Fontes principais

1. J. Jaja, An introduction to Parallel Algorithms, Addison Wesley, 92
   - ▷ Algoritmos paralelos
2. E. Cáceres, H. Mongeli, S. Song: Algoritmos paralelos usando CGM/PVM/MPI: uma introdução
   `http://www.ime.usp.br/~song/papers/jai01.pdf`

O modelo PRAM e algoritmos paralelos

Modelo de desenvolvimento de algoritmos paralelos

para $x \in X$ **faça em paralelo**
  instrução 1
  ...
  instrução k

Significado: X é um conjunto. Cada elemento $x \in X$ é associado a um processador. No total são utilizados $|X|$ processadores. Cada processador executa instruções de 1 a k, sequencialmente, em paralelo com os demais processadores, para cada x.

O modelo PRAM

Exemplos:

▷ Zerar um vetor A de n elementos:

para $1 \leq i \leq n$ **faça em paralelo**
$A[i] := 0$

▷ Utiliza n processadores.

▷ Utiliza 1 passo de tempo

O modelo PRAM

- ▷ Zerar uma matriz M de $n \times n$ elementos:

  para $1 \leq i, j \leq n$ **faça em paralelo**

  $M[i, j] := 0$

- ▷ Utiliza n^2 processadores.
- ▷ Utiliza 1 passo de tempo

Submodelos PRAM

Leitura e escrita concorrente

Como o modelo PRAM é SIMD, todos os processadores ativos executam a mesma instrução ao mesmo tempo, as quais podem ser leitura ou escrita em memória.

Quando dois ou mais processadores acessam um mesmo endereço de memória (ao mesmo tempo), dizemos que está ocorrendo uma leitura ou escrita concorrente (simultânea).

Leitura e escrita concorrente

O que acontece nessas situações?

O modelo PRAM foi dividido em submodelos, que tratam o problema acima de maneiras diferentes.

▷ EREW, CREW, CRCW (fraco, comum, arbitrário, com prioridade, forte)

Submodelos

▷ EREW: Exclusive Read Exclusive Write

Não existe leitura ou escrita simultâneas

▷ CREW: Concurrent Read Exclusive Write

Existe leitura simultânea. Todos os processadores obtêm o mesmo valor.

Não existe escrita simultânea

Submodelos

- CRCW: Concurrent Read Concurrent Write

  Existe leitura simultânea. Todos os processadores ativos obtêm o mesmo valor.

  Existe escrita simultânea.

  Como é realizado a escrita simultânea? O submodelo CRCW foi subdividido em submodelos, que realizam a escrita simultaânea de maneiras diferentes.

Submodelos

▷ CRCW Fraco: Na escrita simultânea. Todos os processadores escrevem o mesmo valor 0 ou 1.

▷ CRCW modo comum: Na escrita simultânea, todos os processadores escrevem o mesmo valor qualquer.

▷ CRCW vencedor arbitrário: Os valores escritos simultaneamente podem ser diferentes. Qual destes valores ficará armazenada na posição de memória? Qualquer um. Não sabemos qual.

Submodelos

▷ CRCW com prioridade: Os valores escritos simultaneamente podem ser diferentes. Ficará armazenado na posição da memória o valor escrito pelo processador de maior prioridade (assumimos que prioridade é o número de identificação)

▷ CRCW forte: Os valores escrito simultaneamente podem ser diferentes. Ficará armazenado na posição da memória o maior valor escrito.

Submodelos PRAM - Exemplos

- EREW:

para $1 \leq i \leq n$ **faça em paralelo**
$A[i] := i$

Submodelos PRAM - Exemplos

▷ CREW:

para $1 \leq i \leq n$ **faça em paralelo**
$A[i] := C$

Leitura simultânea em C

Submodelos PRAM - Exemplos

▷ CRCW fraco:

para $1 \leq i \leq n$ **faça em paralelo**
$B := 0$

Escrita simultânea em B

Submodelos PRAM - Exemplos

▷ CRCW modo comum:

para $1 \leq i \leq n$ **faça em paralelo**
$B := C$

Escrita simultânea em B e leitura simultânea em C

Submodelos PRAM - Exemplos

$\triangleright$ CRCW modo comum:

para $1 \leq i, j \leq n$ **faça em paralelo**
$A[i] := i$

Escrita simultânea em cada A[i].
Ex: Os elementos do par (i, j) sendo $\{(1,1),(1,2),(1,3)\cdots(1,n)\}$ são computados para $i = 1$, e este será atribuído simultaneamente ao elemento $A[1]$ por n processadores.

Submodelos PRAM - Exemplos

- CRCW vencedor arbitrário:
- CRCW com prioridade:
- CRCW forte:

para $1 \leq i \leq n$ **faça em paralelo**
$B := A[i]$

Submodelos PRAM - Exemplos

Escala do submodelo mais fraco para o mais forte

Mais fraco

- EREW
- CREW
- CRCW Fraco
- CRCW Modo Comum
- CRCW Arbitrario
- CRCW com Prioridade
- CRCW Forte

Mais forte

Submodelos PRAM - Exemplos

Um submodelo ser mais forte significa que ele exige mais do hardware

Quase sempre posso executar um algoritmo mais fraco em uma máquina mais forte.

Submodelos PRAM - Exemplos

Se tenho um algoritmo mais forte não posso executá-lo em uma máquina mais fraca sem haver uma simulação

É desejável que o algoritmo seja para um submodelo o mais fraco possível.

Medidas de desempenho de algoritmos paralelos

Medidas de desempenho de algoritmos paralelos

▷ Complexidade de tempo paralelo: Tempo total consumido pelo algoritmo (número de passos do algoritmo), sendo que cada computação efetuada em paralelo contribui com uma unidade para o tempo total, independente do número de processadores envolvidos.

Medidas de desempenho de algoritmos paralelos

$\triangleright$ Complexidade de processadores: Maior número de processadores envolvidos em qualquer passo do algoritmo.

$\triangleright$ Complexidade de espaço: Igual ao caso sequencial

$\triangleright$ Complexidade de custo (work): Número de operações realizadas.

Compl.de custo $=$ compl. de tempo paralelo $\times$ compl. de processadores

Speed-up (aceleração)

$$\text{Speed-up} = \frac{\text{Compl. de tempo do melhor algoritmo sequencial}}{\text{Compl. de tempo paralelo do algoritmo paralelo}}$$

Pode ser uma medida experimental também.

Medidas de desempenho de algoritmos paralelos

O melhor tempo paralelo (mínimo) que pode-se obter é igual ao melhor tempo sequencial dividido pelo número de processadores.

$$\text{tempo paralelo} \geq \frac{\text{melhor tempo sequencial}}{\text{número de processadores}}$$

Speed-up máximo

$$\text{Speed-up máximo} = \frac{\text{melhor tempo sequencial}}{\text{melhor tempo paralelo}}$$

$$\text{Speed-up máximo} = \frac{\text{melhor tempo sequencial}}{\frac{\text{melhor tempo sequencia}}{\text{número de processadores}}}$$

speed-up $\leq$ número de processadores

Exemplo

para $1 \le i \le n$ **faça em paralelo**
    para $1 \le j \le n$ **faça**
        $M[i,j] := 0$

- Tempo paralelo: $O(n)$
- Complexidade de processadores: $O(n)$
- Custo: $O(n^2)$ operações

Exemplo

para $1 \leq i, j \leq n$ **faça em paralelo**
$M[i, j] := 0$

- Tempo paralelo: $O(1)$
- Complexidade de processadores: $O(n^2)$
- Custo: $O(n^2)$ operações

Exemplo

para $1 \leq i \leq n$ **faça**
$A[i] := 0$

para $1 \leq i \leq n$ **faça em paralelo**
$A[i] := 0$

▷ Tempo sequencial: $O(n)$

▷ Tempo paralelo: $O(1)$
▷ Custo: $O(n)$

$$\text{Speed-up} = \frac{n}{1} = n$$

▷ O algoritmo paralelo é n vezes mais rápido que o sequencial

Medidas de desempenho de algoritmos paralelos

O melhor custo (mínimo) que pode-se obter é igual ao tempo do melhor algoritmo sequencial.

$$\text{custo mínimo} = \text{melhor tempo paralelo} \times \text{número de processadores}$$

$$\text{custo mínimo} = \frac{\text{melhor tempo sequencial}}{\text{número de processadores}} \times \text{número de processadores}$$

$$\text{custo} \geq \text{melhor tempo sequencial}$$

Eficiência e Otimalidade

Um algoritmo paralelo é **eficiente** se sua complexidade de tempo é uma função polinomial no logaritmo do tamanho da entrada e sua complexidade de processador é polinomial no tamanho da entrada.

Algoritmos eficientes

Algoritmos eficientes	tempo	processadores
sequenciais	polinomial	1
paralelos	polilogaritmo	polinomial

Algoritmos paralelos	tempo	processadores
eficientes	$O(\log^3 n)$	$O(n^3)$
eficientes	$O(\log n)$	$O(n)$
não eficientes	$O(n)$	$O(\log n)$
não eficientes	$O(\log n)$	$O(2^n)$

Eficiência e Otimalidade

Um algoritmo paralelo é **ótimo** se seu custo é igual ao tempo do melhor algoritmo sequencial para o problema.

Eficiência e Otimalidade

Um algoritmo é **ótimo absoluto** se seu custo é igual ao tempo do melhor algoritmo sequencial, e este tempo é igual ao limite inferior do problema.

Eficiência e Otimalidade

O conceito de ótimo em algoritmos paralelos depende do submodelo PRAM considerado. Um algoritmo pode ser implementado em 2 submodelos diferentes com complexidades diferentes.

Eficiência e Otimalidade

Problema: limite inferior $O(n)$

Algoritmo sequencial: tempo $O(n \log n)$

Algoritmo paralelo: tempo $O(\log n)$

Processadores: $O(n)$

Custo: $O(n \log n)$

Algoritmo ótimo e eficiente!

Eficiência e Otimalidade

Problema: limite inferior $O(n)$

Algoritmo sequencial: tempo $O(n)$

Algoritmo paralelo: tempo $O(\log n)$

Processadores: $O(\frac{n}{\log n})$

Custo: $O(n)$

Algoritmo ótimo e eficiente!

Teorema de Brent

Utilizado para reduzir a complexidade de processadores de um algoritmo

Teorema 1. *Suponha que um problema possa ser resolvido através de um algoritmo paralelo, com complexidade de tempo $O(t)$, custo $O(m)$ operações, e complexidade de processadores maior que $O(p)$. Então este algoritmo pode ser implementado com complexidade de tempo $O(\frac{m}{p} + t)$ e de processadores $O(p)$.*

Prova. O algoritmo possui t passos. Suponha que no passo i o algoritmo realize m_i operações. Assim, $m = m_1 + m_2 + \cdots + m_t$. Se utilizarmos apenas p processadores no passo i, o tempo total gasto neste passo será $\left\lceil \frac{m_i}{p} \right\rceil$ passos.

O tempo total do algoritmo será:

$$\sum_{i=1}^{t} \left\lceil \frac{m_i}{p} \right\rceil \leq \sum_{i=1}^{t} \left(\left\lfloor \frac{m_i}{p} \right\rfloor + 1 \right) = \frac{m_1 + m_2 + \cdots + m_t}{p} + t = \frac{m}{p} + t$$

Logo, a complexidade de tempo fica $O(\frac{m}{p} + t)$ utilizando $O(p)$ processadores.

$\square$

Teorema de Brent

▷ Algoritmo 1

para $1 \leq i, j \leq n$ **faça em paralelo**
$A[i, j] := 0$

▷ tempo: $O(1)$
▷ processadores: $O(n^2)$
▷ custo: $O(n^2)$

Teorema de Brent

Aplicando o teorema de Brent (para $p = O(n)$)

▷ Algoritmo 2

para $1 \leq i \leq n$ **faça em paralelo**
    para $j = 1$ **até** n **faça**
        $A[i, j] := 0$

▷ tempo: $\dfrac{n^2}{n} + 1 = O(n)$
▷ processadores: $O(n)$

Fim