PERFORMANCE

Copyright © 2010, Elsevier
Inc. All rights Reserved

Speedup

- Número de núcleos = p
- Tempo de execução serial = T_{serial}
- Tempo de execução paralelo = $T_{parallel}$

speedup linear

$$T_{parallel} = T_{serial} / p$$

Copyright © 2010, Elsevier Inc. All rights Reserved

Speedup de um program paralelo

$$S = \frac{T_{serial}}{T_{parallel}}$$

Copyright © 2010, Elsevier Inc. All rights Reserved

Eficiência de um programa paralelo

$$E = \frac{S}{p} = \frac{\left(\frac{T_{serial}}{T_{parallel}}\right)}{p} = \frac{T_{serial}}{p \bullet T_{parallel}}$$

Copyright © 2010, Elsevier Inc. All rights Reserved

Speedups e eficiências de um programa a paralelo

p	1	2	4	8	16
S	1.0	1.9	3.6	6.5	10.8
$E = S/p$	1.0	0.95	0.90	0.81	0.68

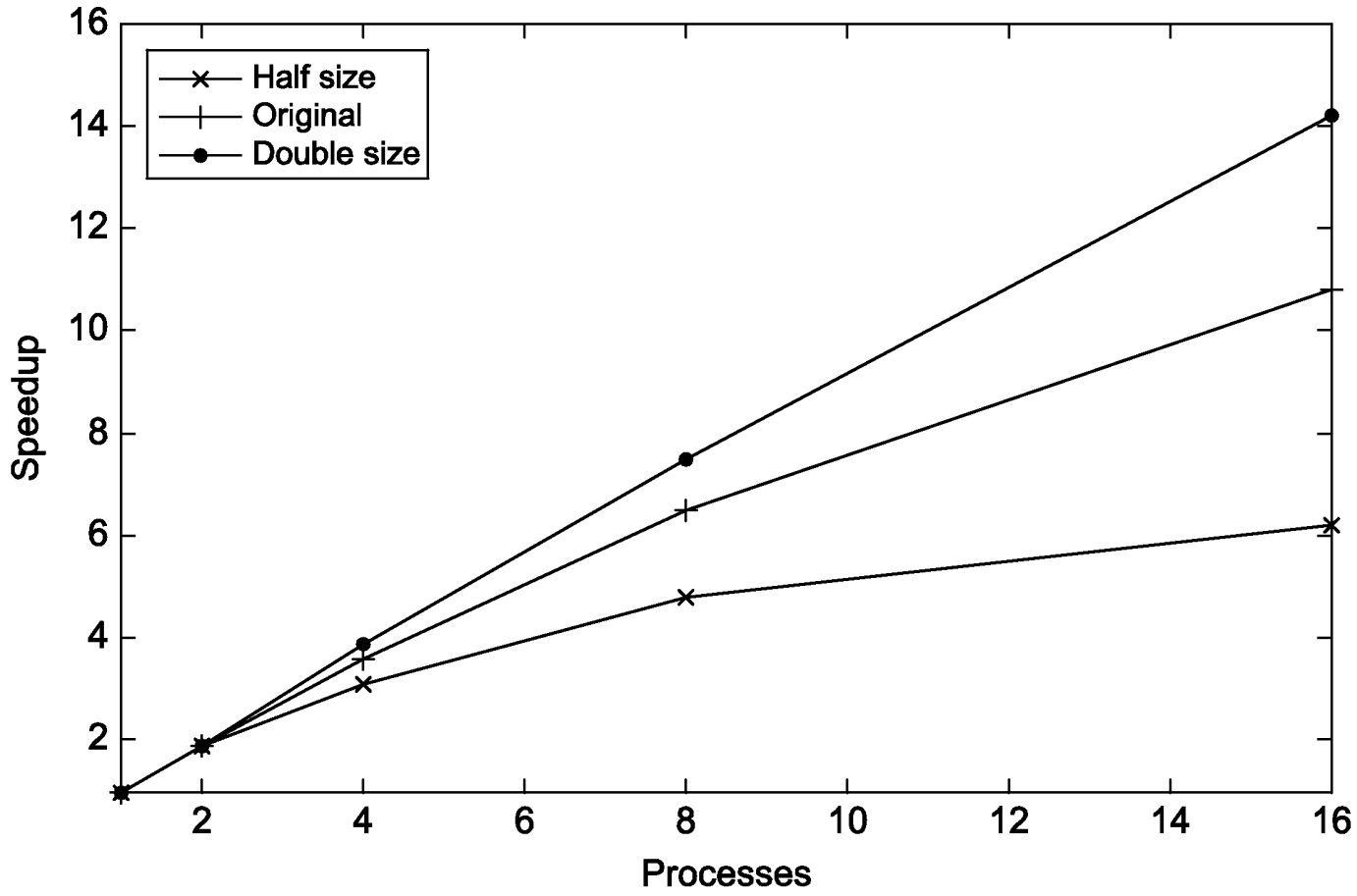
Copyright © 2010, Elsevier Inc. All rights Reserved

Variação de speedups e eficiências de um programa paralelo com o tamanho do programa

	p	1	2	4	8	16
Half	S	1.0	1.9	3.1	4.8	6.2
	E	1.0	0.95	0.78	0.60	0.39
Original	S	1.0	1.9	3.6	6.5	10.8
	E	1.0	0.95	0.90	0.81	0.68
Double	S	1.0	1.9	3.9	7.5	14.2
	E	1.0	0.95	0.98	0.94	0.89

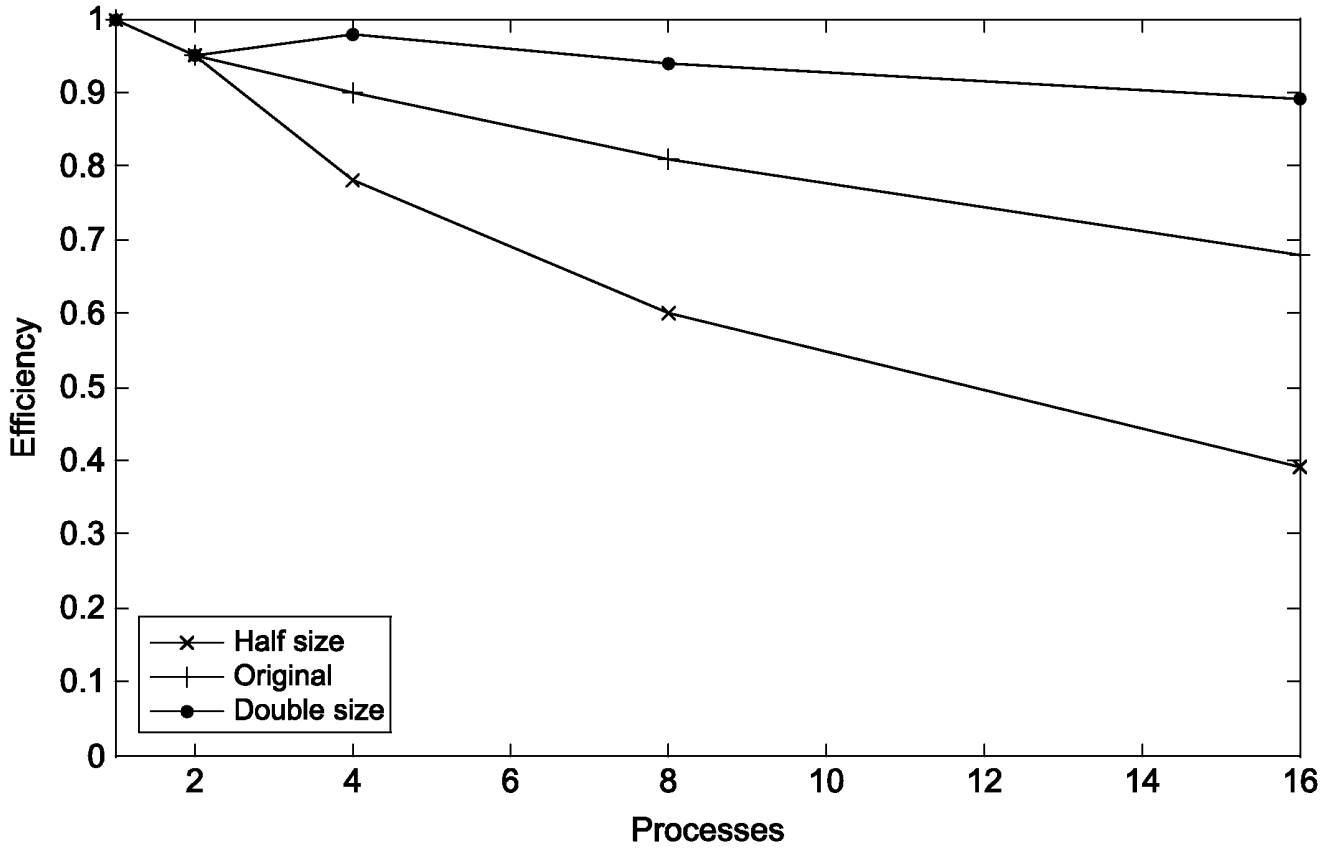
Copyright © 2010, Elsevier Inc. All rights Reserved

Speedup



Copyright © 2010, Elsevier Inc. All rights Reserved

Efficiency



Copyright © 2010, Elsevier Inc. All rights Reserved

Efeito do overhead

$$T_{parallel} = T_{serial} / p + T_{overhead}$$

Copyright © 2010, Elsevier Inc. All rights Reserved

Lei de Amdahl

- Ao não ser que todo um programa serial possa ser paralelizado, o speedup possível será bem limitado – independente do número de núcleos disponíveis.

Copyright © 2010, Elsevier Inc. All rights Reserved

Exemplo

- Podemos paralelizar 90% de um programa serial.
- A paralelização é “perfeita” independente do número *p* de núcleos que usarmos.
- Assuma um programa em que $T_{serial} = 20$ seconds

Copyright © 2010, Elsevier Inc. All rights Reserved

Exemplo (cont.)

- Tempo de execução da parte não “paralelizável” é

$$0.1 \times T_{serial} = 2$$

- Tempo de execução da parte paralelizável é:

$$0.9 \times T_{serial} / p = 18 / p$$

- Tempo de execução total é:

$$T_{parallel} = 0.9 \times T_{serial} / p + 0.1 \times T_{serial} = 18 / p + 2$$

Copyright © 2010, Elsevier Inc. All rights Reserved

Exemplo (cont.)

- Speed up

$$S = \frac{T_{serial}}{0.9 \times T_{serial} / p + 0.1 \times T_{serial}} = \frac{20}{18 / p + 2}$$

Copyright © 2010, Elsevier Inc. All rights Reserved

Fator de Speedup

$$S(p) = \frac{\text{Tempo de execução com um processador (melhor algoritmo sequencial)}}{\text{Tempo de execução com um multiprocessador com } p \text{ processadores}} = \frac{t_s}{t_p}$$

Onde t_s é o tempo de execução em um único processador e t_p é o tempo de execução em um multiprocessador.

$S(p)$ dá o aumento, em velocidade, utilizando-se um multiprocessador.

Tipicamente utiliza-se o melhor algoritmo sequencial com um único processador. O algoritmo adjacente para a implementação paralela pode ser (e usualmente é) diferente.

Fator de Speedup

O fator de speedup pode ser também deduzido em termos de passos computacionais:

$$S(p) = \frac{\text{Número de passos computacionais com um processador}}{\text{Número de passos computacionais paralelos com } p \text{ processadores}}$$

Pode também extender a complexidade de tempo para computações paralelas.

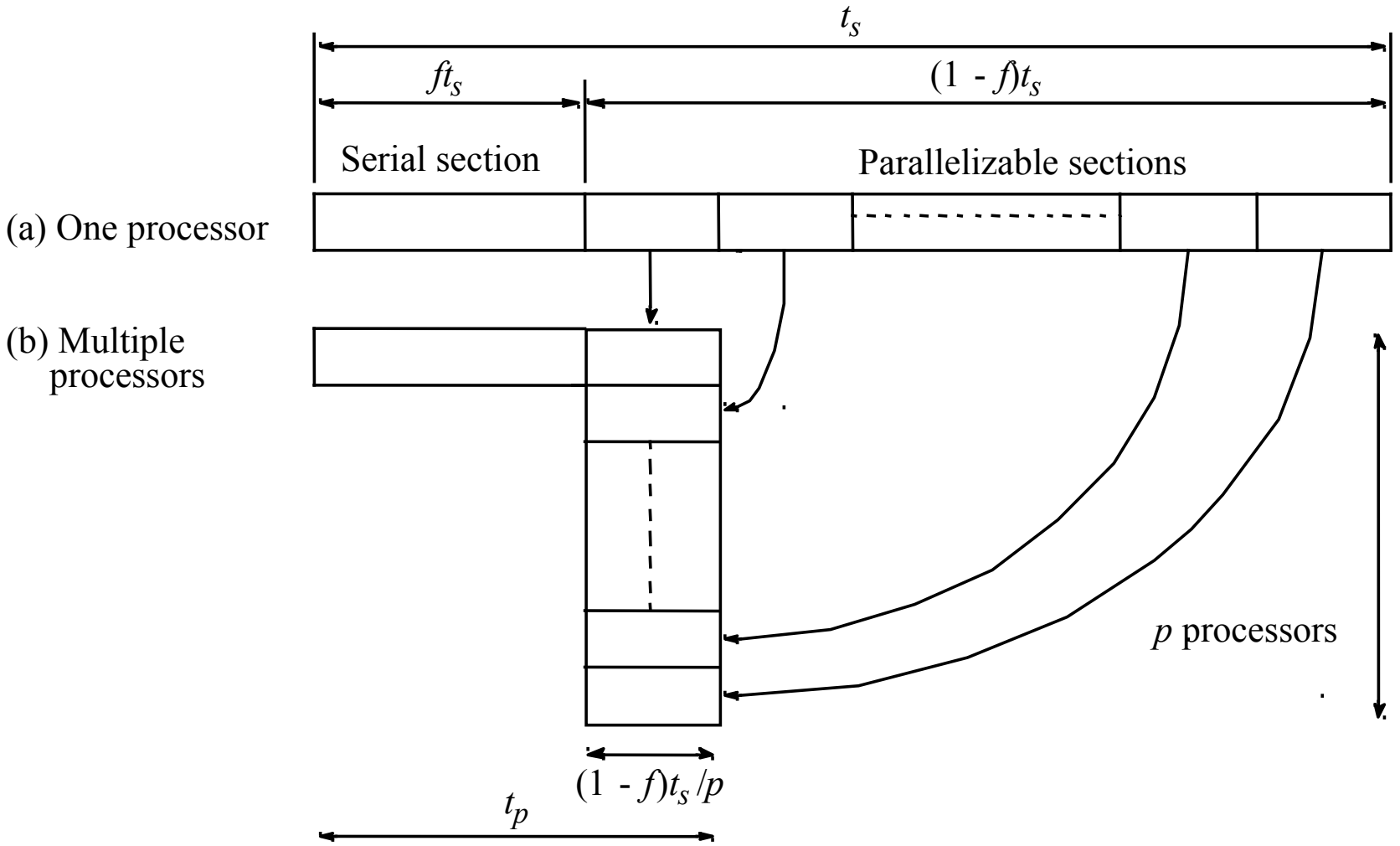
Speedup máximo

O speedup máximo é usualmente *p* com *p* processadores (speedup linear).

É possível se conseguir um speedup superlinear (maior do que *p*) mas usualmente devido a razões específicas, tais como:

- Memória extra no sistema multiprocessado
- Algoritmo não-determinístico

Speedup Máximo
Lei de Amdahl

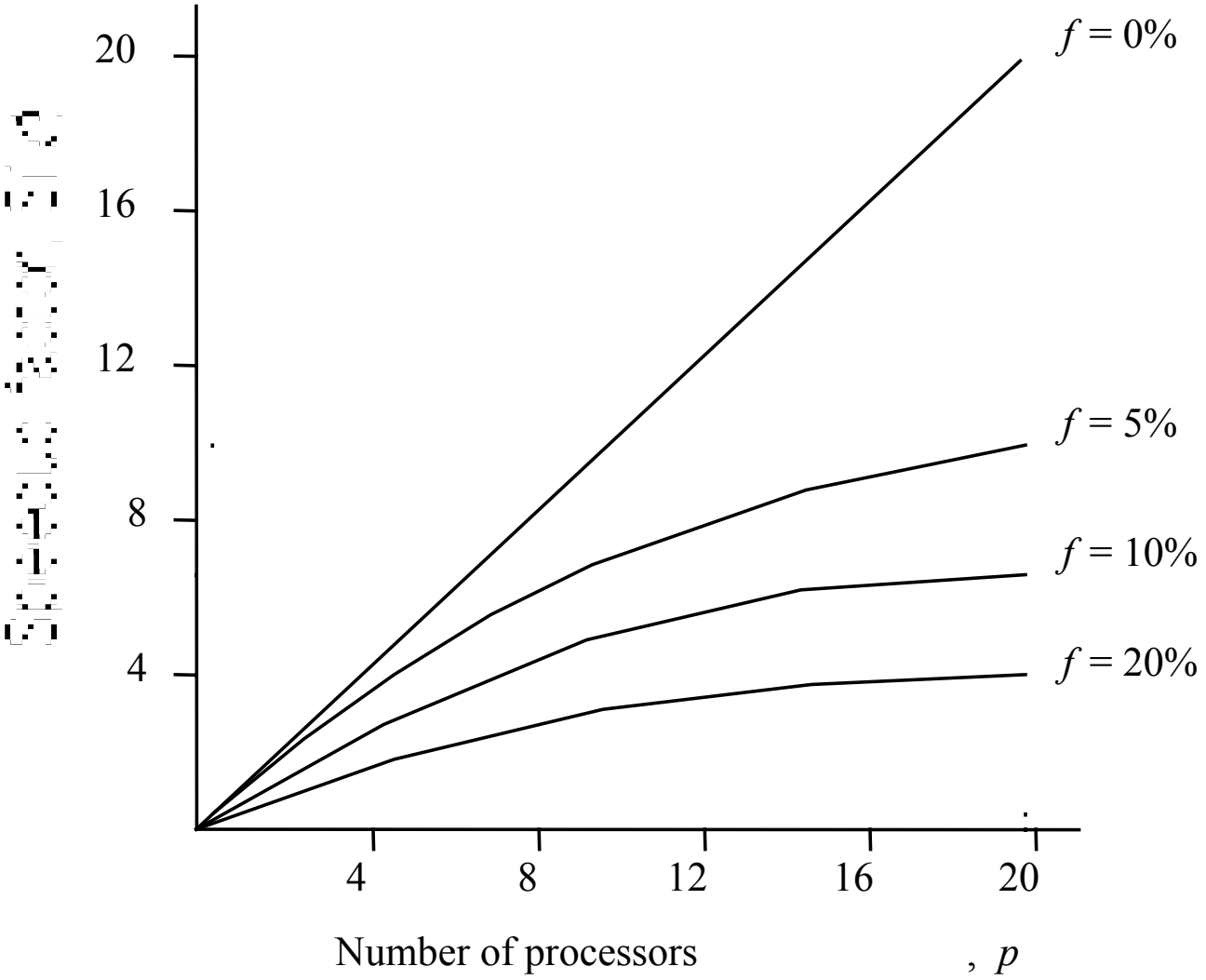


Fator de Speedup é dado por:

$$S(p) = \frac{t_s}{f t_s + (1-f) t_s / p} = \frac{p}{1 + (p-1) f}$$

Esta equação é conhecida como Lei de Amdahl

Speedup contra número de processadores



Mesmo com número infinito de processadores, o speedup máximo é limitado a 1/*f*.

Exemplo

Com somente 5% da computação sendo serial, o speedup máximo é 20, independentemente do número de processadores.

Isto é um resultado desencorajador.

Amdahl usou este argumento para suportar o projeto de sistemas de processadores únicos de ultra-alta velocidade nos anos 60.

Lei de Gustafson

Mais tarde, Gustafson (1988) descreveu como a conclusão da lei de Amdahl pode ser superada considerando-se o efeito de se aumentar o tamanho do problema.

Ele argumentou que quando um problema é portado para um sistema multiprocessado, tamanhos maiores de problema podem ser considerados, ou seja, o mesmo problema, mas com um número maior de valores de dados.

Lei de Gustafson

O ponto de partida para a lei de Gustafson é a computação em sistemas multiprocessados ao invés de em um único computador.

Na análise de Gustafson, o tempo de execução paralela é mantido constante, que assume-se ser um tempo aceitável para se obter a solução

Lei de Gustafson

Computação paralela composta de fração computada sequencialmente, p.explo, *f* ' e uma fração que contem partes paralelas,1 – *f* '.

O *fator de speedup escalado* de Gustafson é dado por:

$$S'(p) = \frac{f' t_p + (1 - f')pt_p}{t_p} = p + (1 - p)f'$$

f ' é a fração da computação no multiprocessador que não pode ser paralelizada. t_p é considerado como constante.

f ' é diferente do *f* anterior, que é a fração da computação em um único computador que não pode ser paralelizada.

Conclusão – aumento quase linear no speedup com o aumento do número de processadores, mas a parte fracional *f'* sequencial precisa diminuir com o aumento do tamanho do problema.

Lei de Gustafson

Por exemplo, se *f* é 5%, o speedup escalado resulta em 19.05 com 20 processadores enquanto que com a Lei de Amdahl, com *f* = 5% o speedup resulta em 10.26.

Gustafson reporta resultados obtidos na prática de speedups muito altos, perto do linear, em um sistema de 1024 processadores (arquitetura de hipercubo)

Escalabilidade

- No geral, um problema é *escalável* se ele pode lidar com tamanhos de problema sempre crescentes.
- Se aumentamos o número de processos/threads, sem aumentar o tamanho do problema e a eficiência permanece fixa então dizemos que o problema é *fortemente escalável*.
- Se aumentando tamanho do problema à medida que aumentamos o número de processos/threads a eficiência permanece fixa, então problema é *fracamente escalável*.

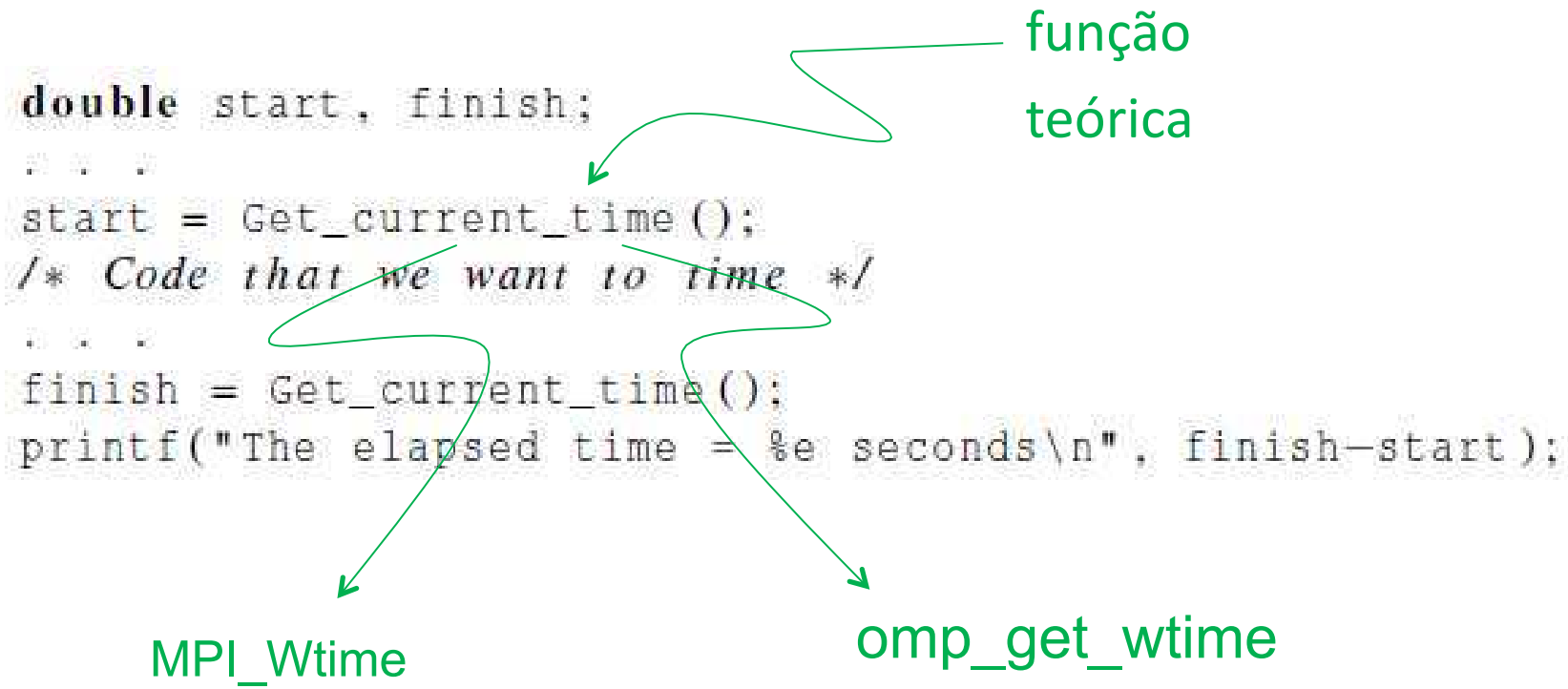
Copyright © 2010, Elsevier Inc. All rights Reserved

Medindo tempos

- Qual é o tempo?
- Do começo até o final?
- Qual o trecho de interesse do programa?
- Usar tempo de CPU?
- Usar o tempo do relógio?

Copyright © 2010, Elsevier
Inc. All rights Reserved

Medindo tempos



Copyright © 2010, Elsevier Inc. All rights Reserved

Taking Timings

```
private double start, finish;
. . .
start = Get_current_time();
/* Code that we want to time */
. . .
finish = Get_current_time();
printf("The elapsed time = %e seconds\n", finish-start);
```

Copyright © 2010, Elsevier Inc. All rights Reserved

Medindo tempos

```
shared double global_elapsed;
private double my_start, my_finish, my_elapsed;
. . .
/* Synchronize all processes/threads */
Barrier();
my_start = Get_current_time();

/* Code that we want to time */
. . .

my_finish = Get_current_time();
my_elapsed = my_finish - my_start;

/* Find the max across all processes/threads */
global_elapsed = Global_max(my_elapsed);
if (my_rank == 0)
   printf("The elapsed time = %e seconds\n", global_elapsed);
```

Copyright © 2010, Elsevier Inc. All rights Reserved

PROJETO DE PROGRAMAS PARALELOS

Copyright © 2010, Elsevier Inc. All rights Reserved

Metodologia de Foster

1. Particionamento
2. Comunicação
3. Agregamção
4. Mapeamento

Copyright © 2010, Elsevier
Inc. All rights Reserved

Metodologia de Foster

1. Particionando: divida a computação a ser realizada e os dados a serem operados em tarefas menores.

   O foco deve ser em executar tarefas que podem ser executadas em paralelo.

Copyright © 2010, Elsevier Inc. All rights Reserved

Metodologia de Foster

2. Communicação: determinar que comunicação precisa ser executada entre as tarefas identificadas nos passos anteriores.

Copyright © 2010, Elsevier Inc. All rights Reserved

Metodologia de Foster

3. Aglomeração ou agregação: combine tarefas e comunicações identificada no primeiro passo em tarefas maiores.

   Por exemplo, se a tarefa A deve ser executada antes que a tarefa B possa ser executada, faz sentido agregá-las em uma única tarefa composta.

Copyright © 2010, Elsevier Inc. All rights Reserved

Metodologia de Foster

4. Mapeamento: atribua a tarefa composta identificada na etapa anterior aos processos/ threads.

   Isto deve ser feito de modo a minimizar a comunicação, e cada processo/thread recebe aproximadamente o mesmo volume de trabalho.

Copyright © 2010, Elsevier Inc. All rights Reserved