Giải pháp hiệu quả đảm bảo nhất quán dữ liệu chia sẻ phân tán trên nền tảng P2P có cấu trúc

An Effective Solution for The Consistency of Data Sharing and Distribution on Structured P2P Substrate

Nguyễn Hồng Minh, Nguyễn Xuân Huy

Abstract: *There are certain difficulties in ensuring the consistency of data sharing and distribution on structured P2P substrate because of the requirements of simultaneous processing interacted by many users and peer's input/output or updated speed. This paper presents a high effective solution which is proposed for structured P2P substrate, uses the updated dissemination tree and proposes a method using buffer and index vectors in order to "condition" between the requests and processes of updating. The experimental results conducted on Oversim are aimed at comparing the efficiency of new proposed solution with that of Nakashima. The experimental results indicate that the new proposed is highly effective in ensuring the consistency (over 90%) and satisfies the requirements of latency of update propagation. Especially, in case the peer's input/output or updated speed is high, the new proposed also achieve greater efficiency.*

Keyword: *P2P structured; data consistency; replica; replica node; updated dissemination tree.*

I. GIỚI THIỆU

Các ứng dụng chia sẻ dữ liệu phân tán xây dựng trên nền mạng phủ P2P như Gnutella [1], KazaA [2], Freenet [3]… ngày càng được quan tâm nghiên cứu trong những năm gần đây. P2P gồm các điểm (peer) liên kết logic tạo thành mạng phủ trên nền của mạng vật lý, chẳng hạn như Pastry [4], Tapestry [5], CAN [6], v.v… Trong đó, điểm không thuần nhất về khả năng xử lý, tốc độ vào/ra, độ trễ truyền thông điệp và băng thông sử dụng. Hơn nữa, P2P cung cấp nền tảng cho các ứng dụng xây dựng phía trên kiến trúc phân tán có khả năng tự tổ chức, khả năng chịu lỗi và đảm bảo yêu cầu sẵn sàng cao của dữ liệu chia sẻ.

Nghiên cứu trước đây tập trung đối với yêu cầu chia sẻ dữ liệu phân tán tĩnh, chủ yếu chỉ có các thao tác đọc, ít cập nhật và node có độ ổn định cao. Ngày nay, yêu cầu dữ liệu chia sẻ có thể được cập nhật thường xuyên hay thậm chí làm việc tương tác, đồng thời bởi nhiều người dùng như P2P WiKi [7], Social Networking [8], P2P collaborative workspace [9],v.v... Hơn nữa, giải pháp cần giải quyết những khó khăn do đặc trưng của mạng P2P.

Trong bài này, điểm chứa bản sao của dữ liệu chia sẻ được gọi là node. Khi cập nhật node, sự thay đổi phải được lan truyền theo phương thức hiệu quả tới các node khác trong hệ thống. Đây chính là lược đồ đảm bảo nhất quán và cũng là khó khăn, thách thức chủ yếu trong các ứng dụng chia sẻ dữ liệu phân tán [10]. Chẳng hạn, cách thức đơn giản là một node chịu trách nhiệm với khóa (được gọi là node chính) lưu trữ thông tin của tất cả các node chứa bản sao (gọi là node sao). Khi thực hiện cập nhật mới, node chính gửi thông báo trực tiếp tới tất cả các node sao. Tuy nhiên, với cách thức này thì node chính dễ trở nên quá tải, nhất là khi số lượng node tăng nhanh, tốc độ cập nhật và vào/ra của node cao.

Các hướng nghiên cứu được chia thành 2 lớp giải pháp như sau:

Một là, lớp giải pháp cho P2P không có cấu trúc để cập nhật cho các node sao, node sử dụng phương thức lan truyền kém tin cậy như: ngẫu nhiên, lan rộng và

làm ngập. Hướng nghiên cứu này có ưu điểm thực hiện đơn giản, đáp ứng yêu cầu sẵn sàng cao của dữ liệu chia sẻ. Tuy nhiên, nhược điểm là chi phí truyền thông kém hiệu quả (do sự dư thừa, trùng lặp), độ trễ lớn và khả năng xảy ra tương tranh do cập nhật dữ liệu đồng thời. Hơn nữa, giải pháp chỉ đảm bảo nhất quán xác suất, ngẫu nhiên và nhất quán yếu.

Hai là, lớp giải pháp cho P2P có cấu trúc: xây dựng phía trên nền mạng phủ P2P cây bổ trợ lan truyền cập nhật (cây cập nhật). Bất kỳ node sao nào cũng có thể cập nhật trên bản sao đang sử dụng. Tuy nhiên sự thay đổi này phải được gửi về node gốc. Node này chịu trách nhiệm gửi cập nhật tới các node sao có yêu cầu nên khắc phục tình trạng dư thừa thông điệp, khả năng xảy ra tương tranh… Các giải pháp xây dựng, xử lý những vấn đề về cấu trúc, thực hiện lan truyền cập nhật có thể khác nhau và đạt được những kết quả như: khắc phục các nhược điểm nêu trên của lớp giải pháp cho P2P không có cấu trúc, đảm bảo độ tin cậy cao và yêu cầu đảm bảo nhất quán theo thiết kế đề ra. Tuy nhiên, còn tồn tại những hạn chế trong xây dựng cây và phương thức cập nhật, dẫn đến tình trạng chưa hiệu quả về yêu cầu độ trễ, sử dụng thông điệp, mức độ nhất quán… Đặc biệt cây cập nhật có thể xảy ra tắc nghẽn làm giảm độ tin cậy, ổn định và phát sinh nhiều chi phí.

Để vượt qua những khó khăn trên, chúng tôi đề xuất giải pháp đảm bảo nhất quán dữ liệu chia sẻ theo mô hình tuyến tính [10]. Trong đó sử dụng cây cập nhật *d*-ary, gồm các node sao của mỗi đối tượng dữ liệu chia sẻ. Node sử dụng vùng đệm (buffer) lưu các bản sao và vectors chỉ số để quản lý cập nhật hiệu quả. Kết quả nghiên cứu đã có những đóng góp mới:

- Đề xuất giải pháp hiệu quả xử lý tốc độ vào/ra hoặc cập nhật của node; phân cấp hợp lý chịu trách nhiệm cập nhật, "điều hòa" giữa yêu cầu cập nhật và thực hiện cập nhật. Hơn nữa, chúng tôi cũng đề xuất phương pháp chống tắc nghẽn linh hoạt, hiệu quả.
- Chúng tôi sử dụng ứng dụng Oversim [11] để thực nghiệm giải pháp mới và giải pháp do Nakashima đề xuất [12] trên nền mạng phủ Pastry. Kết quả chỉ ra rằng giải pháp mới có hiệu quả cao đối với yêu cầu đảm bảo nhất quán (trên 90% bản sao được cập nhật), trong khi đáp ứng được yêu cầu về độ trễ lan truyền cập nhật. Đặc biệt, trong trường hợp node có tốc độ vào/ra hoặc cập nhật lớn, giải pháp mới cũng cho hiệu quả cao hơn.

Phần còn lại của bài báo được trình bày như sau: Phần II giới thiệu tổng quan các nghiên cứu đã đề xuất; phần III mô tả giải pháp; phần IV tiến hành thực nghiệm và so sánh kết quả; phần V trình bày kết luận của bài báo.

II. MỘT SỐ NGHIÊN CỨU LIÊN QUAN

Datta [13] sử dụng cho P2P không có cấu trúc và kém ổn định bằng phương thức lan rộng khi cập nhật. Trong đó, mỗi node có thông tin của một tập các node khác. Node đẩy (Push) bản sao cập nhật tới các node khác mà nó có thông tin. Khi liên kết vào cấu trúc, node thực hiện kéo (Pull) bản sao gần nhất để sử dụng. Giải pháp chỉ đảm bảo nhất quán xác suất, nhất quán yếu và tốn chi phí thông điệp do sự trùng lặp. Wang [14] phát triển hơn khi tổ chức các node sao thành chuỗi liên kết logic. Mỗi node có thông tin (định danh ID và địa chỉ IP) của α node kề nó về mỗi hướng. Như vậy, node có thông tin của 2α node kề, gọi là các node thăm dò. Khi cập nhật, node đẩy bản sao mới tới các node thăm dò hiện đang online. Node thăm dò xa nhất của mỗi hướng nhận được bản sao lại tiếp tục gửi theo hướng đó cho tới khi tất cả các node nhận được. Giải pháp đã giảm được chi phí truyền thông (70%) so với sử dụng phương thức lan rộng.

Li [15] đề xuất xây dựng cây cập nhật động gồm các node có khả năng cao (tốc độ xử lý, độ ổn định, băng thông…). Các node cao chịu trách nhiệm cho tập tối đa α node thấp. Node thấp cập nhật sẽ gửi tới node cao chịu trách nhiệm để lan truyền trong cây. Node cao này là node gốc của cây cập nhật, các node cao khác được liên kết động dựa vào khoảng cách của mạng. Shen [16] đề xuất sử dụng SWARM thực hiện nhóm các node gần nhau và cùng chủ đề như "music", "image", "Book"... (mỗi chủ đề có thể gồm nhiều tập tin chia sẻ) thành một nhóm và tạo một bản sao cho mỗi nhóm. Node có khả năng cao nhất sẽ được chọn là

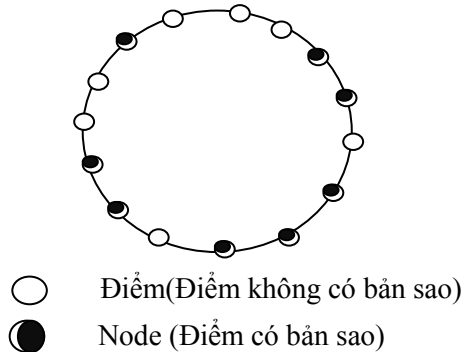
node chịu trách nhiệm (swarm server), các node khác được gọi là node phụ thuộc (client). Khi client cập nhật, nó gửi đến swarm server để lan truyền trong cây cập nhật động được tạo từ các swarm server. Trong đó swarm server gửi cập nhật là node gốc, các swarm server khác được liên kết dựa vào khoảng cách. Các giải pháp sử dụng cây cập nhật động trên có ưu điểm không tốn chi phí duy trì cấu trúc cây, tuy nhiên cũng có khó khăn để xác định khoảng cách và khả năng của node trong thực tế. Vì vậy trường hợp tốc độ cập nhật của node không cao, giải pháp này tỏ ra có hiệu quả về chi phí, độ trễ và số lượng thông điệp. Ngược lại thì giải pháp sẽ kém hiệu quả do tốn chi phí xây dựng cây cập nhật.

Chen [17] đề xuất giải pháp SCOPE phân chia liên tiếp tất cả các node trong mạng phủ P2P có cấu trúc dựa vào không gian định danh thành các phân vùng bằng nhau. Node chịu trách nhiệm đối với khóa trong không gian định danh ban đầu là node gốc. Mỗi phân vùng có node đại diện lưu thông tin vị trí các node. Chỉ node lá mới thực sự chứa các bản sao. Yêu cầu/hủy cập nhật gửi từ node lá tới node gốc thông qua các node đại diện. Node gốc gửi cập nhật trực tiếp tới node lá sau khi xác định được yêu câu thông qua các node đại diện. Giải pháp này giảm được chi phí truyền thông, tránh tương tranh cập nhật. Tuy nhiên, do node không chứa bản sao cũng tham gia vào cây cập nhật, nên số lượng node của cây sẽ rất lớn và node phải tham gia vào nhiều cây khác nhau. Điều này dễ dẫn đến quá tải, tăng chi phí xây dựng, duy trì cấu trúc; tăng độ trễ lan truyền cập nhật. Nakashima [12] đề xuất sử dụng cây cập nhật chỉ gồm các node sao. Các node liên kết vào cấu trúc theo thứ tự thời gian đến. Node gốc chỉ nhận được bản cập nhật mới khi tất cả các node sao đã nhận được bản sao trước đó. Do vậy, mặc dù có hiệu quả hơn so với SCOPE về số lượng thông điệp trao đổi, độ trễ, tuy nhiên giải pháp của Nakashima còn hạn chế về mức độ đảm bảo nhất quán (tốc độ loại bỏ cập nhật thường xấp xỉ 95%) hay độ trễ lan truyền cập nhật khi node có tốc độ vào/ra hoặc cập nhật tăng cao.

III. GIẢI PHÁP

III.1. Khái quát

Chúng tôi sử dụng mạng Pastry để làm ví dụ minh họa cho giải pháp được đề xuất. Pastry sử dụng hàm băm phân tán để định danh *ID* duy nhất cho node và dữ liệu chia sẻ trong cùng không gian định danh 128 bit (tập hợp [0, 2^{128}-1]). *ID* của node *i* (ký hiệu ID_i) băm từ địa chỉ *IP* và *ID* của dữ liệu băm từ tên tập tin. Các node liên kết logic tạo thành mạng phủ Pastry (Hình 1), có thể thực hiện trao đổi thông điệp lẫn nhau nhờ bảng định tuyến. Mỗi node được phân hoạch để chịu trách nhiệm cho một vùng không gian khóa gọi là node chính của khóa.

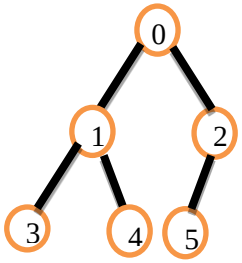


Hình 1. Phân tán dữ liệu chia sẻ trong mạngPastry

III.2. Xây dựng cấu trúc cây cập nhật

Mỗi đối tượng dữ liệu chia sẻ *f*, giải pháp đề xuất xây dựng cây cập nhật tĩnh *d*-ary (mỗi node có tối đa *d* node con) gồm các node chứa bản sao của *f*. Trong đó, node chính là node gốc của cây cập nhật (ký hiệu *R*). Mỗi node có các biến cục bộ *Child, Count* lần lượt là tập các node con và số lượng node ở phía dưới, tính cả chính nó.

Khi một node *P* có yêu cầu dữ liệu chia sẻ *f*, nó gửi yêu cầu được liên kết vào cây cập nhật (request_join) tới *R* theo định tuyến của mạng phủ. *R* nhận được yêu cầu sẽ thi hành thuật toán AGLINK (trình bày dưới đây) để liên kết *P* vào cây cập nhật. Tiếp theo *P* sẽ yêu cầu bản sao từ node cha của nó.



Hình 2 Minh họa xây dựng cây cập nhật 2-Ary

Thuật toán ALGLINK khi node yêu cầu chia sẻ dữ liệu *f*:

ALGLINK*d-Ary* Construction(*R*,*P*)

Input: *P* gửi *request_join*tới *R*

Output: Node cha của *P*

Begin

*P*gửi request_join tới *R*

// P khởi tạo các biến cục bộ

$Child_P$:= {} //tập rỗng

$Count_P$: = 1

If *R có ít hơn d Node con***Then**

$Child_R$=$Child_R \cup Child_P$

$Count_R$ + = $Count_P$

Return*R*

If else

R gửi thông điệp yêu cầu tới node con Q có $Count_Q$ *nhỏ nhất trong số node con của R và lặp lại cho đến khi tìm được node K thỏa mãn (có ít hơn d node con và có* $Count_K$ *nhỏ nhất)*

$Child_K$=$Child_K \cup Child_P$

$Count_K$ + = $Count_P$

Return*K*

End

Hình 2 minh họa phương thức xây dựng cây cập nhật *2*-Ary (mỗi node có tối đa 2 node con) cho dữ liệu chia sẻ *f*. Ban đầu node 0 được chọn là node gốc. Node 1 gửi yêu cầu liên kết vào cây tới node 0 nhờ định tuyến mạng phủ. Node 0 kiểm tra chưa có node con, nên node 1 được liên kết làm node con trái của node 0. Tiếp theo, node 2 gửi yêu cầu tới node 0. Node 0 kiểm tra chỉ có node con trái. Vì vậy, node 2 được liên kết làm node con phải của node 0. Khi node 3 gửi yêu cầu tới node 0, node 0 hiện đã có đủ 2 node con, cho nên nó sẽ gửi yêu cầu xuống cho node 1 để yêu cầu thực hiện tương tự, kết quả node 3 được liên kết làm node con trái của node 1.

III.3. Các thao tác cơ bản

Node lá chỉ có bản sao đang sử dụng. Để điều hòa yêu cầu và thực hiện cập nhật có hiệu quả cao (giảm tắc nghẽn, độ trễ và tăng mức độ nhất quán) mỗi node trong của cây cập nhật sử dụng các biến cục bộ:

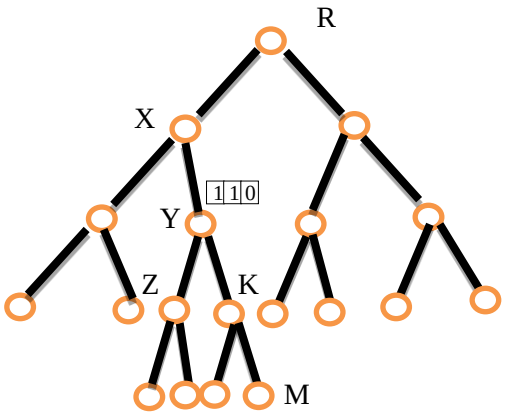
Vectors $d+1$ bit chỉ số: ghi yêu cầu cập nhật từ d node con và chính nó nhằm mục đích có thông tin vị trí node yêu cầu cập nhật. Trong đó, bit đầu tiên ghi yêu cầu của chính node đó, d bit tiếp theo lần lượt cho các node con từ trái qua phải. Bit $1,0$ chỉ ra node tương ứng có/không yêu cầu cập nhật.

Buffer kích thước $b*(d+2)$: buffer có b bản ghi và mỗi bản ghi có $d+2$ phần tử. Phần tử đầu tiên chứa bản sao. Độ lớn của b là độ lệch tối đa giữa các bản sao đang sử dụng. Buffer nhận các bản sao từ node cha hoặc xóa đi những bản sao cũ khi đã cập nhật cho tất cả các node con và cho chính nó. Thành phần $d+1$ bit gồm $1,0$ hoặc để trống tiếp theo chỉ ra phiên bản đó đã/chưa cập nhật cho node tương ứng hoặc không có node con tại vị trí đó. Như vậy, tất cả các bản ghi đều chứa bit 0, tức là bản sao đó chưa cập nhật cho tất cả các node mà nó chịu trách nhiệm.

- **Yêu cầu cập nhật*:*** node lá gửi yêu cầu cập nhật mới lên node cha khi có yêu cầu. Node trong chỉ gửi yêu cầu cập nhật lên node cha khi buffer của nó không có bản sao yêu cầu. Node cha nhận yêu cầu ghi bit 1 vào vectors, tương ứng vị trí của node yêu cầu.

Trong Hình 3, vectors có 3 bit của node Y chỉ ra: yêu cầu cập nhật từ node Z và chính node Y (tương ứng với bit 1 trong vectors), node K không yêu cầu cập nhật (tương ứng với bit 0).

- **Cập nhật*:*** node cập nhật trên bản sao cục bộ và gửi trực tiếp tới node gốc theo định tuyến trên mạng phủ Pastry. Node gốc lưu các bản sao vào buffer theo thứ tự thời gian đến $v_1, v_2, \ldots v_n$.



Hình 3. Minh họa các thao tác cơ bản

Bảng 1. Buffer của node Y trước và sau cập nhật

v_{24}	0	0	1
v_{23}	0	0	1
v_{22}	1	0	1
v_{21}	1	0	1
v_{20}	1	0	1

(a)

v_{24}	0	0	1
v_{23}	1	0	1
v_{22}	1	0	1
v_{21}	1	0	1

(b)

- **Lan truyền cập nhật:** node gốc chịu trách nhiệm gửi các bản sao xuống phía dưới theo yêu cầu và các node trong cũng thực hiện tương tự. Các node kiểm tra vectors để biết yêu cầu cập nhật. Nếu yêu cầu bản sao có trong buffer, node sẽ gửi chính xác dựa vào thông tin trong vectors. Khi đó node đồng thời ghi bit 1 vào vị trí tương ứng với node nhận cập nhật, trong bản ghi của bản sao. Nếu phiên bản yêu cầu chưa có trong buffer và buffer còn trống, node sẽ gửi yêu cầu lên node cha. Mỗi node chỉ chịu trách nhiệm cập nhật cho node con. Phương thức này đã "điều hòa" được giữa yêu cầu và thực hiện cập nhật. Do vậy node không trở nên quá tải cũng như xảy ra tương tranh hoặc tắc nghẽn khi các node cập nhật.

Ví dụ trong Hình 3 và Bảng 1a, b, node Y kiểm tra vectors biết node Z và chính node Y có yêu cầu cập nhật. Node Y cập nhật bản sao v_{23} cho chính nó đồng thời ghi bit 1 vào vị trí tương ứng trong bản ghi chứa v_{23}. Node Y cập nhật bản sao v_{20} cho node Z. Hơn nữa node Y đồng thời xóa bản ghi chứa v_{20}. Chẳng hạn node K có yêu cầu cập nhật, node Y kiểm tra trong buffer thấy rằng v_{24} là bản sao mới nhất và đã cập nhật tới node K. Vì vậy, node Y gửi yêu cầu cập nhật lên node cha của nó là node X.

III.4. Duy trì cấu trúc cây

Mỗi node sử dụng ancestor lưu thông tin của m_i node cha tính từ node cha trực tiếp cho tới node gốc. Ancestor được cập nhật đồng thời khi thực hiện cập nhật bản sao. Node sử dụng thông điệp trao đổi thường xuyên với node cha để phát hiện sự rời bỏ cấu trúc của node cha. Khi node cha rời bỏ, mỗi node con độc lập phát hiện ra và tìm cách liên kết trở lại cấu trúc cây dựa vào thông tin ancestor và thực hiện theo trình tự từ dưới lên và có thể tới node gốc. Trường hợp node gốc rời bỏ, mạng phải chịu trách nhiệm tìm node mới để thay thế. Giả sử xác suất node rời bỏ cấu trúc cây là η theo phân phối Poisson. Vậy xác suất để node con tìm được liên kết trở lại là 1- η^{m_i}.

Trường hợp node liên kết trở lại là node đại diện cho một cây con sẽ làm tăng chiều cao lớn hơn 1, nên không thể xây dựng cây cập nhật bằng thuật toán cây cân bằng truyền thống. Hơn nữa, các ứng dụng có tốc độ vào/ra của node lớn thì việc tối ưu chiều cao của cây rất khó khăn, phức tạp và không cần thiết. Trong giải pháp mới được đề xuất, mỗi node duy trì, cập nhật dễ dàng các biến cục bộ *Child, Count* nhằm tính toán sao cho cây cân bằng về số lượng node, giảm chiều cao của cây cập nhật khi node liên kết vào hệ thống (node đơn hoặc một cây con). Phương pháp này thực hiện đơn giản tuy nhiên vẫn đảm bảo hiệu quả cao so với giải pháp tối ưu hóa chiều cao của cây cập nhật.

III.5. Giải quyết tắc nghẽn

Ngay cả khi buffer đầy nhưng chỉ có các yêu cầu cập nhật các bản sao mà node đang có thì cũng không xảy ra tắc nghẽn, node vẫn đảm bảo phục vụ. Vì vậy, giải pháp đề xuất chỉ xảy ra tắc nghẽn khi buffer của một node đầy và nhận được thêm yêu cầu cập nhật bản sao mới không có trong buffer. Điều này là do tốc độ cập nhật không cân bằng giữa các node trong cây

con. Cụ thể hơn, có hai trường hợp dẫn đến buffer đầy như ví dụ trong Bảng 2 sẽ được trình bày sau đây. Để phòng ngừa tắc nghẽn, giải pháp đề xuất xử lý ngay khi buffer đầy mà không đợi đến khi có yêu cầu bản sao mới không có trong buffer bằng cách hoãn đổi liên kết của các node sao cho phù hợp nhằm giải phóng buffer đầy. Nhờ vậy, chúng ta có thể phòng ngừa được hiện tượng tắc nghẽn.

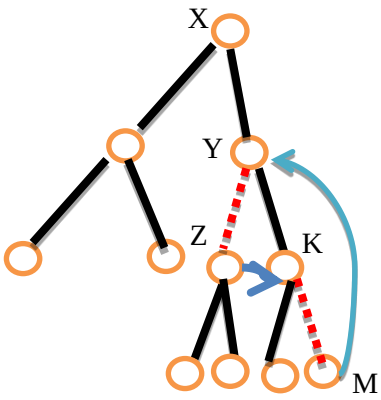
Bảng 2. Buffer của node Y đầy

v_{27}	1	0	0
v_{26}	1	0	0
v_{25}	1	0	0
v_{24}	1	0	0
v_{23}	1	0	1
v_{22}	1	0	1
v_{21}	1	0	1
v_{20}	1	0	1

(a)

v_{27}	0	0	1
v_{26}	0	0	1
v_{25}	0	0	1
v_{24}	0	0	1
v_{23}	0	1	1
v_{22}	0	1	1
v_{21}	0	1	1
v_{20}	0	1	1

(b)



Hình 4. Minh họa trường hợp 1 chống tắc nghẽn

Thứ nhất, khi tốc độ cập nhật của node Y nhanh hơn nhiều (Bảng 2a) so với 2 node con Z, K dẫn tới buffer đầy. node Y biết được node Z là node con có tốc độ yêu cầu cập nhật chậm nhất. Do vậy node Y tìm kiếm trong cây con của nó node M có tốc độ yêu cầu cập nhật tương đồng với node K để hoán đổi liên kết với node Z (Hình 4). Node M và node Z đồng thời hoán đổi chỉ số tương ứng trong buffer của nhau. Lưu ý rằng node M có tốc độ cập nhật nhanh hơn node Z, do vậy các bản sao v_{20}, v_{21}, v_{22}, ... đã được cập nhật. Do vậy, buffer lúc này của node Y có thể xóa đi các bản sao đã cập nhật cho các Node Y, K, M (Bảng 3 a, b).

Thứ hai, khi một node con yêu cầu cập nhật nhanh. Ví dụ trong Bảng 2b, node K có tốc độ cập nhật nhanh hơn nhiều so với node Y và Z. Node Y chọn node K để thay thế nó. Node Y sẽ được liên kết vào cây con của K tại vị trí phù hợp với tốc độ yêu cầu cập nhật của nó (Hình 5). Giải phóng và truyền thông tin trong buffer thực hiện tương tự trường hợp 1.

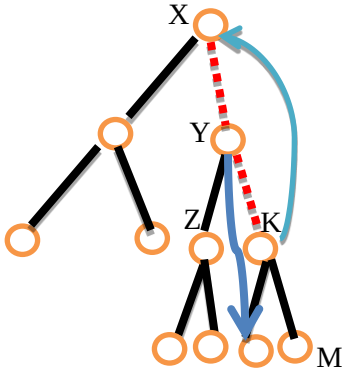
Bảng 3. Giải phóng buffer của node Y

v_{27}	1	0	0
v_{26}	1	0	0
v_{25}	1	0	0
v_{24}	1	0	0
v_{23}	1	0	1
v_{22}	1	0	1
v_{21}	1	0	1
v_{20}	1	0	1

(a)

v_{27}	0	0	1
v_{26}	0	0	1
v_{25}	0	0	1
v_{24}	0	0	1
v_{23}	0	1	1

(b)



Hình 5. Minh họa trường hợp 2 chống tắc nghẽn

IV. ĐÁNH GIÁ THỰC NGHIỆM

Thực nghiệm sử dụng Oversim mô phỏng giải pháp mới và giải pháp do Nakashima đề xuất trên nền mạng phủ Pastry. Chúng tôi lựa chọn so sánh với Nakashima vì đây là nghiên cứu tiêu biểu, hiệu quả đối với các mạng phủ P2P có cấu trúc. Kết quả thực nghiệm đánh giá ảnh hưởng của các tham số đặc trưng bởi P2P như: gia tăng số lượng node sao, tốc độ cập nhật và vào/ra của node, đối với hiệu quả của mỗi giải pháp dựa trên 2 tiêu chí đánh giá quan trọng đối với hệ thống chia sẻ dữ liệu phân tán:

- **Độ trễ lan truyền cập nhật**: thời gian trung bình nhận bản sao của tất cả các node.

- **Mức độ đảm bảo nhất quán:** tỷ lệ các bản sao được cập nhật cho tất cả các node trên tổng số cập nhật gửi tới node gốc.

Tham số cấu hình cho mạng phủ Pastry gồm: Mạng có 10^4 node. Mỗi node có bảng định tuyến gồm 40 hàng, mỗi hàng có 15 thực thể, 32 node lá. Sự không thuần nhất của các node được sinh bởi phân phối Pareto [18] với thiết lập $a = 1$ và $b = 10^4$ (a và b là tham số cận dưới và cận trên của hàm phân phối Pareto) để có 10^4 node có khả năng khác nhau.

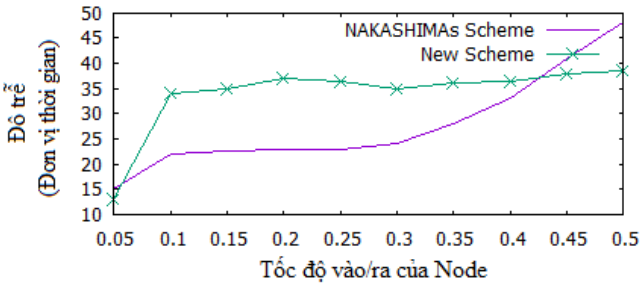
Tham số của ứng dụng và mô phỏng: Độ dài mỗi mô phỏng 1000 đơn vị thời gian. Mỗi file dữ liệu có số lượng bản sao từ 1 đến 5000 theo phân phối Zipf [19]. Tốc độ cập nhật, tốc độ vào/ra của các node sao theo phân phối Poisson. Bậc của node $d = 16$. Buffer có độ lớn $b = 20$. Kết quả chỉ ra trong các biểu đồ là kết quả trung bình của 10 lần thử.

IV.1. Độ trễ lan truyền cập nhật

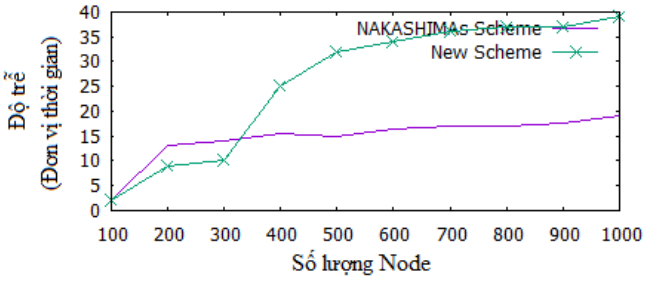
Tốc độ vào/ra của node được tính bằng tỷ lệ thời gian node tham gia chia sẻ dữ liệu trên độ dài thời gian tiến hành thực nghiệm. Trong giải pháp của Nakashima, khi tốc độ vào/ra của node nhỏ thì hầu như bản sao sẽ được gửi thành công từ node gốc tới tất cả các node sao. Ngược lại, giải pháp sẽ cần thêm nhiều chi phí do thời gian dừng hệ thống để xử lý yêu cầu vào/ra của node. Với giải pháp đề xuất, node con chủ yếu nhận các bản sao sẵn có từ node cha. Vì vậy hệ thống luôn duy trì được sự ổn định cao độ trễ lan truyền cập nhật và chỉ khi node cha rời bỏ mới có sự tác động đáng kể trong cây con. Kết quả trong Hình 6 chỉ ra, giải pháp đề xuất có độ trễ ổn định, không tăng nhanh khi tốc độ vào/ra của node tăng. Hơn nữa, giải pháp mới có hiệu quả tốt hơn giải pháp do Nakashima đề xuất khi node có tốc độ vào/ra lớn hơn 0.425.

Khi số lượng node chia sẻ dữ liệu tăng thì chiều cao cây cập nhật tăng nên độ trễ trung bình tăng theo trong cả hai giải pháp. Tuy nhiên, do độ trễ lan truyền qua mỗi bước nhỏ, nên trong giải pháp của Nakashima, khi số lượng node tăng thì độ trễ vẫn ổn định và nhỏ. Trong khi đối với giải pháp mới phải tốn thêm độ trễ vì cập nhật chỉ được gửi khi có yêu cầu. Chính vì vậy, kết quả chỉ ra trong Hình 7, giải pháp của Nakashima có kết quả ổn định và tốt hơn.

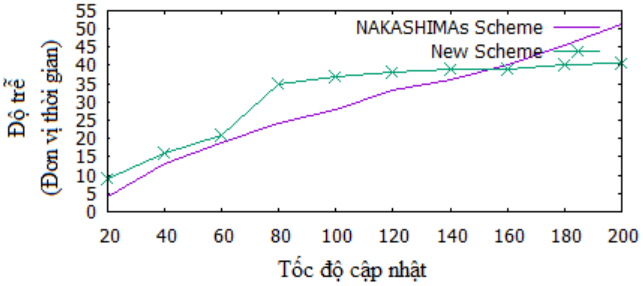
Chúng tôi giả sử trong thời gian thực nghiệm, trung bình mỗi node thực hiện tối đa 200 cập nhật. Kết quả trong Hình 8 chỉ ra: Khi tốc độ cập nhật dưới 160 thì hai giải pháp cho kết quả khá tương đồng. Tuy nhiên, khi tốc độ lớn hơn thì giải pháp đề xuất có độ trễ tốt hơn. Đặc biệt, giải pháp của Nakashima có độ trễ tăng đều so với tốc độ cập nhật, trong khi giải pháp đề xuất giữ được độ trễ ổn định. Lý do bởi vì, trong giải pháp của Nakashima, tốc độ cập nhật tỷ lệ thuận với số lượng node sao cần cập nhật bởi node gốc. Từ đó dẫn đến độ trễ luôn tăng. Tuy nhiên trong giải pháp được đề xuất thì ít chịu ảnh hưởng bởi tốc độ cập nhật do node con nhận các bản sao từ node cha.



Hình 6. Ảnh hưởng của tốc độ vào/ra



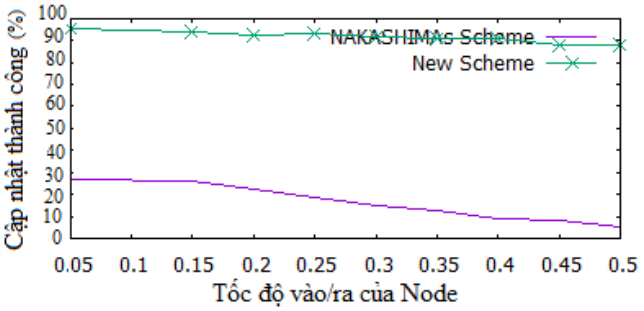
Hình 7. Ảnh hưởng của gia tăng số lượng node



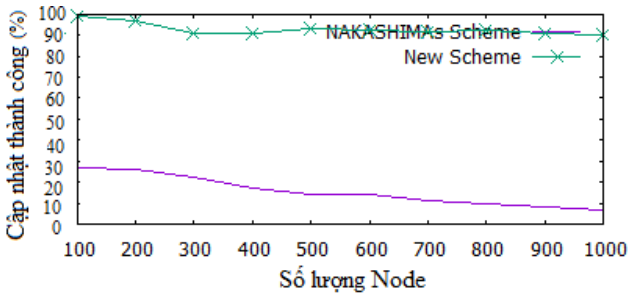
Hình 8. Ảnh hưởng của tốc độ cập nhật

IV.2. Mức độ đảm bảo nhất quán

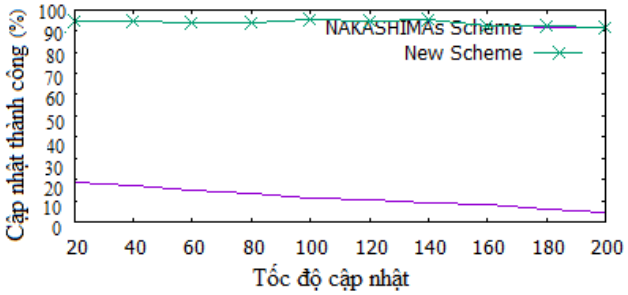
Khi tốc độ vào/ra, số lượng, tốc độ cập nhật của node tăng thì độ trễ lan truyền cập nhật cũng tăng theo dẫn đến mức độ đảm bảo nhất quán trong cả hai giải pháp đều giảm. Kết quả trong Hình 9, Hình 10, Hình 11 chỉ ra, ngay cả khi node có tốc độ vào/ra lớn (0.5), số lượng node tăng (1000) hay tốc độ cập nhật cao thì giải pháp đề xuất vẫn có kết quả cao (trên 90%) do bản sao vẫn có thể được gửi tới node gốc để thực hiện cập nhật cho đến khi buffer của node gốc đầy (tối đa *b* bản sao). Nếu tăng/giảm độ lớn của *b* sẽ làm tăng/giảm mức độ đảm bảo nhất quán. Điều này đồng nghĩa với tăng/giảm độ lệch giữa các bản sao trong mô hình nhất quán tuyến tính dẫn đến không thỏa mãn yêu cầu của ứng dụng. Vì vậy, phải căn cứ yêu cầu của ứng dụng để sử dụng giá trị *b* phù hợp. Ngược lại, giải pháp của Nakashima có tỷ lệ loại bỏ cập nhật rất xấu trong tất cả các trường hợp. Nguyên nhân do việc chỉ sử dụng một bản sao tại cùng một thời điểm, nên bản sao mới sẽ bị loại bỏ khi bản sao trước đó chưa được gửi tới tất cả các node sao (do độ trễ lan truyền). Đặc biệt khi tốc độ cập nhật cao, hiệu quả của hai giải pháp đối với mức độ đảm bảo nhất quán càng rõ rệt.



Hình 9. Ảnh hưởng của tốc độ vào/ra



Hình 10. Ảnh hưởng của gia tăng số lượng node



Hình 11. Ảnh hưởng của tốc độ cập nhật

V. KẾT LUẬN

Bài báo trình bày một giải pháp mới hiệu quả đảm bảo nhất quán dữ liệu xây dựng trên nền mạng phủ P2P có cấu trúc. Trong đó dữ liệu chia sẻ phân tán, có thể được cập nhật thường xuyên, tương tác bởi nhiều người dùng. Kết quả thực nghiệm chỉ ra rằng, giải pháp mới có hiệu quả tốt hơn giải pháp do Nakashima đề xuất về mức độ đảm bảo nhất quán với trên 90% cập nhật được thực hiện, trong khi độ trễ lan truyền cập nhật đảm bảo tương đối ổn định không tăng nhiều. Đặc biệt, trường hợp node có tốc độ cao vào/ra hoặc yêu cầu cập nhật, giải pháp mới cho hiệu quả cao hơn.

TÀI LIỆU THAM KHẢO

[1] Gnutella Org Gnutella, http://www.gnutella.org.

[2] NATHANIEL S, AND AARON KREKELBERG, GOOD, *Usability and privacy: a study of Kazaa P2P file-sharing*, Proceedings of the SIGCHI conference on Human factors in computing systems,ACM,pp. 137-144, 2003.

[3] IAN, et al CLARKE, *Freenet: A distributed anonymous information storage and retrieval system*, Designing Privacy Enhancing Technologies, Springer Berlin Heidelberg, pp. 46-66, 2001.

[4] A. ROWSTRON and P. DRUSCHEL, *Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems*,IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing, Springer Berlin Heidelberg, pp. 329-350, 2001.

[5] B. Y . ZHAO, J. D. KUBIATOWICZ, and A. D. JOSEPH, *Tapestry: An infrastructure for fault-resilient wide-area location and routing*, Technical Report UCB//CSD-01-1141, Vol. 214, U. C. Berkeley, April 2001.

[6] S. RATNASAMY, P . FRANCIS, M. HANDLEY, R. KARP, and S. SHENKER, *A Scalable Content-Addressable Network*, Proc. of ACM SIGCOMM, Vol. 31, No. 4, pp. 161-172, Aug. 2001

[7] G. PIERRE, and M. V. STEEN G. URDANETA, *A decentralized wiki engine for collaborative wikipedia hosting*, WEBIST,pp. 156-163, 2007.

[8] D. SCHIOBERG, L. H. VU, and A. DATTA S. BUCHEGGER, *Peerson: P2p social networking early experiences and insights*, Proceedings of the Second ACM EuroSys Workshop on Social Network Systems, ACM, pp. 46-52, 2009.

[9] JUN, et al WANG, *Distributed collaborative filtering for peer-to-peer file sharing systems*, Proceedings of the 2006 ACM symposium on Applied computing, ACM, pp. 1026-1030, 2006.

[10] DAVID, and Jörn KUHLENKAMP. BERMBACH, *Consistency in distributed storage systems*, Networked Systems, Springer Berlin Heidelberg, pp. 175-189, 2013.

[11] INGMAR, BERNHARD HEEP, and STEPHAN KRAUSE. BAUMGART, *OverSim: A flexible overlay network simulation framework*, IEEE Global Internet Symposium,IEEE, pp. 79-84, 2007.

[12] TAKAYOSHI, and SATOSHI FUJITA. NAKASHIMA, *Tree-Based Consistency Maintenance Scheme for Peer-to-Peer File Sharing Systems*, Computing and Networking (CANDAR), 2013 First International Symposium on, IEEE, pp. 187-193, 2013.

[13] M. H., AND ABERER, K A. DATTA, "*Updates in highly unreliable, replicated peer-to-peer systems*",Distributed Computing Systems, 2003. Proceedings. 23rd International Conference,IEEE, pp. 76-85, 2003.

[14] ZHIJUN, et al WANG, *An efficient update propagation algorithm for P2P systems*, Computer Communications, pp 1106-1115, 2007(30.5).

[15] ZHENYU LI, GAOGANG XIE, and ZHONGCHENG LI, *Efficient and scalable consistency maintenance for heterogeneous peer-to-peer systems*, IEEE Transactions on Parallel and Distributed Systems, pp 1695-1708, 2008(19.12).

[16] HAIYING, GUOXIN LIU, and HARRISON CHANDLER SHEN, *Swarm intelligence based file replication and consistency maintenance in structured P2P file sharing systems*, IEEE Transactions on Computers, pp. 2953-2967, 2015.

[17] XIN, et al CHEN, *SCOPE: Scalable consistency maintenance in structured P2P systems*, in 24th Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1502-1513, 2005.

[18] JOSEF. STEINDL, *The Pareto Distribution*, Palgrave Macmillan UK, Economic Papers, pp. 321-327, 1990.

[19] LADA A., and BERNARDO A. HUBERMAN. ADAMIC, *Zipf's law and the Internet*, Glottometrics, pp. 143-150, 2002.

Nhận bài ngày: 23/11/2016

SƠ LƯỢC VỀ TÁC GIẢ

NGUYỄN HỒNG MINH

Sinh năm 1982.

Tốt nghiệp Cử nhân khoa học máy tính, Học viện An ninh Nhân dân năm 2005; Nhận bằng Thạc sỹ Hệ thống phân tán và mạng, Đại học Besancon Pháp, năm 2010.

Lĩnh vực nghiên cứu: Hệ thống phân tán.

Email: hongminhnguyen1982@gmail.com

NGUYỄN XUÂN HUY

Sinh năm 1944 tại Nam Định.

Tốt nghiệp Cử nhân Toán, Đại học Sư Phạm Leningrad, Liên Xô năm 1973. Nhận bằng Tiến sĩ CNTT năm 1981, Tiến sĩ khoa học CNTT năm 1990 tại Trung tâm Tính toán, Viện Hàn lâm Khoa học Liên Xô.

Lĩnh vực nghiên cứu: Công nghệ phần mềm, Cơ sở dữ liệu lớn, phân tán.

Email: nxhuy564@gmail.com