

VI. OPERASI MATRIKS (Part 1)

Oleh Dr. Asep Juarna

1. Transposisi Matriks (*Matrix Transpose*)

Diberikan matriks A berukuran $m \times n$ (yaitu m baris dan n kolom) berikut:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n-1} & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m-1,1} & a_{m-1,2} & \cdots & a_{m-1,n-1} & a_{m-1,n} \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n-1} & a_{m,n} \end{bmatrix}$$

Transposisi matriks A , dinotasikan dengan A^T , adalah pertukaran posisi elemen-elemen matriks sedemikian rupa sehingga setiap elemen yang menempati baris ke- i kolom ke- j di A (yaitu $a_{i,j}$) menjadi menempati baris ke- j kolom ke- i di A^T (yaitu $a_{j,i}^T = a_{i,j}$).

Dengan kata lain, setiap *baris* di A menjadi *kolom* di A^T . Jelaslah elemen-elemen matriks $a_{i,i}$ tidak bertransposisi sehingga susunan elemen-elemen matriks A^T adalah:

$$A^T = \begin{bmatrix} a_{1,1} & a_{2,1} & \cdots & a_{m-1,1} & a_{m,1} \\ a_{1,2} & a_{2,2} & \cdots & a_{m-1,2} & a_{m,2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{1,n-1} & a_{2,n-1} & \cdots & a_{m-1,n-1} & a_{m,n-1} \\ a_{1,n} & a_{2,n} & \cdots & a_{m-1,n} & a_{m,n} \end{bmatrix}$$

Contoh: Berikut ini adalah dua buah matriks A dan B dengan masing-masing transposisinya.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}, \quad A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, \quad B^T = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Algoritma transposisi matriks berikut dirancang untuk matriks bujur sangkar (*square matrix*) $n \times n$, yaitu matriks dengan jumlah baris sama dengan jumlah kolom.

1.1. Algoritma Sekuensial

Algoritma sekuensial transposisi matriks adalah sangat sederhana, yaitu:

procedure *seq_transpose* (A)

(1) **for** $i = 1$ **to** n **do**

(2) **for** $j = 1$ **to** $i - 1$ **do**

(3) $a_{i,j} \leftrightarrow a_{j,i}$

(4) **end for**

(5) **end for**

Dalam algoritma di atas, $a_{i,j} \leftrightarrow a_{j,i}$ adalah **procedure** *swap* berikut:

procedure swap (x, y)

- (1) $temp \leftarrow x$
- (2) $x \leftarrow y$
- (3) $y \leftarrow temp$

Pada **procedure seq_transpose**, banyaknya iterasi pada *loop* (2) bergantung pada nilai i yang diberikan pada *loop* (1); banyaknya iterasi akibat kedua *loop* tersebut adalah:

$$0 + 1 + 2 + 3 + \dots + (n - 1) = (n - 1) \times n/2 = (n^2 - n)/2$$

Sementara itu *running time* **procedure swap** adalah konstan. Dengan demikian *running time* **procedure seq_transpose** adalah $O(n^2)$; nilai *running time* ini optimal.

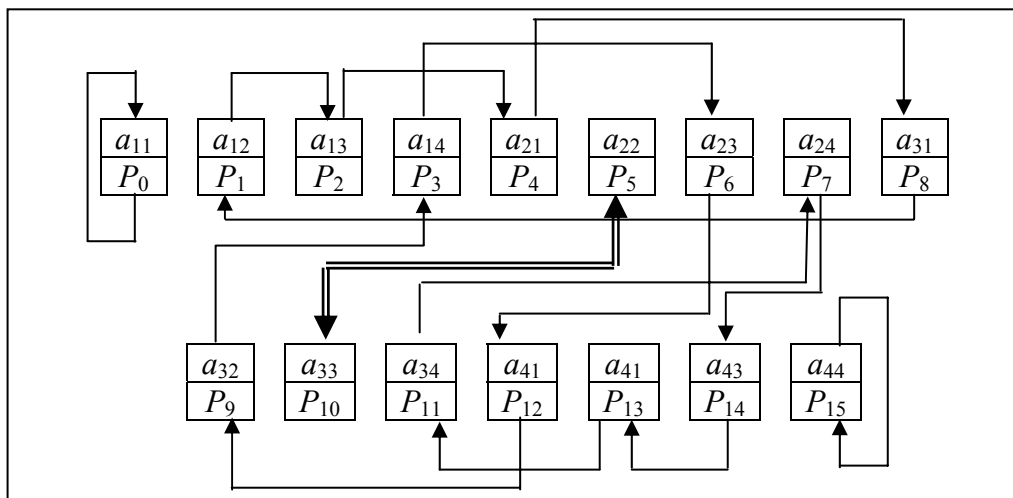
1.2. Algoritma Paralel

1.2.1. IN (Interconnection Network) Perfect Shuffle

Dalam algoritma ini, $n = 2^q$ dengan $q \in \{1, 2, 3, \dots\}$. IN *Perfect Shuffle* yang akan digunakan terdiri dari n^2 prosesor, yaitu: $P_0, P_1, P_2, \dots, P_{2^{2q}-1}$ (ingat, karena ada n^2 prosesor, $n = 2^q$, indeks prosesor pertama adalah 0, yaitu P_0 , maka indeks prosesor terakhir, atau ke- n^2 , adalah $n^2 - 1$ atau $2^{2q} - 1$, yaitu $P_{2^{2q}-1}$), yaitu sebanyak elemen matriks yang akan dicari transpose-nya, dengan $n = 2^q, q \in \{1, 2, 3, \dots\}$. Elemen a_{ij} dari matriks **A** dipegang oleh prosesor P_k , dengan $k = 2^q(i - 1) + (j - 1)$.

Contoh: Untuk $q = 2$, maka $n = 2^2 = 4$, dengan demikian matriks **A** berukuran 4×4 , dan prosesor IN *Perfect Shuffle* adalah $P_0, P_1, P_2, \dots, P_{15}$. Elemen matriks a_{23} dipegang oleh prosesor P_k dengan $k = 2^2(2 - 1) + (3 - 1) = 6$ yaitu P_6 , elemen matriks a_{32} dipegang oleh prosesor P_k dengan $k = 2^2(3 - 1) + (2 - 1) = 9$ yaitu P_9 , elemen a_{44} dipegang oleh prosesor P_k dengan $k = 2^2(4 - 1) + (4 - 1) = 15$ yaitu P_{15} .

Untuk melihat koneksi antar prosesor, indeks-indeks prosesor dinyatakan dalam biner. Untuk $q = 2$ kita mempunyai prosesor-prosesor beserta indeksnya sebagai berikut: $P_{0000}, P_{0001}, P_{0010}, P_{0011}, \dots, P_{1111}$. Prosesor P_k terhubung ke prosesor P_m , dengan k dan m adalah representasi biner, jika m diperoleh dengan **menggeser** satu posisi ke kiri secara **siklus** (*cyclic left shifting*). Dengan aturan ini $P_{0000} = P_0$ terhubung ke dirinya sendiri, $P_{0001} = P_1$ terhubung ke $P_{0010} = P_2$, $P_{0010} = P_2$ terhubung ke $P_{0100} = P_4$, $P_{0100} = P_4$ terhubung ke $P_{1000} = P_8$, dan $P_{1000} = P_8$ terhubung ke $P_{0001} = P_1$. Lihat Gambar 6.1.



Gambar 6.1. Perfect Shuffle 16 prosesor

Dengan $n = 2^q$, setelah q **pergeseran siklus** bit-bit indeks prosesor, elemen a_{ij} dari matriks A yang semula dipegang prosesor P_k , dengan $k = 2^q(i - 1) + (j - 1)$, sekarang dipegang oleh P_r , dengan $r = 2^q(j - 1) + (i - 1)$, yaitu prosesor yang semula memegang elemen a_{ji} dari matriks A .

Contoh: Kembali, misalkan $q = 2$; IN *Perfect Shuffle* terkait seperti gambar di atas. Pada gambar tersebut mula-mula $P_1 = P_{0001}$ memegang elemen a_{12} . Pada pergeseran pertama elemen a_{12} ini berpindah ke $P_2 = P_{0010}$. Pada pergeseran kedua elemen a_{12} ini akhirnya berpindah ke $P_4 = P_{0100}$, yaitu prosesor yang semula memegang a_{21} . Di lain pihak, pada pergeseran pertama elemen a_{21} yang mula-mula dipegang oleh prosesor $P_4 = P_{0100}$ berpindah ke $P_8 = P_{1000}$, dan pada pergeseran kedua elemen a_{21} ini akhirnya berpindah ke $P_1 = P_{0001}$ yang mula-mula memegang a_{12} . Perhatikan bahwa setelah 2 pergeseran ini P_0 dan P_{15} sama sekali tidak memindahkan elemen matriks yang dipegangnya sedangkan elemen yang dipegang P_5 dan P_{10} kembali kepada keadaan semula setelah dua pergeseran ini, seperti terlihat pada Gambar 6.1.

Dengan demikian, procedure transpose matriks A menggunakan IN *Perfect Shuffle* adalah sebagai berikut:

procedure *par_shuffle_transpose* (A)

- (1) **for** $i = 1$ **to** q **do**
- (2) **for** $k = 1$ **to** $2^{2q} - 2$ **do in parallel** {'-2' karena P_0 dan $P_{2^{2q}}$ tidak diikutsertakan}
- (3.1) $r \leftarrow 2^{k \bmod (2^{2q} - 1)}$
- (3.2) P_k mengirimkan elemen yang dipegangnya ke P_r
- (4) **end for**
- (5) **end for**

Langkah (3.1) dan (3.2) memerlukan waktu konstan, sedangkan iterasi (1) terdiri dari q langkah atau $\log n$ langkah mengingat $n = 2^q$. Dengan demikian *running time* **procedure *par_shuffle_transpose*** di atas adalah $t(n) = O(\log n)$. Jumlah prosesor yang digunakan adalah $p(n) = n^2$ sehingga *cost* procedure ini adalah $c(n) = O(n^2 \log n)$ yang **tidak optimal** karena *running time* procedure sekuensial adalah **lebih kecil**, yaitu $O(n^2)$.

Latihan:

(Kerjakan dan kirim hasilnya ke ajuarna@staff.gunadarma.ac.id)

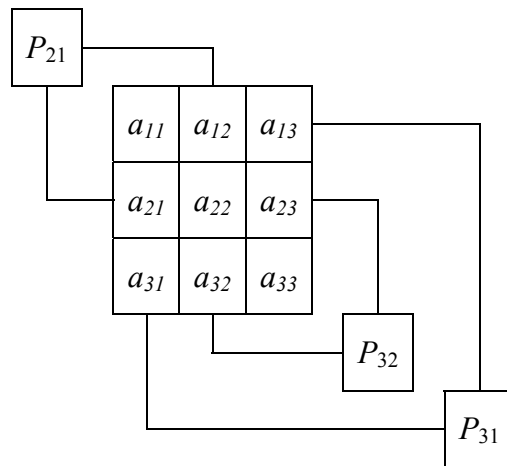
1. Periksa Gambar 6.1 dan tunjukkan dalam tabel bahwa memang benar prosesor P_k terhubung ke prosesor P_m , dengan k dan m adalah representasi biner, jika m diperoleh dengan **menggeser** satu posisi ke kiri secara **siklus** (*cyclic left shifting*).

2. Tabel di samping menyatakan elemen matriks A yang dipegang oleh masing-masing dari 16 prosesor IN *perfect shuffle* pada kondisi, $q = 1$, dan $q = 2$. Lengkapi tabel tersebut berdasarkan Gambar 6.1. Periksa, apakah kondisi akhir ($q = 2$), jika dibandingkan dengan kondisi awal, memperlihatkan peristiwa transpose matriks?

	Elemen yang dipegang		
	Awal	$q = 1$	$q = 2$
P_0	a_{11}	a_{11}	a_{11}
P_1	a_{12}	a_{31}	a_{21}
P_2	a_{13}		
\vdots			
P_{15}	a_{44}		

1.2.2. SIMD-EREW

Pada awalnya seluruh elemen matriks A berukuran $n \times n$ tersimpan di dalam *shared memory* bersama dengan informasi posisi baris dan kolomnya (indeks) (i, j) . **Tugas** setiap prosesor SIMD-EREW adalah menukarkan elemen matriks berindeks (i, j) dengan elemen matriks berindeks (j, i) . Selanjutnya adalah **fakta** bahwa dalam transposisi matriks A semua elemen matriks A berindeks (i, i) tidak mengalami transposisi. Dengan tugas dan fakta ini maka jumlah prosesor SIMD-EREW yang diperlukan adalah $p(n) = (n^2 - n)/2$. Untuk $n = 3$, proses transposisi ini diperlihatkan pada Gambar 6.2. Dalam gambar tersebut prosesor-prosesor P_{21} , P_{32} , dan P_{31} berturut-turut bertugas menukarkan elemen-elemen matriks a_{21} dengan a_{12} , a_{32} dengan a_{23} , dan a_{31} dengan a_{13} .



Gambar 6.2. Transposisi matriks A berukuran 3×3 dengan 3 prosesor.

Dengan demikian, procedure transpose matriks A menggunakan SIMD-EREW adalah sebagai berikut:

procedure *par_SIMD-EREW_transpose* (A)

- (1) **for** $i = 2$ **to** n **do in parallel**
- (2) **for** $j = 1$ **to** $i - 1$ **do in parallel**
- (3) $a_{ij} \leftrightarrow a_{j,i}$
- (4) **end for**
- (5) **end for**

Running time procedure swap (yaitu $a_{ij} \leftrightarrow a_{j,i}$) adalah konstan sehingga *running time procedure par_SIMD-EREW transpose* adalah konstan, atau $t(n) = O(1)$, sehingga *cost procedure ini* adalah $c(n) = p(n) \times t(n) = O(n^2)$ yang berarti **procedure par_SIMD-EREW transpose** adalah *cost optimal*.