

Evaluación de Algoritmos Paralelos

- ❑ **Objetivo de la evaluación**
- ❑ **Estrategias de evaluación**
- ❑ **Figuras de mérito**
- ❑ **Ejemplo: OCEAN en paso de mensajes**

Objetivo de la evaluación

Dar respuesta a preguntas tales como:

- ✓ A: ¿Qué algoritmo es más rápido, entre varias alternativas ?
- ✓ B: ¿Con qué eficacia estoy usando la máquina?
- ✓ C: ¿Cómo se comporta el algoritmo al variar alguna de las características de la máquina o del problema?

Estrategias de evaluación

❑ **Evaluación experimental**

- ✓ Requiere la disponibilidad del hardware y la implementación del algoritmo
- ✓ No permite especular sobre el comportamiento del algoritmo ante modificaciones de la máquina

❑ **Evaluación mediante simulación**

- ✓ Requiere una especificación detallada de muchos detalles de la máquina y del algoritmo
- ✓ Puede ser lento

❑ **Evaluación mediante modelo analítico**

- ✓ La construcción del modelo puede ser difícil, si se desea mucha precisión
- ✓ Es rápido
- ✓ Facilita la comparación de algoritmos

No son estrategias incompatibles

Estrategias de evaluación

□ Evaluación mediante modelo analítico

$$\text{figura de mérito} = f(P_m, P_p, P_a)$$

P_m : Parámetros de la máquina (T_{sup} , T_e , T_{op} , etc)

P_p : Parámetros del problema (N , número de condición, etc)

P_a : Parámetros del algoritmo (tamaño de bloque)

- ✓ Un modelo preciso es difícil de construir, particularmente para un modelo de variables compartidas

Figuras de mérito

❑ Tiempo de ejecución

- ✓ Suele ser la figura de interés para el usuario
- ✓ No permite evaluar la eficacia del algoritmo (pregunta B)

❑ Speed Up

$$S = \frac{T_1}{T_p}$$

T_1 es el tiempo de ejecución en un procesador

T_p es el tiempo de ejecución en p procesadores

Speed Up absoluto

T_1 se calcula usando el mejor algoritmo secuencial

Speed Up relativo

T_1 se calcula ejecutando el algoritmo paralelo en un solo procesador

- ✓ El Speed Up permite responder a la pregunta B



Figuras de mérito

□ Speed Up

Sea:

$$T_1 = T_{seq} + T_{par}$$

T_{seq} es el tiempo empleado en cálculos no paralelizables.

T_{par} es el tiempo empleado en cálculos perfectamente paralelizables entre cualquier número de procesadores.

Entonces:

$$T_p = T_{seq} + \frac{T_{par}}{p} + O(p)$$

$O(p)$ es el overhead debido a:

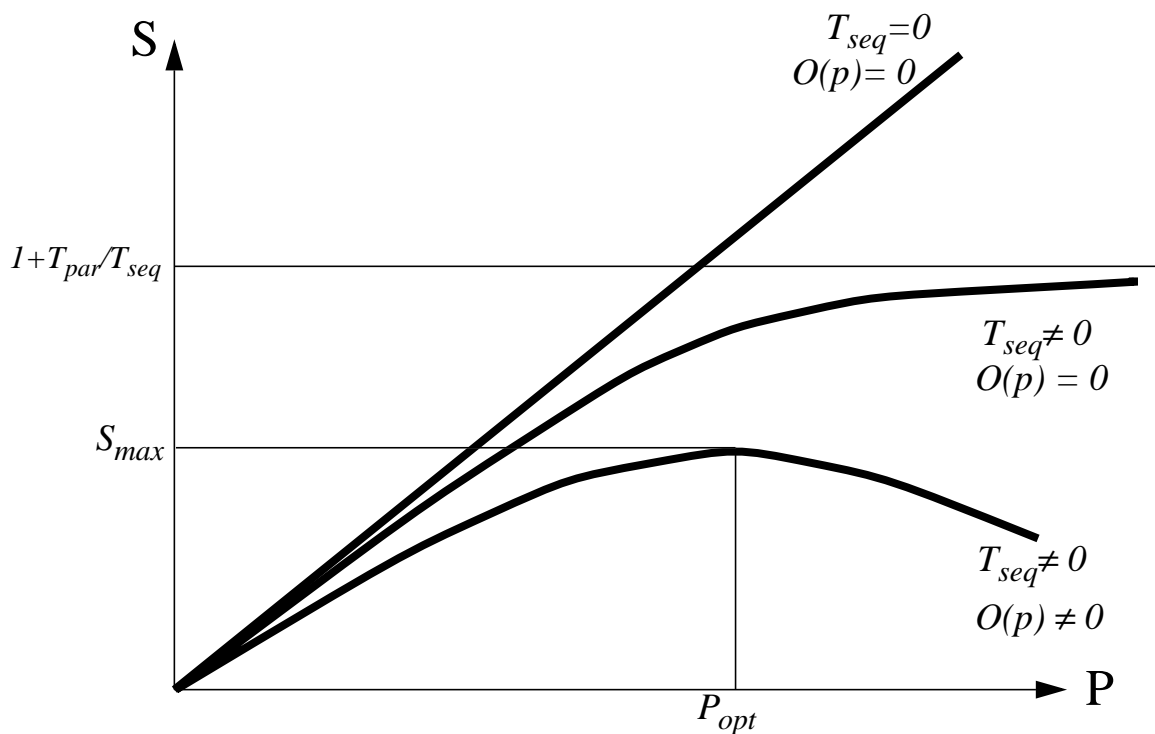
Desequilibrio de carga

Sobrecarga de comunicación y/o sincronización

Figuras de mérito

□ Speed Up. La ley de Amdahl.

$$S = \frac{T_{seq} + T_{par}}{T_{seq} + \frac{T_{par}}{p} + O(p)}$$



El valor de P_{opt} está limitado por el nivel de concurrencia.

Figuras de mérito

□ Speed Up escalado

Generalmente, cuando un usuario amplia el número de procesadores amplia también el tamaño del problema a resolver.

Fixed-Time Speed Up

El usuario está trabajando con un sistema mediante el que resuelve un problema en un tiempo T . Al aumentar el tamaño del sistema, aumenta también el tamaño del problema de forma que el tiempo de ejecución se mantiene constante.

Sea $T_i(n)$ el tiempo en procesar un problema de tamaño n en i procesadores.

Si T es el tiempo de referencia que debe mantenerse constante, en un sistema con p procesadores se resolverá un problema de tamaño $n'(p)$ tal que:

$$T_p(n'(p)) = T$$

El speed up se define como:

$$\text{SpeedUp} = \frac{T_1(n'(p))}{T_p(n'(p))} = \frac{T_1(n'(p))}{T}$$

El valor de $n'(p)$ está limitado por: (a) el nivel de concurrencia y (b) el tamaño de la memoria del sistema.

Figuras de mérito

□ Speed Up escalado

Memory Bounded Speed Up

Al aumentar el tamaño del sistema aumenta también el tamaño del problema de forma que se ocupa toda la memoria disponible.

Sea K el tamaño de la memoria de un nodo (procesador + memoria local).

Sea $M(n)$ la cantidad de memoria requerida para resolver un problema de tamaño n .

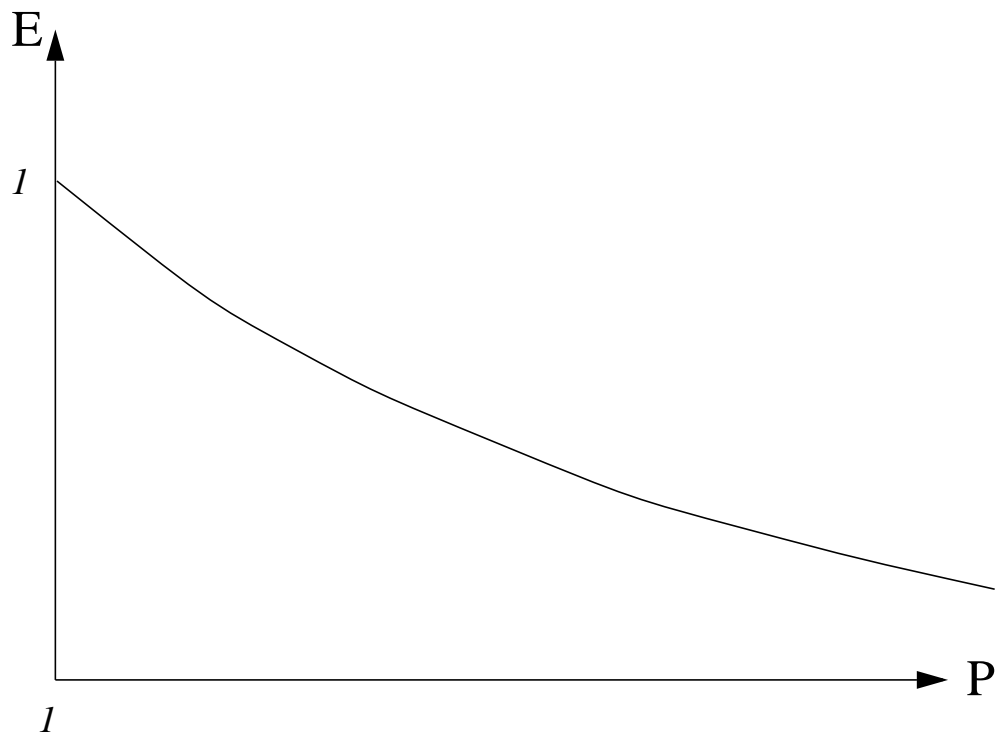
Cuando se dispone de p procesadores se resuelve un problema de tamaño $M^{-1}(Kp)$. El speed up se define como:

$$\text{SpeedUp} = \frac{T_1(M^{-1}(Kp))}{T_p(M^{-1}(Kp))}$$

Figuras de mérito

□ Eficiencia

$$E(p) = \frac{\text{SpeedUp}(p)}{p} = \frac{T_1(n)}{T_p(n)p}$$



Figuras de mérito

□ La Fracción Serie

Objetivo

Se tiene que:

$$T_1 = T_{seq} + T_{par}$$
$$T_p = T_{seq} + \frac{T_{par}}{p} + O(p)$$

Si se evalúa la eficiencia para varios procesadores se observa, en general que decrece al aumentar el número de procesadores.

Eficiencias de Linpack para Cray Y-MP

p	s	e
1	-	-
2	1.95	0.975
3	2.88	0.960
4	3.76	0.940
8	6.96	0.870

La pregunta que se pretende responder es: ¿La caída de la eficiencia se debe a falta de paralelismo en el problema/algoritmo (valor elevado de T_{seq}), o a un crecimiento importante del overhead (valor creciente de $O(p)$) ?.

Figuras de mérito

□ La Fracción Serie

Definición:

$$f_p = \frac{T_{seq} + O(p)}{T_1}$$

Midiendo f_p para diferentes valores de p es posible determinar la evolución del overhead al crecer p . Si f_p permanece constante entonces el overhead es nulo y, por tanto, la pérdida de eficiencia se debe a una falta de paralelismo del problema/algoritmo.

Medición

La fracción serie puede medirse de forma experimental. Supongamos que $O(p) = 0$ (caso ideal).

$$f_p = \frac{T_{seq}}{T_1} \Rightarrow T_p = T_1 f_p + \frac{T_1(1 - f_p)}{p} \Rightarrow$$

$$\frac{1}{s} = f_p + \frac{1 - f_p}{p} \Rightarrow$$

$$f_p = \frac{1/s - 1/p}{1 - 1/p}$$

Figuras de mérito

□ La Fracción Serie

Conclusión

Si f_p permanece constante entonces fue correcto suponer que $O(p) = 0$ y, por tanto, la caída de la eficiencia se deba a falta de paralelismo.

Si f_p crece con p entonces no es verdad que $O(p) = 0$. El overhead crece con el número de procesadores.

Ejemplos

Eficiencias de Linpack para Cray Y-MP

p	s	e	f_p
1	-	-	-
2	1.95	0.975	0.024
3	2.88	0.960	0.021
4	3.76	0.940	0.021
8	6.96	0.870	0.021

En este ejemplo, la pérdida de eficiencia se debe a falta de paralelismo del problema/algorithm.

Figuras de mérito

□ La Fracción Serie

Ejemplos

Eficiencias de Linpack para Alliant FX/40

p	s	e	f_p
1	-	-	-
2	1.90	0.950	0.053
3	2.65	0.883	0.066
4	3.22	0.805	0.080

En este caso, el overhead crece con el número de procesadores.

Uno de los ganadores del Bell Award

p	s	e	f_p
4	3.95	0.9885	0.0039
16	15.46	0.9663	0.0023
64	57.46	0.8978	0.0018
256	177.5	0.6934	0.0017
1024	351.2	0.3430	0.0019

Se hizo un buen trabajo de paralelización (valor pequeño de T_{seq}).

Figuras de mérito

□ Escalabilidad

Definición

Se dice que un sistema (arquitectura + algoritmo) es escalable si es capaz de usar de forma satisfactoria un número creciente de procesadores.

Existen muchas propuestas de métodos para determinar si un sistema es escalable o no. Cada método toma como punto de partida una definición más precisa de “uso satisfactorio de un número creciente de procesadores”.

Un ejemplo: Isoeficiencia

Un sistema es escalable si, al aumentar el número de procesadores es posible mantener la eficiencia constante, aumentando para ello el tamaño del problema (número de operaciones a realizar) en la medida en que sea necesario.

La escalabilidad se caracteriza mediante la función $F(p)$ que indica el tamaño de problema necesario para mantener la eficiencia constante con p procesadores.



Ejemplo: OCEAN en paso de mensajes

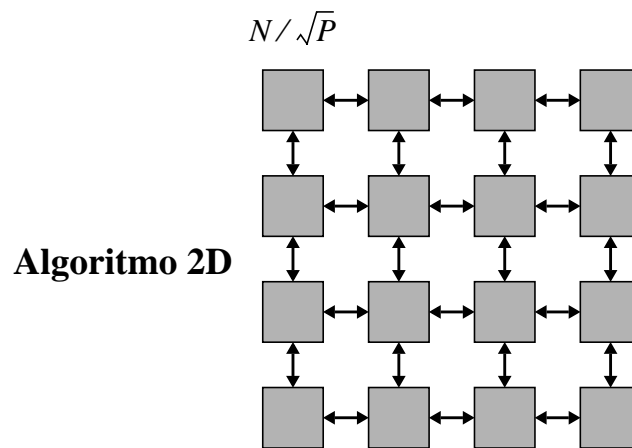
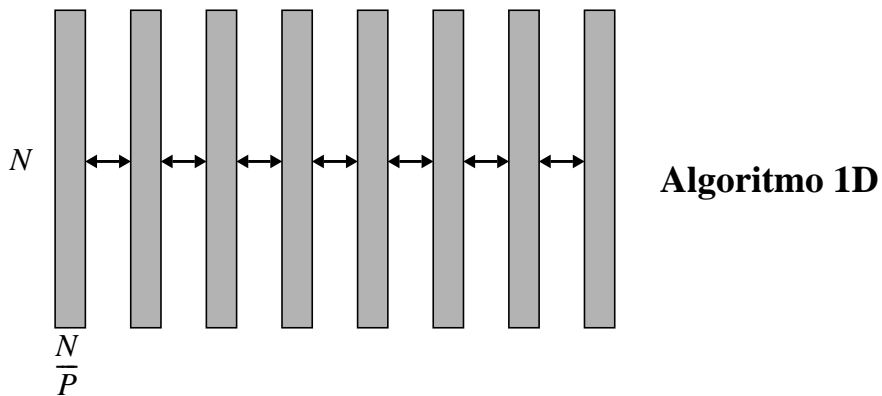
❑ Algoritmo secuencial

$$T_1(N) = 6N^2T_c$$

N es la dimensión de la matriz

T_c es el tiempo de producto o suma

❑ Algoritmos paralelos



Asumimos un multicomputador con store-and-forward

Ejemplo: OCEAN en paso de mensajes

□ Modelo del multicomputador

- ✓ Coste de operación aritmética: T_c (incluye el acceso a los datos)
- ✓ Comunicación store-and-forward y all-port
- ✓ Coste de enviar k mensajes de N números cada uno por enlaces diferentes:

$$k \times T_s + N \times T_e$$

Para los ejemplos supondremos:

T_c	1
T_e	1
T_s	200

Ejemplo: OCEAN en paso de mensajes

□ Modelo para el algoritmo 1D

Intercambio de las columnas de las fronteras

$$2T_s + NT_e$$

Actualización de la submatriz local

$$6\frac{N^2}{p}T_c$$

Determinación de la convergencia

$$\frac{p}{2}(2T_s + T_e + 2T_c)$$

Ejemplo: OCEAN en paso de mensajes

□ **Análisis del algoritmo 1D**

Fixed-problem size

Fixed-time speed-up

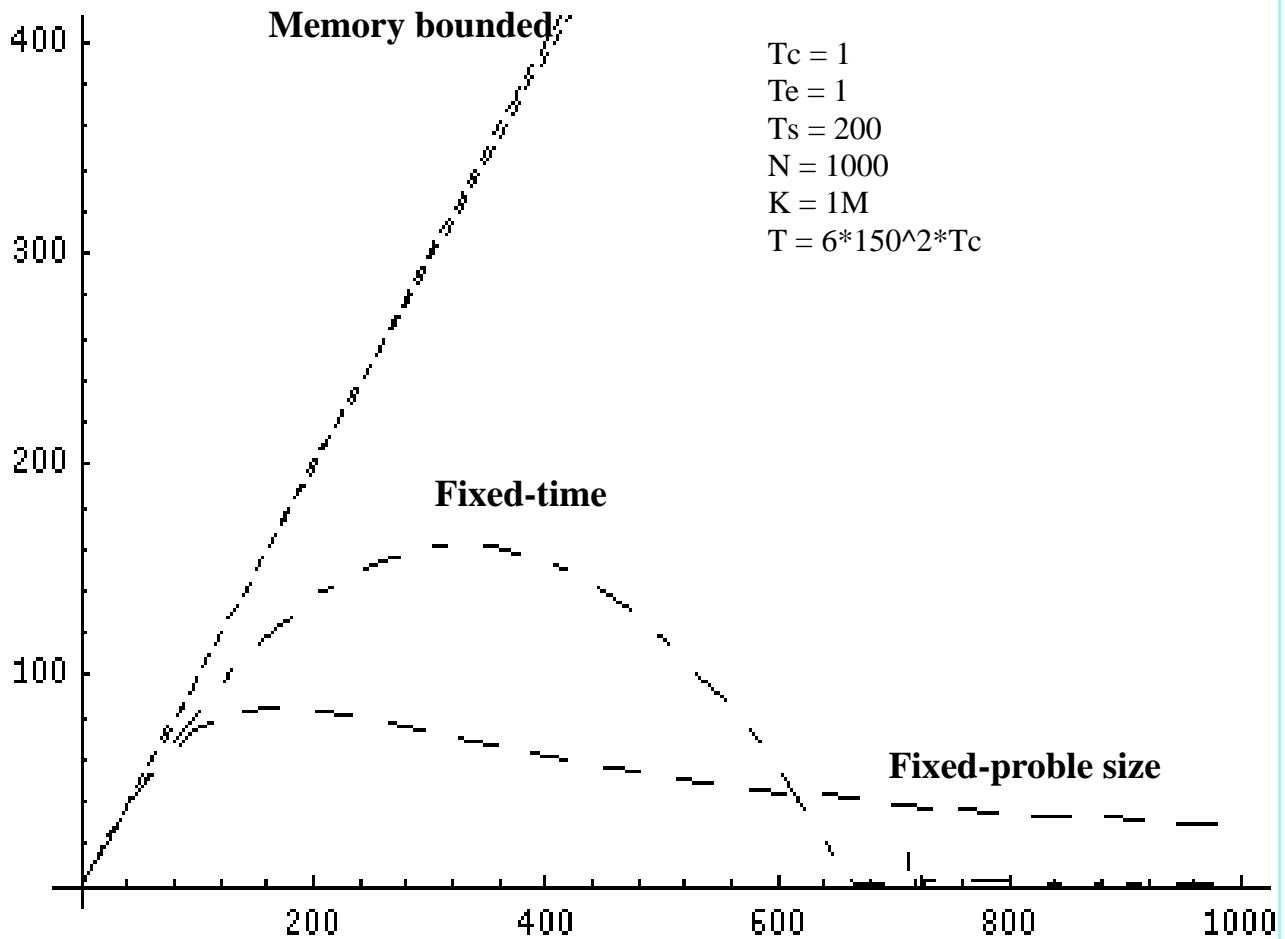
Memory bounded speed-up

Escalabilidad



Ejemplo: OCEAN en paso de mensajes

□ Speed-up para el algoritmo 1D



Ejemplo: OCEAN en paso de mensajes

□ Modelo para el algoritmo 2D

Intercambio de las columnas de las fronteras

$$4T_s + \frac{N}{\sqrt{p}}T_e$$

Actualización de la submatriz local

$$6\frac{N^2}{p}T_c$$

Determinación de la convergencia

$$\sqrt{p}(2T_s + T_e + 2T_c)$$

Ejemplo: OCEAN en paso de mensajes

□ **Análisis del algoritmo 2D**

Fixed-problem size

Fixed-time speed-up

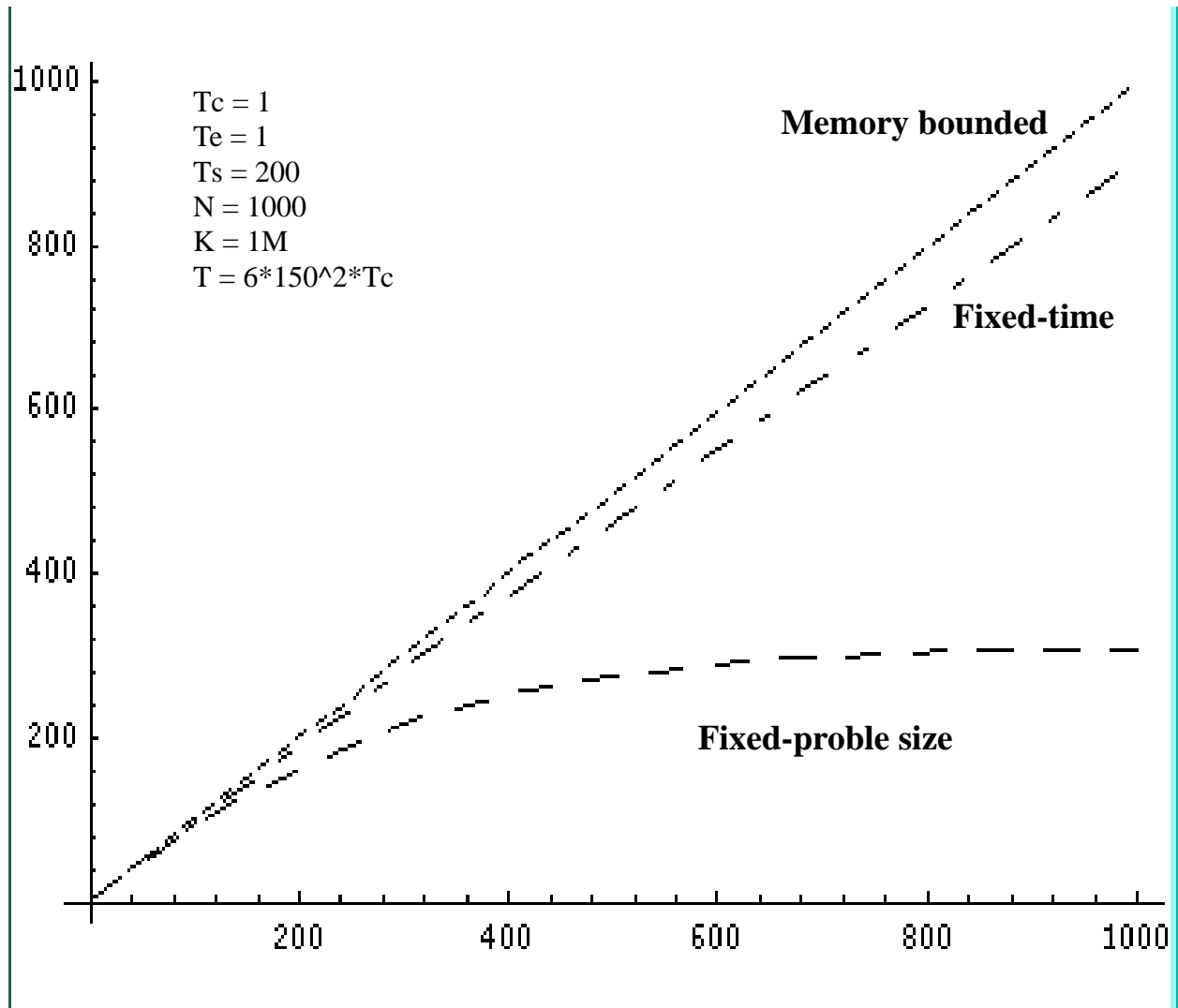
Memory bounded speed-up

Escalabilidad



Ejemplo: OCEAN en paso de mensajes

□ Speed-up para el algoritmo 2D



Evaluación de Algoritmos Paralelos

- ❑ **Objetivo de la evaluación**
- ❑ **Estrategias de evaluación**
- ❑ **Figuras de mérito**
- ❑ **Ejemplo: OCEAN en paso de mensajes**

Objetivo de la evaluación

Dar respuesta a preguntas tales como:

- ✓ A: ¿Qué algoritmo es más rápido, entre varias alternativas ?
- ✓ B: ¿Con qué eficacia estoy usando la máquina?
- ✓ C: ¿Cómo se comporta el algoritmo al variar alguna de las características de la máquina o del problema?

Estrategias de evaluación

❑ **Evaluación experimental**

- ✓ Requiere la disponibilidad del hardware y la implementación del algoritmo
- ✓ No permite especular sobre el comportamiento del algoritmo ante modificaciones de la máquina

❑ **Evaluación mediante simulación**

- ✓ Requiere una especificación detallada de muchos detalles de la máquina y del algoritmo
- ✓ Puede ser lento

❑ **Evaluación mediante modelo analítico**

- ✓ La construcción del modelo puede ser difícil, si se desea mucha precisión
- ✓ Es rápido
- ✓ Facilita la comparación de algoritmos

No son estrategias incompatibles

Estrategias de evaluación

□ Evaluación mediante modelo analítico

$$\textit{figura de mérito} = f(P_m, P_p, P_a)$$

P_m: Parámetros de la máquina (T_{sup}, T_e, T_{op}, etc)

P_p: Parámetros del problema (N, número de condición, etc)

P_a: Parámetros del algoritmo (tamaño de bloque)

- ✓ Un modelo preciso es difícil de construir, particularmente para un modelo de variables compartidas

Figuras de mérito

❑ Tiempo de ejecución

- ✓ Suele ser la figura de interés para el usuario
- ✓ No permite evaluar la eficacia del algoritmo (pregunta B)

❑ Speed Up

$$S = \frac{T_1}{T_p}$$

T_1 es el tiempo de ejecución en un procesador

T_p es el tiempo de ejecución en p procesadores

Speed Up absoluto

T_1 se calcula usando el mejor algoritmo secuencial

Speed Up relativo

T_1 se calcula ejecutando el algoritmo paralelo en un solo procesador

- ✓ El Speed Up permite responder a la pregunta B

Figuras de mérito

□ Speed Up

Sea:

$$T_1 = T_{seq} + T_{par}$$

T_{seq} es el tiempo empleado en cálculos no paralelizables.

T_{par} es el tiempo empleado en cálculos perfectamente paralelizables entre cualquier número de procesadores.

Entonces:

$$T_p = T_{seq} + \frac{T_{par}}{p} + O(p)$$

$O(p)$ es el overhead debido a:

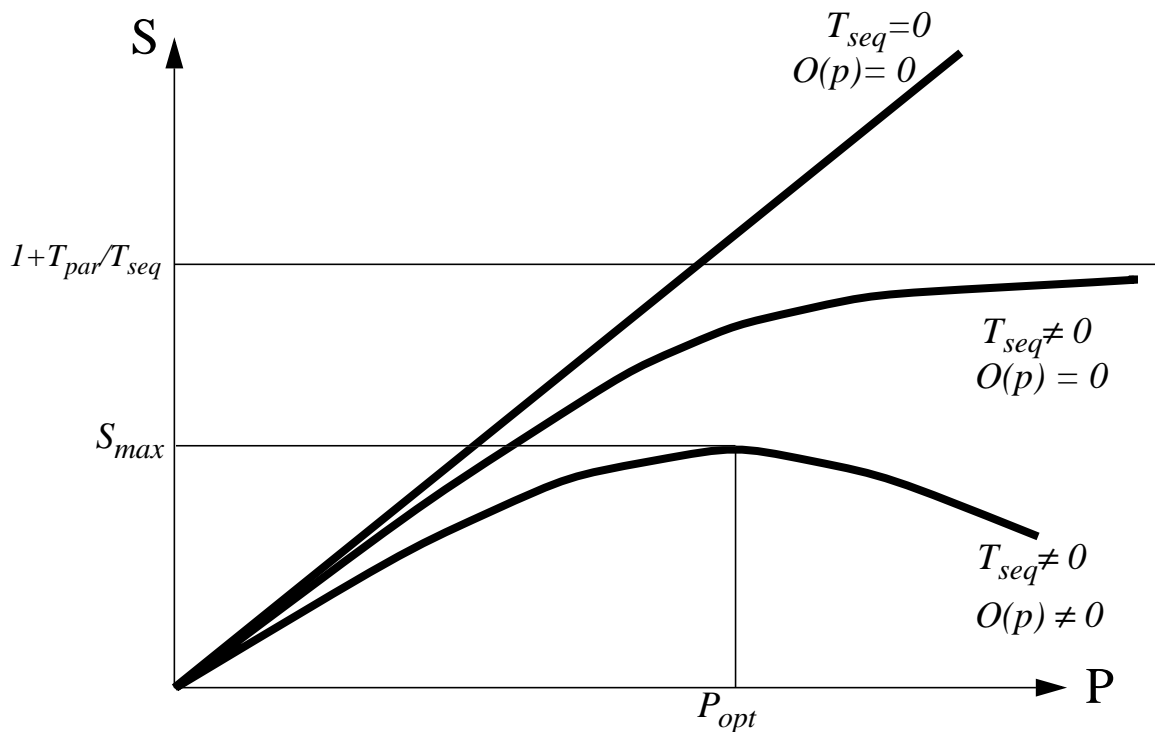
Desequilibrio de carga

Sobrecarga de comunicación y/o sincronización

Figuras de mérito

□ Speed Up. La ley de Amdahl.

$$S = \frac{T_{seq} + T_{par}}{T_{seq} + \frac{T_{par}}{p} + O(p)}$$



El valor de P_{opt} está limitado por el nivel de concurrencia.

Figuras de mérito

□ Speed Up escalado

Generalmente, cuando un usuario amplía el número de procesadores amplía también el tamaño del problema a resolver.

Fixed-Time Speed Up

El usuario está trabajando con un sistema mediante el que resuelve un problema en un tiempo T . Al aumentar el tamaño del sistema, aumenta también el tamaño del problema de forma que el tiempo de ejecución se mantiene constante.

Sea $T_i(n)$ el tiempo en procesar un problema de tamaño n en i procesadores.

Si T es el tiempo de referencia que debe mantenerse constante, en un sistema con p procesadores se resolverá un problema de tamaño $n'(p)$ tal que:

$$T_p(n'(p)) = T$$

El speed up se define como:

$$\text{SpeedUp} = \frac{T_1(n'(p))}{T_p(n'(p))} = \frac{T_1(n'(p))}{T}$$

El valor de $n'(p)$ está limitado por: (a) el nivel de concurrencia y (b) el tamaño de la memoria del sistema.

Figuras de mérito

□ Speed Up escalado

Memory Bounded Speed Up

Al aumentar el tamaño del sistema aumenta también el tamaño del problema de forma que se ocupa toda la memoria disponible.

Sea K el tamaño de la memoria de un nodo (procesador + memoria local).

Sea $M(n)$ la cantidad de memoria requerida para resolver un problema de tamaño n .

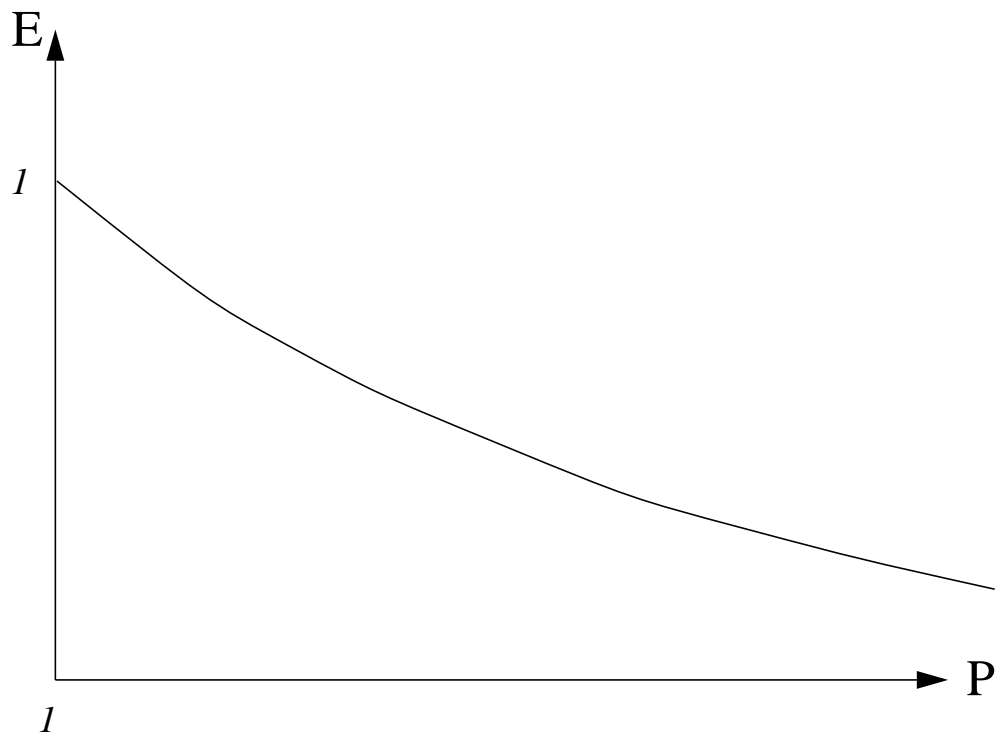
Cuando se dispone de p procesadores se resuelve un problema de tamaño $M^{-1}(Kp)$. El speed up se define como:

$$\text{SpeedUp} = \frac{T_1(M^{-1}(Kp))}{T_p(M^{-1}(Kp))}$$

Figuras de mérito

□ Eficiencia

$$E(p) = \frac{\text{SpeedUp}(p)}{p} = \frac{T_1(n)}{T_p(n)p}$$



Figuras de mérito

□ La Fracción Serie

Objetivo

Se tiene que:

$$T_1 = T_{seq} + T_{par}$$
$$T_p = T_{seq} + \frac{T_{par}}{p} + O(p)$$

Si se evalúa la eficiencia para varios procesadores se observa, en general que decrece al aumentar el número de procesadores.

Eficiencias de Linpack para Cray Y-MP

p	s	e
1	-	-
2	1.95	0.975
3	2.88	0.960
4	3.76	0.940
8	6.96	0.870

La pregunta que se pretende responder es: ¿La caída de la eficiencia se debe a falta de paralelismo en el problema/algoritmo (valor elevado de T_{seq}), o a un crecimiento importante del overhead (valor creciente de $O(p)$) ?.

Figuras de mérito

□ La Fracción Serie

Definición:

$$f_p = \frac{T_{seq} + O(p)}{T_1}$$

Midiendo f_p para diferentes valores de p es posible determinar la evolución del overhead al crecer p . Si f_p permanece constante entonces el overhead es nulo y, por tanto, la pérdida de eficiencia se debe a una falta de paralelismo del problema/algorithm.

Medición

La fracción serie puede medirse de forma experimental. Supongamos que $O(p) = 0$ (caso ideal).

$$f_p = \frac{T_{seq}}{T_1} \Rightarrow T_p = T_1 f_p + \frac{T_1(1 - f_p)}{p} \Rightarrow$$

$$\frac{1}{s} = f_p + \frac{1 - f_p}{p} \Rightarrow$$

$$f_p = \frac{1/s - 1/p}{1 - 1/p}$$

Figuras de mérito

□ La Fracción Serie

Conclusión

Si f_p permanece constante entonces fue correcto suponer que $O(p) = 0$ y, por tanto, la caída de la eficiencia se deba a falta de paralelismo.

Si f_p crece con p entonces no es verdad que $O(p) = 0$. El overhead crece con el número de procesadores.

Ejemplos

Eficiencias de Linpack para Cray Y-MP

p	s	e	f_p
1	-	-	-
2	1.95	0.975	0.024
3	2.88	0.960	0.021
4	3.76	0.940	0.021
8	6.96	0.870	0.021

En este ejemplo, la pérdida de eficiencia se debe a falta de paralelismo del problema/algorithm.

Figuras de mérito

□ La Fracción Serie

Ejemplos

Eficiencias de Linpack para Alliant FX/40

p	s	e	f_p
1	-	-	-
2	1.90	0.950	0.053
3	2.65	0.883	0.066
4	3.22	0.805	0.080

En este caso, el overhead crece con el número de procesadores.

Uno de los ganadores del Bell Award

p	s	e	f_p
4	3.95	0.9885	0.0039
16	15.46	0.9663	0.0023
64	57.46	0.8978	0.0018
256	177.5	0.6934	0.0017
1024	351.2	0.3430	0.0019

Se hizo un buen trabajo de paralelización (valor pequeño de T_{seq}).

Figuras de mérito

□ Escalabilidad

Definición

Se dice que un sistema (arquitectura + algoritmo) es escalable si es capaz de usar de forma satisfactoria un número creciente de procesadores.

Existen muchas propuestas de métodos para determinar si un sistema es escalable o no. Cada método toma como punto de partida una definición más precisa de “uso satisfactorio de un número creciente de procesadores”.

Un ejemplo: Isoeficiencia

Un sistema es escalable si, al aumentar el número de procesadores es posible mantener la eficiencia constante, aumentando para ello el tamaño del problema (número de operaciones a realizar) en la medida en que sea necesario.

La escalabilidad se caracteriza mediante la función $F(p)$ que indica el tamaño de problema necesario para mantener la eficiencia constante con p procesadores.



Ejemplo: OCEAN en paso de mensajes

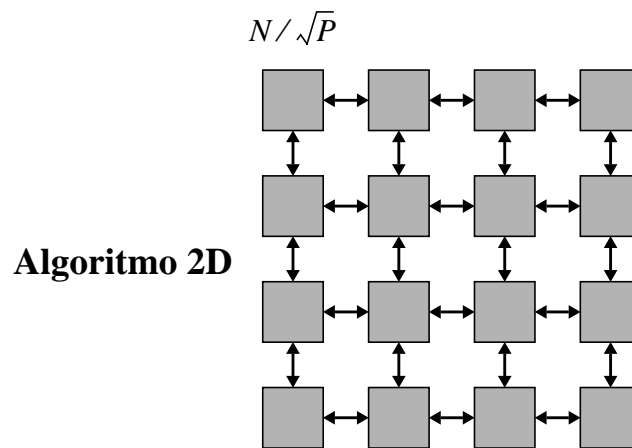
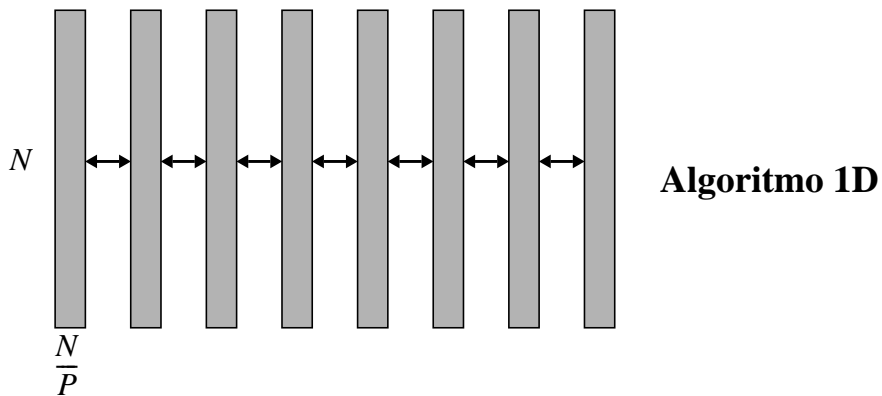
❑ Algoritmo secuencial

$$T_1(N) = 6N^2T_c$$

N es la dimensión de la matriz

T_c es el tiempo de producto o suma

❑ Algoritmos paralelos



Asumimos un multicomputador con store-and-forward

Ejemplo: OCEAN en paso de mensajes

□ Modelo del multicomputador

- ✓ Coste de operación aritmética: T_c (incluye el acceso a los datos)
- ✓ Comunicación store-and-forward y all-port
- ✓ Coste de enviar k mensajes de N números cada uno por enlaces diferentes:

$$k \times T_s + N \times T_e$$

Para los ejemplos supondremos:

T_c	1
T_e	1
T_s	200

Ejemplo: OCEAN en paso de mensajes

□ Modelo para el algoritmo 1D

Intercambio de las columnas de las fronteras

$$2T_s + NT_e$$

Actualización de la submatriz local

$$6\frac{N^2}{p}T_c$$

Determinación de la convergencia

$$\frac{p}{2}(2T_s + T_e + 2T_c)$$

Ejemplo: OCEAN en paso de mensajes

□ **Análisis del algoritmo 1D**

Fixed-problem size

Fixed-time speed-up

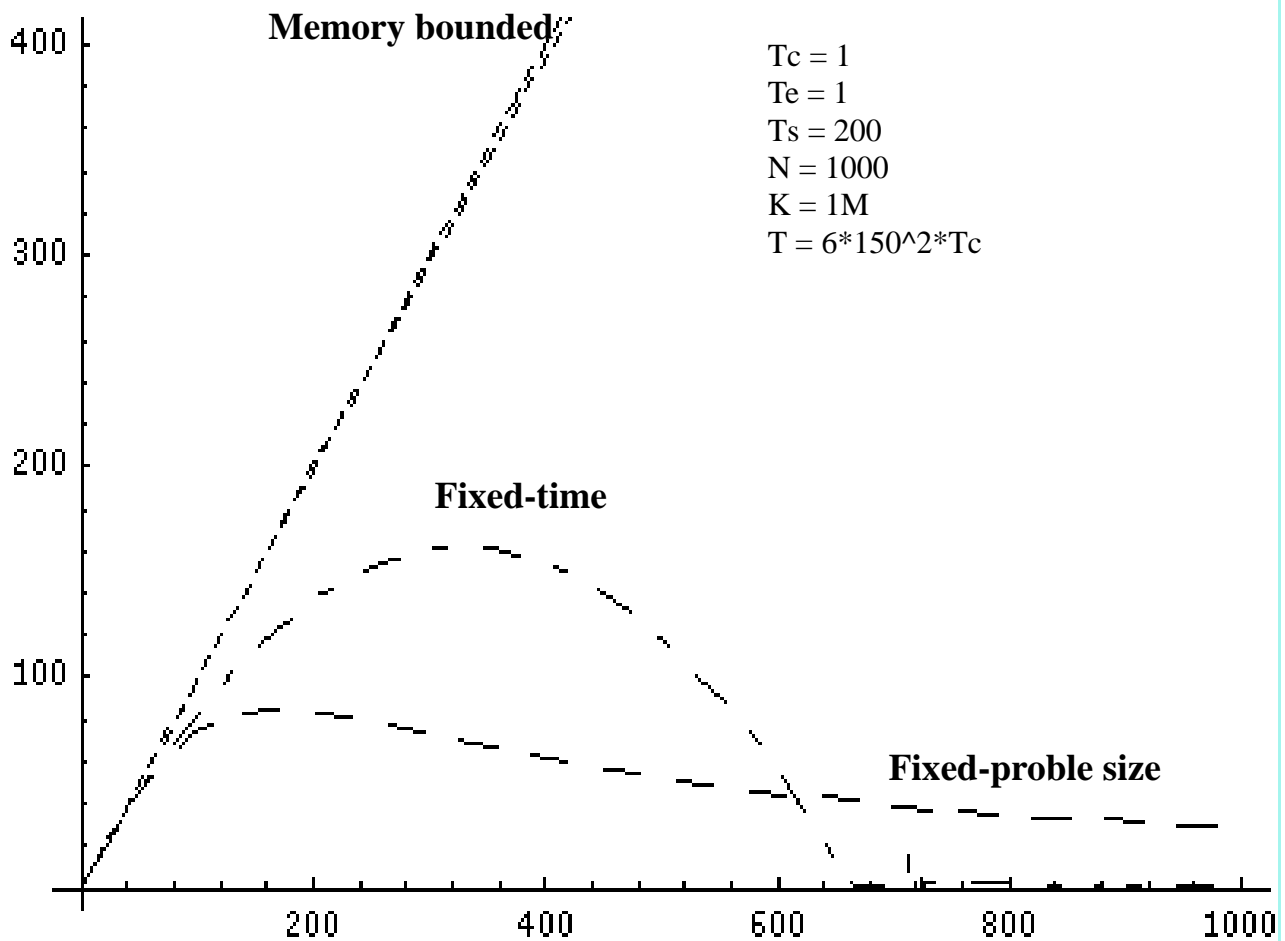
Memory bounded speed-up

Escalabilidad



Ejemplo: OCEAN en paso de mensajes

□ Speed-up para el algoritmo 1D



Ejemplo: OCEAN en paso de mensajes

□ Modelo para el algoritmo 2D

Intercambio de las columnas de las fronteras

$$4T_s + \frac{N}{\sqrt{p}}T_e$$

Actualización de la submatriz local

$$6\frac{N^2}{p}T_c$$

Determinación de la convergencia

$$\sqrt{p}(2T_s + T_e + 2T_c)$$

Ejemplo: OCEAN en paso de mensajes

□ **Análisis del algoritmo 2D**

Fixed-problem size

Fixed-time speed-up

Memory bounded speed-up

Escalabilidad



Ejemplo: OCEAN en paso de mensajes

□ Speed-up para el algoritmo 2D

