Tạp chí Tin học và Điều khiển học, T.17, S.1 (2001), 40–45

THUẬT TOÁN CẬP NHẬT LAN TRUYỀN TRONG CƠ SỞ DỮ LIỆU NHIỀU BẢN SAO

NGUYỄN ĐÌNH THUÂN

Abstract. Applications that rely on replicated data have different requirements for how their data is managed. Some applications may require that updates propagate amongst replicas with tight time constraints, whereas other applications may be able to tolerate longer propagation delays. Some applications only require replicas to interoperate with a few centralized replicas for data synchronization purposes, while other applications need communication between arbitrary replicas. This paper describes two approaches to replicated data management. The first approach is based on the concept of aggressive replication by causal message delivery, while the second approach uses the concept of history maintenance using log maintenance techniques.

Tóm tắt. Những ứng dụng sử dụng dữ liệu nhiều bản sao có những yêu cầu khác nhau đối với cách quản lý dữ liệu. Có ứng dụng yêu cầu việc cập nhật phải thực hiện với thời gian rất ngắn. Trong khi đó lại có những ứng dụng cho phép cập nhật với thời gian lâu hơn. Một số ứng dụng lại chỉ đòi hỏi cập nhật trên một vài bản sao với mục đích đồng bộ dữ liệu, nhưng một số ứng dụng khác cần việc truyền tin giữa các bản sao nào đó. Bài báo mô tả hai phương pháp quản lý dữ liệu nhiều bản sao.

1. MỞ ĐẦU

Trong các ứng dụng có sử dụng dữ liệu nhiều bản sao ở các vị trí trên mạng tùy theo mục tiêu mà có thể có nhiều cách quản lý khác nhau. Chẳng hạn, có ứng dụng yêu cầu việc cập nhật phải thực hiện với thời gian ngắn nhất, cũng có ứng dụng cho phép việc cập nhật với thời gian được thực hiện lâu hơn. Nhiều ứng dụng chỉ đòi hỏi thao tác cập nhật trên một vài bản sao với mục đích đồng bộ dữ liệu, nhưng có ứng dụng cần phải cập nhật trên một số lượng rất lớn các bản sao. Ngoài ra, tùy theo sự đụng độ giữa các dữ liệu khi cập nhật mà có các kỹ thuật giải quyết khác nhau.

Để đảm bảo tính nhất quán của dữ liệu khi cập nhật, với yêu cầu thời gian không quá hạn hẹp, chúng ta có thể dùng nghi thức truyền giao 2 pha (2PC: Two-Phase Commit) hoặc nghi thức truyền giao 3 pha (3PC: Three-Phase Commit) [6]. Trong các thuật toán này, khi mỗi vị trí trên mạng cần cập nhật thì gửi cho tất cả các vị trí khác trên mạng, sau khi nhận được các thông tin từ các vị trí này thì nếu tất cả đều đồng ý thì mới cập nhật, nếu có ít nhất 1 vị trí chưa sẵn sàng thì chưa cập nhật. Thuật toán này luôn đảm bảo tính nhất quán dữ liệu, tuy nhiên sẽ gặp khó khăn khi số bản sao là khá lớn hoặc có đường truyền xa hoặc có một vị trí nào đó trong chúng là chưa sẵn sàng.

Đối với cách tiếp cận lan truyền dữ liệu, trong [2] các tác giả đã đưa ra giải pháp thiết kế điều khiển dữ liệu tập trung. Trong [3] các tác giả cung cấp giải pháp quản lý lan truyền các E-mail. Trong bài này chúng tôi trình bày và đánh giá thuật toán cập nhật dữ liệu nhiều bản sao bằng cách dùng thời nhãn, gọi là thuật toán cập nhật lan truyền. Thuật toán này rất có hiệu quả khi số bản sao của dữ liệu rất lớn và việc cập nhật cần phải thực hiện trong thới gian ngắn.

2. THUẬT TOÁN CẬP NHẬT LAN TRUYỀN

2.1. Thuật toán 1

Tư tưởng của thuật toán này như sau: Giả sử D là CSDL chia sẻ. Khi một vị trí s cần cập nhật đơn vị dữ liệu $d \in D$, nó sẽ gán một thời nhãn t cho d. Khi 2 vị trí liên lạc với nhau, chúng sẽ so sánh các thời nhãn của các đơn vị dữ liệu trong D. Nếu vị trí nào có đơn vị dữ liệu được cập

nhật sau (có thời nhãn lớn hơn), thì d sẽ được gửi sang máy kia để cập nhật.

Khi một vị trí cần cập nhật đơn vị dữ liệu d, thì tạo ra tiến trình xử lý lan truyền việc cập nhật và khi một vị trí khác nhận biết được có cập nhật cũng sẽ thực hiện việc lan truyền cập nhật. Mặt khác, nếu một vị trí s_1 (đã cập nhật d) kết nối với s_2 và biết rằng s_2 đã cập nhật d thì khả năng lan truyền của s_1 phải giảm đi.

Gọi $u(d)$ là thao tác cập nhật d, các vị trí trên mạng được phân lớp thành các trạng thái như sau:

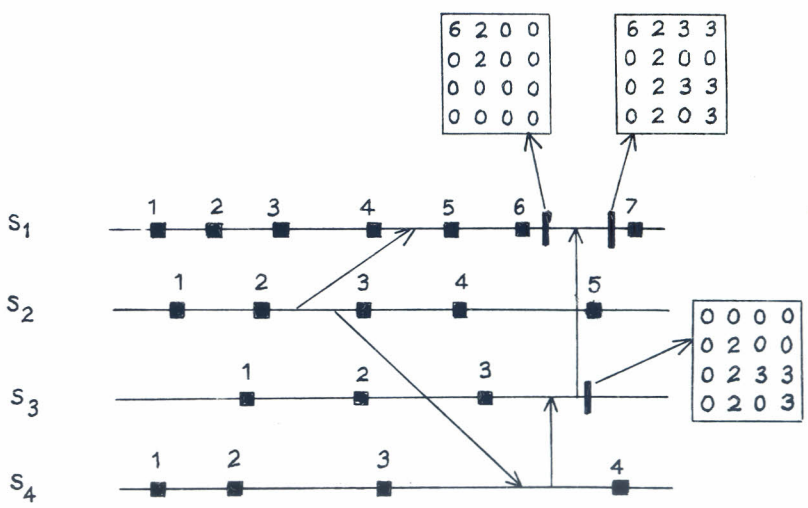
- Trạng thái 1 (chưa cập nhật): là trạng thái chưa từng nhận biết thao tác $u(d)$.
- Trạng thái 2 (lan truyền): là trạng thái đã thực hiện thao tác $u(d)$ và đang lan truyền thao tác $u(d)$.
- Trạng thái 3 (không còn lan truyền): là trạng thái đã thực hiện thao tác $u(d)$ và không còn lan truyền $u(d)$ nữa.

Thuật toán 1 như sau:

- Một vị trí ở trạng thái 1 chưa cập nhật), nếu được lan truyền thao tác $u(d)$ thì sẽ chuyển sang trạng thái 2 (lan truyền cho vị trí khác).
- Một vị trí ở trạng thái 2 (lan truyền) sẽ lặp lại việc nối kết ngẫu nhiên với các vị trí khác trên mạng và lan truyền thao tác $u(d)$.
- Một vị trí ở trạng thái 2 (lan truyền), khi tiến hành kết nối gặp vị trí ở trạng thái 2 hoặc trạng thái 3 (đã cập nhật xong và không còn lan truyền), với xác suất p cho trước, sẽ chuyển sang trạng thái 3. (Xác suất p tùy thuộc vào độ rộng của miền lan truyền mà cho lan truyền nhanh hay lâu).

2.2. Thuật toán 2

Trong Thuật toán 1 ở trên, chúng ta đã xét trường hợp đơn giản là chỉ tiến hành thao tác cập nhật là ghi chồng lên đơn vị dữ liệu d tại mỗi vị trí, trong đó các thao tác cần cập nhật d có thời nhãn nhỏ hơn đều bị bỏ qua. Chẳng hạn, nếu $u_2(d)$ cập nhật $u_1(d)$ thì không có một vị trí nào thực hiện thao tác $u_1(d)$ sau khi đã nhận $u_2(d)$. Tuy nhiên, trong trường hợp thao tác cập nhật chỉ là sửa đổi dữ liệu thì $u_1(d)$ phải được thực hiện trên tất cả các vị trí trước khi thực hiện $u_2(d)$. Để cho thứ tự của tất cả các bản sao được nhất quán, mỗi vị trí cần ghi lại quá trình thực hiện việc cập nhật (history) của mình trong sổ ghi (log). Mỗi log là danh sách có thứ tự các bản ghi, mỗi bản ghi ứng với 1 sự kiện cập nhật. Để minh họa thuật toán này, ta xét khái niệm vecto thời nhãn và ma trận thời nhãn như sau:



Ví dụ về ma trận thời nhãn

Định nghĩa 1. Cho $s_1, s_2, \ldots, s_n$ là các vị trí trên mạng.

- Với mỗi vị trí ta gọi vecto thời nhãn là vecto $(T_1, T_2, \ldots, T_n)$, trong đó T_i là số message chứa thao tác cập nhật nhận được từ vị trí s_i với $i = 1, \ldots, n$.

- Với mỗi vị trí i ta gọi ma trận thời nhãn là ma trận (T_{ij}) cấp $n \times n$ trong đó T_{ij} là số message chứa thao tác cập nhật của vị trí i nhận được từ vị trí j với $i, j = 1, \ldots, n$.

Định nghĩa 2

- Cho sổ ghi L, ta gọi k bản ghi đầu tiên của L là $L[k]$.
- Với mỗi sự kiện $e \in L$, vị trí của e trong L là Index(i).
- Ta kí hiệu $L[e]$ thay cho L[Index(e)].

Định nghĩa 3 (Về tính nhất quán của các sổ ghi)

Cho e là sự kiện được cập nhật đầu tiên tại vị trí s, khi đó với mỗi vị trí $j = 1, \ldots, N$ và mỗi sự kiện f, ta có:

$$f \in L_s[e] \Leftrightarrow f \in L_j[e].$$

Khai báo mỗi bản ghi sự kiện trong log gồm 3 thành phần:

```
Type Event_Rec = Record
        op   : operation;   {Phép toán cập nhật và các tham số của nó}
        s    : site;        {Vị trí đầu tiên thực hiện việc cập nhật}
        t    : time;        {Thời nhãn gán cho sự kiện cập nhật này}
     End;
```

Các khai báo biến:

```
Var  Count: integer;        {Biến đếm số sự kiện được tạo ra tại vị trí hiện hành}
     L    : Log             {Sổ ghi các sự kiện đang nhận}
     Min_TS: Array [1..N, 1..N] of Time;
                            {Min_TS[q]: cận dưới của thời nhãn lớn nhất trong L_q}
     Stabe: Array[1..N] of integer;
                            {Stable[q]: số sự kiện nhiều nhất được tạo ra bởi q mà
                                        tất cả các vị trí khác đã nhận}
```

Thuật toán 2.1.

Khi một vị trí thực hiện thao tác cập nhật đơn vị dữ liệu u, trước hết vị trí này ghi thao tác vào log của nó.

```
Procedure  Update(u);
   Begin   count:=count+1;
           Min_TS[Current][Current]:=Count;
           New(e);
           e.op:=u;
           e.t:=Min_TS[Current];
           e.s:=Current;
           Append e vào L;
   end;
```

Thuật toán 2.2 (Lan truyền cập nhật log)

Khi vị trí p lan truyền log đến vị trí q, thì p chỉ cần gửi các sự kiện mà q chưa có trong log. Nếu sự kiện e trong log L của p, giả sử rằng e được thực hiện lần đầu tiên tại vị trí $r = e.p$ và $k = e.\mathrm{TS}[r]$ là sự kiện được tạo ra tại vị trí r. Khi đó sự kiện e được gửi cho q nếu Min_TS$[q][r] < k$.

```
Procedure  Send_Log(q);
    Begin
        Khởi tạo L':=∅;
        ∀e ∈ L sao cho Min_TS[q][e.s] < e.TS[e.s] thì Append e vào L';
        Gửi L' cho q;
        Gửi Min_TS cho q;
    end;
```

Thuật toán 2.3 (Nhận được lan truyền log và tiến hành cập nhật)

```
Procedure  Receive_Log(p)
    Begin
        Nhận L' từ p;
        Nhận Min_TS từ p;
        ∀e ∈ L' nếu e ∉ L thì Apend e vào L
        For k:=1 to N {Cập nhật các thời nhãn}
            Min_TS[Current] [k]:=Max(Min_TS[Current][k], Min_TS[p][k])
        For i:=1 to N và i <> Current {Cập nhật các cận dưới thời nhãn}
            For j:=1 to N
                Min_TS[i][j]:=Max(Min_TS[i][j], Min_TS[i][j])
        For i:=1 to N
            Stable[i]:=Min(Min_TS[1][i],..., Min_TS[N][i])
        ∀e ∈ L' nếu e.TS[e.s] ≤ Stable[e.s] thì Xóa e trong L
    end;
```

Nhận xét. Việc lan truyền dữ liệu trong Thuật toán 1 tương tự như mô hình số lượng quần thể dự trữ và số lượng quần thể đã sử dụng trong sinh học. Trong đó áp dụng Mô hình Toán học cho Sinh thái học (theo [4]), nếu gọi:

Gọi S là tỷ lệ số vị trí ở trạng thái 1 (chưa được cập nhật) tương ứng với số lượng quần thể dự trữ.

Gọi I là tỷ lệ số vị trí ở trạng thái 2 (đang lan truyền) tương ứng với số lượng quần thể đã sử dụng. Khi đó qui luật biến thiên của S và I theo t được mô hình theo các phương trình vi phân sau [4]:

$$\begin{cases} \dfrac{dS}{dt} = -SI & (1) \\ \dfrac{dI}{dt} = SI - \dfrac{(1-S)I}{k} & (2) \end{cases}$$

Phương trình (1) nói lên một vị trí sẽ ở trạng thái 1 sẽ chuyển sang trạng thái 2 khi có một vị trí ở trạng thái 2 nối kết với nó.

Trong phương trình (2), số vị trí ở trạng thái 1 chuyển thành trạng thái 2 và số vị trí ở trạng thái 2 chuyển thành trạng thái 3 nếu chúng nối kết với 1 vị trí không phải trạng thái 1 với xác suất $1/k$. Tham số k nhằm điều khiển thời gian lan truyền.

3. ĐÁNH GIÁ

Định lý 1. *Khi áp dụng Thuật toán 1 để cập nhật lan truyền trên, theo thời gian t thì số message trung bình cần truyền giữa các vị trí là tuyến tính theo k.*

Chứng minh. Mỗi vị trí sau khi trở thành trạng thái 2, sau một thời gian nào đó sẽ gửi các message cho đến các vị trí khác cho đến khi nối kết với một số vị trí ở trạng thái 2. Một vị trí sè trở thành trạng thái 3 với xác suất $1/k$ khi chúng nối kết với một vị trí trạng thái 2, và sẽ trở thành trạng thái 3 sau khi nối với i vị trí ở trạng thái 2 với xác suất $(1-1/k)^{i-1}\,(1/k)$. Do đó, số message trung bình là:

$$E(X) = N(1-S)\left[1 + \sum_{i=1}^{\infty} i\left(1-\frac{1}{k}\right)^{i-1}\frac{1}{k}\right]$$
$$= N(1-S)(k+1).$$

Như vậy số lượng thêm vào của sự lan truyền trên là tăng theo lũy thừa k, tuy nhiên số trung bình là tuyến tính theo k.

Định lý 2. *Thuật toán 1 sẽ lan truyền tốt khi bắt đầu khởi tạo việc lan truyền và sẽ không tốt khi phải cập nhật một số vị trí cuối cùng.*

Chứng minh. Từ hệ phương trình (1)–(2) ở trên ta có:

$$\frac{dI}{dS} = \frac{1}{kS} - \frac{k+1}{k}. \tag{4}$$

Giải phương trình (3), ta được:

$$I(S) = \frac{\ln(S)}{k} - \frac{k+1}{k}S + C. \tag{5}$$

Để tìm C ta chọn một trường hợp đặc biệt trong điều kiện ban đầu là toàn bộ các vị trí đều ở trạng thái 1, với $S = 1$ và $I = 0$, thế vào (5) ta có:

$$C = \frac{k+1}{k}. \tag{6}$$

Hay:

$$I(S) = \frac{\ln(S)}{k} + (1-S)\frac{k+1}{k}. \tag{7}$$

Hiệu quả của Thuật toán 1 được đo bởi số vị trí chưa được ở trạng thái 2 khi việc lan truyền đã kết thúc (nghĩa là $I = 0$, hầu hết các vị trí ở trạng thái 3) và tổng số message đã gửi. Chúng ta có thể tính được phần còn lại S_0 là tỷ lệ của các vị trí chưa được cập nhật. Giải phương trình (7) với $I = 0$, ta được:

$$S_0(k) = e^{-(k+1)(1-S)}$$
$$\approx e^{-(k+1)}.$$

Theo trên, số lượng thêm vào của sự lan truyền trên là tăng theo lũy thừa k, tuy nhiên, số trung bình là tuyến tính theo k. Ta xét thêm trường hợp thay đổi tham số lan truyền từ k thành $k+1$.

Ta có, số các vị trí ở trạng thái 2 khi tăng k lên 1 được tính như sau:

$$N\big(S_0(k+1) - S_0(k)\big) \approx N\big(e^{-(k+2)} - e^{-(k+1)}$$
$$= N\left(\frac{1}{e} - 1\right)e^{-(k+1)}.$$

Vậy số message cần thiết để lan truyền đến các vị trí mới tăng theo lũy thừa với các vị trí được thêm vào. Do đó, thuật toán lan truyền này tốt khi bắt đầu khởi tạo phân tán ($k = 2$ sẽ có 96% số vị trí được lan truyền), tuy nhiên thuật toán này sẽ không tốt khi phải cập nhật một số vị trí cuối cùng.

Định lý 3. *Thuật toán 2 đảm bảo tính chất nhất quán của các sổ ghi giữa các vị trí.*

Chứng minh. Giả sử rằng sự kiện e được thực hiện đầu tiên tại vị trí p và lan truyền đến q, khi đó e được ghi vào sổ ghi L_p của p theo Thuật toán 2.1. Ngoài ra, tất cả các sự kiện là nguyên nhân trước e đều có trong sổ ghi của p theo Thuật toán 2.2, khi lan truyền sang q thì tất cả các sự kiện này sẽ được lan truyền sang q theo Thuật toán 2.3. Sau đó nếu e đã được lan truyền tiếp tục từ p hoặc q thì tất cả các sự kiện trước e đều có trong sổ ghi và như vậy chúng cũng được lan truyền kèm theo các tham số của chúng.

4. KẾT LUẬN

Trong bài báo này chúng tôi đã giới thiệu và đánh giá thuật toán lan truyền. Tuy chưa đảm bảo tính nhất quán dữ liệu tại mỗi thời điểm rất gần, tuy nhiên Thuật toán 1 rất hữu hiệu trong trường hợp số bản sao là rất lớn và các vị trí trên mạng không phải lúc nào cũng sẵn sàng. Ngoài ra, số message cần thiết để lan truyền đến các vị trí mới tăng theo lũy thừa với các vị trí được thêm vào nhưng chi phí là tuyến tính. Thuật toán 2 bảo đảm tính nhất quán dữ liệu, các sổ ghi và tất cả vị trí được lan truyền, tuy nhiên số message phải truyền là khá lớn. Vấn đề còn mở có thể làm tối ưu hai thuật toán trên là cần lan truyền theo chiều nào để hiệu quả hơn.

Lời cám ơn

Tôi xin chân thành cám ơn GS. Nguyễn Đình Ngọc và PGS. Nguyễn Xuân Huy đã có những gợi ý định hướng quan trọng cho tôi nghiên cứu thuật toán, xin chân thành cám ơn TS. Ngô Quốc Tạo đã tận tình góp những ý kiến quí báu giúp tôi hoàn thành bài báo này.

TÀI LIỆU THAM KHẢO

[1] A. Silberschatz, H. F. Korth, S. Sudarshan, *Database System Concepts*, McGraw-Hill, 1997.

[2] A. Raikar, C. L. Sun, M. Guduru, W. Tu, *Evaluation and Comparison of Replicated Data Management Algorithms*, 1997, URL http://www.umr.edu.

[3] D. B. Terry, K. Petersen, M. Spreitzer, M. Theimer, The case for non-transparent replication, *Data Engineering Bulletin* **21** (1998) 12–20, http://dblp.uni-trier.de.

[4] Iu. M. Xviregiev, *Các mô hình toán học trong sinh thái học*, Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội, 1988, trang 26–85.

[5] Ausubel J. and Marchetti C., *Elektron, Population Growth Algorithm*, Elsevier Science Inc., New York, 1996, URL http://phe.rockefeller.edu/Bi-Logistic.

[6] R. Chow, J. Johnson, *Distributed Operating System & Algorithms*, Addision Wesley Longman, 1997.

[7] T. Ozsu and P. Valdurier, *Principles of Distributed Database Systems*, Prentice Hall, 1992.

Nhận bài ngày 20 tháng 12 năm 1999
Nhận bài sau khi sửa ngày 4 tháng 11 năm 2000

Đại học Thủy sản Nha Trang