Tạp chí Tin học và Điều khiển học, T. 13, S. 4 (1997) (103–114)

VỀ MỘT ỨNG DỤNG CỦA VECTOR TIME TRONG CÁC HỆ CSDL PHÂN TÁN

NGUYỄN NAM HẢI[1)], PHÍ MẠNH LỢI[2)]

Abstract. For the concurrence control algorithms using timestamp technique in Distributed Database Systems, the technique of generating labels for the events is very important. This paper presents the method of generating label that bases on the technique of Vector time for concurrence control algorithms in Distributed Database Systems. An algorithm basing on this method has been presented. The evaluation of performance of algorithm is also examined.

I. ĐẶT VẤN ĐỀ

Trong các hệ CSDL phân tán, kỹ thuật gắn thời gian thường được sử dụng trong các thuật toán điều khiển trướng tranh. Mục đích của việc gắn nhãn thời gian là để tạo cho các phép toán truy nhập CSDL một thứ tự tổng thể. Tuy nhiên, việc tạo ra một cơ chế sinh nhãn lý tưởng là một việc hết sức khó khăn. Do vậy, người ta cố gắng tạo ta những cơ chế sinh nhãn càng gần với cơ chế sinh nhãn lý tưởng càng tốt. Thông thường để tạo nhãn người ta thường sử dụng đồng hồ logic hoạt động theo nguyên lý của Lamport đề xuất năm 1978 [4]. Tuy nhiên kỹ thuật này không chính xác, phản ánh không đầy đủ thông tin thời gian cho các sự kiện xẩy ra trong hệ thống. Vector time là kỹ thuật được đề xuất bởi Fidge năm 1991 [2] sử dụng khá tốt cho việc xác định trạng thái tổng thể trong các hệ tính toán phân tán truyền thông báo. Trong bào này, chúng tôi trình bày một cơ chế sinh nhãn dựa trên kỹ thuật Vector time cho thuật toán điều khiển tương tranh BTO trong các hệ CSDL phân tán. Kết quả nhận được thuật toán hoạt động tốt hơn với việc sử dụng cơ chế sinh nhãn bằng đồng hồ logic của Lamport. Đánh giá hiệu quả của thuật toán mới này cũng được chỉ ra.

II. MỘT SỐ KHÁI NIỆM CƠ BẢN

Hệ phân tán là một tập các quá trình tuần tự $P_1, P_2, ..., P_n$ có thể liên lạc với nhau thông qua trao đổi thông báo. Ở đây các kênh truyền thông được giả định là hoàn toàn tin cậy, độ trễ truyền thông là hữu hạn nhưng không đoán trước được.

Tính toán phân tán mô tả được thực hiện của một chương trình phân tán gồm một tập các quá quá trình.

Thực hiện của một quá trình tuần tự được biểu diễn như một dãy các sự kiện. Một sự kiện có thể là nội tại trong quá trình đó và chỉ thay đổi trạng thái cục bộ trong quá trình đó. Một sự kiện cũng có thể kéo theo những quá trình khác bằng việc gửi hoặc nhận thông báo.

Tập các sự kiện của tính toán phân tán được sắp bởi thứ tự theo qui tắc sau:

- *Các sự kiện xảy ra trong cùng một quá trình là được sắp toàn bộ.*
- *Sự kiện gửi thông báo xảy ra trước sự kiện nhận thông báo đó.*

Ký hiệu tập E là tập các sự kiện của một tính toán phân tán và $\rightarrow$ là thứ tự xác định trên E.

Cho $e, f \in E$, nếu Not $e \rightarrow f$ và Not $f \rightarrow e$ khi đó ta nói e và f xảy ra đồng thời và ký hiệu là $e//f$.

Quan hệ thứ tự được sinh ra bởi tập E với thứ tự $\rightarrow$ được ký hiệu là $(E, \rightarrow)$.

Cho $e \in E$, khi đó ta định nghĩa:

- Past$(e) = \{f \in E | f \rightarrow e\}$
- Future$(e) = \{f \in E | e \rightarrow f\}$

Khái niệm hệ CSDL phân tán ở đây bao gồm cả khái niệm CSDL phân tán và hệ quản trị CSDL phân tán, trong đó: *CSDL phân tán là một tập các CSDL có quan hệ với nhau về mặt logic và được phân bố trên một mạng máy tính. Hệ quản trị CSDL phân tán là hệ thống phần mềm cho phép quản trị CSDL phân tán và làm cho sự phân tán đó trong suốt đối với người sử dụng [7].*

Giao tác là khái niệm để chỉ đơn vị tính toán tin cậy và tương thích trong các hệ CSDL phân tán.

III. SINH NHÃN DỰA TRÊN NGUYÊN LÝ CỦA LAMPOT

Một kỹ thuật sinh nhãn truyền thống thường hay được sử dụng cho các giải thuật điều khiển tương tranh dựa trên kỹ thuật gắn nhãn thời gian là kỹ thuật được xây dựng trên những nguyên lý của Lamport đề xuất 1978. Trước hết sẽ trình bày nguyên lý làm việc sau đó độ chính xác của kỹ thuật sẽ được xem xét.

1. Nguyên lý làm việc

Xét một hệ phân tán gồm n quá trình $P_1, P_2, ..., P_n$ hoạt động theo nguyên lý như sau:

- Các sự kiện xảy ra trong cùng một quá trình là được sắp toàn bộ.
- Giữa các quá tình có thể trao đổi thông tin bằng việc gửi thông báo và sự kiện gửi thông báo xảy ra trước sự kiện nhận thông báo đó.

Mỗi quá trình P_1 được gắn một đồng hồ tuyến tính C_i là một biến nguyên nhận các giá trị tăng dần. Các C_i làm việc như sau:

1) Mọi thông báo của một quá trình bất kỳ gửi cho một quá trình khác đều mang theo giá trị thời gian của quá trình đó tại thời điểm gửi thông báo.

2) Trước khi xử lý một sự kiện nội tại xảy ra trong mỗi quá trình thực hiện:

$$C_i = C_i + d \ (d \text{ nguyên dương}).$$

3 Khi một quá trình P_i nhận được một thông báo từ quá trình P_j, C_i được xử lý như sau:

$$C_i = \max(C_i, C_j),$$
$$C_i = C_i + d.$$

Giá trị thời gian gán cho các sự kiện xảy ra trong hệ thống được sử dụng làm nhãn thời gian để xem xét quan hệ giữa các sự kiện xảy ra trong hệ thống.

2. Độ chính xác

Bây giờ ta xét độ chính xác của kỹ thuật này khi phản ánh mối quan hệ giữa các sự kiện trong hệ thống phân tán. Sự không chính xác này tập trung vào những sự kiện xảy ra đồng thời. Ta sẽ cố gắng định lượng xem đồng hồ tuyến tính sẽ phản ánh được chính xác bao nhiêu quan hệ giữa các cặp sự kiện và bao nhiêu quan hệ phản ánh không chính xác. Xét E là tập các sự kiện, và $\rightarrow$ là thứ tự xác định như trên tập các sự kiện. Định nghĩa *hạng của một sự kiện e là chiều dài lớn nhất của các xích kết thúc tại e và ký hiệu là* $r(e)$. Ở đây xích là một tập con của E các sự kiện từng đôi một so sánh được với nhau theo thứ tự $\rightarrow$.

Tập $R = \bigcup_{i=1}^{n} R_i$ với $\forall i \geq 0$, $R_i = \{e \in E | r(e) = i\}$ được gọi là phân tích hạng của $(E, \rightarrow)$.

Ta dễ dàng chứng minh được:

- *Nếu* $ef \in E$ *và* $r(e) = r(f)$ *thì* $e//f$. (1)
- *Nếu* $ef \in E$ *và* $r(e) = r(f)$ *thì* $C(e) = C(f)$. (2)

Từ các mệnh đề (1) và (2) tập các cặp các sự kiện của E xảy ra đồng thời và có hạng bằng nhau thì có giá trị đồng hồ tuyến tính bằng nhau. Tuy nhiên, ta có nếu $e//f$ thì có thể có những sự kiện g có quan hệ nhân quả đối với f mà $e//g$ và với những trường hợp này ta có $r(e) \neq r(g)$ kéo theo $C(e) \neq C(g)$. Tương tự, một cách đối xứng có thể có những sự kiện h có quan hệ nhân quả với f mà $f//h$ và với những trường hợp này ta có $r(f) \neq r(h)$ kéo theo $C(f) \neq C(h)$. Số những sự kiện như vậy có thể định lượng trong công thức sau:

$$||\text{Past}(f)| - |\text{Past}(f) \cap \text{Past}(e)|| + ||\text{Future}(f)| - |\text{Future}(f) \cap \text{Future}(e)||+$$
$$||\text{Past}(e)| - |\text{Past}(e) \cap \text{Past}(f)|| + ||\text{Future}(e)| - |\text{Future}(e) \cap \text{Future}(f)||$$

Như vậy đồng hồ tuyến tính chỉ đặc trưng được một phần quan hệ đồng thời. Tổng số những cặp quan hệ đồng thời phản ánh được bởi đồng hồ tuyến tính có

thể định lượng được bởi công thức sau:

$$I = \sum_{i=1}^{n} C_{|R_i|}^2 \tag{3}$$

Nếu gọi N là tổng số những cặp (e, f) mà $e//f$ thì số những cặp phản ánh không chính xác bởi công thức đồng hồ tuyến tính sẽ là:

$$N_{\text{err}} = N - L$$

N_{err} sẽ phục thuộc vào từng trường hợp cụ thể, nó sẽ đạt cực đại khi giữa các quá trình không có trao đổi thông báo với nhau và đạt cực tiểu khi tập các sự kiện là được sắp toàn bộ.

Sự phản ánh không chính xác của đồng hồ tuyến tính đã làm giảm hiệu quả của những giải thuật điều khiển tương tranh dựa trên kỹ thuật gắn nhãn thời gian. Sự phản ánh không chính xác này đã dẫn tới trường hợp có những trường hợp các sự kiện xảy ra đồng thời nhưng dựa vào các nhãn thời gian, giải thuật có khi lại nhầm hiểu chúng có quan hệ nhân quả với nhau dẫn đến những trường hợp phải thực hiện lại giao tác phi lý.

Vấn đề đặt ra là cần phải xác định một cơ chế sinh nhãn tốt cho những giải thuật điều khiển tương tranh dựa trên kỹ thuật gắn nhãn thời gian. Cơ chế được chọn sẽ là Vector time đã được trình bày trong phần tiếp theo.

IV. SỬ DỤNG VECTOR TIME TRONG CÁC HỆ CSDL PHÂN TÁN

1. Định nghĩa Vector time

Để thuận tiện cho việc trình bày, chúng tôi sẽ nhắc lại định nghĩa Vector time [3]. Cho N là tập các số nguyên, E là tính toán phân tán. Định nghĩa một ánh xạ đơn vị E vào N^n.

$$C : E \to N^n$$

thỏa mãn hai điều kiện sau:

1) $e, f \in E,\ e \to f \Leftrightarrow C(e) \Rightarrow C(f)$,

2) $e, f \in E,\ e//f \Leftrightarrow \text{Not}C(e) \Rightarrow C(f)$ và $\text{Not}C(f) \Rightarrow C(e)$.

Thứ tự *Rightarrow* được xác định như sau:

$C(e) \Rightarrow C(f) \Leftrightarrow C(e)[i] \leq C(f)[i]$, $i = 1, ..., n$ và tồn tại một giá trị k để $C(e)[k] < C(f)[k]$.

$C(e)[i]$ là tọa độ thứ i của vector $C(e)$ trong không gian N^n.

Trường hợp $C(e)$ và $C(f)$ không so sánh được với nhau, để đơn giản chúng tôi cũng sử dụng ký hiệu $C(e)//C(f)$.

Ta gọi $C(e)$ là Vector time của sự kiện e và tập $C(E)$ được gọi là tập các Vectơ time tương ứng với các tập sự kiện E.

2. Ứng dụng

a. *Mô tả hệ thống*

CSCL phân tán gồm các item dữ liệu $x_1, x_2, ..., x_m$ được phân bố trên mạng máy tính gồm các trạm có tên $id_1, id_2, ..., id_n$. Các item dữ liệu có thể lặp trên một số trạm. Các tên trạm là duy nhất và được lấy từ một tập được sắp toàn bộ nào đó, để đơn giản có thể lấy tên trạm trùng với số thứ tự của trạm trên mạng. Tại mỗi trạm trên mạng máy tính được gắn một đồng hồ logic là một vectơ gồm hai phần:

- Một vectơ n chiều $C^i(C_1^i, C_2^i, ..., C_n^i) \in N^n$,
- Tên trạm tương ứng.

Như vậy đồng hồ logic tại mỗi trạm i $(i = 1, ..., n)$ là một cặp (C^i, id_i).

Trên tập các *Vector time* xác định một thứ tự tổng thể $\rightarrow$ như sau. (C, id) và (C', id') là hai *Vector time* bất kỳ. Ta nói:

$(C, id) \rightarrow (C', id') \Leftrightarrow$ hoặc $(C \Rightarrow C')$ hoặc $(C//C'$ và $id < id')$.

Tập các sự kiện xác định trong hệ thống gồm các loại sự kiện.

- Sự kiện khởi tạo giao tác,
- Sự kiện gửi thông báo,
- Sự kiện nhận thông báo.

Cơ chế làm việc của đồng hồ được xác định như sau:

- Khi có một giao tác được khởi tạo tại một trạm i nào đó thì tọa độ thứ i trong thành phần thứ nhất của đồng hồ tại trạm đó được tăng lên 1. Nhãn thời gian được gán cho giao tác là giá trị của đồng hồ của trạm tại thời điểm đó.
- Khi một trạm i gửi một thông báo cho một trạm nào đó $(i \neq j)$ thì tạo độ thứ i trong thành phần thứ nhất của đồng hồ tại trạm i tăng lên 1. Giá trị thời gian tại thời điểm này cũng được gửi cùng thông báo tới trạm j.
- Khi một trạm j nhận được một thông báo từ trạm i thì đồng hồ tại trạm j được xác định như sau:

  $(C_j, id_j) = (\text{Sup}\,(C^j, C^i), id_j)$.

  Giá trị tọa độ thứ j trong thành phần thứ nhất của đồng hồ tại trạm j được tăng lên 1.

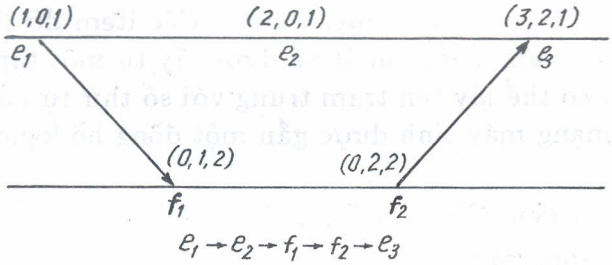
Phép toán Sup ở đây được xác định như sau:

x, y là các vectơ n chiều, khi đó $\text{Sup}\,(x, y) = w$ là vectơ được xác định như sau: $w[i] = \max(x[i], y[i])$, $i = 1, ..., n$.

Ta dễ dàng thấy được với cách xây dựng đồng hồ như thế này đảm bảo phản ánh được các hiện tượng:

+ Trong một trạm thì sự kiện xảy ra sau có giá trị thời gian lớn hơn.

+ Sự kiện nhận thông báo có giá trị thời gian lớn hơn thời gian xảy ra sự kiện gửi thông báo tương ứng.



Hình 1. Ví dụ về thứ tự các sự kiện hệ thống

b. Giải thuật điều khiển truy nhập CSDL

Giải thuật được xây dựng trên cơ sở giải thuật BTO. Tư tưởng chủ đạo ở đây là ta xét giải thuật BTO với cơ chế gắn nhãn thời gian sử dụng *Vector time* và chúng tôi gọi là BVtO. Nhãn thời gian được gán cho mỗi giao tác tại thời điểm khởi tạo là giá trị *Vector time* tại thời điểm đó.

Với mỗi *item* dữ liệu x lưu trữ tại trạm. SC sẽ sử dụng các biến sau:

- $R\text{-}Vt(x)$: $R\text{-}Vt(x)[0]$ là tổng hợp toàn bộ tri thức và thời gian mà SC biết được về các *dm-read*(x) đã được xử lý. $R\text{-}Vt(x)[1]$ là số hiệu của trạm tương ứng với *dm-read*(x) đã được xử lý gần nhất.
- $W\text{-}Vt(x)$: Biến có kiểu *Vector time*, trong đó $W\text{-}Vt(x)[0]$ là tổng hợp toàn bộ tri thức về thời gian mà SC biết được và các *dm-write*(x) đã được xử lý. $W\text{-}Vt(x)[1]$ là số hiệu của trạm tương ứng với *dm-write*(x) đã được xử lý gần nhất.
- min-$R\text{-}Vt(x)$: là *Vector time* nhỏ nhất trong số các Vt(*dm-write*(x)) đặt trong buffer chờ thực hiện.
- min-$P\text{-}Vt(x)$: là *Vetor time* nhỏ nhất trong số các Vt(*prewrite*(x)) đặt trong buffer.

$Vt(x)$ là hàm lấy *Vector time* của biến x.

Quá trình đồng bộ được thực hiện theo các luật sau:

R1. Khi có *dm-read*(x) đến trạm, SC xử lý như sau:

Đặt $R = Vt$(*dm-read*(x))

if $(R[1] = W\text{-}Vt(x)[1])$ then

begin

if $(R[0][R][1] \geq W\text{-}Vt(x)[0]W\text{-}Vt(x)[1]$ then

```
        begin
            Đặt dm-read(x) vào buffer chờ thực hiện
        end
        else begin
            Loại dm-read(x), //lệnh đến muộn thời gian//
            Loại giao tác tương ứng.
        end
    else begin
        if (R[0] ⇒ W-Vt(x)[0] then
        begin
            Loại dm-read(x)
            Hủy bỏ giao tác tương ứng.
        end
        else begin
            Đặt dm-read(x) vào buffer chờ thực hiện
        end
    end
    end
```

Khi $dm\text{-}read(x)$ được đặt vào *buffer* nó sẽ được sắp xếp vào vị trí tương ứng theo thứ tự tăng dần của các $Vt(dm\text{-}read(x))$ đang chờ thực hiện trong *buffer*.

Kiểm tra các $Vt(dm\text{-}read(x))$ trong *buffer* theo giá trị tăng dần xem có lệnh nào thỏa mãn điều kiện:

$$Vt(dm\text{-}read(x)) \rightarrow \text{min-}P\text{-}Vt(x).$$

Nếu thỏa mãn thực hiện các hành động:

+ Chuyển $dm\text{-}read(x)$ cho DP thực hiện,
+ $R\text{-}Vt(x)[0] = \text{Sup}(R\text{-}Vt(x)[0], Vt(dm\text{-}read(x)[0])$.
+ $R\text{-}Vt(x)[1] = Vt(dm\text{-}read(x)[1])$.

R2. Khi SC nhận được lệnh $prewrite(x)$ từ một trạm i nào đó, nó xử lý như sau:

```
Đặt P = Vt(prewrite(x))
if (P[1] = W-Vt(x)[1]) then
begin
    if (P[0][P[1]] ≥ W-Vt(x)[0][W-Vt(x)[1]]) then
    begin
        Đặt prewrite(x) vào buffer
    end
    else begin
        Loại prewrite(x), //lệnh đến muộn thời gian//
        Loại giao tác tương ứng
    end
else
```

```
begin
    if (P[0] ⇒ W-Vt(x)[0]) then
    begin
        Loại prewrite(x)
        Hủy bỏ giao tác tương ứng.
    end
    else
    begin
        Đặt prewrite(x) vào buffer
    end
end
end
if (P[1] = R-Vt(x)[1] then
begin
    if (P[0][P[1]] ≥ R-Vt(x)[0][R-Vt(x)[1]]) then
    begin
        Đặt prewrite(x) vào buffer
    end
    else begin
        Loại prewrite(x), //Lệnh đến muộn thời gian//
        Loại giao tác tương ứng.
    end
else
begin
    if (P[0] ⇒ R-Vt(x)[0]) then
    begin
        Loại prewrite(x)
        Hủy bỏ giao tác tương ứng.
    end
    else
    begin
        Đặt prewrite(x) vào buffer
    end
end
end
```

R3. Khi nhận được *dm-write*(x) SC sẽ xử lý như sau:

+ *dm-write*(x) được đặt vào buffer ở vị trí tương ứng so sánh với các *dm-write*(x) khác đang chờ xử lý trong buffer.

+ Kiểm tra xem có *dm-write*(x) nào thỏa mãn:

$(Vt(dm\text{-}write(x)) \rightarrow \text{min-}R\text{-}Vt(x)$ or $Vt(dm\text{-}write(x)) = \text{min-}R\text{-}Vt(x))$

và

$(Vt(dm\text{-}write(x)) \rightarrow \text{min-}P\text{-}Vt(x)$ or $Vt(dm\text{-}write(x)) = \text{min-}P\text{-}Vt(x))$,

nếu thỏa mãn, thực hiện các hành động sau:

1) Chuyển $dm\text{-}write(x)$ cho DP thực hiện,

2) $W\text{-}Vt(x)[0] = \text{Sup}(W\text{-}Vt(x)[0], Vt(dm\text{-}write(x))[0])$.

3) $W\text{-}Vt(x)[1] = Vt(dm\text{-}write(x)[1]$,

4) Loại $prewrite(x)$ tương ứng với $dm\text{-}write(x)$ vừa được thực hiện khỏi *buffer*. Xác định lại min-P-$Vt(x)$. Kiểm tra xem có $dm\text{-}read(x)$ nào đủ điều kiện thực hiện không. Khi có một $dm\text{-}read(x)$ nào đó trong buffer được htực hiện, xác định lại $R\text{-}Vt(x)$ và min-R-$Vt(x)$. Nếu min-R-$Vt(x)$ thay đổi, kiểm tra xem có $dm\text{-}write(x)$ nào trong *buffer* có đủ điều kiện thực hiện không.

Trên cơ sở các luật R1, R2, R3 giải thuật BVtO được trình bày như sau.

Giải thuật BVtO

```
declare-var
    msg: Message;
begin
    repeat
        wait(msg);
        case of msg
            dm-read:    begin
                            Xử lý theo luật R1
                        end
            prewrite:   begin
                            Xử lý thoe luật R2
                        end
            dm-write:   begin
                            Xử lý theo luật R3;
                        end
        end-case
    until forever
end
```

V. HIỆU QUẢ

Trong phần này sẽ chỉ ra rằng sử dụng Vector time để sinh nhãn hiệu quả hơn sử dụng kỹ thuật đồng hồ tuyến tính. Để giải quyết vấn đề này, chúng tôi sẽ so sánh hiệu quả giữa các giải thuật BLtO là giải thuật BTO sử dụng kỹ thuật đồng hồ tuyến tính của Lamport để sinh nhãn với giải thuật BVtO là giải thuật BTO sử dụng kỹ thuật *Vector time* để sinh nhãn. Mức độ hiệu quả ở đây được

xem dưới góc độ các log mà nó có thể sinh ra được.

Trước hết sẽ chứng minh tập các log được sinh bởi giải thuật BVtO không nhỏ hơn tập các log được sinh bởi giải thuật BLtO. Kết quả này được biểu diễn thông qua định lý sau.

Định lý 1. *Nếu log L được sinh bởi giải thuật BLtO thì L cũng được sinh bởi giải thuật BVtO.*

Chứng minh: Giả sử ngược lại. Tồn tại log$L = s_1, s_2, ..., s_m$ được sinh bởi giải thuật BLtO nhưng log này không thể đươc sinh bởi giải thuật BVtO.

Do tồn tại ít nhất một cặp s_1, s_2 sao cho:

hoặc $s_i\, s_j$ xung đột $r - w$

hoặc $s_i\, s_j$ xung đột $w - w$

mà $s_i s_j$ không xử lý được bởi BVtO nhưng lại xử lý được bởi BLtO.

Trước hết chúng tôi sẽ chỉ ra rằng điều này không thể xảy ra được với trường hợp xung đột $r - w$.

Dựa vào luật R1, ta có với cặp s_i s_j không xử lý được bởi giải thuật BVtO chỉ có thể là một trong các trường hợp sau.

a) $Vt(s_j)[1] = Vt(s_i)[1]$ và $Vt(s_j)[0][Vt(s_j)[1] - 1] < Vt(s_i)[0][Vt(s_i)[1] - 1]$. Dẫn đến s_i và s_j xảy ra tại cùng một trạm và s_j xảy ra trước s_i. Do đó theo nguyên lý làm việc của đồng hồ tuyến tính ta có:

$$Lt(s_j) < Lt(s_i) \tag{1}$$

b) $Vt(s)j)[0] \Rightarrow Vt(s_i)[0]$,

do đó $Vt(s_j)[0][i] \leq Vt(s_i)[0][i]$, $i = 1, ..., n$ và tồn tại ít nhất một giá trị nào đó sao cho:

$$Vt(s_j)[0][k] < VtPs_i)[0][k] \quad \text{suy ra } s_j \in \text{Past}(s_i)$$

Theo nguyên lý làm việc của đồng hồ tuyến tính ta có:

$$Lt(s_j) < Lt(s_i) \tag{2}$$

Từ (1) và (2) suy ra không thể xảy ra trường hợp $s_i s_j$ xung đột $r - w$ và s_i s_j xử lý được bởi BLtO mà lại trường hợp xung đột $r - w$.

Vậy nếu log$L$ được sinh bởi BLtO thì L cũng được sinh bởi BVtO.

Định lý được chứng minh.

Định lý 2. *Tập các log sinh bởi BLtO là tập con thực sự của tập các log được sinh bởi BVtO.*

Chứng minh: Để chứng minh định lý này ta chỉ cần chỉ ra tồn tại một log được sinh bởi BVtO nhưng lại không sinh được bởi BLtO.

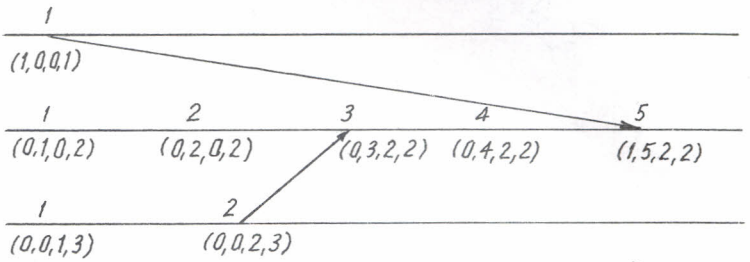
log$L$ được xây dựng không khó khăn. Ta chỉ cần xây dựng trong L có hai phép toán xung đột s_i s_j thuộc các giao tác T_i, T_j phân biệt sao cho thỏa mãn:

$$Lt(s_j) < Lt(s_i)$$

và $Vt(s_j)[0]$ // $Vt(s_i)[0]$ và $Vt(s_j)[0][Vt(s_j)[1]-1] > Vt(s_i)[0][Vt(s_i)[1]-1]$.

Cặp phép toán này sẽ bị từ chối bởi BLtO nhưng lại xử lý được bởi BVtO. Định lý được chứng minh.

Một kết quả dễ dàng nhận thấy là trong số các phép toán bị BLtO không chấp nhận dẫn đến giao tác tương ứng buộc phải thực hiện lại không chỉ là những phép toán đến muộn thời gian. Theo nguyên lý của đồng hồ tuyến tính thì xảy ra trường hợp có những phép toán được gắn nhãn thời gian nhỏ nhưng không có nghĩa đó là những phép toán đến muộn. Những trường hợp như vậy đã giải quyết tốt khi chúng ta sử dụng kỹ thuật *Vector time* để sinh nhãn thời gian.



Hình 2. Ví dụ trường hợp BVtO xử lý nhưng BLtO không xử lý được

Với những quan điểm đã áp cho thuật toán BTO và những kết quả thu được từ việc áp dụng này có thể mở rộng cho các giải thuật sử dụng kỹ thuật gắn nhãn thời gian khác. Và ta cũng nhận thấy với những cách sinh nhãn thời gian khác nhau, ta có thể thu được những giải thuật hoạt động với những hiệu quả khác nhau. Hiệu quả của giải thuật phụ thuộc nhiều vào độ chính xác của sự phản ánh thời gian mà các kỹ thuật sinh nhãn khác nhau có thể đạt được.

TÀI LIỆU THAM KHẢO

1. Berstein P. A., Goodman N., *Concurency control in distributed database system*, Computing Serveys, Vol. 13, No. 2 (1981).
2. Fidge C., *Logical time in distributed computing systems*, IEEE Trans. On Computer, 24, 8 (1981).
3. Hai N. N. and Hai N. T., *The notion of Vector time in distributed computations*, Proceedings of NCST of Vietnam, Vol. 7, No. 1 (1995).

4. Lamport L. *Time, clocks and the ordering of events in distributed systems*, Com. ACM, Vol. 21, No. 7 (1978).
5. Raynal M., *About logical clocks for Distributed system*, R.R., INRIA, No. 1534 (1991).
6. Valot C., *Accuracy of distributed timestamps*, R.R., INRIA, No. 1804 (1992).
7. Ozsu M. T., Valduriez P., *Principles of distributed database system*, Prentice Hall, Englewood, Cliffs, M. jersey 07632.

1) Khoa Công nghệ thông tin
Trường Đại học Bách khoa Hà Nội.
2) Viện Công nghệ thông tin
Trung tâm KHTN và CNQG.

Nhận bài ngày 6-1-1997