Transposição de Matrizes no Hipercubo

Li Kuan Ching

Orientador: Siang Wun Song

Apresentaremos, neste trabalho, um algoritmo para o hipercubo com aplicações como transposição de matrizes e inversão de um conjunto de bits.

Antes, vamos introduzir algumas características.

Um hipercubo de dimensão n (também chamado n-cubo) é constituído de 2^n processadores chamados nós ou vértices. Assim, um 3-cubo tem 8 nós, um 4-cubo tem 16 nós. Cada nó de um n-cubo é identificado por um endereço binário de n bits. Dois nós são vizinhos se os seus endereços binários diferem de um só bit. Por exemplo, no 4-cubo, 0101 e 1101 são nós vizinhos. (Maiores detalhes, ver [2] e [3]).

Seja $\oplus$ o operador lógico OU-Exclusivo e a função $Sum(j)$ o número de 1's na representação binária de j. O número de arestas que separam o nó i do nó j é dado por $Sum(i \oplus j)$.

O número de arestas num n-cubo é $n \cdot 2^{n-1}$, onde em cada uma destas o fluxo de informações é bidirecional. Aqui, suporemos que entre dois nós têm-se duas ligações, cada uma destas unidirecional. Portanto, teremos um total de $n \cdot 2^n$ ligações. A razão da escolha é otimizar o paralelismo existente e diminuir a fila do tráfego em cada uma das arestas.

Detalharemos um pouco mais a aplicação do algoritmo na transposição de matrizes.

Seja A uma matriz $N \times N$, onde $N = 2^n$, disposta dentro de um n-cubo, da seguinte maneira. Cada elemento a_{ij} de A está posicionado no nó i e memória local j do mesmo. Denotaremos este endereço por $i|j$. Transpor a matriz A significa direcionar as N informações contidas no nó i $\{i|j : j = 0, 1, 2, \cdots, N-1\}$ para as N memórias locais i dos nós j $\{j|i : i = 0, 1, 2, \cdots, N-1\}$.

A transposição requer, no mínimo, 2^{n-1} passos de comunicação entre os nós do n-cubo (ver [1]).

Algoritmo

Seja W uma tabela de entradas w_{ij}, $i = 0, 1, \cdots, 2^{n-1}-1$, $j = 0, 1, \cdots, n-1$, de tamanho $2^{n-1} \times n$, onde o k-ésimo bit de w_{ij} é denotado por w_{ij}^k, satisfazendo as seguintes condições:

1. Existência da ligação: $w_{ik}^k = 1$, para todo i, k.

2. Unicidade da coluna: $w_{i_1 j} \neq w_{i_2 j}, i_1 \neq i_2$.

3. Unicidade da linha: $w_{ij_1} \neq w_{ij_2}, j_1 \neq j_2$ para todo i.

Gerada a tabela W, temos que, para $i = 0, 1, 2, \cdots, 2^{n-1} - 1$, a expressão $w_{ij} \oplus$ <endereço-do-nó> especifica a posição da memória local para a informação mandada na ligação j. A informação que chega pela ligação j toma o lugar deste último.

Algoritmo prático para construção da tabela W

Seja $m_i = 2i + 1, i = 0, 1, \cdots, 2^{n-1} - 1$ e o representaremos na forma binária. Cada w_{ij} pode ser calculado da seguinte forma:

i) Complementar o bit $j + 1$ da representação; (se $j \neq n - 1$).

ii) Trocar os bits 0 e j entre si.

Exemplo:
$n = 3$:

$$m_i = 2i + 1, \quad i = 0, 1, 2, 3$$

$$m_0 = 1 = 001 \qquad m_1 = 3 = 011 \qquad m_2 = 5 = 101 \qquad m_3 = 7 = 111$$

$$m_0 = 1 = 001$$

$$j = 0: \quad 011$$
$$j = 1: \quad 110$$
$$j = 2: \quad 100$$

m_1, m_2, e m_3 obtém-se de forma análoga, aplicando o algoritmo prático dado.

Tabela W

	0	1	2
1	011	110	100
2	001	111	110
3	111	010	101
4	101	011	111

Obtida a tabela W, calculamos para cada passo de comunicação ($i = 0, 1, 2, 3$), $w_{ij} \oplus$ <endereço-do-nó>, $j = 0, 1, 2$. Assim, por exemplo, para o nó 0 (end. binário 000).

1º **passo:**

0	1	2
011	110	100

000|011 terá locomoção na dimensão 0

000|110 terá locomoção na dimensão 1

000|100 terá locomoção na dimensão 2

2º **passo:**

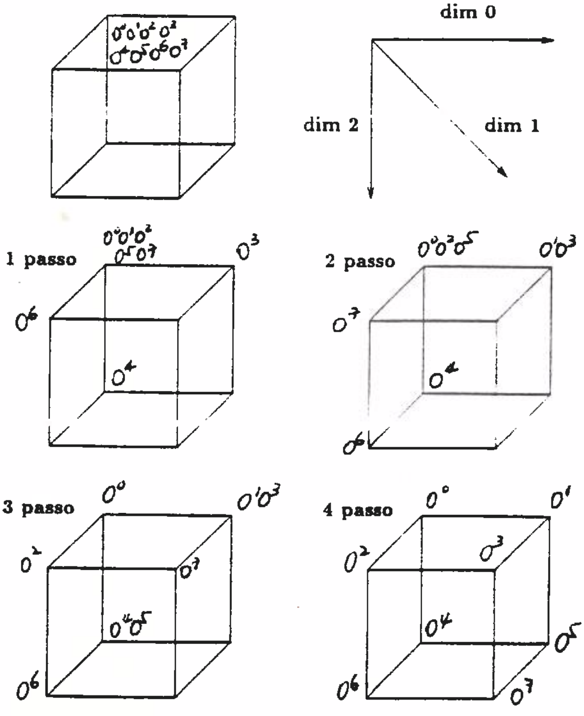
0	1	2
001	111	110

000|001 terá locomoção na dimensão 0

000|111 terá locomoção na dimensão 1

000|100 terá locomoção na dimensão 2

Analogamente para os outros passos de comunicação.
Daremos, a seguir, uma visualização para o nó 0.



Bibliografia

[1] Bertsekas. D. P., and Tsitsiklis, J. N., *Parallel and Distributed Computation - Numerical Methods*, **Prentice - Hall International, Inc., 1989.**

[2] Edelman. Alan., *Optimal Matrix Transposition and Bit Reversal on Hypercubes: All-to-All Personalized Communication*, **Journal of Parallel and Distributed Computing** 11,pp 328-331, 1991.

[3] Kung, S. Y., *VLSI Array Processors.* **Prentice Hall.** 1988.

[4] Ullman, J. D., *Computational Aspects of VLSI.* **Computer Science Press.** 1984.