Aplicação de um método de imersão para problemas matriciais no hipercubo[1]

C. Y. Takemoto and S. W. Song
Universidade de São Paulo
Instituto de Matemática e Estatística
Departamento de Ciência da Computação
05508-900 São Paulo, SP, Brasil

1 Introdução

O hipercubo vem se destacando como uma das mais populares redes de interconexão para máquinas paralelas, devido, entre outras razões, a seu pequeno diâmetro e grau ($\log n$, onde n é o seu número de nós), a facilidade de roteamento e a sua capacidade de simular eficientemente outras arquiteturas paralelas. Além disso, o hipercubo pode ser totalmente dividido em subcubos permitindo a implementação de algoritmos do tipo *divisão e conquista*.

A imersão (*embedding*) de uma rede de interconexão em outra é uma questão muito importante no desenvolvimento e na análise de algoritmos paralelos. Através destas imersões, os algoritmos originalmente desenvolvidos para uma determinada arquitetura podem ser mapeados para uma outra arquitetura.

Os resultados de imersão de um m-cubo r-ário em um hipercubo já são conhecidos na literatura, como o método do *código de Gray refletido* de Saad e Schultz [SS88]. Entretanto, utilizaremos aqui o método do *dobramento recursivo* proposto por Song e Hamdi [HS96] para este tipo de imersão. Neste caso, *dobramos* recursivamente o hipercubo tal que a disposição final contém o m-cubo r-ário. Esta disposição, chamada de $T(r,m)$, possui a seguinte propriedade: Todos os subcubos, formados pelos nós com os mesmos primeiros km bits, para $1 \leq k < \log r$, também estão dispostos em m-cubos $(r/2^k)$-ários. O método de imersão do dobramento recursivo possibilita a identificação imediata de todos os subcubos menores de um m-cubo r-ário. Isto não se verifica no método do código de Gray refletido.

Como nas publicações anteriores sobre o método do dobramento recursivo não há exemplos que ilustrem a aplicabilidade e utilidade desta propriedade, apresentaremos neste artigo um algoritmo recursivo proposto para um m-cubo r-ário que necessite de todos os subcubos menores. Assim, se implementarmos este algoritmo no hipercubo utilizando a disposição $T(r,m)$, podemos obter de maneira fácil os endereços destes subcubos através da mencionada propriedade.

2 Transposição de matriz

A transposta de uma matriz $A = (a_{ij})$, $n \times n$, é uma matriz $A^T = (a^T_{ij})$ de mesmo tamanho, tal que $a^T_{ij} = a_{ji}$, para $0 \leq i,j < n$.

[1] Apoio da FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) Proc. No. 97/04638-5, CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) Proc. No. 52 3778/96-1 e Commission of the European Communities through Project ITDC-207-82167.

Seja A uma matriz quadrada $n \times n$, (supor n uma potência de 2):

$$A = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix},$$

onde $A_{i,j}$ são submatrizes $\frac{n}{2} \times \frac{n}{2}$. Podemos calcular a transposta da matriz A trocando inicialmente os elementos correspondentes das submatrizes $A_{0,1}$ e $A_{1,0}$ entre si, conforme a Figura 1, e então transpondo recursivamente e em paralelo os elementos dentro de cada

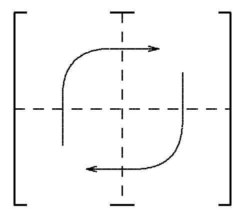

Figura 1: $A_{0,1} \longleftrightarrow A_{1,0}$.

submatriz $A_{i,j}$. Este algoritmo foi baseado no texto de Kumar et al. [KGGK94].

Supomos que a matriz A esteja armazenada em uma grade $[n \times n]$, ou seja, 2-cubo n-ário, de modo que cada processador armazene um único elemento da matriz.

O algoritmo recursivo **Transposta** recebe uma matriz A, a ordem n desta matriz, e a coordenada (x, y) do elemento que se encontra no canto superior esquerdo (veja a Figura 2.(a)), e obtêm a matriz transposta de A. Assim, a primeira chamada deste algoritmo deve ser o seguinte: **Transposta**$(A, n, 0, 0)$.

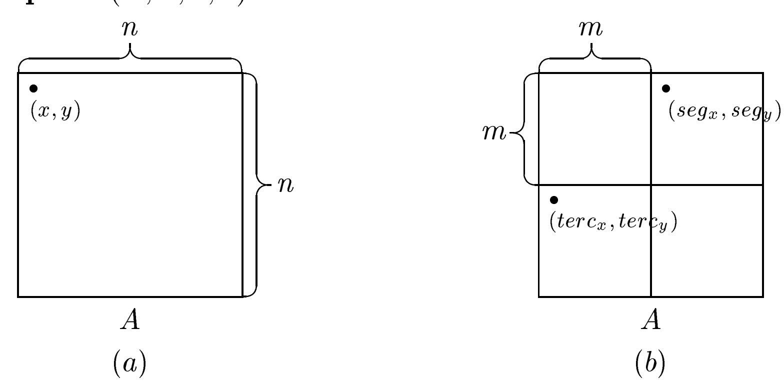

Figura 2: Entradas dos Algoritmos 2.1 e 2.2.

Algoritmo 2.1 *Entrada: uma matriz A, a ordem de A, n, e a coordenada do canto superior esquerdo da matriz A, (x, y);*

Saída: a matriz transposta de A, A^T.

Transposta(A, n, x, y)
begin
  if $n \neq 1$ **then**
    begin

Troca$(A, \frac{n}{2}, x, y+\frac{n}{2}, x+\frac{n}{2}, y)$**;**
parbegin
Transposta$(A, \frac{n}{2}, x, y)$**;**
Transposta$(A, \frac{n}{2}, x, y+\frac{n}{2})$**;**
Transposta$(A, \frac{n}{2}, x+\frac{n}{2}, y)$**;**
Transposta$(A, \frac{n}{2}, x+\frac{n}{2}, y+\frac{n}{2})$
parend
end
end

Os comandos (separados por **;**) colocados entre **parbegin** e **parend** são executados em paralelo. O algoritmo **Transposta**, por sua vez, faz a chamada do procedimento **Troca**. Este procedimento executa a troca dos elementos da submatriz do segundo quadrante (quadrante superior direito) pelos do terceiro quadrante (quadrante inferior esquerdo), Figura 1. Como entrada deste procedimento temos a matriz A, a ordem m das submatrizes em questão, as coordenadas do canto superior esquerdo do segundo e do terceiro quadrantes, respectivamente (seg_x, seg_y) e $(terc_x, terc_y)$. Veja Figura 2.(b).

Algoritmo 2.2 *Entrada: a matriz A, a ordem das submatrizes em questão, m, a coordenada do canto superior esquerdo da submatriz que se encontra no segundo quadrante, seg_x e seg_y, e a coordenada do canto superior esquerdo do terceiro quadrante, $terc_x$ e $terc_y$;*

Saída: a troca entre os elementos correspondentes do segundo e terceiro quadrantes.

Troca$(A, m, seg_x, seg_y, terc_x, terc_y)$
parbegin
doall $seg_x \leq i < m + seg_x$
doall $seg_y \leq j < m + seg_y$
mover o dado de $P_{i,j}$, m linhas para baixo $(i+m)$
e m colunas para esquerda $(j-m)$
endoall
endoall;
doall $terc_x \leq i < m + terc_x$
doall $terc_y \leq j < m + terc_y$
mover o dado de $P_{i,j}$, m linhas para cima $(i-m)$
e m colunas para direita $(j+m)$
endoall
endoall
parend

Usamos **doall** para indicar laços sem dependência entre iterações permitindo a execução de todas as iterações em paralelo. O algoritmo de transposição recursivo para uma matriz 8×8 está ilustrado na Figura 3.

3 Conclusão

O método do dobramento recursivo permite a imersão de um m-cubo r-ário em um hipercubo de mesmo número de nós com dilatação 1. Este método possui uma propriedade adicional não encontrada no método do código de Gray refletido, possibilitando a identificação imediata de todos os subcubos menores de um m-cubo r-ário mapeado no hipercubo.

Mostramos neste artigo que o método do dobramento recursivo pode ser útil para uma classe de problemas matriciais do tipo divisão-e-conquista. Mais especificamente, apresentamos um algoritmo recursivo para a obtenção da transposta de uma matriz, proposto para um 2-cubo r-ário que necessitem de todos os subcubos menores. Outros algoritmos deste tipo, como multiplicação de matrizes e ordenação por intercalação *Par-Ímpar*, podem ser encontrados em [Tak98].

Devemos observar que não estamos sugerindo que estes algoritmos para os problemas específicos sejam bons algoritmos para hipercubos, pois sabemos que existem outros melhores. Entretanto, um dos objetivos da imersão de outras redes de interconexão no hipercubo é o de permitir a utilização de algoritmos originalmente desenvolvidos para estas redes no hipercubo. Isto geralmente implica numa subutilização do hipercubo. Tendo isto em mente, as aplicações acima ilustram tão somente a aplicabilidade da propriedade do método do dobramento recursivo no aproveitamento de algoritmos propostos para grades no hipercubo.

Referências

[HS96] M. Hamdi and S. W. Song. On Embedding Various Networks into the Hypercube Using Matrix Transformations. *10th International Parallel Processing Symposium, IEEE Computer Society*, pages 650–654, April 15–19, 1996.

[KGGK94] V. Kumar, A. Grama, A. Gupta, and G. Karypis. Introduction to Parallel Computing - Design and Analisys of Algorithms. The Benjamin/Cummings Publishing Company, Inc., 1994.

[SS88] Y. Saad and M. H. Schultz. Topological properties of hypercubes. *IEEE Transactions on Computers*, 37(7), pages 867–872, July 1988.

[Tak98] C. Y. Takemoto. Algoritmos de Imersão de Redes de Interconexão em Hipercubos. *Dissertação de Mestrado, Instituto de Matemática e Estatística da Universidade de São Paulo*, 1998.

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3.6)	(3,7)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5.6)	(5,7)
(6,0)	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
(7,0)	(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)

(a)

(0,0)	(0,1)	(0,2)	(0,3)	(4,0)	(4,1)	(4,2)	(4,3)
(1,0)	(1,1)	(1,2)	(1,3)	(5,0)	(5,1)	(5,2)	(5,3)
(2,0)	(2,1)	(2,2)	(2,3)	(6,0)	(6,1)	(6,2)	(6,3)
(3,0)	(3,1)	(3,2)	(3,3)	(7,0)	(7,1)	(7,2)	(7,3)
(0,4)	(0,5)	(0,6)	(0,7)	(4,4)	(4,5)	(4,6)	(4,7)
(1,4)	(1,5)	(1,6)	(1,7)	(5,4)	(5,5)	(5.6)	(5,7)
(2,4)	(2,5)	(2,6)	(2,7)	(6,4)	(6,5)	(6,6)	(6,7)
(3,4)	(3,5)	(3.6)	(3,7)	(7,4)	(7,5)	(7,6)	(7,7)

(b)

(0,0)	(0,1)	(2,0)	(2,1)	(4,0)	(4,1)	(6,0)	(6,1)
(1,0)	(1,1)	(3,0)	(3,1)	(5,0)	(5,1)	(7,0)	(7,1)
(0,2)	(0,3)	(2,2)	(2,3)	(4,2)	(4,3)	(6,2)	(6,3)
(1,2)	(1,3)	(3,2)	(3,3)	(5,2)	(5,3)	(7,2)	(7,3)
(0,4)	(0,5)	(2,4)	(2,5)	(4,4)	(4,5)	(6,4)	(6,5)
(1,4)	(1,5)	(3,4)	(3,5)	(5,4)	(5,5)	(7,4)	(7,5)
(0,6)	(0,7)	(2,6)	(2,7)	(4,6)	(4,7)	(6,6)	(6,7)
(1,6)	(1,7)	(3.6)	(3,7)	(5.6)	(5,7)	(7,6)	(7,7)

(c)

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(7,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)	(6,2)	(7,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	(5,3)	(6,3)	(7,3)
(0,4)	(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	(6,4)	(7,4)
(0,5)	(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	(6,5)	(7,5)
(0,6)	(1,6)	(2,6)	(3.6)	(4,6)	(5.6)	(6,6)	(7,6)
(0,7)	(1,7)	(2,7)	(3,7)	(4,7)	(5,7)	(6,7)	(7,7)

(d)

Figura 3: Transposição de uma matriz 8×8.