Um algoritmo $O(logn)$ para Multiplicação de Matrizes no Hipercubo

Li Kuan Ching
Orientador : Prof. Dr. Siang Wun Song §

§ Depto de Ciência da Computação, IME,USP

e-mail: li@ime.usp.br, song@ime.usp.br

Sumário

O objetivo deste trabalho será apresentar um algoritmo de multiplicação de matrizes de ordem $O(logn)$, para máquinas de organização tipo *Hipercubo*.

1 Introdução

Muitas topologias de interconecção têm sido propostos na literatura com o propósito de conectar centenas ou milhares elementos de processamento. Uma das topologias mais populares destes propostos é a organização denominado **hipercubo** ou **n-cubo binário**. Esta arquitetura tem se destacado em termos da obtenção de excelente desempenho computacional paralelo[2].

Será apresentado, a seguir, um algoritmo para multiplicação de matrizes, da ordem de $O(logn)$. É um resultado de importância, visto a grande utilização da aplicação nos meios científicos de pesquisa e de aplicação e comercial[1].

2 Noções do *Hipercubo*

O hipercubo foi introduzido no meio científico por Charles Seitz[3] no início da década de 80. Este é uma conexão de pequenos computadores ou elementos de processamento, denominados nós ou vértices, que se comunicam entre si através da troca de mensagens.

Esta organização é de dimensão k (ou ordem k) se este tem $n = 2^k$ nós, sendo k um inteiro. Cada um destes nós é conectado a k outros nós e é identificado por um endereço binário de k bits. Desta forma, dois nós desta organização são vizinhos se os seus endereços binários diferem de um só bit. Um hipercubo pode também ser definido recursivamente[2]. O de ordem 0 é um nó somente. Para construir um hipercubo de ordem $n+1$, basta conectar os nós de dois hipercubos de ordem n, seguindo a propriedade mencionada acima. O 4-cubo, para exemplificar, tem $2^4 = 16$ nós ou vértices. Toma-se um nó deste quaisquer, supondo 1101. Os nós vizinhos a este são quatro, a saber 0101, 1001, 1111 e 1100, que corresponde exatamente a ordem do hipercubo em questão[4, 5].

3 O problema

Seja A e B duas matrizes $n x n$. Assim, cada elemento destas matrizes pode ser representado por a_{ij} e b_{ij}, respectivamente $(0 \leq i,j \leq n-1)$. Seja c_{ij}, $0 \leq i,j \leq n-1$, um elemento da matriz C, resultante da multiplicação das matrizes A e B.

Para resolver este problema, será usado hipercubo de 2^{3q} nós, que satisfaz a condição $n^3 = 2^{3q}$, $q \in N$. Cada nó ou vértice tem 3 registradores, que são: A(m), B(m) e C(m). Antes da aplicação do algoritmo, os a_{ij} e b_{ij} estão armazenados nos registradores $A(2^q i + j)$ e $B(2^q i + j)$ e posteriormente à aplicação, os c_{ij} estão armazenados em $C(2^q i + j)$, $0 \leq i,j \leq n-1$.

Seja m o endereço binário de um nó dado. O *complementar do endereço m no bit b*, representado por $m^{(b)}$, é simplesmente substituir o complementar do bit na posição b do endereço. Para exemplificar, seja 1100 o endereço binário. Assim, o complementar $1100^{(1)}$ do endereço em questão é 1110.

4 O algoritmo

O algoritmo consiste de três fases no total. A primeira fase é denominada propagação, ou seja, transmite os valores de a_{ij} e b_{ij} para os registradores A e B dos respectivos nós, divididas em três subfases. Na segunda fase é feita multiplicações em cada um dos nós e na terceira fase, a soma dos produtos obtidos. Será apresentado a seguir mais detalhadamente.

▷Primeira fase : Propagação

- Primeira subfase:
  $A(2^{2q}k + 2^q i + j) = a_{ij}$
  $B(2^{2q}k + 2^q i + j) = b_{ij}, 0 \leq k \leq n-1$
- Segunda subfase:
  $A(2^{2q}k + 2^q i + j) = a_{ik}, 0 \leq j \leq n-1$

- Terceira subfase:
  $B(2^{2q}k + 2^{q}i + j) = b_{kj}, 0 \leq i \leq n-1$

▷**Segunda fase : Multiplicação em paralelo**

É efetuada a operação de multiplicação em todos os nós, simultaneamente, isto é, C(m) := A(m)*B(m).

▷**Terceira fase : Soma dos Produtos Obtidos**

É feita a soma dos produtos em todos os nós, que é equivalente dizer efetuar C(m) := C(m) + C($m^{(t)}$), $2q \leq t \leq 3q-1$.

Algoritmo detalhado

/* Fase 1 : Propagação de a_{ij} e b_{ij}

```
begin
      for t := 3q-1 downto 2q do
            Node(m), todo m        /* checar em todos os nós
            if m^(t) = 1 then
                  A(m^(t)) := A(m)
                  B(m^(t)) := B(m)
      endfor
      for t := q-1 downto 0 do
            Node(m), todo m        /* checar em todos os nós
            if m^(t) ≠ m^(2q+t) then
                  A(m^(t)) := A(m)
      endfor
      for t := 2q-1 downto q do
            Node(m), todo m        /* checar em todos os nós
            if m^(t) ≠ m^(q+t) then
                  B(m^(t)) := B(m)
      endfor
end
```

/* Fase 2 : Multiplicação feita paralelamente em todos os processadores

```
begin
      C(m) := A(m)*B(m)
end
```

/* Fase 3 : Soma dos produtos Obtidos

```
begin
      for t := 2q to 3q-1 do
            Node(m), todo m      /* todos os nós
            C(m) := C(m) + C(m^(t))
      endfor
end
```

5 Análise do algoritmo

Será feita aqui uma análise da complexidade do algoritmo apresentado.

Na primeira fase, para a execução do primeiro *for* foi necessário 2q passos. O segundo e o terceiro *for* desta fase foram necessários q passos cada. A terceira fase precisou de q passos também para a sua execução. Além destes, foram necessários para 1 passo de multiplicação (o executado em paralelo em todos os processadores). Com o resultado da soma do número de operações e da condição estabelecida inicialmente, $n^3 = 2^{3q}$, é obtido a complexidade final do algoritmo, de $O(logn)$.

Referências

[1] D.P. Bertsekas, J.N. TsiTsiklis 'Parallel and Distributed Computations - Numerical methods', Prentice-Hall, 1989.

[2] N. Mokhoff - Senior Editor 'Hypercube architecture leads the way for commercial supercomputers in scientific applications', Computer Design, May 1, 1986

[3] Charles Seitz, 'A Cosmic Cube', Communications of the ACM, vol.28, no.1, Jan 1985, pp 22-33.

[4] J.D. Ullman, 'Computacional Aspects of VLSI', Computer Scientific Press, 1984.

[5] Li K.C. 'Hipercubo e algumas características', Ata do V Colóquio de Iniciação Científica. IME-USP, 1989.