MỘT GIẢI PHÁP HIỆU QUẢ CHO VIỆC ĐỒNG BỘ HÓA DỮ LIỆU TRÊN THIẾT BỊ DI DỘNG

Nguyễn Dũng

Khoa Công nghệ Thông tin, Trường Đại học Khoa học – Đại học Huế

Email: nguyendung622@gmail.com

TÓM TẮT

Việc sử dụng phổ biến các thiết bị cầm tay như điện thoại thông minh hay máy tính bảng trong các hoạt động hàng ngày khiến cho việc đồng bộ dữ liệu trở thành một nhu cầu bức thiết. Đồng bộ đảm bảo cho dữ liệu trong các thiết bị cá nhân hoặc tổ chức được nhất quán. Các thách thức quan trọng là băng thông thấp, khả năng xử lý và giới hạn dung lượng lưu trữ của các thiết bị. Trong bài báo này chúng tôi nghiên cứu lý thuyết đồng bộ dữ liệu và đề xuất thuật toán cho việc đồng bộ dữ liệu.

Từ khóa: *dữ liệu, di động, đồng bộ.*

1. MỞ ĐẦU

Với sự bùng nổ và phát triển ngày càng mạnh mẽ của các thiết bị di động, dữ liệu của người sử dụng không còn tập trung trên một thiết bị mà nó bị phân tán rải rác trên nhiều thiết bị khác nhau. Khi tiến hành sửa đổi dữ liệu trên một thiết bị sẽ dẫn đến tình trạng dữ liệu không còn nhất quán. Do đó nhu cầu đồng bộ hóa dữ liệu trở thành vấn đề đáng quan tâm.

Đồng bộ hóa dữ liệu là quá trình trao đổi và đồng bộ hóa thông tin giữa hai nguồn dữ liệu theo thời gian. Ứng dụng của đồng bộ hóa dữ liệu rất đa dạng, có thể là đồng bộ hóa tập tin, đồng bộ hóa lịch... Việc đồng bộ dữ liệu có thể diễn ra trên nhiều loại thiết bị khác nhau, có thể là: máy tính cá nhân, điện thoại thông minh, máy tính bảng,…

Một số mô hình lý thuyết về đồng bộ hóa dữ liệu đã được công bố trong một số nghiên cứu khoa học, và vấn đề cơ bản của việc đồng bộ hóa liên quan đến bài toán mã hóa Slepian-Wolf của ngành lý thuyết thông tin. Các mô hình lý thuyết này được phân loại tùy theo việc chúng xem xét dữ liệu được đồng bộ hóa như thế nào:

- Dữ liệu không có thứ tự: Bài toán đồng bộ hóa dữ liệu không có thứ tự (còn gọi là bài toán hòa hợp tập hợp - set reconciliation problem) được mô hình hóa thành cách tính mức chênh lệch đối xứng $S_A \oplus S_B = (S_A - S_B) \cup (S_B - S_A)$ giữa hai tập xa nhau S_A và S_B. Một số cách xử lý tiêu biểu là:
    - Chuyển toàn bộ (wholesale transfer): Trong trường hợp này toàn bộ dữ liệu được truyền tới một nơi để tiến hành so sánh cục bộ. Phương pháp này dễ cài đặt và thực hiện, tuy nhiên phương pháp này có nhược điểm rất lớn

là tốn băng thông và thời gian truyền tải lớn đối với các tập dữ liệu có kích thước lớn.

- Đồng bộ hóa theo dấu thời gian (timestamp synchronization): Trong trường hợp này mọi thay đổi đối với các dữ liệu được đánh dấu bằng các dấu thời gian (timestamp). Việc đồng bộ hóa được tiến hành bằng cách chép các dữ liệu có dấu thời gian mới nhất so với lần đồng bộ hóa trước đó[1]. Phương pháp này tỏ ra hiệu quả hơn hẳn khi mà chúng ta chỉ cần truyền những thay đổi, thay đổi được ghi nhận bằng dấu thời gian, từ nguồn đến đích một cách dễ dàng. Tuy nhiên có hai vấn đề lớn cần quan tâm: một là làm thế nào để ghi nhận những thay đổi của tập dữ liệu trên nguồn này với nguồn còn lại, hai là hòa hợp những thay đổi này vào tập dữ liệu đích.
- Đồng bộ hóa kiểu toán học (mathematical synchronization): Trong trường hợp này dữ liệu được xem như những đối tượng toán học và đồng bộ hóa tương ứng với một quá trình xử lý toán học[1].

- Dữ liệu được xếp thứ tự: Trong trường hợp này, hai chuỗi xa nhau $\sigma_A$và σ_B cần được hòa hợp với nhau. Thông thường, các chuỗi này được giả định là khác nhau tới một số cố định các sửa đổi nào đó (tức là các thao tác thêm, xóa, sửa các ký tự). Sau đó quá trình đồng bộ hóa dữ liệu là việc giảm dần khoảng cách sửa đổi giữa $\sigma_A$và σ_B, cho đến khi khoảng cách sửa đổi bằng không.

Đã có nhiều nhà khoa học tiến hành xây dựng các thuật toán đồng hóa dữ liệu như: Palm HotSync, Intellisync, SyncML, CPISync, … Tuy nhiên hiện nay việc xây dựng thuật toán đồng bộ dữ liệu cần chú ý đến việc dữ liệu phân tán trên nhiều thiết bị, trong đó có các thiết bị di động[3]. Các thiết bị di động này xét về khả năng lưu trữ, khả năng xử lý và băng thông còn thấp. Do đó chúng ta tôi tiến hành nghiên cứu và xây dựng giải thuật đồng bộ nhằm để giải quyết các vấn đề nêu trên.

2. GIẢI THUẬT

Phát biểu bài toán: Giả sử chúng ta có hai thiết bị A và B kết nối với nhau với băng thông thấp và độ trễ của mạng cao. Tại thời điểm bắt đầu chuyển dữ liệu, máy A chứa một tập tin có kích thước là a_i và máy B có một tập tin có kích thước b_i (giả sử $0 \leq i < n$, với n là kích thước lớn nhất giữa hai tập tin). Mục đích của giải thuật là cho B nhận được một bản sao của tập tin từ A.

Cấu trúc cơ bản của giải thuật như sau:

1 B gửi dữ liệu S của b_i đến A

2 A đối sánh dữ liệu nhận được với a_i và gửi dữ liệu D đến B

3 B cấu trúc lại tập tin dựa vào b_i, S và D

Câu hỏi được đặt ra là khuôn dạng của *S* là gì, làm thế nào A sử dụng dữ liệu *S* để đối sánh với a_i và làm thế nào để B tái cấu trúc lại a_i.

Với cấu trúc đơn giản này, chúng ta dễ dàng nhận dữ liệu *S* mà B gửi đến A cần phải có kích thước nhỏ hơn kích thước của tập tin hoàn chỉnh để tăng tốc độ truyền tải[1].

Chúng ta có thể thực hiện một số phép thử trên thuật toán này để tìm ra lời giải tối ưu:

- Phép thử thứ nhất:

1 B chia b_i thành *N* khối với kích thước là b'_j và tính toán một chữ ký S_j cho mỗi khối. Các chữ ký này được gửi đến A

2 A chia a_i thành *N* khối có kích thước là a'_k và tính S'_k cho mỗi khối

3 A tìm S_j đối sánh với S'_k với mọi khối k

4 Với mỗi k, A có thể gửi đến B hoặc là chỉ số khối *j* khi S_j khớp với S'_k hoặc dữ liệu của khối a'_k trong trường hợp ngược lại.

5 B cấu trúc lại a_i bằng cách sử dụng các khối từ b_i hoặc các khối từ a_i.

Giải thuật này rất đơn giản nhưng lại gặp một vấn đề là nếu tập tin trên máy A giống với tập tin trên máy B nhưng chỉ khác một byte đầu tiên của tập tin thì sẽ không có khối nào so khớp với nhau và giải thuật sẽ truyền toàn bộ tập tin.

- Phép thử thứ hai:

Để giải quyết vấn đề còn tồn tại trong phép thử thử nhất chúng ta có thể cho A tạo các chữ ký không chỉ cho các khối mà còn tạo chữ ký cho tất cả các byte ở vùng biên của khối. Khi A so sánh chữ ký tại mỗi byte vùng biên với mỗi chữ ký S_j của b_i nó có thể tìm khớp trên khối này. Điều này cho phép chúng ta có thể chèn thêm và xóa bớt byte một cách tùy ý trên các tập tin.

Phương pháp này chúng ta có thể thực hiện được nhưng nó không khả thi vì chi phí tính toán một chữ ký hợp lý cho các khối là rất lớn, vì không những tạo chữ ký cho khối và còn tạo chữ ký cho tất cả byte của khối. Để thuật toán trở nên khả thi chúng ta có thể sử dụng thuật toán tạo chữ ký đơn giản nhưng như thế sẽ làm cho chữ ký trở nên yếu đi. Một chữ ký yếu sẽ làm cho thuật toán đồng bộ không còn chính xác nữa. Ví dụ: Chữ ký có thể là 4 byte đầu tiên của mỗi khối. Điều này sẽ rất dễ để tạo nên một chữ ký, nhưng giải thuật đồng bộ sẽ cho kết quả sai vì hai khối có thể không khác nhau ở 4 byte đầu tiên.

[1] Trừ khi kết nối giữa A và B là không đối xứng. Nếu liên kết từ B đến A là rất nhanh nhưng kết nối từ A đến B là rất chậm thì kích thước của S là bao nhiêu là không thành vấn đề.

Giải pháp cho vấn đề này là chúng ta không chỉ sử dụng một chữ ký cho một khối mà là hai. Nếu chúng ta gọi 2 chữ ký đó là *R* (rolling checksum, tính checksum của khối dữ liệu) và *H* (hash, tạo bảng băm cho các khối dữ liệu), thì giải thuật sẽ trở thành:

1	B chia b_i thành *N* khối với kích thước là b_j' và tính toán chữ ký R_j và H_j cho mỗi khối. Các chữ ký này được gửi đến A
2	Với mỗi byte thứ *i* trong a_i, A tính R_i' cho khối bắt đầu tại *i*.
3	A so sánh R_i' với R_j nhận được từ B
4	Với mỗi *j*, khi R_i' khớp với R_j, A tính H_i'và so sánh nó với H_j
5	Nếu H_i' khớp với H_j thì A gửi một thẻ bài đến B để xác định khối khớp và vị trí khối khớp, ngược lại A gửi byte đó đến B
6	B nhận các byte và thẻ bài từ A và sử dụng nó để tái tạo lại a_i

Như vậy để giải thuật trên được hiệu quả chúng ta cần các điều kiện sau:

- Chữ ký *R* cần chi phí tính toán thấp. Ở đây chúng ta có thể sử dụng giải thuật MD4 hoặc MD5 hoặc IDEA để tính. Hơn nữa những giải thuật này có tính bảo mật cao.
- Chữ ký *H* cần có xác suất đụng độ thấp. Bảng băm này có thể tạo một cách dễ dàng dựa vào lý thuyết bảng băm.
- A cần thực hiện thuật toán đối sánh chữ ký của các khối nhận từ B một cách hiệu quả.

Với phép thử thứ hai, chúng ta có thể giải quyết vấn đề đồng bộ dữ liệu một cách hiệu quả và ít tốn chi phí nhất. Tuy nhiên trong trường hợp tồi nhất chúng ta có thể dễ dàng nhận ra là giải thuật phải truyền toàn bộ tập tin từ A đến B khi mà không có khối dữ liệu nào so khớp với nhau.

3. MỘT SỐ VẤN ĐỀ KHÁC CỦA THUẬT TOÁN

Tái cấu trúc tập tin: Đây là một trong những phần đơn giản của giải thuật. Sau khi gửi các chữ ký đến A, B nhận được thông tin là các luồng byte từ A nếu không so khớp. Để tái cấu trúc tập tin, B cần ghi tuần tự các luồng byte nhận được hoặc khối dữ liệu của B nếu so khớp lên tập tin mới. Tất nhiên để làm được việc này ứng dụng cần được cấp phép truy xuất tập tin và tạo tập tin mới[2].

Chọn kích thước của khối dữ liệu, L ($0 < L \leq n$*):* Kích thước của các khối dữ liệu sau khi chia tập tin là vấn đề hết sức quan trọng của thuật toán. Việc chọn kích thước phù hợp phù thuộc vào các yếu tố[2]:

- Kích thước của khối phải lớn hơn kích thước của các chữ ký của khối cộng lại.
- Một khối có kích thước lớn sẽ làm giảm thông tin của chữ ký gửi từ B đến A

- Một khối có kích thước nhỏ thì xác suất so khớp trên A sẽ cao hơn, do đó nó sẽ làm giảm có lượng byte truyền từ A đến B

Do đó để xác định kích thước tối ưu của khối dữ liệu, chúng tôi giả định hai tập tin sai khác nhau một số cố định nào đó.

Chẳng hạn, chúng tôi giả định hai tập tin là giống nhau ngoại trừ một chuỗi Q byte, thì tổng số byte sẽ truyền sấp xỉ là:

$$t(L) = \frac{(s_R + s_H)n}{L} + QL + s_t\left(\frac{n}{L} - Q\right)$$

Trong đó:

- s_R là kích thước của chữ ký R
- s_H là kích thước của chữ ký H
- s_t là kích thước của token
- n là kích thước lớn nhất trong hai tập tin

Giả sử $s_R = 4, s_H = 16, s_t = 4$ thì (1) trở thành

$$t(L) = \frac{24n}{L} + Q(L - 4)$$

Trong trường hợp này giá trị tối ưu cho L là $\sqrt{24n/Q}$. Điều này có nghĩa rằng đối với các tập tin có kích thước phổ biến khoảng một vài kilobyte đến một vài megabyte và với khoảng một chục khác biệt trong hai tập tin thì kích thước khối tối ưu là khoảng vài trăm ngàn byte.

4. KẾT LUẬN

Trong bài báo này, chúng tôi đã tiến hành phân tích và xây dựng thuật toán đồng bộ dữ liệu dựa trên ràng buộc đồng bộ trên thiết bị di động. Giải thuật có thể được áp dụng hiệu quả cho các bài toán như đồng bộ lịch cá nhân và đơn vị dựa trên nhiều nguồn khác nhau.

LỜI CẢM ƠN

Đầu tiên tôi xin gửi lời cảm ơn đến trường Đại học Khoa học đã cấp kinh phí cho bài báo này (bài báo là một phần trong đề tài cấp cơ sở trường). Cám ơn Khoa Công nghệ Thông tin, trường Đại học Khoa học đã tạo mọi điều kiện để công trình này hoàn thành.

TÀI LIỆU THAM KHẢO

[1]. Minsky, Y.; Trachtenberg, A.; Zippel, R. (2003). "Set reconciliation with nearly optimal communication complexity". *Information Theory*, IEEE Transactions on 49 (9): 2213–2218. doi:10.1109/TIT.2003.815784. ISSN 0018-9448

[2]. A.Tridgell (February 1999). "Efficient algorithms for sorting and synchronization" (PDF), PhD thesis. The Australian National University

[3]. S. Agarwal, D. Starobinski, A. Trachtenberg (Aug 2002). "On the Scalability of Data Synchronization Protocols for PDAs and Mobile Devices", Department of Electrical and Computer Engineering Boston University

AN EFFECTIVE SOLUTION FOR DATA SYNCHRONIZATION ON MOBILE DEVICES

Nguyen Dung

Department of Information Technology, Hue University College of Sciences

Email: nguyendung622@gmail.com

ABSTRACT

The popular usages of portable devices such as smart phones or tablets in daily activites makes data synchronization become an urgent need. Synchronization keeps data in personal and/or organization devices in consistent state. The most important chalenges are the low bandwith, capablity of processing, and limit of storage in devices. In this paper, we review theory of data synchronization and propose an algorithm for effective synchronization of data.

Keywords: *data, mobile, synchronization.*