Tạp chí Tin học và Điều khiển học, T.27, S.2 (2011), 107–118

GIẢI CÁC BÀI TOÁN TRÊN CÂY TOÁN TỬ ĐƯỜNG ỐNG BẰNG MA TRẬN ĐẶC TRƯNG

LÊ HUY THẬP

Viện Công nghệ thông tin, Viện Khoa học và Công nghệ Việt Nam

Abstract. The paper describes a representation of POT by $n \times n$ characteristical square matrix and therefore, every operations on the POT such as cutting of the child vertices from the father's vertex, collapsing of child vertices to the father's vertex, local cuttings, finding the optimal query schedule and the cost balance,..., on the POT that are performed easier using the algorithms and processing of array and model programming language.

Tóm tắt. Bài báo đề xuất cách biểu biễn POT bằng ma trận đặc trưng vuông $n \times n$. Từ đó mọi thao tác trên POT như tách các đỉnh con khỏi đỉnh cha, gộp các đỉnh con vào đỉnh cha, nhát cắt cục bộ, tìm lịch truy vấn tối ưu và cân bằng tải,..., cho POT sẽ được thực hiện dễ dàng khi sử dụng các thuật toán và cách xử lý mảng và ngôn ngữ lập trình hiện đại.

1. MỞ ĐẦU

Khi cần xử lý một vấn đề nào đó bằng máy tính, nếu vấn đề đó có thể chia nhỏ được và các phần đó được thực hiện song song thì sẽ rút ngắn thời gian hoàn thành, tiết kiệm các chi phí tiềm năng, giải quyết được các vấn đề lớn và phức tạp.

Các kiến trúc máy tính hiện tại đang ngày càng dựa vào khả năng song song hóa phần cứng để cải thiện hiệu suất như: Có nhiều đơn vị thực hiện, dùng các chỉ lệnh đường ống (Pipelined Instructions), đa nhân (Multi-core). Hơn nữa các thiết bị phần cứng hiện rất sẵn có và rẻ tiền.

Các hệ điều hành đa bộ xử lý quản trị các tiến trình, quản trị bộ nhớ, quản trị tài nguyên và quản trị các tệp: hoặc là mở rộng và phát triển từ những hệ đơn bộ xử lý để chạy được trên các kiến trúc song song như VMS, VNIX, hoặc là được thiết kế riêng cho các kiến trúc song song như hệ Hydra, Medusa của Carnegie Mellon University và các hệ điều hành tổng hợp được thiết kế để cài đặt được trên các kiến trúc song song khác nhau, ví dụ như MACH Multi processor.

Một số ngôn ngữ lập trình có thể dùng để lập trình song song như C^{++}, FORTRAN90, ORACLE, LOTUSNOT,..., đã được sử dụng để thể hiện các bài toán xử lý song song và phân tán.

Giải các bài toán trên cây toán tử đường ống đã được sử dụng để giải các vấn đề lớn, phức tạp như dự báo thời tiết, bão, động đất, sóng thần, mô hình sinh thái, ... Việc lập lịch tối ưu cho cây toán tử đóng góp một phần không nhỏ trong mục đích đó.

Cây toán tử là cây mà mỗi đỉnh là một toán tử, cạnh nối hai đỉnh cho biết hai hai toán tử đó có trao đổi dữ liệu. Như đã biết [5,6], một cây toán tử tương ứng một - một với một ma trận, chúng ta gọi ma trận tương ứng với cây toán tử là ma trận đặc trưng của cây đó.

Cây toán tử mà một số toán tử của nó có thể thực hiện song song, dữ liệu ra của toán tử này có thể là dữ liệu vào của toán tử kia được gọi là cây toán tử dạng đường ống POT (Pipelined Operator Tree).

Lập lịch tối ưu cho cây toán tử POT là tìm kiếm một cách phân chia hợp lý các đỉnh của cây toán tử cho các bộ xử lí để thời gian trả lời truy vấn ít nhất. Lập lịch cho cây toán tử POT là bài toán NP-khó. Bài toán loại này được đưa về dạng đơn giản hơn bằng cách dùng các thuật toán cắt các cạnh, gộp các đỉnh để chuyển từ một cây phức tạp thành các cây toán tử đơn giản hơn mà vẫn không làm mất ý nghĩa của việc song song hóa.

Cho POT $T = (V, E)$, trong đó V là tập các đỉnh, E là tập các cạnh.

Ma trận đặc trưng của POT được gọi là POM (Pipelined Operator Matrix). Khi đã có ma trận đặc trưng POM chúng ta có thể tìm lịch truy vấn tối ưu, cân bằng tải trên POT ,... bằng cách thực hiện trên ma trậ. Điều này tạo ra khả năng xử lý POT tốt hơn vì khi đã có ma trận thể hiện cây toán tử, ta dùng các thuật toán xử lý ma trận để thay cho xử lý trên cây sẽ dễ dàng và nhanh hơn.

Giải các bài toán trên cây toán tử đường ống bằng ma trận đặc trưng của nó, hiện chưa có tài liệu nào đề cập giải quyết.

2. THỂ HIỆN CÂY TOÁN TỬ POT BẰNG MA TRẬN ĐẶC TRƯNG *POM*

Cho cây toán tử đường ống POT $T = (V, E)$, trong đó,

$V = \{i | i \in N\}$ là tập các đỉnh, hay còn là toán tử;

$WT = \{t_i | i \in V\}$ là tập các trọng số tại đỉnh i (chi phí ở toán tử i);

$E = \{(i, j)\}$ là cạnh nối i với j (cạnh này tồn tại khi và chỉ khi có sự truyền số liệu từ i đến j);

$WC = \{c_{i,j} | i < j; i, j \in V\}$ là tập các trọng số trên cạnh (chi phí truyền dữ liệu từ toán tử i đến toán tử j).

Ta xây dựng ma trận đặc trưng POM của POT như sau (xem POT trên Hình 1 và POM trên Bảng 1):

1) Hàng thứ nhất và cột thứ nhất thể hiện đỉnh của POT;

2) Hàng thứ hai và cột thứ hai thể hiện trọng số đỉnh của POT;

3) Giao của hàng i với cột j là trọng số $c_{i,j}$. Khi viết $c_{i,j}$ hoặc (i, j) nghĩa là hàng i là hàng con của hàng cha j. Vì cấu trúc của POT nên trong POM mỗi con i chỉ có một cha j

duy nhất, vì vậy chỉ có một j sao cho $c_{i,j} \neq 0$ và $c_{i,k} = 0$ với mọi $k \neq j$.

Nhận xét

$c_{i,i} = 0\ \forall i$.

Nếu m là lá thì $c_{i,m} = 0\ \forall i$.

Mỗi hàng i có nhiều nhất một m để $c_{i,m} \neq 0$ (do mỗi đỉnh con i có nhiều nhất một đỉnh cha). Chẳng hạn, trên Hình 1 đỉnh con 3 chỉ có một đỉnh cha là 8 nên trong Bảng 1 chỉ có $c_{3,8} \neq 0$, còn $c_{3,j} = 0\ \forall j \neq 8$).

Mỗi cột m có thể có nhiều i để $c_{i,m} \neq 0$ (do mỗi đỉnh cha m có thể có nhiều đỉnh con). Chẳng hạn, trên Hình 1 đỉnh cha 8 chỉ có 3 đỉnh con là 3, 4, 5 nên trong Bảng 1 có $c_{3,8} = 2,\ c_{4,8} = 1$ và $c_{5,8} = 3$).

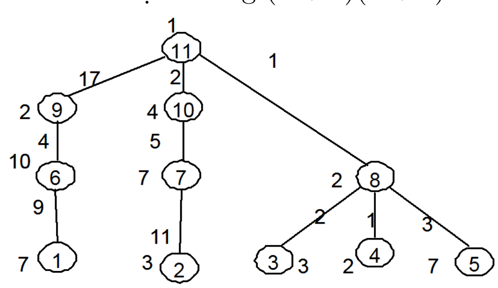
Trong POM chỉ có thể có $c_{ij} \neq 0$ khi $i < j$.

Hai hàng i và j được gọi là có liên đới với nhau nếu tồn tại dãy $c_{ik_1} \neq 0,\ c_{k_1k_2} \neq 0$, hoặc $c_{k_2k_1} \neq 0, ...,\ c_{k_mj} \neq 0$, hoặc $c_{j,k_m} \neq 0$.

Chẳng hạn, trên Hình 1 đỉnh 2 và 3 liên thông nên trong Bảng 2 có $c_{27} = 11 \neq 0;\ c_{7,10} = 5 \neq 0; c_{10,11} = 2 \neq 0; c_{11,8} = 0$ nhưng $c_{8,11} = 1 \neq 0; c_{8,3} = 0$; nhưng $c_{3,8} = 2 \neq 0$. Vậy theo quy ước các hàng 2 và 3 liên đới với nhau.

Các hàng trong POM đều liên đới với nhau, điều này tương đương với nghĩa POT bao giờ cũng là cây liên thông.

+ Nếu cây có n đỉnh thì ma POM trận vuông $(n+2)(n+2)$.



Hình 1. POT

Bảng 1. POM ứng với POT

	Hàng	1	2	3	4	5	6	7	8	9	10	11
Hàng	ti	7	3	3	2	7	10	7	2	2	4	1
1	7	0					9					
2	3		0					11				
3	3			0					2			
4	2				0				1			
5	7					0			3			
6	10						0			4		
7	7							0			5	
8	2								0			1
9	2									0		17
10	4										0	2
11	1											0

3. BÀI TOÁN LẬP LỊCH

3.1. Một số định nghĩa, thuật toán gộp, tách và khái niệm lập lịch

Định nghĩa 1. Tải tại hàng i là số

$$l_i = t_i + c_{i,m} + \sum_{k<i} c_{k,i},$$

trong đó m là hàng cha, công thức này thể hiện việc tính tải của hàng i là phép cộng cột i (trừ hàng 1 là ký hiệu hàng) và cộng với chi phí truyền thông từ hàng i lên hàng cha m nếu như i còn hàng cha m. Để biết hàng i còn hàng cha m hay không, ta chỉ cần kiểm tra có tồn tại m nào đó để $c_{i,m} \neq 0$ hay không mà thôi. Chẳng hạn trong Bảng 1 xét hàng 8 có $t_8 = 2, c_{3,8} = 2; c_{4,8} = 1; c_{5,8} = 3$; khi kểm tra hàng 8 thấy có $c_{8,11} = 1$, thì $m = 11$ là hàng cha của hàng 8, vậy $l_8 = 2 + 1 + 2 + 1 + 3 = 9$.

Định nghĩa 2. Gọi F_q là một tập con bấy kỳ các hàng i của POM, tải tại trên F_q, ký hiệu Tai(F_q), là số

$$L_q = \sum_{i \in F_k} (t_i + c_{i,m} + \sum_{k<i} c_{k,i}).$$

Chẳng hạn, trong POM ỏ Bảng 1, xét tập $F_1 = \{7, 8\}$ thì tải F_1 là

$$\begin{aligned} L_1 &= \sum_{i \in \{7,8\}} (t_i + c_{i,m} + \sum_{k<i} c_{k,i}) \\ &= t_7 + c_{7,10} + \sum_{k<7} c_{k,7} + (t_8 + c_{8,11} + \sum_{k<8} c_{k,8}) \\ &= (7 + 5 + 11) + (2 + 1 + 2 + 1 + 3) = 35. \end{aligned}$$

Định nghĩa 3. Lịch truy vấn trên POM là việc phân chia các hàng của POM thành p tập $(F_l, ..., F_p)$ (như vậy F_q là tập các toán tử được định vị trên bộ xử lý q), sao cho $L = \max_q L_q$ là nhỏ nhất.

Định nghĩa 4. Phép $Gop(i, m)$, là cách gộp hàng con i vào hàng cha m với $t_m^{new} = t_m^{old} + t_i$ và nếu các hàng i và m đã liên kết ([5,6]) với những hàng nào thì m sẽ liên kết với những hàng đó.

Thuật toán 1. $Gop(i, m)$ gộp hàng con i vào hàng cha m

Giả sử POM có n hàng (n cột)
Input: Hàng con i hàng cha m
Ouput: POM đã gộp hàng con i vào hàng cha m
Begin
  $t_m = t_m + t_i$
    For $k = 1$ to n

$c_{m,k} += c_{i,k}$

End For

Ghi nhãn hàng con i vào bên cạnh hàng cha m

Xóa hàng i và cột i

End

Độ phức tạp thuật toán Gop là $O(n)$.

Ví dụ 1. Áp dụng Thuật toán 1 vào POM ở Bảng 1 khi gộp hàng con 1 vào hàng cha 6 và gộp hàng con 9 vào hàng cha 11, ta được POM ở Bảng 2.

Bảng 2. POM sau khi gộp

	Hàng	2	3	4	5	6,1	7	8	10	11,9
Hàng	t_i	3	3	2	7	17	7	2	4	3
2	3	0					11			
3	3		0					2		
4	2			0				1		
5	7				0			5		
6,1	17					0				4
7	7						0		5	
8	2							0		1
10	4								0	2
11,9	3									0

Định nghĩa 5. Phép $Tach(i, m)$ cho phép tách hàng con i ra khỏi hàng cha m. Khi đó:

$t_m^{new} = t_m^{old} + c_{i,m}$,

$t_i^{new} = t_i^{old} + c_{i,m}$.

Phép tách sẽ tạo $POM_{\#}$ chứa hàng con i và $POM_{\#\#}$ chứa hàng cha m.

Thuật toán 2. $Tach(i, m)$ tách hàng con i ra khỏi hàng cha m

Giả sử đã cho POM. Khi tách hàng con i ra khỏi hàng cha m sẽ tạo ra $POM_{\#}$ chứa hàng con i và $POM_{\#\#}$ chứa hàng cha m với số hàng của $POM_{\#}$ là Card($POM_{\#}$) và của $POM_{\#\#}$ là Card($POM_{\#\#}$) tương ứng.

Nếu Card($POM_{\#}$) > 1 thì ta có thể tiếp tục tách $POM_{\#}$ nếu cần (tương tự cho Card($POM_{\#\#}$) > 1).

Input: POM cần tách

Ouput: Các POM đã được tách

Begin

$t_m = t_m + c_{i,m}$ { i là hàng con, m là hàng cha}

$t_i = t_i + c_{i,m}$ { i là hàng con, m là hàng cha}

End

Ví dụ 2. Áp dụng hai thuật toán tách và gộp.

Tách POM

Từ POM ở Hình 1, $Tach(8,11)$ vah $Tach(9,11)$ ta được các $POM\ _1$, $POM\ _2$, $POM\ _3$ như các Bảng 3, Bảng 4, Bảng 5.

Bảng 3. $POM\ _1$

	Hàng	3	4	5	8
Hàng	t_i	3	2	7	3
3	3	0			2
4	2		0		1
5	7			0	3
8	3				0

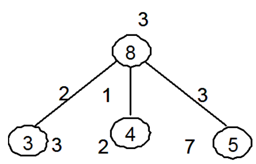
Bảng 4. $POM\ _2$

	Hàng	1	6	9
Hàng	t_i	7	10	19
1	7	0	9	
6	10		0	4
9	19			0

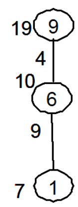
Bảng 5. $POM\ _3$

	Hàng	2	7	10	11
Hàng	t_i	3	7	4	19
2	3	0	11		
7	7		0	5	
10	4			0	2
11	19				0

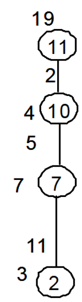
Các POT$_s$ tương ứng trên Hình 2, 3, 4.



Hình 2. *POT$_1$*



Hình 3. *POT$_2$*



Hình 4. *POT$_3$*

Gộp các $POM\ _1$ *,* $POM\ _2$ *và* $POM\ _3$

Bây giờ, ta dùng thuật toán $Gop(i,m)$ để gộp $POM\ _1$, $POM\ _2$ và $POM\ _3$ và cho thực hiện trên các bộ xử lí p_1, p_2 và p_3 với giả thiết khả năng xử lí của các bộ xử lí này là không hạn chế.

Ta sẽ gộp $POM\ _2$, hàng 1 là hàng con của hàng cha 6, $Gop(1,6)$ được $POM\ _{2\#}$ ở B6. Trong $POM\ _{2\#}$ hàng 6,1 là hàng con của hàng cha 9, $Gop((1,6),9)$ ta được POM 2a ở B7.

Bảng 6. $POM\ _{2\#}$

	Hàng	6,1	9
Hàng	ti	17	19
6,1	17	0	4
9	19		0

Bảng 7. $POM\ _{2a}$

	Hàng	9,(6,1)
Hàng	ti	23
9,(6,1)	23	0

Tương tự ta có $POM\ _{1a}$ và $POM\ _{3a}$ trên B8 và B9 tương ứng

Bảng 8. $POM\ _{1a}$

	Hàng	((8,3),4),5
Hàng	ti	15
(((8,3),4),5)	15	0

Bảng 9. $POM\ _{3a}$

	Hàng	11,(10, (7,2))
Hàng	t_i	33
11,(10, (7,2))	33	0

Rõ ràng, tải của $POM\ _{1a}$ là 15, tải của $POM\ _{2a}$ là 23, tải của $POM\ _{3a}$ là 33.

Lịch truy vấn ở đây gồm 3 bộ xử lý p_1, p_2 và p_3 để xử lí $POM\ _{1a}$, $POM\ _{2a}$, $POM\ _{3a}$ tương ứng và thời gian trả lời của POM là $L = 33$.

3.2. Mở rộng bài toán lập lịch trên *POM*

Lập lịch tối ưu cho *POM* là tìm kiếm một cách phân chia các hàng cột (tức là các toán tử) của nó cho các bộ xử lí để thời gian trả lời truy vấn ít nhất. Bài toán được thực hiện bằng cách dùng các thuật toán cắt các liên kết hàng cột và gộp các hàng cột để chuyển từ *POM* phức tạp thành *POM* đơn giản hơn mà vẫn không làm mất ý nghĩa của việc song song và tối ưu hóa. Việc xử lý bằng máy tính trên *POM* bao giờ cũng đơn giản hơn trên POT.

3.2.1. Loại bỏ $c_{i,j}$ truyền thông lớn và POM tiền xử lý

Như chúng ta đã biết trọng số $c_{i,j}$ chính là chi phí truyền thông giữa toán tử i và toán tử j.

Định nghĩa 6. $c_{i,j}$ của *POM* được gọi là truyền thông lớn nếu $c_{i,j} \geq t_i + \sum_{k\neq j} c_{i,k}$ hoặc $c_{j,i} \geq t_j + \sum_{k\neq i} c_{k,j}$.

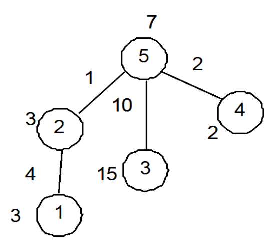
Ví dụ 3. Trên *POM* ở Bảng 10, $c_{1,2} = 4 > t_1 = 3$, $c_{3,5} = 10 < 15 = t_3$ nhưng, $c_{3,5} = 10 = t_5 + c_{2,5} + c_{4,5} = 7 + 1 + 2$ nên $c_{1,2}$ và $c_{3,5}$ truyền thông lớn.

Bảng 10. POM có trọng số lớn

	Hàng	1	2	3	4	5
Hàng	t_i	3	3	15	2	7
1	3	0	4	0	0	0
2	3		0	0	0	1
3	15			0	0	10
4	2				0	2
5	7					0

Bảng 11. POM tiền xử lí

	Hàng	1	2,1	4	5,3
Hàng	t	3	6	2	22
1	3	0	4	0	0
2,1	6		0	0	1
4	2			0	2
5,3	22				0



Hình 5. POT ứng với *POM* trên Bảng 10

Nhận xét 1. Nếu $c_{i,j}$ có truyền thông lớn của *POM*, và p là số bộ xử lý, thì luôn luôn tồn tại một POM' tối ưu của *POM* với p bộ xử lý, sao cho các toán tử i và j được thực thi trên cùng một bộ xử lý, tức là hai hàng này được gộp lại [1, 2, 4, 5]. Như vậy ta sẽ tìm và loại bỏ các $c_{i,j}$ truyền thông lớn. Với *POM* đã loại các $c_{i,j}$ truyền thông lớn, được gọi là *POM* đã qua tiền xử lý. Để tạo ra *POM* tiền xử lý, ta dùng thuật toán Tien_XL_*POM* sau.

Thuật toán 3. Tien_XL_*POM*

Input: *POM* cho trước chưa qua tiền xử lí

Output: POM tiền xử lý

For $i = 1$ to n

For $j = 1$ to n

If $(c_{i,j} \geq t_i + \sum_{k \neq j} c_{i,k})$ then

$Gop(i, j)$ {Gọi thuật toán $Gop(i, j)$ để gộp i vào j}

End If

End For

End For

Độ phức tạp thuật toán là $O(n^3)$.

Ví dụ 4. Với của POM ở Bảng 10, vì $c_{1,2}$ và $c_{3,5}$ truyền thông lớn, nên gộp 1 vào 2 và 3 vào 5 ta được POM tiền xử lý như Bảng 11.

3.2.2. Cân bằng tải và phân chia công việc

Cân bằng tải

Một POM được gọi là tối ưu về cân bằng tải nếu mọi bộ xử lý tham gia trong quy trình xử lý POM có thời gian thực hiện gần như nhau.

Phân chia công việc

Thuật toán phân chia công việc để tạo ra cân bằng tải giữa các bộ xử lý.

Giả sử có p bộ xử lý, và tập các công việc $W = \{w_l, ..., w_n\}$ có thời gian thực hiện là $T = \{t_1, ..., t_n\}$ tương ứng. Công việc w_i phải được thực hiện trọn vẹn trên bộ xử lý bất kỳ với thời gian t_i.

Nội dung thuật toán

Tìm công việc w_i có thời gian thực hiện t_i lớn nhất trong các công việc chưa được phân công và giao cho bộ xử lý hiện có tải ít nhất. Quá trình sẽ được lặp lại cho đến khi không còn công việc nữa.

Thuật toán 4. Chia_Viec

Input : W, T

Output: Phân hoạch $\{F_1, F_2, ..., F_p\}$

Thuật toán:

Begin

$\{F_1, F_2, ..., F_p\} \leftarrow \phi$

$W \leftarrow \{w_1, ..., w_n\}$

$T \leftarrow \{t_1, ..., t_n\}$

Do

Select $F_i : Tai(F_i) = \min_{1 \leq k \leq p} Tai(F_k)$

Select $w_j : \max_{w_k \in W} T$

$F_i \leftarrow F_i \cup \{w_j\}$
$W \leftarrow W \cup \{w_j\}$
While $W \neq \phi$
Save $\{F_1, ..., F_p\}$

End

Thuật toán này có độ phức tạp $O(n^2)$.

Thuật toán bảo đảm cân bằng tải giữa các bộ xử lý nên thường được sử dụng kết hợp với các thuật toán khác đề cho những kết quả tốt hơn.

3.2.3. Thuật toán TachGop dựa vào trọng số

Thuật toán tiến hành thực hiện việc gộp các hàng và tách các hàng dựa trên thông tin về trọng số của hàng con và trọng số $c_{i,j}$. Giả sử cho trước hằng số α, với $\alpha > 1$, ta sẽ sử dụng thuật toán $Tach(i, m)$ nếu $t_i > \alpha c_{i,m}$ và sử dụng thuật toán toán $Gop(i, m)$ trong trường hợp ngược lại.

Thuật toán 5. TachGop

Input : POM tiền xử lý và tham số $\alpha > l$
Output: Phân hoạch $\{POM\ _1, POM_2, ...\}$
Thuật toán:
While <Còn hàng cha m có hàng con i >
If $t_i > \alpha c_{i,m}$ Then
$Tach(i, m)$
Else
$Gop(i, m)$
End If
End While
Save $\{POM_1, POM_2, ...\}$

Nhận xét

+ Thuật toán có độ phức tạp $O(n)$, n là số hàng của POM tiền xử lý.
+ Kết quả thuật toán là một tập các POM với số lượng không thể đoán trước được nên thông thường thuật toán này sẽ cùng đi đôi với thuật toán phân chia công việc để phân phối các POM này cho các bộ xử lý.
+ Thuật toán chỉ xem xét sử dụng toán tử Gop hay $Tach$ cho một hàng con và hàng cha của nó nên quyết định này độc lập với trọng số của hàng cha, do đó trong một số trường hợp sẽ làm tăng trọng số của hàng cha lên một cách đáng kể.

3.3. Ví dụ tổng hợp

Cho POM như bảng sau.

Bảng 12. Dữ liệu của POM

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	t_i	7	3	3	2	7	12	10	2	2	1	5	10	7	2	1	2	4	2	1	5	5
1	7	0											6									
2	3		0											11								
3	3			0											2							
4	2				0										1							
5	7					0									3							
6	12						0														5	
7	10							0				1										
8	2								0			1										
9	2									0									12			
10	1										0					1						
11	5											0							2			
12	10												0				4					
13	7													0				5				
14	2														0					1		
15	1															0			5			
16	2																0			17		
17	4																	0		2		
18	2																		0		3	
19	1																			0		2
20	5																				0	1
21	5																					0

Áp dụng thuật toán Thuật toán 1, TachGop với $\alpha = 3,56$, ta nhận được các kết quả sau.

Bảng 13. POM đã được tiền xử lý

		1	3	4	5	6	7	8	11	12	13,2	14	17	18,9,15,10	19,16	20	21
	t_i	7	3	2	7	12	10	2	5	10	10	2	4	6	3	5	5
1	7	0								6							
3	3		0									2					
4	2			0								1					
5	7				0							3					
6	12					0										5	
7	10						0		1								
8	2							0	1								
11	5								0					2			
12	10									0					4		
13,2	10										0		5				
14	2											0			1		
17	4												0		2		
18,9,15,10	6													0		3	
19,16	3														0		2
20	5															0	1
21	5																0

POM1	Hàng	1	12	(19,16) cha 12 cha 12	T?i
Hàng	ti	7	10		21
1	7				
12	10			4	
T?i	**21**				

POM2	Hàng	-13,2	17	(19,16) cha 17	Tải
Hàng	t_i	10	4		**16**
-13,2	10				
17	4			**2**	
Tải	**16**				

POM3	Hàng	3	4	5	14	(19,16) cha 14	Tai
Hàng	ti	3	2	7	2		**15**
3	3						
4	2						
5	7						
14	2					**1**	
Tải	**15**						

POM4	Hàng	7	11 cha 7	Tải
Hàng	ti	10		11
7	10		1	
Tải	**11**			

POM5		8	11	18,9,15,10 cha 11	Tải
	ti	2	5		**10**
8	2				
11	5			**2**	
7 con 11			**1**		
Tải	**10**				

POM6		6	18,9,15,10	20	21 cha 20	Tải
	ti	12	6	5		26
6	12					
18,9,15,10	6					
20	5		2			
11 con			2			
18,9,15,10						
Tải	26					

POM7		(19,16)	21	Tải
	ti	3	5	16
(19,16)	3			
21	5			
20 con 21			1	
12 con (19,16)		4		
14 con (19,16)		1		
17 con (19,16)		2		
Tải	16			

Hoặc viết gọn hơn như bảng sau.

STT(POM)	Hàng trong POM	Tải	STT(POM)	Hàng trong POM	Tải
1	{1,12}	21	5	{8,11}	10
2	{(13,2),17}	16	6	{(18,9,15,10),20}	26
3	{3,4,5,14}	15	7	{(19,16),21}	16
4	{10}	11			

Giả sử chỉ có 4 bộ xử lý được đánh số như sau P_0, P_1, P_2, P_3. Bây giờ áp dụng Thuật toán 4 ta có:

Lần chia thứ nhất:

P0 = POM6= {(18,9,15,10) , 20}	Tải(P0)=26	P1 = POM1 = {1,12}	Tải(P1)=21
P2 = PoM2= {(13,2),17}	Tải(P2)=16	P3 = POM7= {(19, 16) , 21}	Tải(P3)=16

Lần chia thứ hai:

$P2 = \{POM2, POM3\}$ $= \{\{(13,2),17\},\{3,4,5,14\}\}$	Tải(P2)=16+15= 31	P3 = {POM7, POM4} = {{(19, 16) , 21},{10}	Tải(P3)=16+11= 27
$P1 = \{POM1, POM5\}$ = {{1,12},{8,11}}	Tải(P1)=21+10=31	P0 = {POM6} = {(18,9,15,10) , 20}	Tải(P0)=26

4.KẾT LUẬN

Chuyển POT sang POM, từ đó dùng các phép tính ma trận để tiến hành gộp, tách và lập lịch truy vấn có thể thực hiện hoàn toàn trên mảng thông qua phần tử mảng. Việc phân bố các toán tử cho các bộ xử lý được thực hiện thông qua thuật toán phân chia công việc. Vấn đề sẽ phức tạp hơn khi các bộ xử lý lại được phân phối trên các mạng khác nhau, vì khi đó các $c_{i,j}$ không đơn thuần chỉ là truyền dữ liệu giữa các bộ xử lý mà còn các thông tin phụ trợ khác.

Lập trình tự động tìm và rút trích xâu con để tìm ra các toán tử của SQL và cho tương ứng với các toán tử đại số quan hệ, chuyển sang câu vấn tin đại số, từ đó xây dựng thành POM không còn qua POT và áp dụng các thuật toán đã được trình bày. Ứng dụng cho một số bài toán trong thực tế dựa trên POM và các thuật toán đã được chỉ ra trên đó...

TÀI LIỆU THAM KHẢO

[1] Barry Wlkingson, Michael Allen, *Parallel Programming, Technique and Applications Using Netwworked Workstations and Parallel Computers*, Prentice Hall New Jersey, 1999.

[2] Đoàn Văn Ban, Nguyễn Mậu Hân, *Xử lý song song và phân tán*, NXB Khoa học và Kỹ thuật, 2006.

[3] Robert Sedgewick, *Cẩm nang thuật toán*, Vol.1, Vol.2, NXB Khoa học và Kĩ thuật, 2001.

[4] Lê Huy Thập, *Giáo trình Kỹ thuật lập trình*, Tập 1, NXB Khoa học Tự nhiên và Công nghệ, 2008.

[5] Lê Huy Thập, *Cơ sở lý thuyết song song*, NXB Thông tin và Truyền thông, 2010.

[6] Seyed H. Roo, *Parallel processing and Parallel Algorithms, Theory and Coputation*, Springer, 1999.

[7] http://www.gralib.hcmuns.edu.vn/sachmoi/2011/02-11/cslythuyet.pdf

[8] http://gralib.hcmuns.edu.vn/sachmoi/2007/08-07/xulysongsong.pdf

Nhận bài ngày 24 - 3 - 2011
Nhận lại sau sửa ngày 27 - 6 -2011