Programação Paralela

Alcides Calsavara

Referências

- *Foundations of Multithreaded, Parallel, and Distributed Programming*. Gregory R. Andrews. Addison-Wesley, 2000. Capítulo 11.
- *An Introduction to Parallel Algorithms*. Joseph JáJá. Addison-Wesley, 1992.

Objetivo

- Reduzir o **tempo de execução** através da divisão das tarefas em muitos processadores
  - redução em comparação com a implementação sequencial, isto é, executada em apenas um processador
  - permite resolver grandes problemas em menos tempo
  - permite resolver muitos problemas no mesmo tempo gasto para resolver somente um
  - A *performance* (ou *desempenho*) de um programa é o seu tempo total de execução.
- Exemplo: previsão do tempo
  - envolve o tratamento de um grande volume de dados
  - exige precisão no resultado e conclusão do processamento com razoável antecipação para que seja útil

Speedup

- Sejam:
  - T_1 : tempo de execução para resolver um problema usando um programa sequencial que é executado em um único processador
  - T_p : tempo de execução para resolver o mesmo problema usando um programa paralelo que é executado em *p* processadores.
- O *speedup* do programa paralelo é definido por:

$$\textbf{Speedup} = T_1 / T_p$$

Speedup

- Exemplo: O tempo de execução de um programa sequencial é de 40 segundos, enquanto o tempo de execução do correspondente programa paralelo executado em 10 processadores é de 5 segundos. Então, o speedup do programa paralelo para 10 processadores é 8.
- Speedup:
  - *linear* (ou *perfeito*) : tempo de execução inversamente proporcional ao número de processadores.
  - *sublinear* : abaixo do linear (caso mais comum)
  - *superlinear* : acima do linear (caso raro; ocorre quando a divisão dos dados permite fazer cache)

Speedup


Speedup
12
10
8
6
4
2
0
1
2
3
4
5
6
7
8
9
10
linear
superlinear
sublinear

Eficiência

- Mede quão bem um programa paralelo utiliza processadores extras.
- Definida por:

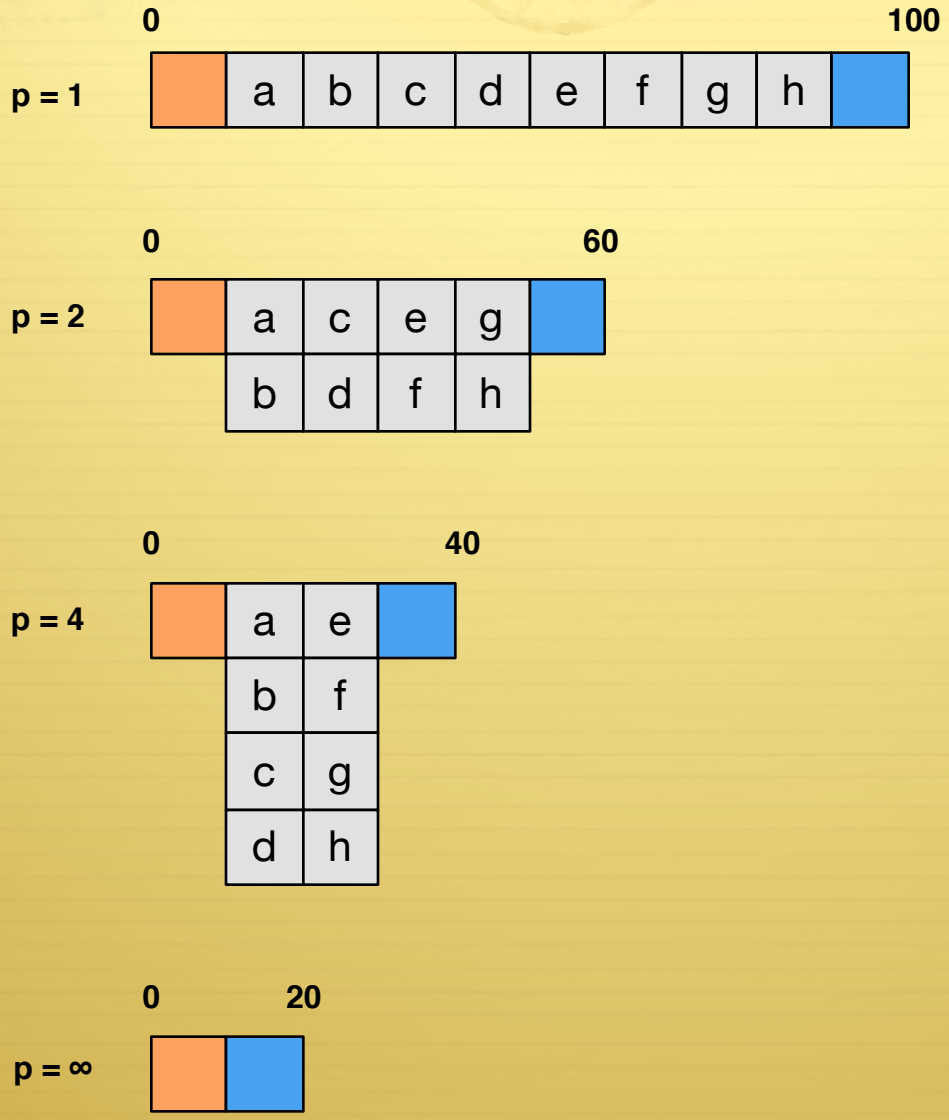
  $$Eficiência = Speedup \ / \ p$$

  $$Eficiência = T_1 \ / \ (p * T_p)$$

- Relação com speedup:
  - speedup linear → eficiência = 1.0
  - speedup sublinear → eficiência < 1.0
  - speedup superlinear → eficiência > 1.0

Limitação devido a I/O

- Programa típico tem três fases:
  1. Input
  2. Processamento
  3. Output
- As fases de input e output são, normalmente, sequenciais, o que limita o speedup de um programa paralelo.
- Exemplo: programa sequencial que tenha a seguinte distribuição com relação ao tempo de execução:
  1. Input: 10%
  2. Processamento: 80%
  3. Output: 10%


0
100
p = 1
a b c d e f g h
0
60
p = 2
a c e g
b d f h
speedup = 100 / 60 = 1,66
0
40
p = 4
a e
b f
c g
d h
speedup = 100 / 40 = 2,5
0
20
p = ∞
speedup = 100 / 20 = 5

Lei de Amdahl

- *O ganho obtido com a aceleração da execução de uma **parte** da computação é limitada pela **fração de tempo** correspondente àquela parte.*

  $Speedup_{geral} = 1 / (1 - Fração_{parte} + Fração_{parte} / Speedup_{parte})$

- No exemplo, a fração da parte que pode ser paralelizada, isto é, acelerada é 0,8. Assim, supondo speedup linear para a parte paralelizada, temos:
  - para p = 2, $Speedup_{geral} = 1 / (1 - 0,8 + 0,8 / 2) = 1,66$
  - para p = 4, $Speedup_{geral} = 1 / (1 - 0,8 + 0,8 / 4) = 2,5$
  - para p = ∞, $Speedup_{geral} = 1 / (1 - 0,8) = 5$

Relatividade

- Speedup e eficiência são medidas relativas, pois dependem de:
  - número de processadores
  - tamanho do problema
  - algoritmo que é usado
- Um programa paralelo tem ***escalabilidade*** se a sua eficiência é constante para um grande intervalo de número de processadores e de tamanhos de problema.
- ***Eficiência absoluta*** (ou ***Speedup absoluto***) é calculado usando-se o melhor algoritmo sequencial conhecido para T_1.

Tipos de Paralelismo

1. **Paralelismo Iterativo**
   - Programa é composto por vários processos, normalmente idênticos, cada um com um ou mais loops; cada processo é um programa iterativo.
   - Os processos trabalham juntos para resolver um único problema.
   - Ocorre frequentemente em computação científica que executa em múltiplos processadores (comunicação por variáveis compartilhadas). Exemplo: Multiplicação de matrizes.

Tipos de Paralelismo

2. **Paralelismo Recursivo**

   - Programa tem um ou mais procedimentos recursivos e as chamadas são independentes quanto ao acesso aos dados (cada chamada trabalha em uma parte diferente do dado compartilhado).
   - Usado frequentemente em algoritmos de divisão-e-conquista (ex: Mergesort) e de backtracking (ex: xadrez).
   - Usado para resolver problemas de combinatória, tais como ordenação, escalonamento (ex: caixeiro viajante) e jogos (ex: xadrez).

Multiplicação de Matrizes

Algoritmo sequencial:

Input: `a` *e* `b`*, duas matrizes n x n*
Output: `c`*, uma matriz n x n*

```
for [i = 0 to n-1]
    for [j = 0 to n-1]
        c[i, j] = 0.0;
        for [k = 0 to n-1]
            c[i, j] = c[i, j] + a[i, k] * b[k, j];
```

Exercício: Escrever o algoritmo **Worker$_i$** que calcula os elementos da matriz c correspondente a uma parte das p partes de c, sendo p o número de processadores disponíveis para o processamento paralelo.