TẠP CHÍ TIN HỌC VÀ ĐIỀU KHIỂN HỌC, TẬP I (1991), Số 2 (1–8)

THUẬT TOÁN SYSTOLIC CHO CÁC PHÉP TOÁN CƠ SỞ DỮ LIỆU QUAN HỆ

ĐOÀN VĂN BAN
Viện Tin học
NGUYỄN NGỌC THUẦN
Đại học Sư phạm Vinh

I – GIỚI THIỆU

Cấu trúc Systolic (thuật toán Systolic) được H. T. Kung và C. E. Leiserson [4] đề xuất năm 1978. Trước đó đã có nhiều công trình đề cập đến cấu trúc tương tự cấu trúc Systolic, song H. T. Kung là người đầu tiên chỉ ra rằng những thiết kế theo kiểu Systolic: các phần tử xử lý được liên kết với nhau theo quan hệ láng giềng là cách tốt nhất trong việc tận dụng kỹ nghệ VLSI để tạo thành những hệ xử lý có tốc độ cao và là cấu trúc thích hợp cho các quá trình tính toán. Cấu trúc Systolic đã được sử dụng rất hữu hiệu trong nhiều lĩnh vực như: xử lý ảnh [5], qui hoạch tuyến tính và nhất là các phép tính ma trận [6], trong tìm kiếm sắp xếp thông tin [1, 2],...

Trong bài này, chúng tôi xây dựng một số thuật toán Systolic để thực hiện một số phép toán cơ sở của mô hình dữ liệu quan hệ do Codd đưa ra [3]: phép tính tích Đề-các, kết nối và tìm kiếm sắp xếp. Các phép toán này đều thực hiện với độ phức tạp tuyến tính. Điều này có ý nghĩa quan trọng trong cài đặt những hệ cơ sở dữ liệu lớn và thích hợp cho những máy cơ sở dữ liệu, những kiến trúc máy tính song song. Đặc biệt, cùng với transputer và OCCAM [2], cấu trúc Systolic dễ dàng cài đặt tất cả các phép toán cơ sở của mô hình dữ liệu quan hệ.

II – MỘT SỐ KHÁI NIỆM CƠ SỞ

1. Cấu trúc và các tính chất Systolic

Cấu trúc Systolic gồm một tập các bộ xử lý, trong đó mỗi bộ chỉ trao đổi thông tin với những bộ lân cận gần nhất. Thông thường, dữ liệu được bơm qua tất cả các bộ xử lý và hầu như các bộ trong hệ là giống nhau, ngoại trừ một số ít ở biên. Các bộ xử lý có thể liên kết với nhau theo nhiều kiểu: tuyến tính, bảng, hình sao, mạng,... [4]. Để tiện theo dõi, chúng tôi xin giới thiệu một số đặc trưng cơ bản của cấu trúc Systolic.

a. Hệ Systolic là một hệ gồm các bộ xử lý được nối với nhau theo một kiểu chính tắc, các bộ xử lý được tổ chức đồng đều theo chức năng.

b. Các bộ xử lý quan hệ với nhau theo quan hệ láng giềng gần nhất. Tính chất láng giềng gần nhất đóng vai trò quan trọng trong việc phân biệt hệ Systolic với các hệ đa xử lý khác. Do

cấu trúc chính tắc và truyền thông tin trong khoảng cách ngắn, nên hệ Systolic rất thích hợp cho việc cài đặt VLSI.

c. Hầu hết các bộ xử lý trong hệ là cùng chủng loại. Trong các hệ đã được đề xuất, thì chỉ có một ít bộ xử lý khác kiểu, điều đó rất quan trọng cho việc sản xuất và thiết kế.

d. Dữ liệu được chuyển đổi trong hệ giống như hệ tuần hoàn. Mỗi một bộ xử lý trong hệ nhận dữ liệu vào từ bộ xử lý láng giềng, thực hiện việc xử lý và đưa dữ liệu ra, làm đầu vào cho bộ xử lý bên cạnh, rồi lại tiếp tục nhận dữ liệu vào mới, và xử lý tiếp ...

e. Dữ liệu được truyền đi qua suốt cả hệ, tất nhiên có thể tạo ra một số vòng lặp dữ liệu trong hệ.

Từ mô tả các tính chất ở trên, hệ Systolic có những ưu điểm sau:

Chúng ta có thể tiến hành nhiều tính toán trên mỗi một dữ liệu vào của hệ, sử dụng trực tiếp những kết quả tính toán trước đó chuyển tới. Do không cần lưu trữ dữ liệu, nên tránh được việc truy nhập thường xuyên vào bộ nhớ, làm tăng thêm tốc độ tính toán.

Hệ Systolic có thể thực hiện đồng thời một số lớn các phép toán trên các bộ xử lý mà dòng điều khiển là đơn giản và thuần nhất.

Trong những năm gần đây, nhiều thuật toán Systolic đã được xây dựng và một số mô hình xử lý kiểu Systolic đã được thiết kế [1, 4, 5]. Trong tương lai, máy tính phải có cơ chế Systolic để giải quyết những bài toán cực lớn, cần xử lý nhanh cũng như giải quyết các vấn đề trong các lĩnh vực tính toán như:

- Các bài toán về đại số tuyến tính,
- Các hệ xử lý tín hiệu, xử lý ảnh,
- Các bài toán phi số, hệ hỏi đáp trong hệ cơ sở dữ liệu, sắp xếp, tìm kiếm,...

2. Quan hệ

Cho các tập $E_1, E_2, ..., E_n$ (không nhất thiết phải khác nhau). Ta nói rằng R là một quan hệ trên n tập vừa nêu, nếu nó là tập các n-bộ $(e_1, e_2, ..., e_n)$ sao cho $e_i \in E_i$ $(i = 1, 2, ..., n)$. Các tập $E_1, E_2, ..., E_n$ gọi là các miền của R, còn n gọi là bậc của R. Số lượng bộ của R gọi là lực lượng (bản số). Thuộc tính là tính chất của đối tượng, lấy giá trị từ một miền nào đó. Mỗi tập E_i ở trên có thể xem là một miền của thuộc tính A_i nào đó.

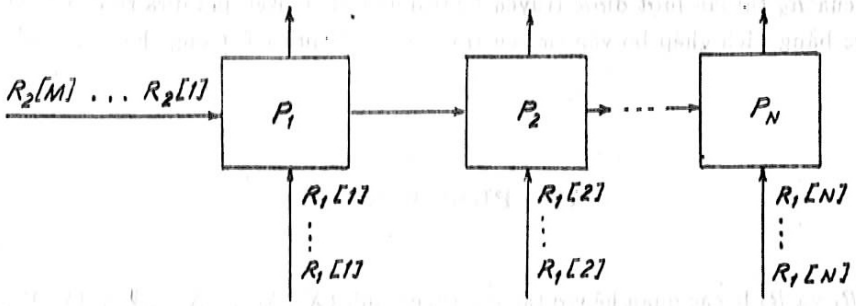
Từ đây về sau, nếu không có gì giải thích thêm, ta ký hiệu $R, R_1, R_2,$ là các quan hệ, bộ thứ k của R được ký hiệu là $R[k]$ (theo thứ tự lưu trữ).

III – TÍCH ĐỀ-CÁC CỦA CÁC QUAN HỆ

Giả sử R_1, R_2 là các quan hệ, các số n, m, N, M tương ứng là bậc, bản số của chúng. Tích Đề-các của R_1 và R_2, ký hiệu $R_1 \times R_2$, là tập $(M \times N)$ bộ, mà n thành phần đầu tiên của mỗi bộ thuộc R_1 còn m thành phần còn lại là một bộ thuộc R_2, nghĩa là

$$R_1 \times R_2 = \{t \mid t = (s_1, s_2) : s_1 \in R_1,\ s_2 \in R_2\}$$

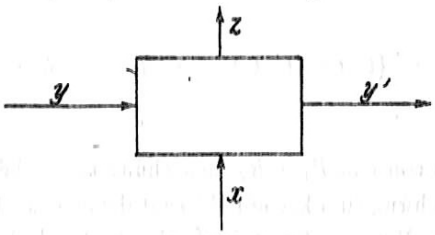
Thông thường, để tính tích Đề-các $R_1 \times R_2$ chúng ta cần tiến hành $(N \times M)$ phép toán. Sử dụng cấu trúc Systolic, chúng ta có thể thực hiện $R_1 \times R_2$ với N bộ xử lý song song ghép nối như trong hình 1. Ở đây, các bộ của R_1 được đưa vào từ các bộ xử lý tương ứng theo dưới lên, còn các bộ của R_2 lần lượt chuyển vào các bộ xử lý từ trái qua phải theo từng nhịp.



Hình 1: Cấu trúc Systolic tính tích Đề-các

Nhịp thứ nhất, bộ thứ nhất của R_2 ($R_2[1]$) được đưa vào bộ xử lý thứ nhất. Nhịp thứ hai, bộ thứ nhất của R_2 ($R_2[1]$) ra khỏi bộ xử lý thứ nhất P_1 và đi thẳng vào bộ xử lý thứ hai P_2, đồng thời bộ thứ hai R_2 ($R_2[2]$) được đưa vào bộ xử lý thứ nhất P_1, quá trình như thế cứ tiếp tục cho tới khi bộ cuối cùng $R_2[M]$ được đưa vào bộ xử lý cuối cùng P_N để xử lý thì dừng.

Hoạt động của mỗi bộ xử lý được mô tả như trong hình 2.



Hình 2: Hoạt động của bộ xử lý tính tích Đề-các

Sau khi các đầu vào x và y được nạp vào bộ xử lý, bộ z được tạo nên do việc ghép x và y và được đẩy ra khỏi bộ xử lý về phía trên. Phần tử y không thay đổi, được đưa sang bên phải. Dữ liệu ra khỏi bộ xử lý được xác định như sau:

$$y' := y$$
$$z := x \odot y$$

Theo hình 1 và 2, các bộ của $R_1 \times R_2$ tạo thành một ma trận P hàng N cột. Các bộ thu được ở mỗi nhịp tạo thành một hàng của ma trận đó, và do vậy đó là một ma trận M chéo, nghĩa là chỉ có M đường chéo gồm các bộ của $R_1 \times R_2$, còn các phần tử khác là rỗng, số nhịp của hệ chính là độ phức tạp thời gian và cũng là số hàng P của ma trận kết quả, $P = M + N - 1$. Vậy để thu được $N \times M$ bộ của $R_1 \times R_2$ bằng cách sử dụng N bộ xử lý song song theo cấu trúc Systolic thì số phép toán cần thực hiện là $N + M - 1$, ít hơn đáng kể so với các cách tính truyền thống $(M \times N)$.

Từ thuật toán Systolic được mô tả ở hình 1, chúng ta nhận thấy dữ liệu vào từ dưới lên

tại bộ xử lý thứ i ở mỗi nhịp đều là $R_1[i]$ $(i = 1, 2, ..., N)$. Do vậy, chúng ta có thể cải tiến thuật toán trên như sau:

Mỗi bộ xử lý P_i có một thanh ghi được dùng để lưu trữ bộ thứ i của R_1. Khi hệ thống bắt đầu thực hiện tính tích Đề-các của R_1, R_2 thì nạp các bộ $R_1[i]$ vào thanh ghi tương ứng của P_i, còn các bộ của R_2 thì lần lượt được truyền từ trái qua phải. Kết quả đưa ra ở mỗi nhịp là một bộ thu được bằng cách ghép bộ vào từ bên trái với bộ đã lưu trữ trong thanh ghi của bộ xử lý đó.

IV – PHÉP KẾT NỐI

Giả sử R_1 và R_2 là các quan hệ với tập các thuộc tính $\{X_1, X_2, ..., X_j, ...\}$ và $\{Y_1.Y_2, ..., Y_k, ...\}$ tương ứng. Phép θ-kết nối của quan hệ R_1 theo thuộc tính x_j với quan hệ R_2 theo thuộc tính y_k, ký hiệu

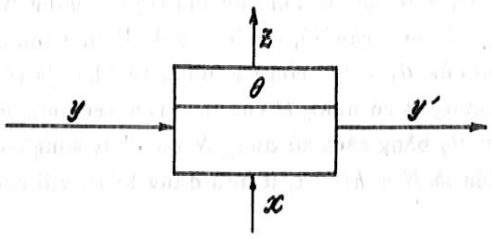
$$R_1 \underset{X_j\ \theta\ Y_k}{\bowtie} R_2$$

là tập tất cả các bộ t sao cho t là dãy ghép một bộ t_1 thuộc R_1 với một bộ t_2 thuộc R_2, trong đó thành phần thứ j của t_1 và thành phần thứ k của t_2 thỏa mãm điều kiện θ-kết nối, nghĩa là nó là tập các bộ của tích Đề-các $R_1 \times R_2$ thỏa mãn điều kiện θ-kết nối (θ là điều kiện kết nối)

$$R_1 \underset{X_j\ \theta\ Y_k}{\bowtie} R_2 = \{t \mid t = (t_1, t_2) : t_1 \in R_1, t_2 \in R_2, t_1[X_j]\ \theta\ t_2[Y_k]\}$$

Vì θ-kết nối là một tập con của $R_1 \times R_2$, nên chúng ta có thể cải tiến tuật toán tính tích Đề-các, đưa thêm phần kiểm chứng tính kết nối khi xuất dữ liệu ra. Thuật toán Systolic cho phép θ-kết nối được mô tả như sau: Ngay từ đầu, tại mỗi thanh ghi của bộ xử lý chúng ta lưu trữ sẵn một bộ tương ứng của R_1, đồng thời tại mỗi bộ xử lý chúng ta sử dụng thêm một thanh ghi để lưu trữ phép toán θ-kết nối. Sự hoạt động tiếp theo của hệ cũng giống như sự hoạt động đã được mô tả ở phần 3., chỉ khác một điều là phải kiểm tra điều kiện trước khi kết ghép và xuất thông tin. Các bộ của R_2 được lần lượt đưa vào các bộ xử lý, tại đây phép kiểm chứng được thực hiện đối với hai bộ của R_1 và R_2. Nếu $t_1[X_j]$ và $t_2[Y_k]$ thỏa mãn điều kiện θ-kết nối, thì kết quả của việc ghép chúng lại được đưa ra ở phía trên của mỗi bộ xử lý, trường hợp ngược lại thì kết quả được đưa ra là rỗng (ký hiệu là Λ).

Hoạt động của mỗi bộ xử lý trong hệ Systolic thực hiện phép kết nối được mô tả như sau:



Hình 3: Hoạt động của θ-kết nối

IF $x\ \theta\ y$ THEN $z := x \odot y$ ELSE $z := \Lambda$;
$y' := y$.

Từ cách phân tích ở mục 3, chúng ta nhận thấy rằng các bộ của $R_1 \underset{\theta}{\bowtie} R_2$ chỉ khác rỗng ở một số vị trí trên M đường chéo của $R_1 \times R_2$ và thời gian cần thiết để tính chúng là tuyến tính đối với dữ liệu.

V – THUẬT TOÁN SẮP XẾP

Việc sắp xếp các thành phần của một quan hệ dữ liệu, nhất là những quan hệ dữ liệu lớn, theo một thuộc tính nào đấy là rất quan trọng, giúp cho việc xử lý thông tin được nhanh và có hiệu quả hơn. Dựa vào thuật toán sắp xếp tuần tự với độ phức tạp $O(N \log N)$ của Miranker [7], chúng tôi xây dựng hệ Systolic gồm N bộ xử lý song song để sắp xếp các quan hệ dữ liệu theo kiểu lan truyền (pipelined) với độ phức tạp $O(N)$, trong đó N là độ dài dữ liệu.

Giả sử quan hệ R gồm N bộ cần được sắp. Có thể giả thiết N là chẵn, nếu N lẻ thì bổ sung thêm một bộ gồm các thuộc tính có giá trị cực đại trong miền xác định. Bộ sắp (sorter) được xây dựng sẽ cần $N/2$ bộ xử lý. Việc sắp xếp các thành phần của R được tiến hành qua 2 giai đoạn là giai đoạn vào và giai đoạn ra

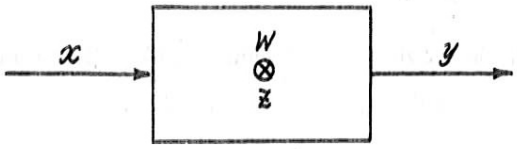
Hình 4: Thuật toán sắp xếp

⊗ Phép so sánh, ⟶ Dòng dữ liệu vào, ⟵ Dòng dữ liệu ra

Giai đoạn vào:

Chúng ta giả thiết là sắp xếp theo thứ tự tăng dần (theo thuộc tính bất kỳ nào đó). Mỗi bộ xử lý của bộ sắp có hai thanh ghi, được gọi là thanh ghi trên và thanh ghi dưới, sử dụng để lưu trữ dữ liệu. Bắt đầu khởi động hệ thống, các thanh ghi được nạp giá trị lớn nhất có thể lưu trữ được (ký hiệu là ∞) của thuộc tính cần sắp. Ngoài ra cần một thanh ghi cho bộ đếm và một thanh ghi lưu trữ trạng thái của mỗi bộ xử lý, điều khiển việc chuyển đổi dòng dữ liệu. Trong quá trình thực hiện, các bộ xử lý tiến hành so sánh hai giá trị trong hai thanh ghi trên và dưới, giá trị lớn hơn sẽ được đưa khỏi bộ xử lý và chuyển thành dữ liệu vào của bộ xử lý đứng ngay bên phải, nhường chỗ cho kết quả đối sánh từ bộ xử lý bên trái. Nếu hai giá trị ở hai thanh ghi này bằng nhau thì giá trị ở thanh ghi trên đi ra khỏi bộ xử lý. Quá trình trên sẽ được tiến hành ở tất cả các bộ xử lý cho đến khi bộ cuối cùng của R được đưa vào bộ xử lý trái nhất, lúc đó bộ sắp đổi chiều để thực hiện quá trình ngược lại – quá trình ra. Giai đoạn vào ở mỗi bộ xử lý được mô tả như sau:

```
IF z > w THEN
BEGIN
      y := z;
      z := x
END ELSE
BEGIN
      y := w;
      w := x
END;
```



Hình 5: Hoạt động của bộ xử lý ở giai đoạn vào

Giai đoạn ra:

Ngay khi bộ cuối cùng của R được đưa vào bộ xử lý trái nhất thì dòng dữ liệu đổi chiều để thực hiện quá trình ra. Lúc đó tại mỗi bộ xử lý đầu vào bên trái chuyển thành đầu ra và đầu ra bên phải chuyển thành đầu vào. Từ thuật toán ở giai đoạn vào, chúng ta nhận thấy rằng bộ có giá trị bé nhất của thuộc tính cần sắp của R hiện đang được lưu trữ tại bộ xử lý thứ nhất (P_1) của bộ sắp, giá trị tiếp theo nằm ở một trong hai bộ xử lý đầu,.... Để có được một dãy tăng dần như mong muốn thì tại mỗi bộ xử lý chúng ta phải tiến hành như sau: Giá trị nhỏ hơn trong từng bộ xử lý được đưa ra làm đầu vào mới của bộ đứng ngay bên trái. Bộ xử lý trái nhất trong bộ sắp sẽ xuất ra lần lượt các bộ của R đã được sắp xếp theo yêu cầu.

Hoạt động của mỗi bộ xử lý ở giai đoạn ra được mô tả như sau:

```
IF z > w THEN
BEGIN
      x := w;
      w := y
END ELSE
BEGIN
      x := z;
      z := y
END;
```

Hình 6: Quá trình ra của thuật toán sắp xếp

Vấn đề nảy sinh ở đây là lúc nào thì Bộ sắp thực hiện quá trình chuyển từ giai đoạn vào sang giai đoạn ra một cách đúng đắn và kịp thời? Để giải quyết vấn đề này, chúng ta có thể cải tiến thuật toán trên như sau:

Nạp m_i là số lần đối sánh của bộ xử lý thứ i (P_i) vào thanh ghi trạng thái. m_i được xác định như sau

$$m_{i+1} = m_i - 2 \quad (i = 1, 2, ..., N/2)$$

trong đó m_1 là số lần đối sánh của R.

Khi khởi động hệ thống, các thanh đếm l_i được gán giá trị 0 ($i = 1, 2, ...$). Sau mỗi nhịp thực hiện, thanh đếm tăng thêm một đơn vị. Như thế thuật toán sắp xếp Systolic chuyển dòng dữ liệu từ giai đoạn vào sang giai đoạn ra tại bất kỳ bộ xử lý nào, mà tại đó thỏa mãn hệ thức $m_i - l_i = 0$ ($i = 1, 2, ..., N/2$).

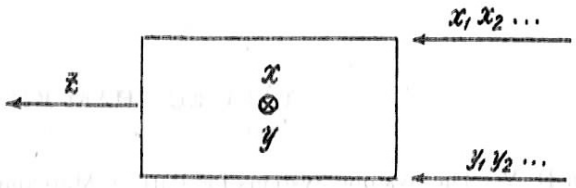
Chúng ta nhận thấy rằng, khi sử dụng bộ sắp gồm $N/2$ bộ xử lý, thì thời gian thực hiện sắp xếp là $2N$ đối với quan hệ có lực lượng N. Thuật toán vừa được mô tả, với $N/2$ bộ xử lý thì chỉ có thể dùng để sắp trực tiếp được cho các quan hệ có không quá N thành phần.

Trong thực tế, chúng ta phải giải quyết vấn đề sắp xếp đối với các quan hệ rất lớn. Một trong nhiều cách giải quyết là sử dụng thuật toán trộn (MERGESORT). Thuật toán này được mô tả một cách hình thức như sau:

Chia quan hệ R cần sắp thành các tập con tùy ý theo số các bộ xử lý thực tế của các bộ sắp. $R \longrightarrow R_1, R_2, ..., R_m$. Đối với các R_i $(i = 1, 2, ..., m)$, chúng ta áp dụng song song thuật toán vừa nêu ở trên, dùng m đầu ra này, trộn với nhau từng đôi một, kết quả thu được lại tiếp tục trộn với nhau từng đôi một,.... Quá trình trên được tiến hành cho tới lúc chỉ còn hai dòng dữ liệu cùng tham gia vào công việc trộn để thu được kết quả cuối cùng.

Để tiện mô tả, chúng ta gọi quá trình sắp xếp ở các tập con R_i $(i = 1, 2, ..., m)$ là quá trình 0, còn các quá trình trộn sau đó theo thứ tự là thứ $1, 2, 3, ..., \log_2 m$. Quá trình 0 được thực hiện theo thuật toán đã nêu ở trên. Các quá trình còn lại sẽ thực hiện theo thuật toán sau đây: Mỗi bộ xử lý chỉ gồm hai thanh ghi trên, dưới (x, y) nhận các giá trị từ hai dòng dữ liệu là các tập con đã được trộn ở quá trình trước. Khi trộn, hai giá trị của x và y sẽ được so sánh, giá trị bé hơn sẽ được đưa ra và giá trị của dòng dữ liệu tương ứng sẽ được đưa vào để đối sánh tiếp.

```
IF x < y THEN
BEGIN
      z := x;
      x := x1
END ELSE
BEGIN
      z := y;
      y := y1
END;
```



Hình 7: Thuật toán trộn

Phân tích hoạt động của thuật toán, chúng ta thu được một số kết luận liên quan đến độ phức tạp như sau.

Mệnh đề. Cho quan hệ R gồm $2ml$ bộ, trong đó $2^k < m \leq 2^{k+1}$ (với k là số nguyên). Sử dụng bộ sắp Systolic có độ dài l để sắp xếp R thì:

a. Số bộ xử lý cần thiết là $(m * k + m/2)$

b. Số quá trình thực hiện sẽ là $(k + 1) \cong \log_2 m$

c. Thời gian tổng thể để sắp sẽ là $f(m, l) = \log_2 m + 2 * l + 2 = O(\log_2 m, l)$.

Từ hoạt động của Systolic, dễ dàng nhận thấy $m * l$ bộ xử lý cần cho bước sắp sơ bộ (quá trình 0). Do cơ chế dừng lại, nên chúng ta chỉ cần thêm $m/2$ bộ cho việc trộn ở các quá trình tiếp theo. Kết luận thứ hai là hiển nhiên nếu chúng ta chú ý đến nhận xét sau: Số quá trình được tăng thêm 1 khi và chỉ khi số dòng dữ liệu mới được thêm vào, đủ để tạo thành một dòng dữ liệu mới, có vai trò tương đương với một quá trình được tạo ra trước đó. Kết luận thứ 3 có được vì các quá trình trộn hai đường ở trên đều được tiến hành song song cho tới quá trình thứ $2 * l$, tức là cho tới khi các bộ sắp $R_1, R_2, ..., R_m$ đẩy các bộ cuối cùng của mình ra khỏi dãy.

Nhận xét: Cho R với lực lượng 2 ml, việc sắp xếp tổng thể theo thuật toán trộn hai đường nêu trên chỉ cần thêm $O(\log_2 m)$ phép toán so với bước sắp sơ bộ $O(l)$.

VI - KẾT LUẬN

Cấu trúc Systolic đã được sử dụng để cài đặt nhiều phép toán phức tạp, như phép toán về ma trận [6], biến đổi các hệ số Fourier nhanh, một số phép biến đổi dữ liệu ứng dụng trong xử lý ảnh [5],.... ở đây, chúng tôi đã xây dựng ba thuật toán Systilic để thực hiện các phép toán tính tích Đề-các, kết nối và sắp xếp quan hệ dữ liệu. Các thuật toán này thực hiện song song và truyền ứng (piperlined) với độ phức tạp thời gian là tuyến tính của độ dài dữ liệu.

Nếu R là một danh sách móc nối đơn các bản ghi, thuộc tính cần sắp là giá trị các trường khóa, thì Gordon Sorber [8] đã đề xuất một phương pháp sắp xếp tuần tự khá hữu hiệu dựa trên việc phân tích đường chạy theo thứ tự tăng hoặc giảm ở bước sắp sơ bộ, còn các quá trình tiếp theo được thực hiện như quá trình trộn đã nêu ở trên. Do duy trì được độ dài các đường chạy gần như nhau, nên thuật toán SORB nêu trong [8] tốt hơn nhiều thuật toán khác, kể cả Quicksort và đạt được độ phức tạp là $1.4N\ln(N)$, với N là độ dài dữ liệu.

Nhận ngày 1-6-1991

TÀI LIỆU THAM KHẢO

1. Ban D. V., The Systolic Systems for Pattern Matching, Proc. of Artif. Intel. App. AI 87, Praha 1987, 278–286.
2. Ban D. V., OCCAM Solves Pattern Matching Problems, Tech. Report MBL–UTK N. 180, Bratislava 1987.
3. Codd E. F., Relational Completeness of Database Sublanguages, Database System, Proc. of Comp. Sci. Symp. Vol. 6, Prentice–Hall 1972.
4. Kung H. T., Leiserson C. E., Systolic Array for VLSI, Introduction to VLSI Systems, Ed. Mead C. A., Addison–Wesley, MASS. 1980.
5. Kung H. T., On the Implementation and Use of Systolic Array Processors, Proc. of Inter. Confe. on Comp. Design: VLSI in Comp. IEEE, Nov. 1983, 370–373.
6. Miklosko J., Systolic System for Linear Equation System and Matrix Inversion, Comp. and Artifi. Intel., Vol. 2, No. 4, (1983).
7. Miranker G., Tang L., Wong C. K., Zero–time VLSI Sorter, IBM Jour. Res. Develop., 27, 1980, 140–148.
8. Sorber G., Optimal List Sorting, Software Development International, Winter 1991, 43–49.

(Xem tiếp trang 14)