Computação Paralela e Distribuída

Ano lectivo 2022-23

Rascunho - Será reorganizado e actualizado

Tema#04
Programação de Sistemas de Memória Distribuída

João José da Costa
joao.costa@isptec.co.ao

Março de 2023

Coordenação de Engenharia Informática
Departamento de Engenharias e Tecnologias
Instituto Superior Politécnico de Tecnologias e Ciências

Análise de Desempenho

Análise de Desempenho de Programa Paralelo

Objectivos

Instrutivos

- Analisar as leis e as métricas de desempenho.
- Prever o desempenho de programas paralelo
- Entender os limites para o desempenho

Educativo

- Sentir a necessidade de analisar o desempenho de programa paralelo para melhor desempenho.

Tópicos

- Análise de desempenho
- Speedup e eficiência
- Lei de Amdahl
- Lei de Gustafson-Barsis
- Métrica de Karp-Flatt
- Métrica de Isoeficiência

Desempenho

Lei de Amdahl, Lei de Gustafson-Barsis e Métrica de Karp-Flatt

Objectivos

- Prever o desempenho de programa paralelo.
- Entender as barreiras para o alto-desempenho.

Speedup

Speedup – medida de quão rápido é a execução de um programa paralelo versos um programa sequencial.

$$Speedup = \frac{Tempo\ de\ execução\ sequencial}{Tempo\ de\ execução\ paralelo}$$

Speedup: $\psi(n, p)$

n: tamanho do problema
p: número de tarefas

Componentes

Componentes do tempo de execução

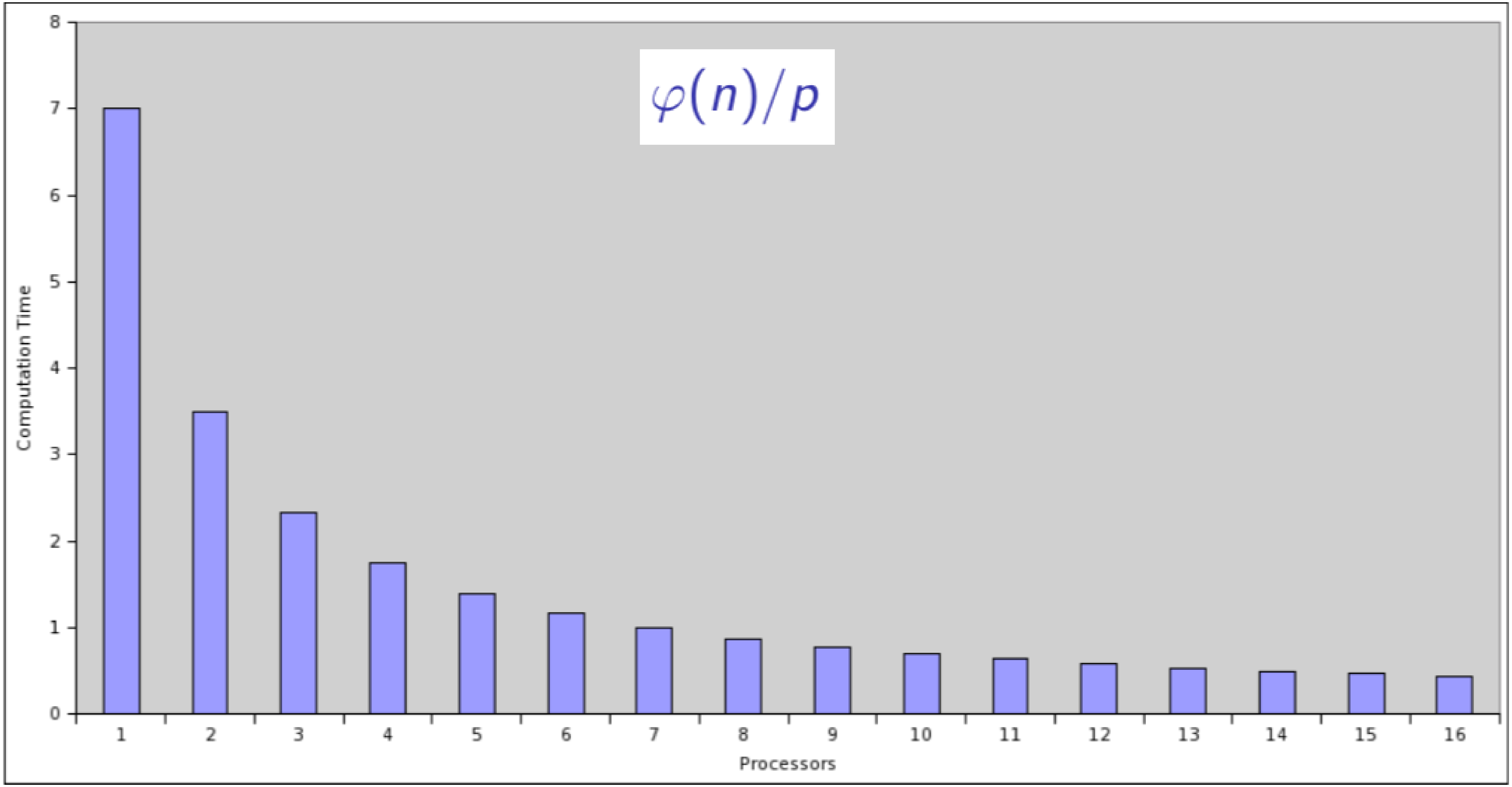
$\sigma(n)$: computações inerentemente sequencial

$\varphi(n)$: computações paralelizável completamente

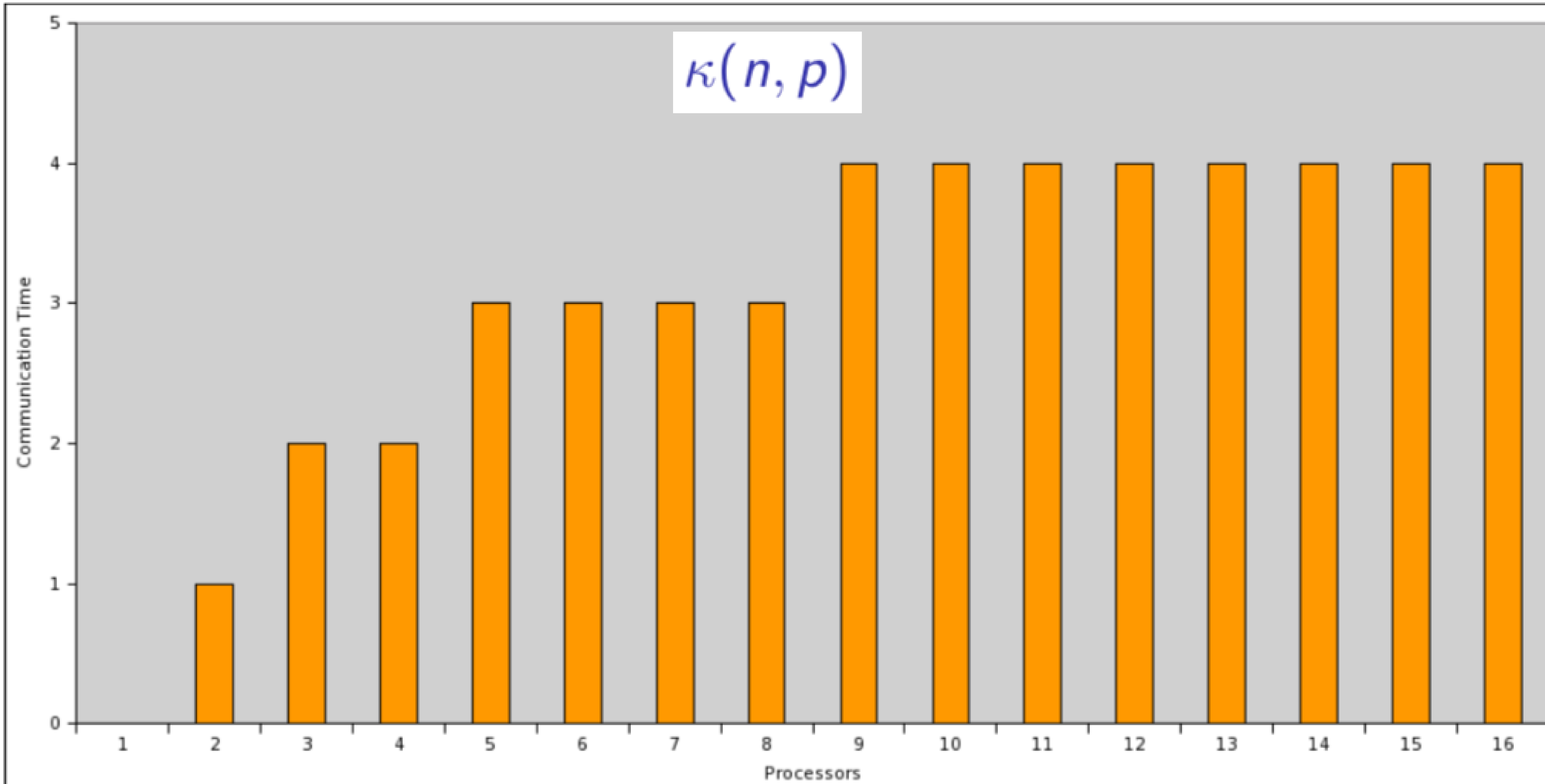
$\kappa(n)$: comunicação / sincronização / operações redundantes

$$\psi(n,p) \leq \frac{\sigma(n) + \varphi(n)}{\sigma(n) + \varphi(n)/p + \kappa(n,p)}$$

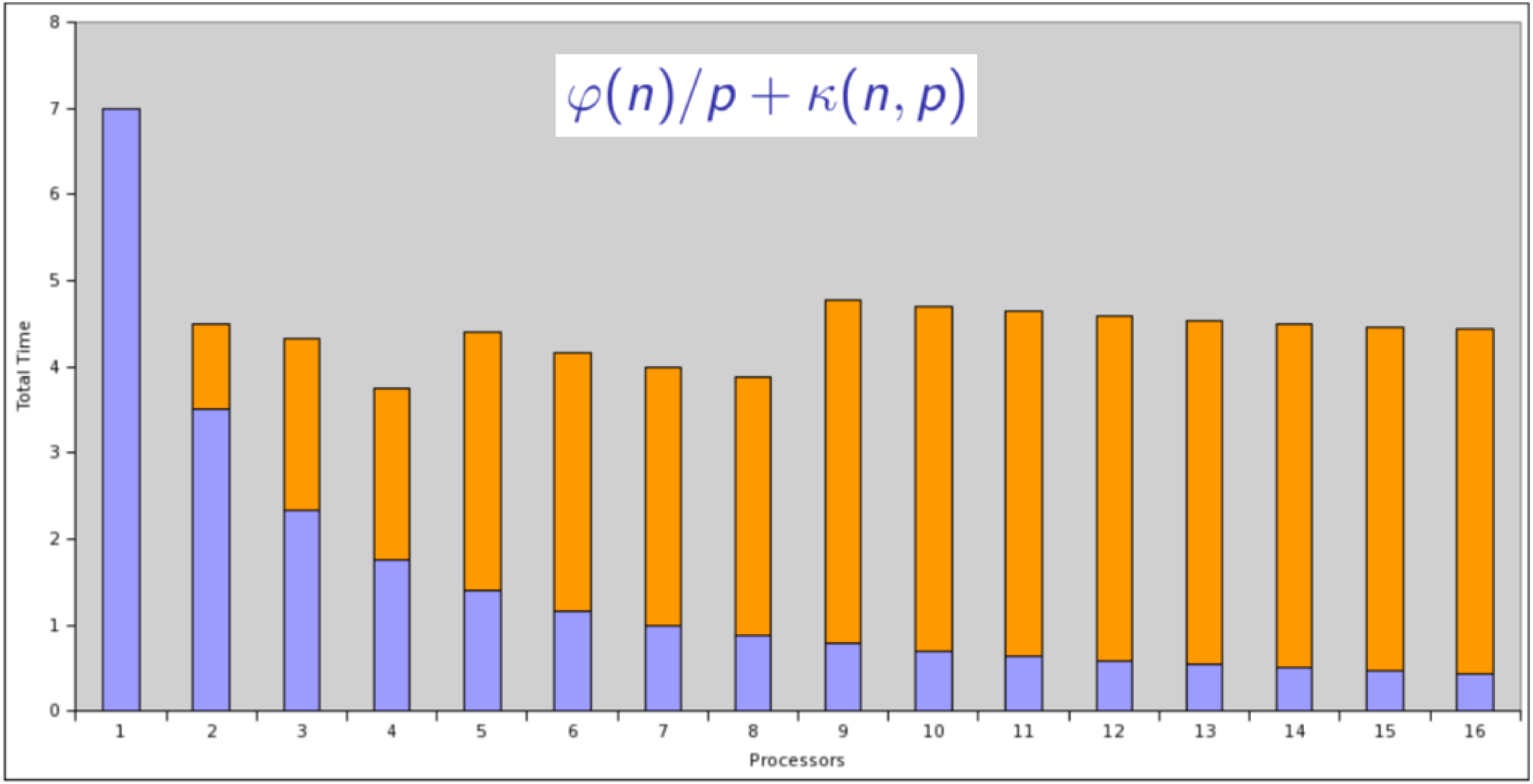
Tempo de computação



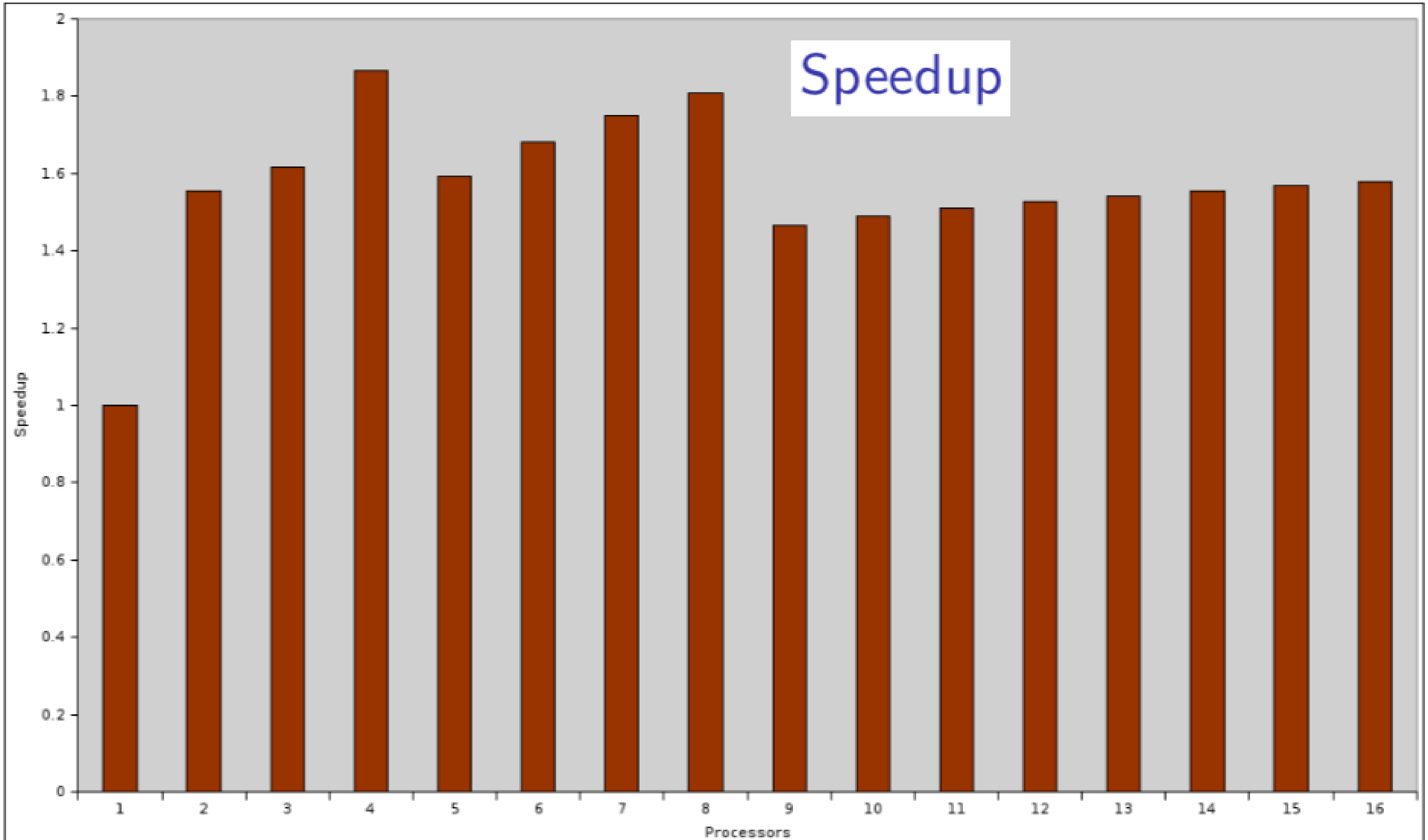
Tempo de comunicação



Tempo de total



Speedup


Speedup
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Processors

Eficiência

Eficiência – medida de utilização de processadores disponíveis.

$$Eficiência = \frac{Tempo\ de\ execução\ sequencial}{Processdores\ usados\ \times\ Tempo\ de\ execução\ paralelo} = \frac{Speedup}{Processadores\ usados}$$

Eficiência: $\varepsilon(n,p)$

n: tamanho do problema
p: número de tarefas

$$\varepsilon(n,p) \leq \frac{\sigma(n) + \varphi(n)}{p\sigma(n) + \varphi(n) + p\kappa(n,p)}$$

$$0 \leq \varepsilon(n,p) \leq 1$$

Lei de Amdahl

Speedup:

$$\psi(n,p) \le \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p+\kappa(n,p)} \le \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p}$$

Seja f a fracção de computação sequencial em um programa sequencial:

$$f(n) = \frac{\sigma(n)}{\sigma(n)+\varphi(n)}$$

$$\boxed{\psi(n,p) \le \frac{1}{f(n)+\frac{1-f(n)}{p}}}$$

Problema

Um programa de animação por computador gera uma longa-metragem quadro a quadro. Cada quadro pode ser gerado independentemente e é enviado para seu próprio ficheiro. Se leva 99 segundos para renderizar um quadro e 1 segundo para exibi-lo, <u>quanto *speedup* pode ser alcançado</u> ao renderizar o filme em 100 processadores?

f(n) = ?????   p = ????

Problema

Um programa de animação por computador gera uma longa-metragem quadro a quadro. Cada quadro pode ser gerado independentemente e é enviado para seu próprio ficheiro. Se leva 99 segundos para renderizar um quadro e 1 segundo para exibi-lo, <u>quanto *speedup* pode ser alcançado</u> ao renderizar o filme em 100 processadores?

$f(n) = 0{,}01 \qquad p = 100$

$$\psi(n,p) \leq \frac{1}{0{,}01 + \frac{0{,}99}{100}} = 50{,}25 \approx 50{,}3$$

Lei de Amdahl

Limitações da lei de Amdahl:

- considera apenas 2 modos de execução.
- não leva em consideração o overhead paralelo, $\kappa(n, p)$

$\Longrightarrow$ Superestima a aceleração alcançável

Efeito de Amdahl

- normalmente, $\kappa(n, p)$ tem menor complexidade do que $\varphi(n)/p$.

- conforme n aumenta, $\varphi(n)/p$ domina $\kappa(n, p)$

- à medida que n aumenta, o speedup aumenta

Medida de optimização alternativa

Lei de Amdahl

- trata o tamanho do problema como uma constante
- mostra como o tempo de execução diminui à medida que o número de processadores aumenta

Uma perspectiva diferente:

- computadores mais rápidos resolvem instâncias de problemas maiores
- considere o tempo como uma constante e permita que o tamanho do problema aumente com o número de processadores

Lei de Gustafson-Barsis

Speeup:

$$\psi(n,p) \leq \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p+\kappa(n,p)} \leq \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p}$$

Seja s a fracção de computação sequencial no programa paralelo:

$$s = \frac{\sigma(n)}{\sigma(n)+\frac{\varphi(n)}{p}}$$

Lei de Gustafson-Barsis:

$$\psi(n,p) \leq p + (1-p)s$$

Prova a equivalência!!!

Lei de Gustafson-Barsis

- começa a partir do tempo de execução paralela;

- estima o tempo de execução sequencial para resolver o mesmo problema;

- tamanho do problema é uma função crescente de p;

- prevê aumento de speedup em escala

Problema

Uma aplicação executada em 10 processadores gasta 3% de seu tempo em código serial. Qual é o aumento de speedup escalado da aplicação?

Speedup escalado: ??????

Problema

Uma aplicação executada em 10 processadores gasta 3% de seu tempo em código serial. Qual é o aumento de velocidade escalado da aplicação?

Speedup escalado:

$$p + (1 - p)s = 9.7$$

Diferente visão, mesmo speedup

❖ Lei de Amdahl:
- Tamanho do problema fixo
- Fracção sequencial do programa serial
- Mede forte escalabilidade

❖ Lei de Gustafson-Barsis:
- Tempo de execução fixo
- Fracção serial do programa paralelo
- Mede escalabilidade fraca

❖ Ambos calculam o mesmo speedup
- (podemos derivar a fracção serial do programa serial da fração serial do programa paralelo e vice-versa)

Métrica de Karp-Flatt

- ❖ A Lei de Amdahl e a Lei de Gustafson-Barsis ignoram $\kappa(n, p)$
- ❖ superestimar o aumento de velocidade ou aumento de escala
- ❖ Karp e Flatt propuseram outra métrica

$\Longrightarrow$ fracção serial determinada experimentalmente

Fracção serial determinada experimentalmente

- Representa a fracção do programa original que não pode ser paralelizado em relação ao tempo de execução sequencial.

$$e = \frac{\sigma(n) + \kappa(n, p)}{\sigma(n) + \varphi(n)}$$

Métrica de Karp-Flatt

$$e = \frac{\sigma(n) + \kappa(n,p)}{\sigma(n) + \varphi(n)}$$

Tempo de execução de um programa paralelo em p processadores:

$$T(n,p) = \sigma(n) + \varphi(n)/p + \kappa(n,p)$$

$$T(n,1) = \sigma(n) + \varphi(n)$$

$$e = \frac{\sigma(n) + \kappa(n,p)}{T(n,1)}$$

$$e = \frac{1/\psi(n,p) - 1/p}{1 - 1/p}$$

Métrica de Karp-Flatt

- ❖ Leva em consideração a sobrecarga paralela

- ❖ Permitir a análise da fonte de ineficiência paralela

  - ➢ oportunidade limitada para paralelismo computacional

  - ➢ sobrecarga paralela (comunicação, sincronização, balanceamento de carga, etc)

Problema 1

p	2	3	4	5	6	7	8
ψ	1,8	2,5	3,1	3,6	4,0	4,4	4,7

Qual é a principal razão para a aceleração de apenas 4,7 em 8 CPUs?

Problema 1

p	2	3	4	5	6	7	8
ψ	1,8	2,5	3,1	3,6	4,0	4,4	4,7

Qual é a principal razão para a aceleração de apenas 4,7 em 8 CPUs?

p	2	3	4	5	6	7	8
e	0,1	0,1	0,1	0,1	0,1	0,1	0,1

Como e é constante, a **grande fracção serial** e a principal razão.

Problema 2

p	2	3	4	5	6	7	8
ψ	1,9	2,6	3,2	3,7	4,1	4,5	4,7

Qual é a principal razão para a aceleração de apenas 4,7 em 8 CPUs?

Problema 2

p	2	3	4	5	6	7	8
ψ	1,9	2,6	3,2	3,7	4,1	4,5	4,7

Qual é a principal razão para a aceleração de apenas 4,7 em 8 CPUs?

p	2	3	4	5	6	7	8
e	0,070	0,075	0,080	0,085	0,090	0,095	0,100

Como e está a aumentar constantemente, a sobrecarga é a principal razão.

Problema 3

p	4	8	12
ψ	3,9	6,5	?

É provável que este programa alcance uma aceleração de 10 em 12 processadores?

Problema 3

p	4	8	12
ψ	3,9	6,5	?

É provável que este programa alcance uma aceleração de 10 em 12 processadores?

p	4	8	12
e	0,009	0,033	0,018

e tipicamente aumenta com p. Speedup provavelmente mais próxima de 8 em 12 processadores.

Escalabilidade

Métrica de isoeficiência

Objectivo

- Avaliar a escalabilidade de um algoritmo paralelo em execução num sistema de computadores paralelos.

Métrica de Isoeficiência

- A escalabilidade de um sistema paralelo mede a capacidade de aumentar o desempenho à medida que o número de processadores aumenta.

(sistema paralelo: programa paralelo a executar em um computador paralelo)

- Um **sistema escalável** mantém a eficiência à medida que os processadores são adicionados.

- A isoeficiência é uma forma de medir a escalabilidade.

Métrica de Isoeficiência

Tempo de execução do programa paralelo em processadores p:

$$T(n,p) = \sigma(n) + \varphi(n)/p + \kappa(n,p)$$

Seja $T_0(n,p)$ o tempo gasto fazendo trabalho não feito pelo algoritmo sequencial:

$$T_0(n,p) = (p-1)\sigma(n) + p\kappa(n,p)$$

$$\begin{aligned}\psi(n,p) &\leq \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\varphi(n)/p+\kappa(n,p)} \\ &= \frac{p(\sigma(n)+\varphi(n))}{\sigma(n)+\varphi(n)+(p-1)\sigma(n)+p\kappa(n,p)} \\ &= \frac{p(\sigma(n)+\varphi(n))}{\sigma(n)+\varphi(n)+T_0(n,p)}\end{aligned}$$

Métrica de Isoeficiência

$$
\begin{aligned}
\varepsilon(n,p) &= \frac{\psi(n,p)}{p} \\
&\leq \frac{1}{1+\frac{T_0(n,p)}{\sigma(n)+\varphi(n)}} \\
&= \frac{1}{1+T_0(n,p)/T(n,1)} \\
\Rightarrow T(n,1) &\geq \frac{\varepsilon(n,p)}{1-\varepsilon(n,p)} T_0(n,p)
\end{aligned}
$$

Para manter a eficiência: constante $\frac{\varepsilon(n,p)}{1-\varepsilon(n,p)} = C$

Relação de Isoeficiência

Para manter o mesmo nível de eficiência à medida que o número de processadores p aumenta, n deve ser aumentado de modo que satisfaçamos:

$$T(n,1) \geq C T_0(n,p)$$

Função de escalabilidade

- Suponha que a relação de isoeficiência seja $n \geq f(p)$.
- Seja $M(n)$ a memória necessária para o problema de tamanho n.
- $M(f(p))/p$ indica como o uso de memória por processador deve aumentar para manter a mesma eficiência.
- $M(f(p))/p$ é chamada de função de escalabilidade.

Função de escalabilidade

Significado da função de escalabilidade

- Para manter a eficiência ao aumentar p, devemos aumentar n;

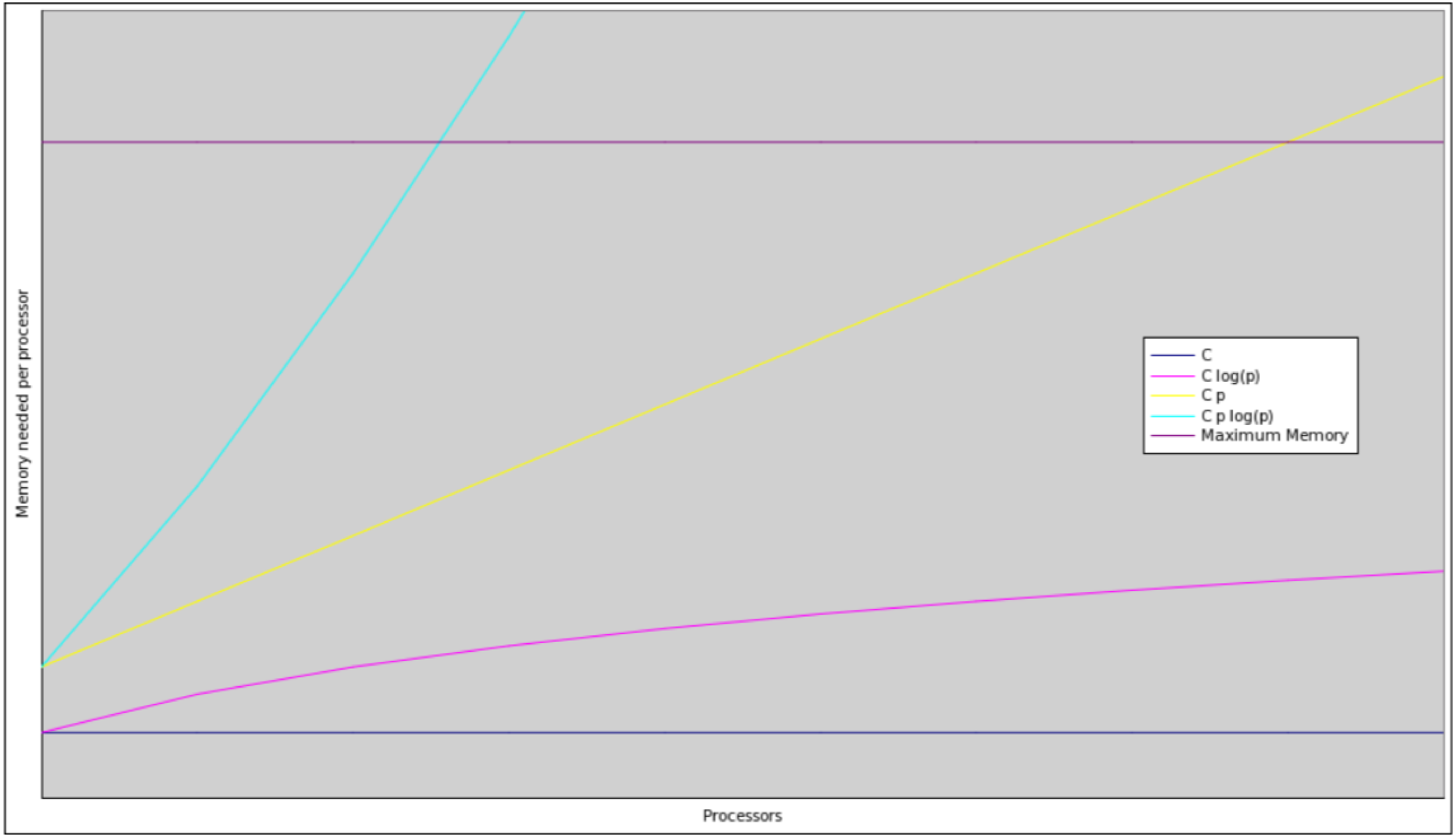
- Tamanho máximo do problema limitado pela memória disponível, que é linear em p;

- Função de escalabilidade mostra como o uso de memória por processador deve crescer para manter a eficiência;

- Função de escalabilidade uma constante significa que o sistema paralelo é perfeitamente escalável.

Função de escalabilidade

Interpretação da função de escalabilidade



Exemplo 1: Redução

Complexidade do algoritmo sequencial: $T(n, 1) = \Theta(n)$

Algoritmo paralelo:

- Complexidade computacional: $\Theta(n/p)$
- Complexidade da comunicação: $\Theta(\log p)$

Sobrecarga paralela: $T_0(n, p) = \Theta(p \log p)$

Exemplo 1: Redução

Relação de isoeficiência: $n \geq Cp \log p$

Para manter o mesmo nível de eficiência, como n deve aumentar quando p aumenta?

Exemplo 1: Redução

Relação de isoeficiência: $n \geq Cp \log p$

Para manter o mesmo nível de eficiência, como n deve aumentar quando p aumenta?

$M(n) = n$

Função de escalabilidade: $M(Cp \log p)/p = C \log p$

O sistema tem boa escalabilidade!

Exemplo 2

Algoritmo Floyd-Warshall

Complexidade do algoritmo sequencial: $T(n, 1) = \Theta(n^3)$

Algoritmo paralelo:

- Complexidade computacional: $\Theta(n^3/p)$
- Complexidade da comunicação: $\Theta(n^2 \log p)$

Sobrecarga paralela: $T_0(n, p) = \Theta(pn^2 \log p)$

Exemplo 2

Algoritmo Floyd-Warshall

Relação de isoeficiência:

$n^3 > Cpn^2 \log p \Longrightarrow n \geq Cp \log p$

Para manter o mesmo nível de eficiência, como n deve aumentar quando p aumenta?

Exemplo 2
Algoritmo Floyd-Warshall

Relação de isoeficiência:
$n^3 > Cpn^2 \log p \Longrightarrow n \geq Cp \log p$

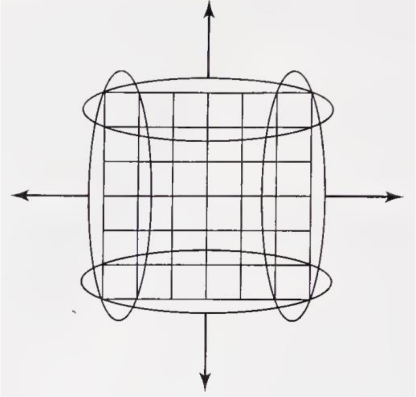
Para manter o mesmo nível de eficiência, como n deve aumentar quando p aumenta?

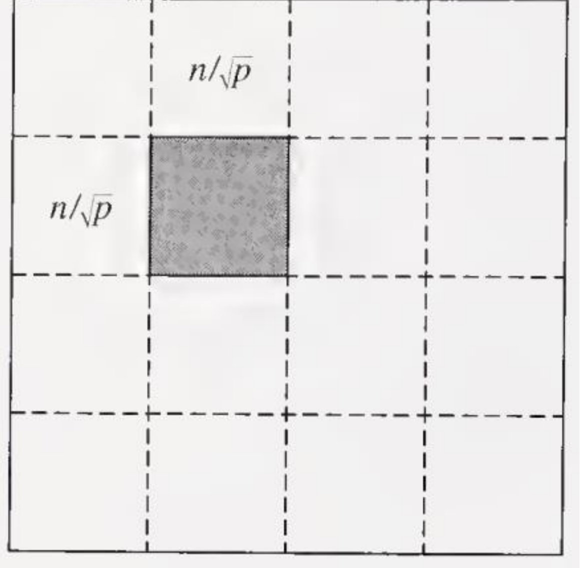
$M(n) = n^2$

Função de escalabilidade: $M(Cp \log p)/p = C^2 p \log^2 p$

O sistema tem baixa escalabilidade!

Exercício





Exercício para resolver na sala de aula

Considere um algoritmo paralelo que implementa um método de diferença finita para a resolução de equação diferencial parcial. O problema é representado por uma grelha $n \times n$. Cada processo é responsável por uma subgrelha de tamanho $(n/\sqrt{p}) \times (n/\sqrt{p})$. Durante cada iteração do algoritmo todos os processos enviam valores limite para seus quatro vizinhos; O tempo necessário para realizar essa comunicação é $\Theta(n/\sqrt{p})$ por iteração.

A complexidade de tempo do algoritmo serial é $\Theta(n^2)$ por iteração.

1. Determine a relação de isoeficiência para este sistema paralelo.
2. Determine a função de escalabilidade.

Revisão

- Análise de desempenho
- Speedup e eficiência
- Lei de Amdahl
- Lei de Gustafson-Barsis
- Métrica de Karp-Flatt
- Métrica de Isoeficiência

Bibliografia

- Consulte dentro da subpasta “references” no repositório da disciplina.