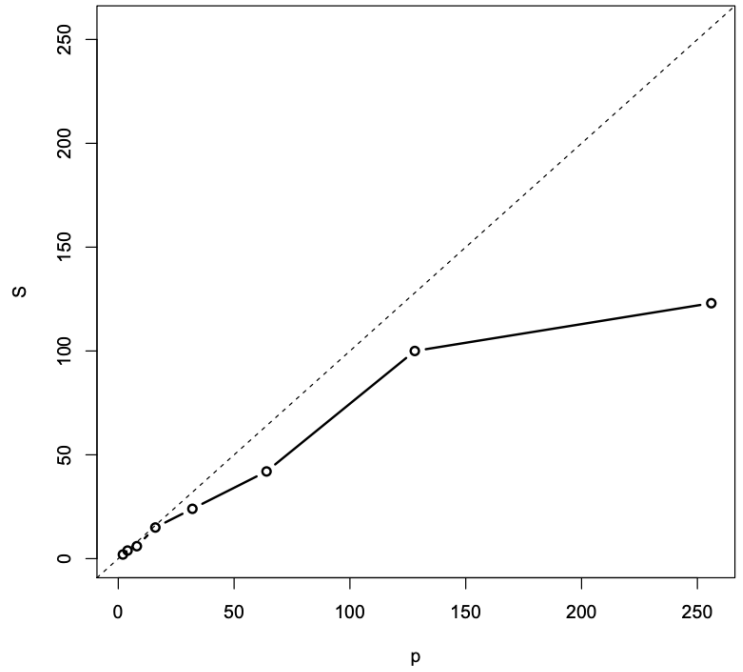
- Evaluación
  - Tiempo de ejecución secuencial (T_{sec}) es una función de:
    - El tamaño del problema y arquitectura
  - Tiempo de ejecución en paralelo (T_{par}) es una función de:
    - El tamaño del problema y arquitectura paralela
    - Número de procesadores usados en la ejecución
  - Desempeño paralelo es principalmente afectado por
    - Algoritmo + arquitectura
- Escalamiento
  - Es la capacidad de un algoritmo paralelo de lograr mejoras en desempeño de forma proporcional al número de procesadores y el tamaño del problema.

Métricas de desempeño

- T_1 es el tiempo de ejecución en un procesador
- T_p es el tiempo de ejecución en p procesadores
- S_p es la aceleración
  - $S_p = \frac{T_1}{T_p}$
- E_p es la eficiencia
  - $E_p = \frac{S_p}{p}$
- C_p es el costo
  - $C_p = p \times T_p$

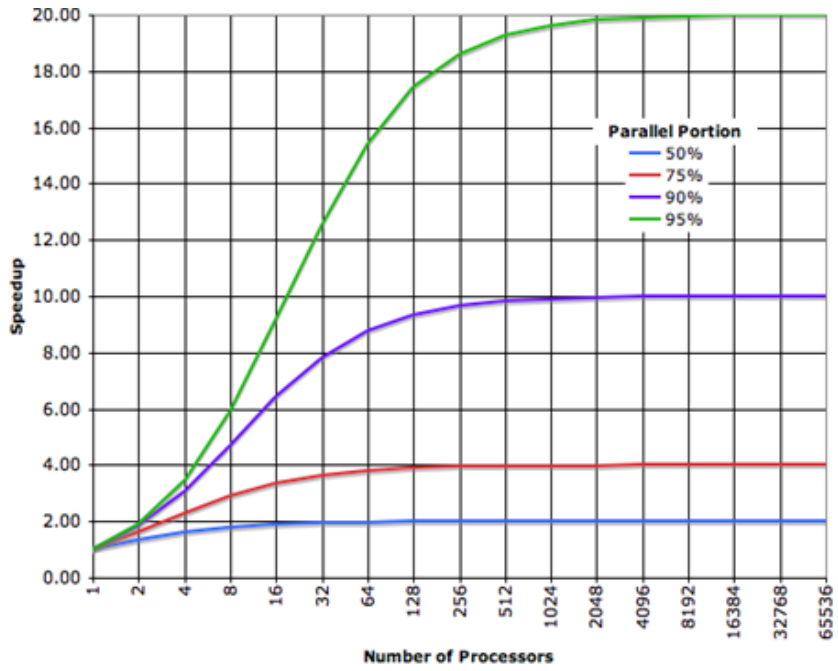
Aceleración ideal versus realidad S_p



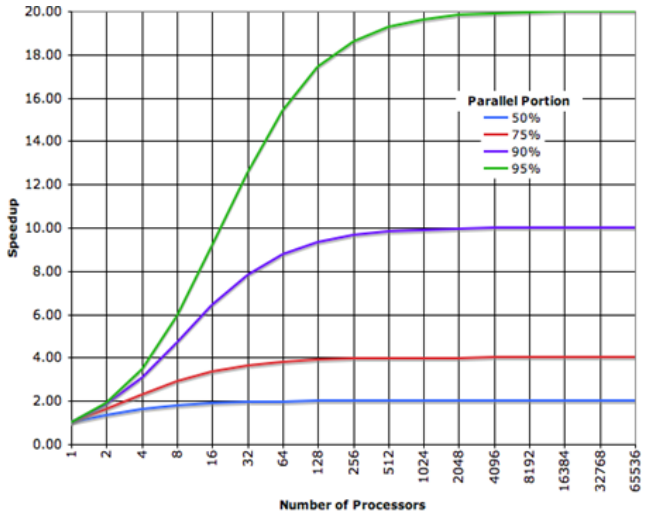


Métricas de desempeño: Ley de Amdahl

- Sea f la fracción de un programa que es secuencial
  - $1 - f$ es la fracción que puede ser paralelizada
- T_1 es el tiempo de ejecución en un procesador
- T_p es el tiempo de ejecución en p procesadores
- S_p es la aceleración
  - $S_p = T_1/T_p$
  - $= T_1/(fT_1 + \frac{(1-f)T_1}{p})$
  - $= 1/(f + \frac{(1-f)}{p})$
- Si $p \to \infty$
  - $S_\infty = 1/f$



Métricas de desempeño: Ley de Amdahl



- ¿Cuándo aplicar la Ley de Amdahl?
  - Cuando el tamaño del problema es fijo.
  - Escalamiento fuerte ($p \to \infty, S_p = S_\infty \to 1/f$)
  - Límite de aceleración es determinada por el grado de ejecución secuencial, **no el número de procesadores!**
  - ¿es esto bueno? ¿Por qué?
  - Eficiencia perfecta es muy difícil de lograr.
- Ver paper de Amdahl adjuntado en la plataforma del curso

Métricas de desempeño: Ley de Gustafson-Barsis

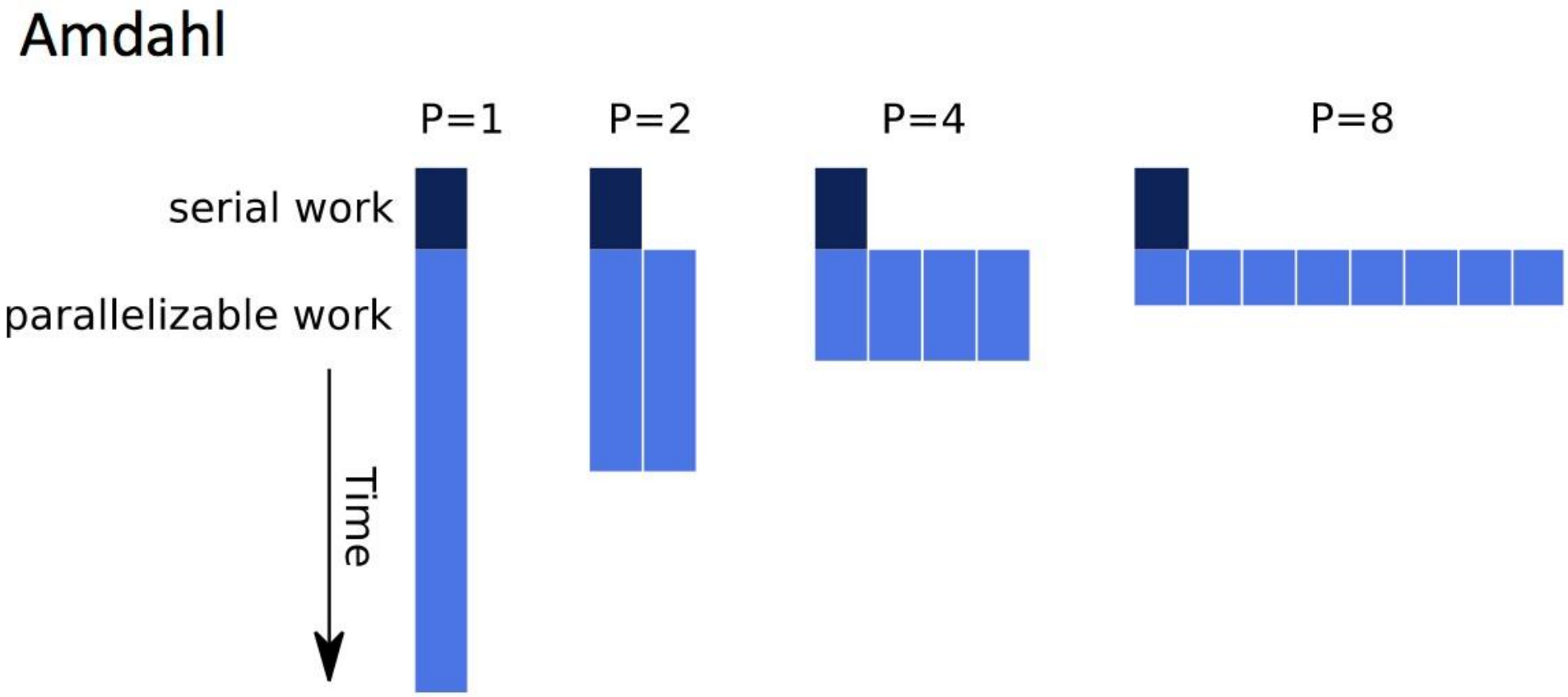
- Asuma que el tiempo paralelo es mantenido constante
  - $T_p = C = \left(f + (1 - f)\right) * C$
  - f_{sec} es la fracción de T_p en ejecución secuencial
  - f_{par} es la fracción de T_p en ejecución paralela
- ¿Cuál es el tiempo de ejecución en un procesador?
  - Si $C = 1$, entonces
  - $T_s = f_{sec} + p(1 - f_{sec}) = 1 + (p - 1)f_{par}$
- ¿Cuál es la aceleración en este caso?
  - $S_p = \frac{T_s}{T_p} = \frac{T_s}{1}$
  - $S_p = 1 + (p - 1)f_{par}$

Métricas de desempeño: Ley de Gustafson-Barsis

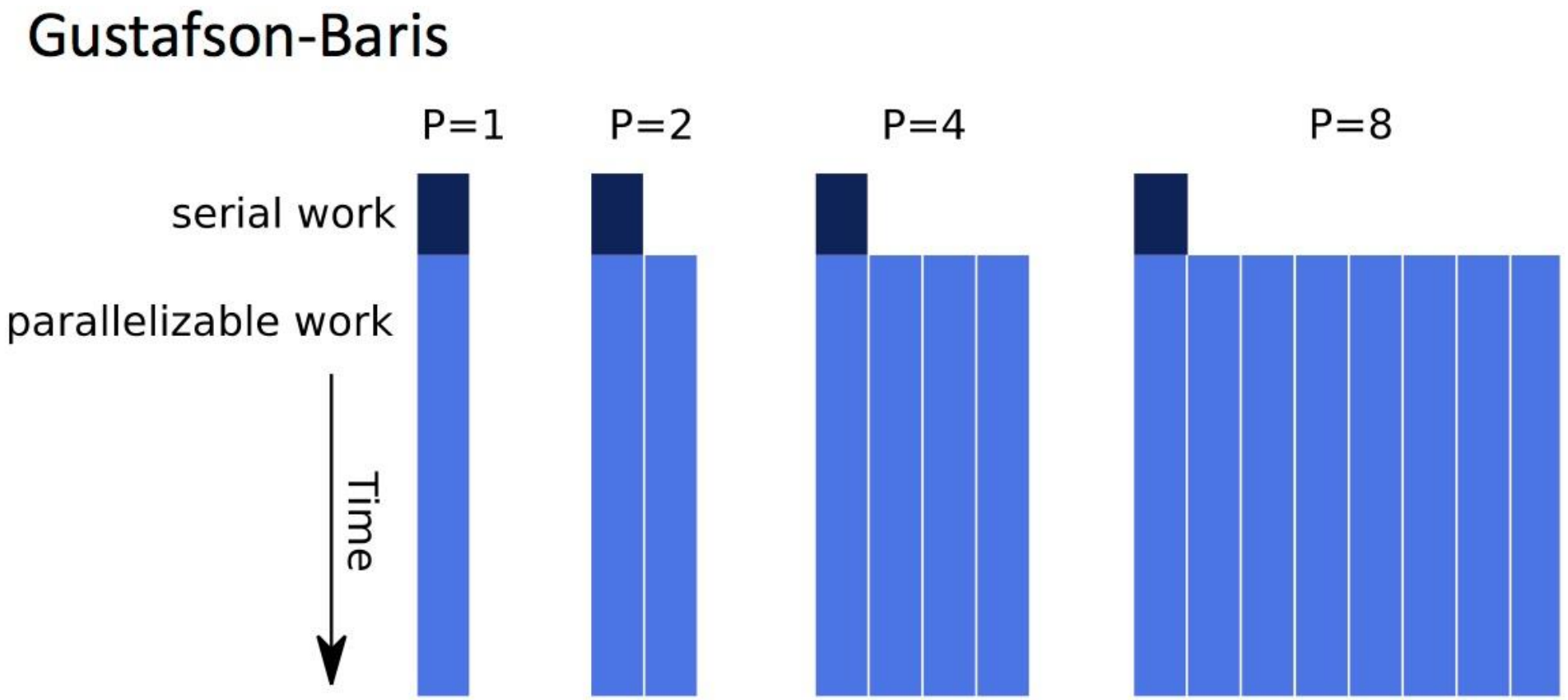
- ¿Cuándo aplicar la Ley de Gustafson-Barsis?
  - Cuando el tamaño del problema puede crecer mientras el número de procesadores también incrementa
  - Escalamiento débil ($S_p = 1 + (p-1)f_{par}$)
  - Función de aceleración incluye el **número de procesadores!**
  - Puede mantener o incrementar eficiencia paralela mientras el problema escala

- Ver paper de Gustafson-Barsis adjuntado en la plataforma del curso

Amdahl versus Gustafson-Barsis



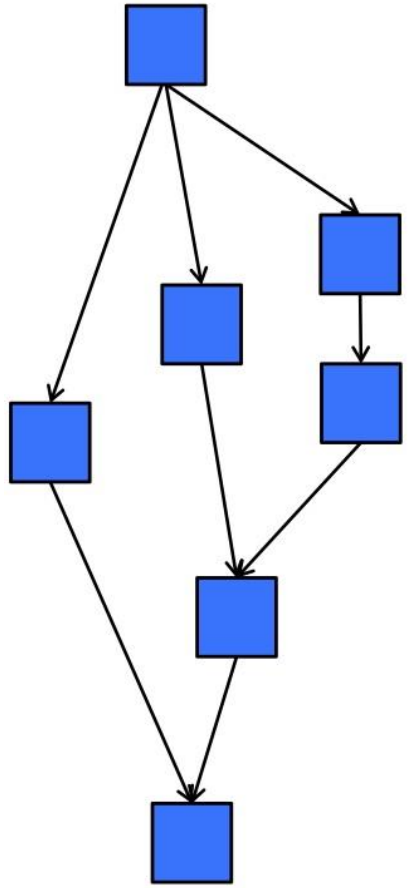
Amdahl versus Gustafson-Barsis



Modelos de Ejecución

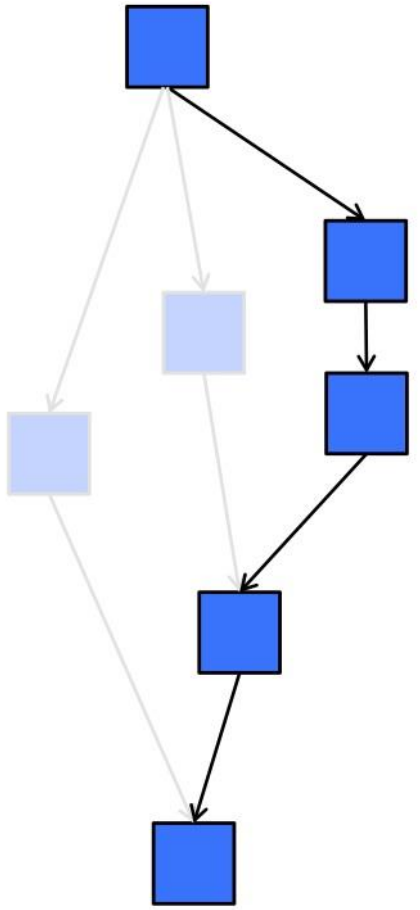
Modelos de ejecución: DAG

- Asuma un programa como un grafo acíclico dirigido (DAG) de tareas
  - Una tarea no puede ejecutar hasta que todos sus inputs estén disponibles
  - Inputs vienen de outpus de otras tareas ejecutadas anteriormente
  - DAG muestra explícitamente la dependencia de tareas
- Considere un planificador de tareas "greedy" (ávaro) para asignar tareas a procesadores
  - No deben haber procesadores ociosos mientras haya tareas por ejecutar



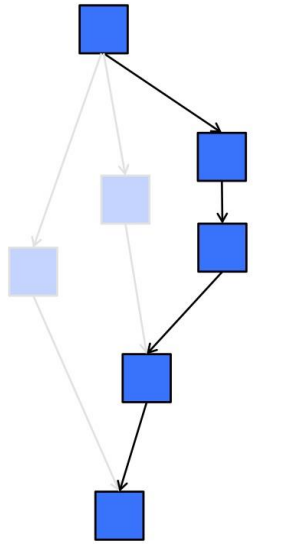
Modelos de ejecución: DAG

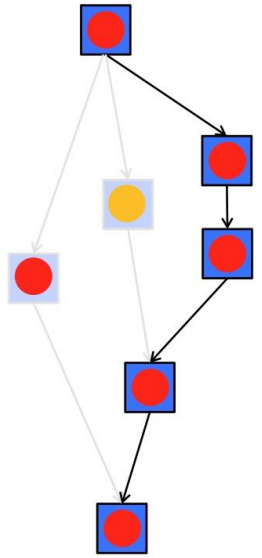
- Ejemplo:
  - Cada tarea tome 1 unidad de tiempo
  - DAG tiene 7 tareas
  - $T_1 = 7$
    - Todas las tareas deben ser ejecutadas
    - Tareas son ejecutadas en orden serial
    - ¿Puede las tareas ser ejecutadas en cualquier orden?
  - $T_\infty = 5$
    - Tiempo por el camino crítico
    - En este caso, es el camino más largo de tareas con dependencias lineales.



Lower/Upper bound con planificación “greedy”

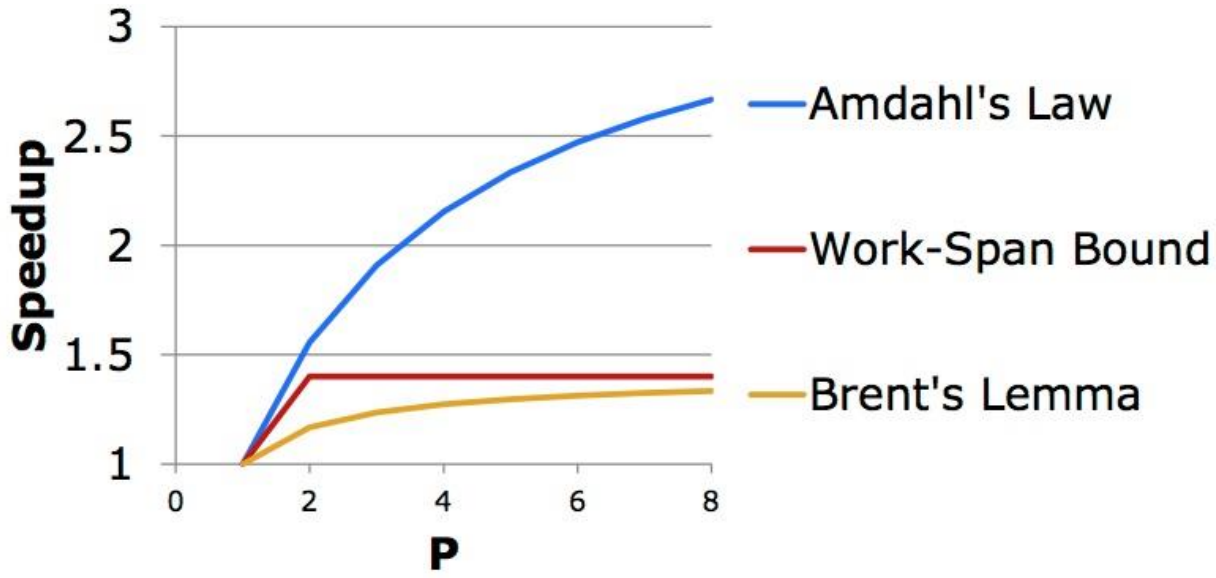
- Suponga que sólo hay p procesadores
- Es posible escribir una fórmula para reflejar el lower bound de T_p
  - $Max(\frac{T_1}{p}, T_\infty) \leq T_p$
- T_∞ es el mejor tiempo de ejecución posible
- Lemma de Brent para el upper bound
  - Capturar el costo adicional de ejecutar otras tareas que no están en el camino crítico
  - Asumir que esto se puede hacer sin grandes costos adicionales
  - $T_p \leq \frac{T_1 - T_\infty}{p} + T_\infty$

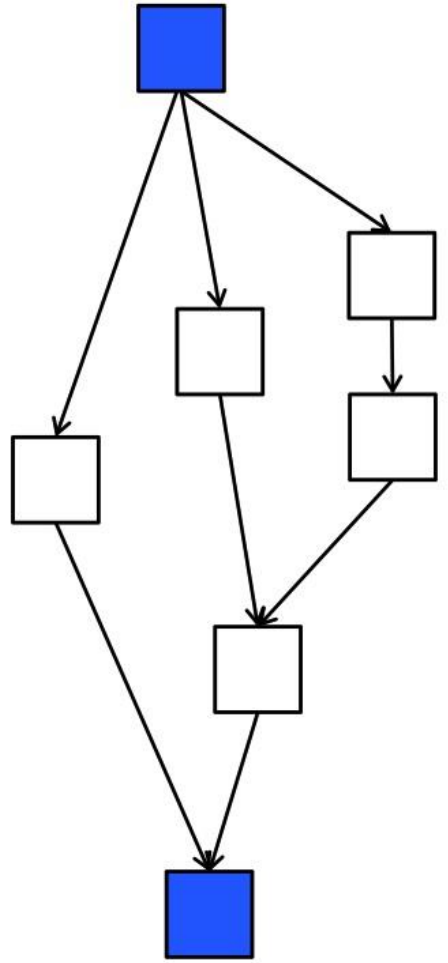




Lower/Upper bound con planificación “greedy”

- Amdahl es optimista





Referencias

- Parallel Computing Center. University of Oregon http://ipcc.cs.uoregon.edu/index.html