Tạp Chí Tin Học và Điều Khiển Học
Tập IX(1994), Số 2, 9-20

Một sơ đồ mới nhằm tăng độ sẵn sàng trong các hệ cơ sở dữ liệu phân tán

Nguyễn Nam Hải
Khoa Tin học, Trường đại học bách khoa Hà nội

I. Giới thiệu

Đối với cơ sở dữ liệu, khả năng sẵn sàng đáp ứng nhu cầu của người dùng khi khai thác là một vấn đề quan trọng. Trong các hệ CSDL phân tán vấn đề này bị chi phối nhiều hơn bởi nhiều yếu tố hơn CDSL tập trung. Do vậy, làm thế nào để tăng độ sẵn sàng cho các hệ CSDL phân tán là một vấn đề quan tâm nhiều [1-4,7,8]. Đã có nhiều giải pháp để tăng độ sẵn sàng cho CSDL phân tán. Một trong những giải pháp đó là tổ chức lặp dữ liệu tại một số trạm làm việc. Giá phải trả cho giải pháp này là phải duy trì tính tương thích của các bản sao CSDL khi sảy ra các sự kiện cập nhật CSDL hoặc sự cố tại các trạm làm việc hoặc kênh truyền nối các trạm làm việc.

Đã có một số giao thức (Protocol) điều khiển lặp được đề xuất và những giao thức này khác nhau ở mức độ và kiểu sự cố mà chúng có thể chịu được. Trong các hệ phân tán có hai loại sự cố: hỏng trạm làm việc và hỏng kênh truyền giữa các trạm làm việc. Những sự cố này có thể dẫn đến sự phân hoạch của mạng thành một vài nhóm trong đó các trạm trong cùng một nhóm có thể liên lạc được với nhau nhưng không thể liên lạc được với các trạm làm việc thuộc nhóm khác. Các giao thức duy trì độ sẵn sàng và tính tương thích của dữ liệu lặp khi mạng bị phân hoạch có thể chia làm hai loại [10]:

+ Loại "lạc quan" (optimistic)
+ Loại "bi quan" (pessimistic)

Đối với loại bi quan, duy trì độ sẵn sàng nhưng không được hy sinh tính tương thích của CSDL [1]. Trong các sơ đồ thuộc loại này việc cập nhật chỉ được thực hiện trên một nhóm trong số các nhóm phân hoạch. Ngược lại, loại lạc quan duy trì độ sẵn sàng ngay cả khi phải hy sinh tính tương thích của dữ liệu lặp [3]. Việc cập nhật cơ sở dữ liệu có thể được thực hiện ở một nhóm và do đó sự không tương thích có thể sảy ra nhưng chúng được tìm ra và giải quyết khi hợp nhất các nhóm. Xét dưới một góc độ nào đó thì các sơ đồ lạc quan có vẻ duy trì độ sẵn sàng của dữ liệu ở mức cao hơn các sơ đồ bi quan, nhưng có những trường hợp không thể hủy bỏ kết quả của những cập nhật khi đã hoàn thành. Và đối với những trường hợp như vậy sơ đồ bi quan thích hợp hơn. Một giải pháp hay được dùng trong các sơ đồ bi quan là "phiếu bầu" (voting) và có thể phân chia làm hai loại:

- Phiếu bầu động.
- Phiếu bầu tĩnh.

Trong các sơ đồ phiếu bầu, giữa các nhóm phân hoạch, người ta chọn lấy một nhóm được phép cập nhật và nhóm này được gọi là nhóm "đa số". Với sơ đồ phiếu bầu tĩnh nhóm có số phiếu bầu lớn nhất sẽ được chọn là nhóm đa số. Với loại phiếu bầu động, quan niệm về nhóm đa số thay đổi cơ bản so với loại phiếu bầu tĩnh. Trong trường hợp này, khi chọn nhóm đa số ứng với mỗi phân hoạch thì nhóm được coi là thiểu số trước đó sẽ không được xét đến. Đối với sơ đồ phiếu bầu động một giao tắc (transaction) có thể cập nhật CSDL tại một trạm làm việc nào đó nếu nhóm chứa hiện thời trạm làm việc đó được coi là nhóm đa số. (Không nhất thiết phải là nhóm có số phiếu bầu cao nhất). Các sơ đồ phiếu bầu động mắc một nhược điểm chung là sau một số lần phân hoạch liên tiếp có thể dẫn đến trường hợp không thể cập nhật CSDL tại bất kỳ nhóm nào. Hạn chế đã được khắc phục rất nhiều trong sơ đồ đã được đề xuất bởi Jian Tang và N. Natarajan [1] nhờ vào việc lưu lại vết của những phân hoạch liên tiếp trong mạng. Tuy nhiên sơ đồ của Jian Tang và N. Natarajan cũng như các sơ đồ phiếu bầu động khác mắc vào tình trạng không giải quyết được là khi nhóm đa số bị phân hoạch thành hai nhóm có số phiếu bầu bằng nhau. Trong trường hợp như vậy khả năng cập nhật CSDL sẽ bị mất. Ta xét ví dụ sau:

Ví dụ ta xét mạng gồm 5 trạm làm việc, tại mỗi trạm làm việc ta gắn cho một phiếu bầu và giải sử các phân hoạch lần lượt sảy ra như sau:

(1) {1,2,3,4,}, {5}.
(2) {1,2} {3,4} {5}.

Theo sơ đồ được đề xuất bởi Jian Tang và N. Natarajan thì sau lần phân hoạch thứ hai các trạm làm việc sẽ có bức tranh về phân hoạch mạng như sau:

Trạm 5 : {1,2,3,4}, {5},
các trạm 1,2,3,4 : {1,2} {3,4} {5}

và với những thông tin được biết về mạng như vậy không chọn được nhóm đa số nên không có trạm nào được phép duy trì cập nhật. Do đó sau lần phân hoạch thứ hai thì độ sẵn sàng của CSDL bị mất, việc cập nhật không thể sảy ra tại bất kỳ nhóm nào.

Trong bài báo này chúng tôi đề xuất một sơ đồ mới trên cơ sở phát triển một số ý tưởng đã được đề xuất trong [1] và [6]. Sơ đồ mới cho phép khắc phục nhược điểm trên của các sơ đồ phiếu bầu động nhờ vào việc xây dựng một quan hệ thứ tự trên tập các nhóm các trạm làm việc.

Trong phần hai sẽ trình bầy một số khái niệm cơ sở cần phục vụ cho việc xây dựng sơ đồ và những chứng minh cần thiết. Phần ba sẽ trình bầy sơ đồ đề xuất. Phần 4 sẽ trình bầy một số chứng minh tính đúng đắn của sơ đồ.

Một số khái niệm cơ sở

CSDL phân tán là một tập có cấu trúc các phần tử dữ liệu được phân bố trên mạng máy tính. Mỗi máy tính trên mạng gọi là một trạm làm việc, các máy có thể liên lạc với nhau bởi kênh liên lạc hai chiều. Các phần tử có thể lặp tại một số trạm làm việc. Trong trường hợp như vậy ta gọi là CSDL phân tán lặp. Trường hợp tại mỗi trạm đều có đầy đủ các bản sao của các phần tử tạo nên CSDL gọi là CSDL phân tán lặp toàn phần. CSDL phân tán lặp toàn phần chính là đối tượng nghiên cứu trong bài báo này.

Đối với CSDL phân tán, tại mỗi thời điểm có thể có nhiều người khai thác CSDL tại các trạm làm việc khác nhau. Mỗi truy nhập tới CSDL có thể xem như bộ ba $< R, W, f >$, trong đó R là tập các phần tử dữ liệu đọc, W là tập các phần tử dữ liệu viết, f là hàm mô phỏng những tính toán trong yêu cầu truy nhập. Để đồng bộ các truy nhập với mục đích duy trì sự tương thích của CSDL có thể sử dụng đồng hồ logic. Đồng hồ logic là cách thức gán một số nào đó cho một sự kiện sảy ra trong hệ thống để đánh dấu thời gian sảy ra sự kiện đó [5]. Các đồng hồ logic trong một hệ thống phân tán phải thỏa mãn hai điều kiện sau:

1. Nếu a và b là hai sự kiện sảy ra tại một trạm S_i nào đó, a sảy ra trước b thì $C_i(a) < C_i(b)$.
2. Nếu a là sự kiện gửi thông báo bởi S_i và b là sự kiện nhận thông báo đó bởi S_j thì $C_i(a) < C_j(b)$.

C_i và C_j là các đồng hồ logic tại các trạm S_i và S_j. Khi có một hệ thống đồng hồ logic trong một hệ thống phân tán thỏa mãn hai điều kiện trên thì ta luôn xây dựng được một thứ tự tổng thể giữa các sự kiện xảy ra trong hệ thống. Thứ tự đó được xác định như sau: a là sự kiện sảy ra tại trạm S_i, b là sự kiện sảy ra tại trạm S_j, ta nói rằng a sảy ra trước b khi và chỉ khi $C_i(a) < C_j(b)$ hoặc $C_i(a) = C_j(b)$ và $S_i <' S_j$. Trong đó quan hệ thứ tự $<'$ là một thứ tự tổng thể tùy chọn giữa các trạm làm việc trong hệ thống đang xét.

III. Mô hình đề xuất

Trong sơ đồ mới do chúng tôi đề xuất, có sự dụng một số ý tưởng và kiến thức trong [1] và [6] sau đây:

+ Kiến trúc truyền thông của hệ thống trong [1] được giữ lại trong sơ đồ mới.

+ Ý tưởng lưu lại vết của những lần phân hoạch mạng liên tiếp cũng được giữ lại trong sơ đồ mới, nhưng tại mỗi trạm chỉ lưu lại hình ảnh của nhóm chứa nó.

+ ý tưởng phân tán tính toán đối với các cập nhật trong [6] cũng được lưu ý trong sơ đồ của chúng tôi.

CSDL được xét trong bài báo này thuộc dạng phân tán lặp toàn phần và mỗi bản sao được gán cho một phiếu bầu.

Sau đây là chi tiết sơ đồ đề xuất.

3.1. Thứ tự xác định trên các nhóm phân hoạch

Gọi:

n: số lượng các trạm làm việc trên mạng,

$N = \{S_1, S_2, ...S_n\}$: tập tất cả các trạm làm việc trên mạng,

2^N: tập tất cả các tập con của N,

Ω: là tập được sắp toàn bộ với quan hệ thứ tự $<'$.

Tại mỗi trạm S_i gắn cho một nhãn x_i, $x_i \in \Omega$ và $x_i \neq x_j$ với $j = 1,n$ và $j \neq i$. Đặt $\sum = \{x_1, x_2, ..., x_n\}$ là tập tất cả các nhãn được gắn trên mạng. Gọi $\sum^*$ là tập tất cả các từ sinh trên bảng chữ $\sum$, $\alpha \in \sum$, ký hiệu len(α) là hàm xác định số các xuất hiện của các phần tử của $\sum$ trong α và gọi là độ dài của α. Ký hiệu ε là từ có độ dài bằng 0.

Xây dựng f như sau, $f: 2^N \to \sum^*$ và thỏa mãn:

- $x \in 2^N$, $x \neq \emptyset$ và $|x| = m$, trong đó $|x|$ là lực lượng của x, thì $f(x) = x'_1x'_2...x'_m$ sao cho $x'_i <' x'_{i+1}$, $i = 1, m-1$ và $x'_1, x'_2, ..., x'_m$ là các nhãn của các trạm làm việc thiết lập nên x.

- $x \in 2^N$ và $x = \emptyset$ thì $f(x) = \varepsilon$.

Giả sử $x, y \in 2^N$, ta định nghĩa

$$\max(f(x), f(y)) = \begin{cases} f(y) & \text{nếu } len(f(x)) < len(f(y)) \\ f(x) & \text{nếu } len(f(y)) < len(f(x)) \\ f(x) & \text{nếu } x_1 > y_1 \text{ và } len(f(x)) < len(f(y)) \\ f(y) & \text{nếu } y_1 > x_1 \text{ và } len(f(x)) < len(f(y)) \\ \text{Không được xác định} & \text{Trong các trường hợp còn lại.} \end{cases}$$

trong đó x_1 và y_1 là hai tiền tố tương úngtrong $f(x)$ và $f(y)$.

Ta nói:

$$f(x) <' f(y) \Longleftrightarrow \max(f(x), f(y)) = f(y).$$

Với hàm f xáx định như trên ta xây dựng được một quan hệ thứ tự trên 2^N. Với quan hệ thứ tự đó, khi một nhóm đa số bị phân hoạch thành hai nhóm x_1 và x_2 có số phiếu bầu bằng nhau thì việc chọn nhóm đa số mới được thực hiện theo giải thuật sau:

if $(f(x_1) < f(x_2))$ *Majority* $= x_1$;
else Majority $= x_2$;
ở đây *Majority* là biến xác định trên 2^N.

Việc xác định một nhóm có phải là một nhóm đa số không sẽ được thực hiện ngay tại thời điểm phân hoạch. Khi hợp nhất, nhóm hợp nhất sẽ chỉ là nhóm đa số nếu một trong hai nhóm tham gia hợp nhất là nhóm đa số.

3.2. Cấu trúc điều khiển tại mỗi trạm

Tại mỗi trạm, việc điều khiển hệ thống được chia thành hai khối:

+ khối CSDL,
+ Khối truyền thông.

Khối CSDL đảm bảo tất cả những thao tác với CSDL như cập nhật CSDL, điều khiển song song, phục hồi dữ liệu ...

Khối truiyền thông đảm bảo tất cả các chức năng truyền thông trong hệ thống. Khi khối CSDL muốn chuyển một thông báo tới một trạm khác, thông báo đó được chuyển xuống cho khối truyền thông và khối truyền thông thực hiện chức năng vận chuyển thông báo đó tới địa chỉ cần thiết trong mạng, cũng vậy khi có một thông báo từ một trạm khác chuyển đến cho trạm. khối truyền thông nhận, xử lý những yêu cầu cần thiết sau đó chuyển cho khối CSDL nếu thông báo đó được chuyển đến cho CSDL. Nếu thông báo gửu đến chỉ liên quan đến khối truyền thông thì thông báo đó được xử lý ngay tại khối truyền thông chứ không cần phải chuyển đến cho khối CSDL. Ngoài ra khối truyền thông còn thực hiện chức năng phát hiện sự cố trong hệ thống theo cơ chế sau: Mỗi trạm sẽ định kỳ gửu tín hiệu kiểm tra tới các trạm láng giềng, nếu sau một thời đoạn xác định mà trạm không nhận được tín hiệu kiểm tra từ một trạm láng giềng nào đó thì nó coi như kênh truyền tới trạm láng giềng đó đã có sự cố và thông báo sự kiện này tới tất cả các trạm trong nhóm chứa nó. Cũng bằng cơ chế đó, mỗi trạm có thể phát hiện được sự phục hồi của các kênh truyền hoặc bản thân các trạm láng giềng và thông báo tới tất cả các trạm khác trong nhóm chứa nó.

3.3 Các biến hệ thống

Để duy trì mối liên hệ giữa các trạm trong cùng một nhóm, tại mỗi trạm làm việc S_i trong mạng lưu giữ các biến sau:

+ $< x_i, L_i, V_i >$ trong đó:
  - x_i là nhãn của S_i,

- V_i là một bộ ba $< M, V, C >$, trong đó

+ M biểu diễn trạng thái của nhó m nhận các giá trị 0 và 1

0 - nhóm thiểu số (Minority),

1 - nhóm đa số (Majority),

+ V số hiệu version cập nhật của các bản sao CSDL trong nhóm,

+ C tập các trạm làm việc trong cùng nhóm.

- L_i là tập các kênh thiết lập nên các đường liên lạcgiữa các trạm trong nhóm chứa S_i có dạng (e, s), trong đó:

+ e tên của kênh,

+ s trạng thái của kênh nhận hai trạng thái 0, 1. Trong đó 0 có nghĩa là kênh còn tốt, 1 có nghĩa là kênh đã hỏng.

+ Label = $\{x_1, x_2, ...x_n\}$ là tập tất cả các nhãn gán cho các trạm làm việc trong toàn mạng.

+ C_i đồng hồ logic nhận các giá trị nguyên $0, 1, ..., \infty$ và được khởi đầu là 0.

3.4 Giao thức xử lý sự cố trênn mạng

3.4.1 Các thông báo và hàm

Để điều khiển hệ thống khi có sự cố, hệ thống sử dụng các hàm và thông báo sau:

- Hàm Group_Update(e), e là tên kênh có sự cố sảy ra. Hàm này được gọi là trạm làm việc phát hiện ra sự cố của kênh nối nó với một trạm láng giềng hoặc nó nhận được thông báo về sự cố của một kênh nào đó trong nhóm chứa nó.

- Thông báo failed(e,ts), e là tên kênh có sự cố sảy ra, ts là tem thời gian được gán cho bởi trạm phát hiện ra sự cố. Thông báo này được sinh ra khi một trạm nào đó phát hiện ra sự kiện kênh có sự cố.

3.4.2 Giao thức xử lý sự cố

Khi một trạm S_i phát hiện kênh nối của nó với trạm láng giềng bị hỏng nó sẽ thực hiện:

- gọi hàm Group_Update(e),

Trong đó e là tên của kênh bị hỏng. Hàm này sẽ thực hiện chức năng cập nhật tại L_i và kiểm tra xem sự hỏng kênh có dẫn đến phân hoạch không. Nếu sự kiện phân hoạch sảy ra nó sẽ xử lý như sau: Nếu nhóm chứa S_i là nhóm đa số thì dựa vào số phiếu bầu có được trong nhóm chứa mới và quan hệ thứ tự $<'$ trên 2^N đã xây dựng ở trên để xác định xem nhóm chứa S_i mới có là nhóm đa số không, đồng thời cập nhật tại V_i. Nếu nhóm chứa S_1 là nhóm thiểu số thì hàm này chỉ phải thực hiện cập nhật V_1.

- $C_i = C_i + 1$.

Gửi thông báo failed(e,ts) tới tất cả các trạm trong nhóm chứa nó, ở đây E là tên của kênh bị hỏng, ts là tem thời gian của S_i, $ts = (C_i, i)$.

Khi một trạm S_i nhận được thông báo này sẽ thực hiện:

+ $C_j = \max(C_j, C_i + 1)$,

+ Gọi hàm Group_Update(e).

Các hàm này sẽ thực hiện song song với các giao tác khác như các giao tác độc lập, không phụ thuộc vào hiện thời trạm đang cập nhật hay hợp nhất.

3.5 Giao thức xử lý cập nhật CSDL

3.5.1 Các cấu trúc dữ liệu điều khiển cập nhật CSDL

a) Các hàng đợi:

- Global_queue: cho các cập nhật từ xa,
- Local_queue: cho các cập nhật phát sinh tại trạm,
- Exec_queue: cho các cập nhật khi có đủ điều kiện để thực hiện.

b) Các dạng thông báo:

- Request(R, W, f): được USER gửi đến khi muốn thực hiện việc truy nhập cơ sở dữ liệu.
- Update($R, W, f, (C_i, i)$): được sinh ra khi nhận yêu cầu cập nhật của USER.
- Reply(ts): được sinh ra để trả lời cho thông báo Update nhận được từ trạm khác.
- Exec(i, t_1, t_2): được sinh ra khi yêu cầu cập nhật tại trạm S_i ở thời điểm t_1 đã đủ điều kiện để chuyển vào hàng đợi thực hiện. t_2 là thời gian sinh ra thông báo Exec.

3.5.2 Mô hình xử lý cập nhật

Khi có yêu cầu truy nhập CSDL tại một trạm S_1 nào đó, S_1 sẽ tăng C_1 lên 1, sau đó kiểm tra W có rỗng hay không. Nếu W rỗng thì yêu cầu được chuyển ngay vào hàng đợi thực hiện, ngược lại S_1 sẽ kiểm tra xem nhóm chứa nó có phải là nhóm đa số không. Nếu nhóm chứa nó là nhóm thiểu số thì yêu cầu cập nhật sẽ bị loại, ngược lại thông báo Update($R, W, f, (C_i, i)$) sẽ được đặt vào hàng Local_queue. Sau đó nó sẽ gửi thông báo Update($R, W < f, (C_i, i)$) tới tất cả các trạm trong nhóm chứa của nó. Trong khoảng thời gian này nếu sảy ra phân hoạch thì các trạm trong nhóm sẽ xử lý sau:

+ Nếu trạm thuộc nhóm thiểu số thì mọi cập nhật trong các hàng đợilocal_queue cũng như trong global_queue đều bị loại bỏ,

+ Nếu trạm thuộc nhóm đa số thì việc kiểm tra chỉ phải xét trong hàng đợi globa_queue. Những cập nhật xuất phát từ những trạm thuộc nhóm thiểu số sẽ bị loại còn những cập nhật xuất phát từ những trạm thuộc nhóm đa số vẫn được giữ lại.

Khi một trạm S_j nào đó nhận được thông báo Update() từ một trạm S_i trong nhóm, nó sẽ thực hiện:

+ $C_j = \max(C_j, t + 1)$,

+ đặt thông báo vào hàng đợi global_queue,

+ gửu thông báo Reply((C_j, j)) cho trạm S_i.

Khi trạm sinh cập nhật S_i nhận được trả lời Reply(t, j) từ trạm S_j nào đó trong nhóm nó thực hiện:

+ $C_i = \max(C_i, t+1)$,

+ Kiểm tra xem đã nhận đủ thông báo Reply cần thiết chưa. Nếu đã nhận đủ S_i thực hiện:

- Chuyển Update() vào hàng đợi thực hiện exec_queue,
- Loại Update() khỏi hàng đợi local_queue,
- Gửi thông báo exec(i, t_1, t_2) tới tất cả các trạm trong nhóm chứa hiện thời của nó. Trong đó t_1 là thời gian khởi sinh cập nhật, t_2 là thời gian hiện tại của S_i.

Một trạm S_j nào đó khi nhận được thông báo exec(i, t_1, t_2) nó thực hiện các hành động sau:

+ $C_j = \max(C_j, t_2 + 1)$,

+ Chuyển thông báo Update$(R, W, f, (i, t_1))$ vào hàng đợi thực hiện Exec_queue, đồng thời loại nó ra khỏi hàng đợi Global_queue.

Một Update$(R, W, f, (t, i))$ trong hàng đợi thực hiện Exec_queue được thực hiện khi không có một Update$(R', W', (t', j))$ nào trong hàng đợi thực hiện Exec_queue mà:

- $t' < t$,
- $R' \cap W \neq \emptyset$ or $W' \cap W \neq \emptyset$ or $W' \cap R \neq \emptyset$ or $W \cap R \neq \emptyset$.

3.6 Giao thức hợp nhất các nhóm phân hoạch

3.6.1 Các cấu trúc dữ liệu điều khiển hợp nhất

- Thông báo Var_merge(ts, L, V)
- Thông báo Group_merge(ts, L, V)
- Thông báo Repare(e, ts)
- Hàm Link_repare(e), e là tên kênh được phục hồi.
- Hàm Adopt_new_var(L, V).

Các thông báo, hàm được sinh ra hoặc gọi tại những thời điểm thích hợp trong quá trình hợp nhất.

3.6.2 Giao thức hợp nhất

Việc khôi phục kênh hoặc trạm có thể dẫn đến việc hợp nhất một vài nhóm với nhau. Khi một trạm hỏng thì bao giờ cũng kéo theo hỏng một vài kênh và khi trạm được sửa chữa có thể kéo theo một vài kênh sửa chữa. Do vậy mọi sự kiện hỏng hoặc sửa chữa trạm ta đều có thể quy về sự kiện hỏng hoặc sửa chữa kênh. Cho nên, ta chỉ cần quan tâm đến các sự kiện đối với kênh là đủ. Khi sảy ra hợp nhất thì nhóm hợp nhất chỉ có thể là nhóm đa số khi một trong các nhóm tham giam gia hợp nhất là nhóm đa số. Ngược lại hợp nhất sẽ là nhóm thiểu số. Khi trạm S_i phát hiện ra kênh liên lạc $(i-j)$ được sửa chữa thì tại trạm S_i sẽ thực hiện:

+ $C_1 = C_1 + 1$,

+ Gọi hàm Link_repare$(i-j)$;

Hàm Link_repare($i-j$) sẽ thực hiện chức năng cập nhật tại L_1 đồng thời kiểm tra xem sự kiện sửa chữa kênh $(i-j)$ có dẫn đến hợp nhất không. Nếu hợp nhất, hàm trả lại giá trị 1 ngược lại trả lại giá trị 0. Việc kiểm tra được tiến hành bằng cách trạm S_i sẽ kiểm tra S_j có nằm trong nhóm chứa nó không (kiểm tra xem có nằm trong $V_i.C$ không).

+ Gửi thông báo sự kiện này tới các trạm trong nhóm chứa nó trừ S_j bằng thông báo Repare($i-j, ts$).

Một trạm S_k nào đó sau khi nhận được thông báo này sẽ thực hiện:

- $C_k = \max(C_k, C_1 + 1)$,

- Gọi hàm Link_Repare($i-j$), nếu hàm này trả lại giá trị 1 có nghĩa là sự kiện hợp nhất xảy ra thì trạm tạm dừng tất cả các yêu cầu cập nhật của người dùng CSDL để chuẩn bị cho quá trình hợp nhất.

Trong S_j và Si chọn lấy một trạm làm trạm phối hợp còn một trạm bị phối hợp việc chọn dựa vào thứ tự các nhãn của S_i và S_j trong mạng. Trạm nào có thứ tự đi trước sẽ được chọn làm trạm phối hợp, chẳng hạn S_i, S_i sẽ gửu thông báo: Var_merge(ts_1, L_1, V_1) cho S_j. S_j sau khi nhận được thông báo Var_merge() sẽ thực hiện:

+ $C_j = \max(C_j, t_i + 1)$.

+ Gửu thông báo Var_merge(ts_j, L_j, V_j) cho S_i. S_1 sau khi nhận đượcthông báo đáp lại sẽ thực hiện:

+ $C_i = \max(C_i, t_i + 1)$.

Sau đó tại mỗi trạm sẽ tiến hành hợp nhất để đượcV(M, V, C) mới như sau:

+ $V.M = \max(V_i.M, V_j.M)$,

+ $V.C = \max(V_i.V, V_j.V)$,

+ $V.C = V_i.C \cup V_j.C$.

Mỗi trạm thuộc nhóm mới hợp nhất có L như sau:

+ $L = l_i \cup L_j \cup \{(i-j).0\}$.

Sau đó so sánh $V_i.V$ và $V_j.V$. Nếu:

a) $V_i.V = V_j.V$

S_i truyền thông báo Group_merge($ts_i, L_i, V-i$) cho các trạm lánh giềng của nó trừS_j. Trạm S_k nào đó sau khi nhận được thông báo Group_merge(ts_i, L_i, V_i) sẽ thực hiện:

+ Gọi hàm adop_new_var(L_i, V_i). Hàm này sẽ thực hiện g'ạ$L_k = L_1$ và $V_k = V_1$.

+ $C_k = \max(C_k, t_1)$.

Trạm S_j cũng thực hiện tương tự.

b) $V_i.V \neq V_j.V$. Gia sử $V_i.V > V_j.V$ thực hiện như sau:

S_i gửi L_i và V_i tới các trạm trong nhóm cù của nó. Đối với S_j thì ngoài việc gửi L_i, V_i nó còn phải lan truyền cả CSDL. Sau đó S_j lại thực hiện với các trạm trong nhóm cù của nó như S_i đà thực hiện với nó. Tại một thời điểm chỉ có nhiều nhất

một tiến trình hợp nhất được thực hiện. Nếu có sự hợp nhất khác sảy ra thì sự kiện đó chưa được thực hiện. trong quá trình hợp nhất nếu sảy ra sự kiện phân hoạch hoặc hỏng kênh, trạm thì sự kiện đó vẫn được xử lý đồng thời. Tại những trạm sau khi đà hoàn thành hợp nhất, người dùng có thể tiến hành cập nhật CSDL nếu nhóm hợp nhất là nhóm đa số.

IV. Chứng minh tính đúng đắn

Để chứng minh được tính đúng đắn của sơ đồ ta phải chứng minh được:

+ Tính nhất quán của CSDL luân được duy trì tại mỗi bản sao CSDL.
+ Sơ đồ phải đảm bảo được tính tương thích lặp của các bản sao CSDL.

Để giải quyết được hai vấn đề trên ta sẽ chứng minh một số định lý sau:

Định lý 1. *Mọi trạm làm việc trong cùng một nhóm có cùng số hiệu version.*

Chứng minh. Theo cách xây dựng một thuật toán điều khiển việc truy nhập CSDL nói trên, khi có yêu cầu truy nhập CSDL tại một trạm nào đó trong nhóm đa số, trạm đó sẽ kiểm tra xem yêu cầu truy nhập đó có cập nhật CSDL không. Nếu có, nó sẽ vận chuyển yêu cầu dó tới mọi trạm trong nhóm. Khi cập nhật có đủ điều khiện để xếp vào hàng đợi thực hiện tại trạm phát sinh nó thì cập nhật cũng được xếp vào hàng đợi thực hiện ở mọi trạm khác trong nhóm chứa trạm phát sinh cập nhật. Hơn nữa thuật toán không sinh ra deadlock. Vấn đề này được giải quyết như sau:

Giả sử $U_1, U_2, ...U_n$ là dày cập nhật trong hàng đợi thực hiện. Xây dựng đồ thị có hướng $G =< V, E >$, trong dó:

+ V là tập đỉnh bao gồm các cập nhật,
+ E là tập cạnh được xây dựng như sau:

$$E = \{(U_i, U_j),\ |U_i < U_j \text{ và}(W_i \cap R_j \neq \emptyset \text{ hoặc} W_i \cap W_j \neq \emptyset \text{ hoặc} R_i \cap W_j \neq \emptyset)\}.$$

Dễ dàng chỉ ra được rằng G không có chu trình. Do vậy thuật toán không sinh ra deadlock, do đó mọi cập nhật khi dã được xếp vào hàng đợi tực hiện đều được thực hiện. Như vậy version của mọi trạm trong cùng một nhóm là giống nhau.

Định lý 2. *Nếu hai trạm bất kỳ trong mạng có số hiệu version giống nhauthì hai bản sao CSDL giống nhau hoàn toàn.*

Chứng minh. Việc chứng minh định lý không phức tạp lắm, nên phần chứng minh định lý này không trình bày ở đây.

Định lý 3. *Giả sử G_1, G_2 là hai nhóm nào đó trên mạng. Tại thời điểm t bất kỳ nếu G_2 có version lớn hơn của G_1 thì luân luân tồn tại một dãy cập nhật tác động trên G_2 và không tác động trên G_1, có thể chuyển G_1 thành G_2 nếu nó tác động trên G_1.*

Chứng minh. Trước hết ta chứng minh định lý với điều kiện không có sự kiện sửa chữa kênh hoặc trạm làm việc, có nghĩa là không có sự kiện hợp nhất nhóm mà chỉ có sự kiện phân chia nhóm.

Ký hiệu Ver(G) là version của nhóm G.

Giả sử Ver(G_1) = n và Ver($G-2$) = m ($m > n$), và tại thời điểm t' nào đó mà ver(G_2) = n. Vì tại thời điểm t', G_2 vẫn có thể cập nhật được nên G_2 là nóm đa số và G_1 là nhóm thiểu số. Do giả thiết không có sự kiện hợp nhất sảy ra nên G_1 không thể trở thành nhóm đa số nên tại thời điểm t' thì Ver(G_1) = n. Theo định lý 2 tại thời điểm t' các bản sao CSDL tại G_1 và G_2 hoàn toàn giống nhau. Gọi $U_1, U_2, ..., U_n$ là dãy cập nhật đã chuyển CSDL từ trạng thái có version n đến trạng thái có version m, chính là dãy cập nhật cần tìm để chuyển G_1 thành G_2.

Xét với trường hợp có sự kiện hợp nhất xẩy ra. Trong trường hợp có sự lan truyền CSDL thì ta coi như là chịu sự tác động của một dãy cập nhật để chuyển sang version cao hơn. Định lý được chứng minh.

Như vậy vấn đề tương thích lặp trong sơ đồ đề xuất đã được giải quyết, bây giờ ta chỉ còn phải giải quyết vấn đề nhất quán dữ liệu đối với mỗi bản sao CSDL nữa là xong. Vấn đề này sẽ được giải quyết bằng định lý 4 dưới đây.

Định lý 4. *Sơ đồ đề xuất duy trì tính nhất quán dữ liệu đối với mỗi bản sao CSDL.*

Chứng minh. Phần chứng minh định lý này dễ dàng thu được bằng cách chỉ ra đồ thị "có thể tuần tự hóa được" của thuật toán điều khiển cập nhật tại mỗi trạm làm việc là không có chu trình. Phần chứng minh chi tiết không trình bày ở đây.

V. Kết luận

Tóm lại, bài báo trình bày một sơ đồ mới với khả năng chấp nhận được thêm những dạng phân hoạch mới mà các sơ đồ trước đây không chấp nhận. Thêm nữa, thuật toán điều khiển cập nhật CSDL phân tán hoàn toàn được áp dụng và phát triển trong điều kiện có sự phân hoạch của mạng đã làm tăng độ tin cậy và độ song song trong sơ đồ đã đề xuất.

Tài liệu tham khảo

1. Jian Tang & Natarajan N., *A scheme for increasing availability in patitioned replicated databases'* Informatic Sciences, v. 53. (1991), p. 1-34.
2. Berstein P.A. & Goodman N., *An algorithm for concurrency and recovery in replicated distributed databases,* ACM trans. Database systems 9(4):596-615 dec. 1984.
3. Davision S.B., *Optimism and consistency in patitioned distributed database systems,* ACM trans. Database systems 9(3):456-481 sept. 1984.

4. Parker D.S.,Popek G.J., Rudsin G. ..., *Detection of mutual inconsistency in distributed systems,* IEEE trans. sofware eng. SE-9 n. 3 May 1983.
5. Lamport L., *Time, clocks and ordering of events in distributed systems'* Comm. ACM, July 1978, p. 558-565.
6. Singhal M., *Update transport: a new technique for update synchronization in replicated database systems,* IEEE trans. sofware eng., v. 16, n. 12 Dec. 1990.
7. Tamer M., GTE laboratories, Patrick Valduricz, INRIA, *Distributed database systems: Where are we now?,* IEEE trans. computer, August 1991.
8. Berstein P.A. & Goodman N., *Concurrency control in distributed database systems,* Computing serveys, v. 13, n. 2, June 1981.
9. Kumar V., *Perfomance comparisionof database concurrency control mechanisms based on two-phase locking, timestamping and Mixed approach,* Information sciences, v.51(1990), p. 221-261.
10. Davidson S.B., Molina H.G. & Skeen D., *Concurrency in partitioned networks,* ACM comput. serveys 17(3) Sept. 1985.

Abstract

A new scheme for enhancing availability in distributed database systems

This paper presents a new scheme that enhances the availability of data in the full replicated distributed database systems. This scheme allows to tolerate a new type of the partition of the network that the old previous schemes in the same type doesn't allow. In addition, the update is based on a full distributed algorithm, i.e. every update is performed by all workstations, so paralellism of computations is higher than schemes using semi-distributed algorithm. This paper also offers a correctness proof of the sch