① 4ª Aula

Serial x Paralelo

- Algoritmo serial/sequencial: executa em 1 único processador.
- Algoritmo paralelo: executa em 2 ou mais processadores.
- Para todo algoritmo paralelo existe um algoritmo sequencial que realiza a mesma tarefa
- É importante entender a solução sequencial para desenvolver uma solução paralela

② Desempenho

- Desempenho de um algoritmo paralelo deve ser superior com relação a versão sequencial para o mesmo problema
- Desempenho = tempo de execução.
- Tempo total de execução = tempo de computação + tempo ocioso + tempo de comunicação.

③ Tempo de execução

- Tempo de computação: tempo de operações de E/S e cálculos do programa
- Tempo ocioso (ou tempo de sincronização): dedicado a sincronizar um algoritmo ou esperar pelos processos mais lentos
- Tempo de comunicação = latência + transmissão
  - latência: tempo de preparação da mensagem (pacotes)
  - tempo de transmissão: velocidade real da transmissão (largura de banda)

④ Projeto de algoritmos paralelos

2 enfoques comuns:

- Paralelismo de dados: os dados são particionados, cada subconjunto de dados é associado a um processo, cada processo executa os mesmos comandos sobre seu conjunto de dados
- Paralelismo de controle: o problema é dividido em tarefas (etapas) independentes, cada tarefa é associado com um processo, cada processo executa comandos diferentes

⑤ Paralelismo de dados

- Implementado mais facilmente que o paralelismo de controle
- Não é prejudicado por dependência entre operações.
- Fácil escalabilidade
- Requer pouca comunicação entre processos

⑥ Paralelismo de controle

- Difícil de implementar
- Preciso considerar dependência entre operações
- Dificulta escalabilidade
- Exige muita comunicação entre processos.

O paralelismo de dados é mais comum, porém a maioria dos algoritmos paralelos envolve as duas abordagens (dados e controle)

(7) Notação

p = número de processos (processadores) ou threads

n = tamanho do problema

→ Avaliação de algoritmos sequenciais utiliza análise assintótica (O, Ω)

→ Análise assintótica não costuma ser aplicada para caracterizar desempenho de algoritmos paralelos

(8) Granularidade

Granularidade $G(n)$ expressa a relação entre a quantidade de computação e a quantidade de comunicação em um algoritmo paralelo

$$G(n) = \frac{T_p}{T_c}$$

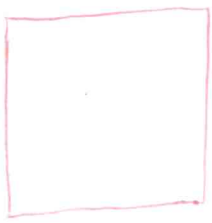
T_p : tempo de computação/processamento

T_c : tempo de comunicação

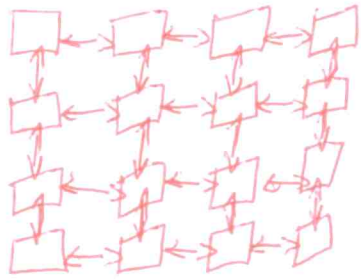
(9)

A granularidade de um problema é inversamente proporcional ao grau de paralelismo do problema

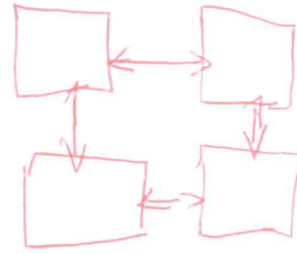
Nº de subproblemas	Computação	Comunicação	Granularidade	Paralelismo
Grande	Pouca	Alta	Fina	Alto
Baixo	Grande	Baixo	Grossa	Baixo



Problema



Granularidade fina



Granularidade grossa

(10) Speedup (Aceleração)

- $T(n)$: tempo de execução de um algoritmo sequencial, depende apenas do tamanho do problema
- $T(n,p)$: tempo de execução do algoritmo paralelo, depende do tamanho do problema e do número de processadores utilizados.
- $T(n,p)$ = tempo total de execução, desde quando o primeiro processador começa a execução até o último processador completar a execução da última instrução

(11)

- Speedup:

$$S(n,p) = \frac{T(n)}{T(n,p)}$$

- De forma geral $0 < S(n,p) \leq p$
- Quando $S(n,p) = p$ temos speedup linear
- Speedup linear é algo raro devido a sobrecarga da maioria das soluções paralelas

(12) Tipos de speedup

- $S(n,p) < 1$, slowdown, situação indesejável (sobrecarga de paralelização muito alta)
- $1 < S(n,p) < p$, sublinear, comportamento geral
- $S(n,p) = p$, linear, ideal (não existe sobrecarga)
- $S(n,p) > p$, superlinear, situação possível.

- Exemplo de Speedup superlinear: Algoritmo de busca

(13) Eficiência

- Eficiência $E(n,p)$ é a medida de utilização dos processos em um algoritmo paralelo em relação ao algoritmo sequencial

$$E(n,p) = \frac{S(n,p)}{p} = \frac{T(n)}{p \cdot T(n,p)}$$

- Tipos de eficiência
  - $E(n,p) < \frac{1}{p}$, slow down
  - $\frac{1}{p} < E(n,p) < 1$, sublinear
  - $E(n,p) = 1$, linear
  - $E(n,p) > 1$, supralinear

(14) Tomadas de tempo

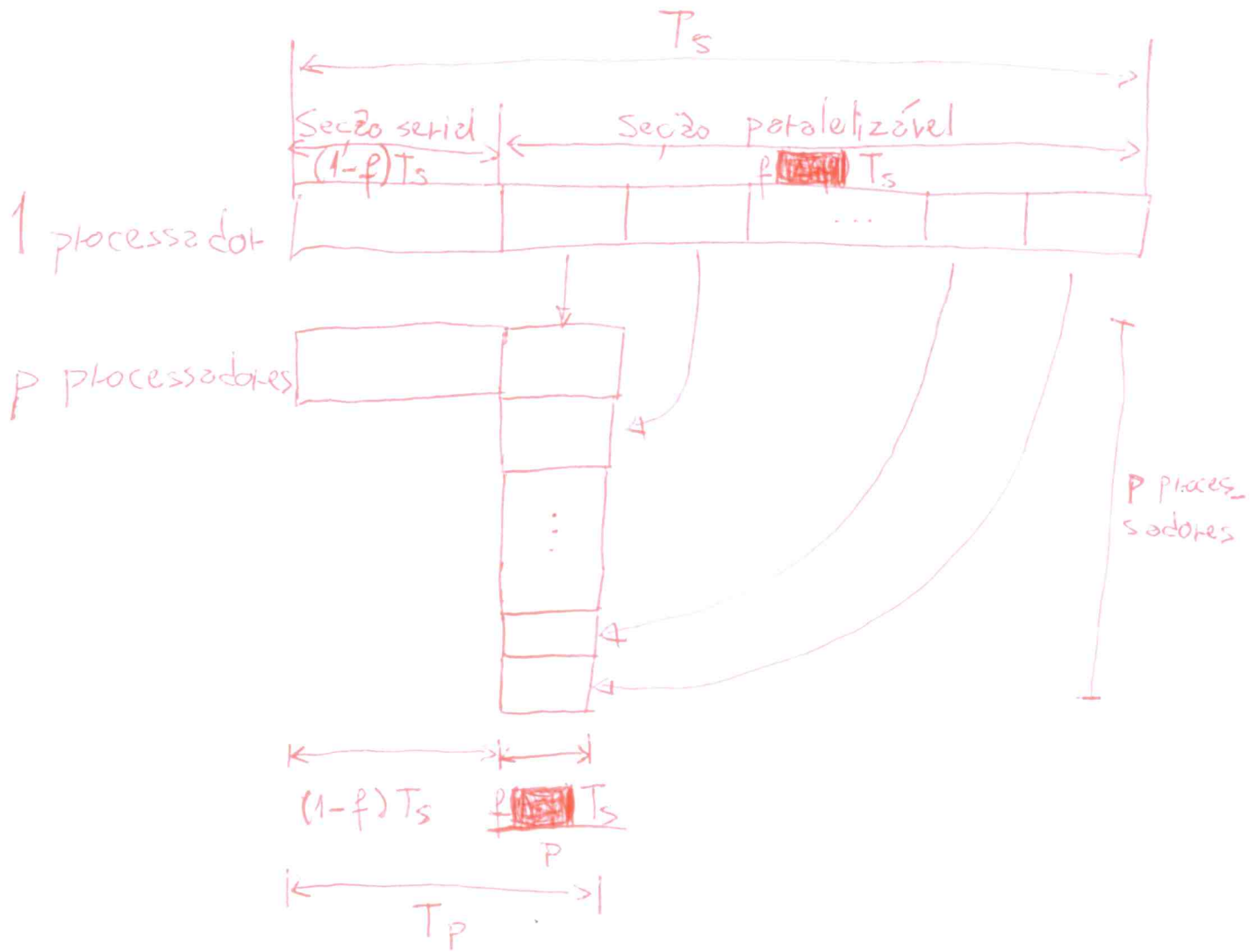
- Avaliação de desempenho de algoritmos/programas/implementações paralelas devem ser realizadas com tomadas de tempo
- Para se obter tomadas de tempo confiáveis deve-se considerar as etapas:
  (1) Garantir o máximo de exclusividade na utilização do processador
  (2) Sincronizar os processos no início do código
  (3) Efetuar tomada de tempo (start)
  (4) Sincronizar os processos no final do código
  (5) Efetuar tomada de tempo (finish)
  (6) Calcular o tempo transcorrido (finish - start)

(15) # Lei de Amdahl

- Formulada por Gene Amdahl em 1967
- Estabelece um limite superior para o speedup de um algoritmo paralelo
- T_s: tempo sequencial para resolver o problema
- Uma fração (1-f) de instruções é inerentemente serial/sequencial
- Uma fração f de instruções são perfeitamente paralelizáveis.

Trocar ordem!

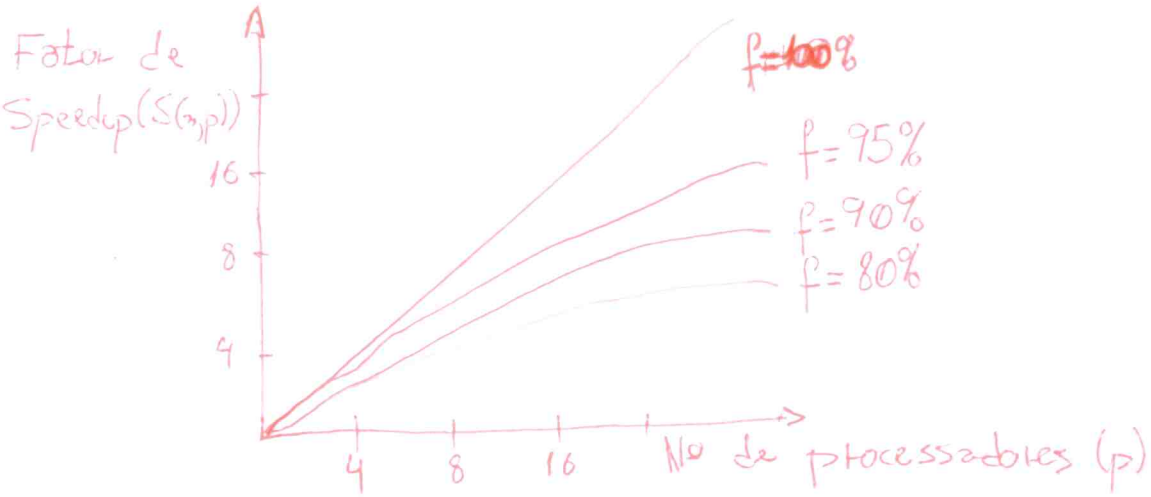
(16)



(17) Speedup máximo pela Lei de Amdahl

$$S(n,p) = \frac{T_s}{(1-f)T_s + f \cdot T_s/p}$$

$$\boxed{S(n,p) = \frac{1}{(1-f) + f/p}}$$ Fator de Speedup



(18) Lei de Gustafson (1988)

- A principal deficiência da Lei de Amdhal é que não considera o tamanho do problema n.
- A Lei de Gustafson reavalia a Lei de Amdahh sob o ponto de vista da escalabilidade
- Escalabilidade ocorre quando é possível manter constante a eficiência $E(n,p)$ incrementando o tamanho do problema n ao mesmo tempo que o número de processos p é incrementado

(19) Trabalho (W) e Sobrecarga (T_o)

- Trabalho de um algoritmo serial/sequencial

$$W(n) = T(n)$$

- Trabalho de um algoritmo paralelo

$$W(n,p) = p \cdot T(n,p)$$

- Sobrecarga de um algoritmo paralelo

$$T_o(n,p) = W(n,p) - W(n) = pT(n,p) - T(n)$$

- Eficiência

$$E(n,p) = \frac{1}{\frac{T_o(n,p)}{T(n)} + 1}$$

(20) Lei de Gustafson

- Speedup:

$$\boxed{S(n,p) = p - (1-f)(p-1)}$$

- Nos últimos anos o conceito de escalabilidade de (Lei de Gustafson) tem aberto novos horizontes ao uso de paralismo massivo