Kỷ yếu Hội nghị Quốc gia lần thứ X về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Đà Nẵng, ngày 17-18/08/2017
DOI: 10.15625/vap.2017.00058

MỘT GIẢI PHÁP HIỆU QUẢ XÂY DỰNG VÀ DUY TRÌ CẤU TRÚC CÂY LAN TRUYỀN CẬP NHẬT ĐẢM BẢO NHẤT QUÁN DỮ LIỆU

Nguyễn Hồng Minh[1], Lê Văn Sơn[2]

[1]Trường Đại học An ninh Nhân dân

[2]Trường Đại học Sư phạm, Đại học Đà Nẵng

hongminhnguyen1982@gmail.com, levansupham2004@yahoo.com

TÓM TẮT: *Sử dụng cấu trúc Cây để lan truyền cập nhật nhằm đảm bảo nhất quán dữ liệu chia sẻ trong các ứng dụng phân tán xây dựng trên nền mạng phủ P2P có cấu trúc là giải pháp phù hợp và hiệu quả cao. Trong bài báo này, Nhóm tác giả đề xuất giải pháp mới xây dựng và duy trì cấu trúc Cây giúp giảm độ trễ khi tốc độ vào/ra của Node cao, không xác định. Thực nghiệm được thực hiện bằng ứng dụng mô phỏng Oversim để so sánh, chứng minh hiệu quả đạt được của giải pháp mới so với các đề xuất đã có.*

Từ khóa: *P2P có cấu trúc, nhất quán dữ liệu, cây lan truyền cập nhật, bản sao.*

I. GIỚI THIỆU

Mạng phủ P2P (chẳng hạn như Pastry [1], Tapestry [2], CAN [3]…) cung cấp nền tảng liên kết logic các điểm (gọi bằng Node) trên nền mạng vật lý. Node hoàn toàn độc lập, tự trị, không thuần nhất về khả năng xử lý, tốc độ vào/ra, độ trễ truyền thông và băng thông sử dụng. Vì vậy, mạng phủ P2P cho phép xây dựng hiệu quả các ứng dụng chia sẻ dữ liệu phân tán trên quy mô rộng lớn, phức tạp… Trong đó nhiều tiến trình sẽ sử dụng các bản sao của một đối tượng dữ liệu (gọi là dữ liệu chia sẻ). Do yêu cầu của ứng dụng và đặc trưng của mạng phủ P2P cho nên đảm bảo nhất quán giữa các bản sao luôn là khó khăn, thách thức chủ yếu [4]. Lược đồ đảm bảo nhất quán gồm giải pháp đối với "cấu trúc và phương thức" lan truyền cập nhật tới các Node khác trong hệ thống. Nhiều nghiên cứu [5] [6] [7] [8] [9] đề xuất xây dựng cấu trúc Cây bổ trợ phía trên nền mạng phủ P2P có cấu trúc để thực hiện lan truyền cập nhật (gọi là Cây cập nhật) đã chứng minh có hiệu quả hơn so với các Cấu trúc lan truyền cập nhật kém tin cậy khác (ngẫu nhiên, làm ngập hay khai khoáng) đối với tình trạng dư thừa, trùng lặp thông điệp; phòng tránh khả năng xảy ra tương tranh cập nhật dữ liệu; đảm bảo độ tin cậy cao và yêu cầu nhất quán theo thiết kế đề ra… Tuy nhiên, khi sử dụng Cây cập nhật thì tham số tốc độ vào/ra của Node (đại lượng theo phân phối Poisson) ảnh hưởng rất lớn đến hiệu quả (độ trễ, số lượng thông điệp, băng thông sử dụng…) do chi phí cao trong xây dựng, duy trì cấu trúc và lan truyền cập nhật... Đây chính là khó khăn khi sử dụng Cây cập nhật. Các giải pháp đã đề xuất còn nhiều hạn chế khi giải quyết trường hợp tốc độ vào/ra của Node cao.

Node liên kết vào Cây theo thứ tự thời gian vào hệ thống như trong các giải pháp [5] [7], nên Node liên kết với nhau nhưng có thể rất xa về địa lý hoặc logic làm tăng chi phí trao đổi thông điệp (lan truyền cập nhật) giữa các Node. Cây cập nhật xây dựng dựa vào tính toán khả năng của Node (độ ổn định, tốc độ xử lý…) hoặc tính toán khoảng cách mạng như trong các giải pháp [8] [9] sẽ rất khó khăn, tốn nhiều chi phí khi tốc độ vào/ra của Node cao.

Để vượt qua những hạn chế nêu trên, Nhóm tác giả đề xuất giải pháp xây dựng Cây cập nhật tĩnh, trong đó gán mỗi Node chịu trách nhiệm một vùng không gian định danh và tiếp tục phân chia nhỏ hơn, liên tiếp cho Node mới. Vì vậy Node liên kết trong Cây cập nhật gần nhau trong không gian định danh (được gọi là gần nhau về logic) nên sự trao đổi thông điệp giữa các Node hiệu quả hơn. Hơn thế nữa, cách thức này cho phép chúng ta giải quyết hiệu quả khi Node vào/ra hệ thống. Vì vậy giải pháp mới đã giúp giảm độ trễ lan truyền cập nhật. Kết quả nghiên cứu đã có những đóng góp mới như sau:

- Đề xuất giải pháp xây dựng Cây cập nhật và duy trì cấu trúc có hiệu quả đối với độ trễ khi Node có tốc độ vào/ra cao.
- Sử dụng Oversim [10] tiến hành thực nghiệm giải pháp mới trên nền mạng phủ Pastry, để chứng minh hiểu quả đạt được về độ trễ so với giải pháp của Nakashima [5] và Li [9].

Phần còn lại của bài báo được trình bày như sau: Phần II giới thiệu một số nghiên cứu liên quan; phần III mô tả giải pháp; phần IV trình bày kết quả thực nghiệm và thực hiện so sánh, đánh giá; phần V trình bày kết luận của bài báo.

II. MỘT SỐ NGHIÊN CỨU LIÊN QUAN

Hu [7] đề xuất xây dựng Cây cập nhật tĩnh, trong đó Node liên kết theo thứ tự thời gian đến và thực hiện hoán đổi vị trí của Node dựa vào tốc độ yêu cầu cập nhật để chống tắc nghẽn khi lan truyền cập nhật. Phương thức xây dựng Cây có ưu điểm là thực hiện đơn giản, tuy nhiên do Node liên kết với nhau có thể rất xa về địa lý hoặc logic, nên hiệu quả truyền thông thấp. Hơn nữa, giải pháp hoán đổi vị trí của Node cũng gặp khó khăn khi tốc độ vào/ra của Node cao.

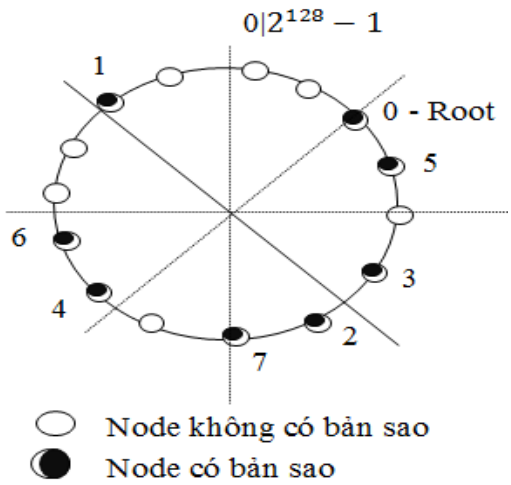
Nakashima [5] xây dựng Cây cập nhật tĩnh, Node liên kết vào Cây theo thứ tự thời gian đến, theo xác xuất Node biết được thông tin của Node gốc và cân bằng số lượng Node trong mỗi Cây con. Node gốc chỉ nhận bản cập nhật mới khi tất cả các Node trong Cây cập nhật đã nhận được bản sao trước đó.

Li [9] đề xuất xây dựng Cây cập nhật động gồm các Node có khả năng cao (tốc độ xử lý, độ ổn định, băng thông…) được liên kết dựa vào khoảng cách của mạng. Mỗi 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. Shen [8] đề xuất giải pháp SWARM, nhóm các Node gần nhau về khoảng cách và có 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 được chọn là Node chịu trách nhiệm (swarm server), các Node khác 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 liên kết dựa vào khoảng cách. Các giải pháp sử dụng Cây cập nhật động có ưu điểm không tốn chi phí duy trì cấu trúc, tuy nhiên trong thực tế rất khó khăn để xác định khoảng cách và khả năng của Node.

III. GIẢI PHÁP

A. Khái quát

Nhóm tác giả sử dụng mạng phủ Pastry làm ví dụ minh họa cho giải pháp đề xuất. Tuy nhiên chúng ta cũng có thể ứng dụng cho các mạng phủ P2P có cấu trúc khác như Tapestry, CAN… 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. Node liên kết logic trong Pastry như Hình 1 và trao đổi thông điệp nhờ định tuyến của mạng phủ. Chi phí để truyền thông điệp giữa hai Node bất kỳ là $\log NP$, trong đó NP là số lượng Node tham gia Pastry.



Hình 1. Node tham gia mạng phủ Pastry

B. Xây dựng 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 $d = 2^b$ Node con). Phương thức xây dựng Cây như sau:

Node i được gán chịu trách nhiệm cho một vùng không gian định danh, ký hiệu ws_i và phân chia d phần liên tiếp bằng nhau ký hiệu là ws_i^1, ws_i^2…ws_i^d (ws_i^m với $m = \overline{1,d}$). N_i^m là ký hiệu của Node con tại vị trí thứ m của Node i.

Node chịu trách nhiệm cho khóa (dữ liệu chia sẻ) ban đầu được chọn là Node gốc (ký hiệu R) của Cây cập nhật, chịu trách nhiệm cho toàn bộ không gian định danh [0, 2^{128}-1]. Khi Node j muốn chia sẻ dữ liệu sẽ gửi yêu cầu tới Node R, Node này kiểm tra $ID_j \in ws_R^m$ ($m = \overline{1,d}$). Nếu Node gốc chưa có Node con chịu trách nhiệm cho không gian định danh ws_R^m, thì j được liên kết làm Node con tại vị trí thứ m (trở thành Node N_R^m) và $ws_j := ws_R^m$. Nếu R đã có Node con N_R^m chịu trách nhiệm cho không gian định danh ws_R^m, thì Node gốc sẽ gửi yêu cầu tới N_R^m để tiếp tục kiểm tra ID_j thuộc về vùng không gian định danh nào trong $ws_{N_R^m}$ (tiếp tục phân chia không gian định danh $ws_{N_R^m}$ thành d phần bằng nhau) và thực hiện tương tự cho tới khi tìm được Node k thỏa mãn: $ID_j \in ws_k^m$ và chưa có Node con N_k^m. Thuật toán ID_LINK giả Code được trình bày như dưới đây. Do vậy, Node i chịu tránh nhiệm không gian định danh ws_i (dĩ nhiên $ID_i \in ws_i$), sẽ là Node cha hay tổ tiên của tất cả các Node có ID thuộc về không gian định danh này nên các Node được liên kết gần nhau về logic trong Cây cập nhật.

Bảng 1. Thuật toán giả code ID_LINK Node liên kết vào Cây cập nhật

ID_LINK d-Ary Construction(R, j)
Input: Node j gửi request_join tới R **Output**: Node cha của Node j

Begin
$ID_j \in ws_R^m$
// R kiểm tra ID của j thuộc vùng không gian định danh m trong ws_R ([0, 2^{128}-1])
If nếu R đã có Node con N_R^m **Then**
R gửi yêu cầu tới N_R^m và N_R^m thực hiện tương tự cho đến khi tới Node k không thỏa mãn
//Node k chưa có Node con thỏa mãn $ID_j \in ws_k^n$
$ws_j := ws_k^n$ // $ID_j \in ws_k^n$ ($n = \overline{1,d}$)
Return k
Else if
$ws_j := ws_R^m$
Return R
End

Ký hiệu p_l là xác suất Node mới sẽ được liên kết vào Cây cập nhật tại mức l ($0 < l \le L$, $L = log_d N$ là chiều cao, N số lượng Node của Cây cập nhật).

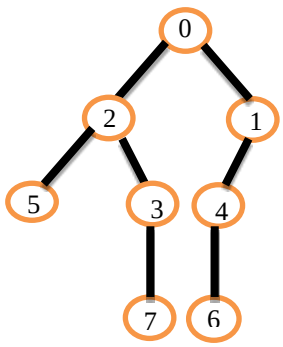
$$p_l = \frac{d^l}{N}, \text{ chú ý rằng: } \sum_{l=0}^{l=L} p_l = 1$$

Ký hiệu $cost_l$ (đơn vị tính bằng số bước (hop)) là chi phí lan truyền thông điệp (cập nhật) từ Node gốc tới Node bất kì tại mức l, ta có

$$\begin{aligned} cost_l &= \log\frac{N}{d^0} + \log\frac{N}{d^1} + \cdots + \log\frac{N}{d^{l-1}} \\ &= \log N + (\log N - b) + \cdots + (\log N - b * (l-1)) \text{ (do } L = \log_d N = \frac{\log N}{b}) \\ &= l * \log N - \frac{b*l*(l-1)}{2} \end{aligned} \quad (1)$$

Như vậy chi phí (đơn vị tính bằng hop) Node mới liên kết vào Cây cập nhật là:

$$cost = p_1 * cost_1 + p_2 * cost_2 + \cdots p_L * cost_L = \sum_{l=1}^{l=L} p_l * cost_l \quad (2)$$



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

Hình 2 minh họa phương thức xây dựng Cây cập nhật *2-Ary*. Nhóm tác giả đánh số các Node là $1, 2 \ldots 7$ theo thứ tự thời gian đến của nó vào Cây cập nhật. Ban đầu Node 0 được chọn là Node gốc. Node 1 gửi yêu cầu liên kết tới Node 0. Node 0 kiểm tra $ID_1 \in ws_0^2$ và Node 0 chưa có Node con nào được chỉ định để chịu trách nhiệm trong vùng định danh ws_0^2. Do vậy Node 1 liên kết làm Node con bên phải của Node 0 và $ws_1 := ws_0^2$. Tiếp theo, Node 2 đến. Node 0 kiểm tra $ID_2 \in ws_0^1$ và Node 0 cũng chưa có Node con nào được chỉ định chịu trách nhiệm trong vùng định danh ws_0^1. Vì thế Node 2 liên kết làm Node con bên trái của Node 0 và $ws_2 := ws_0^1$. Khi Node 3 đến, Node 0 kiểm tra $ID_3 \in ws_0^1$. Tuy nhiên Node 2 đã được chỉ định chịu trách nhiệm cho vùng định danh ws_0^1, do vậy Node 0 gửi yêu cầu cho Node 2. Node 2 kiểm tra $ID_3 \in ws_2^2$ và Node 2 chưa có Node con nào được chỉ định chịu trách nhiệm trong vùng

định danh ws_2^2. Node 3 liên kết làm Node con bên phải của Node 2 và $ws_3 := ws_2^2$. Các Node $4, 5, 6, 7$ cũng được liên kết vào Cây cập nhật theo cách thức tương tự.

C. Duy trì Cây cập nhật

Khi tốc độ vào/ra của Node cao thì yêu cầu duy trì Cây cập nhật là hết sức khó khăn và ảnh hưởng quan trọng đến hiệu quả của ứng dụng. Trong đề xuất mới, mỗi Node giữ thông tin của Node cha, Node ông (có thể cập nhật những thông tin này đồng thời với khi lan truyền bản sao mới) và trao đổi thông điệp thường xuyên với nhau để phát hiện sự rời bỏ cấu trúc (chẳng hạn như sau một số lượng nhất định thông điệp gửi đi mà không có sự phản hồi nào thì cho rằng Node đã rời bỏ). Khi có sự thay đổi, các Node có liên quan sẽ được truyền tham số là vùng không gian định danh mới mà nó chịu trách nhiệm hoặc thay đổi Node con chịu trách nhiệm. Giải pháp mà Nhóm tác giả đề xuất chỉ cần thực hiện rất ít sự thay đổi cấu trúc vì vậy cho hiệu quả cao ngay cả với trường hợp Node lỗi. Có hai trường hợp Node rời bỏ cấu trúc sau:

1. Node chủ động rời bỏ

Nếu i là Node lá rời bỏ, nó chỉ cần thông báo tới Node cha j để biết vùng định danh ws_i hiện không có Node con chịu trách nhiệm. Nếu là Node trong thì nó chọn một Node lá k trong Cây con thay thế cho chính nó.

2. Node bị lỗi

Nếu là Node lá, Node cha sẽ phát hiện ra sự rời bỏ qua trao đổi thông điệp và thực hiện tương tự như trên. Nếu là Node trong, mỗi Node con của Node bị lỗi độc lập phát hiện ra điều này và chúng sẽ cùng gửi thông điệp tới Node ông với những nội dung: yêu cầu liên kết trở lại và đề xuất Node thay thế (Node lá trong mỗi Cây con). Node ông lựa chọn một Node trong số các Node được đề xuất để thay thế cho Node vừa rời bỏ. Trường hợp Node gốc bị lỗi thì mạng phủ chịu trách nhiệm tìm ra Node mới thay thế.

D. Các thao tác cơ bản

1. Thao tác cập nhật

Các Node đều có thể thực hiện cập nhật trên bản sao của nó, tuy nhiên sự thay đổi này cần gửi tới Node gốc qua định tuyến của mạng phủ Pastry, chi phí $\log N$ bước (hop). Node gốc chịu trách nhiệm xuất bản cập nhật nhằm tránh xảy ra tương tranh cập nhật dữ liệu.

2. Lan truyền cập nhật

Node gốc lan truyền cập nhật cho các Node con và thao tác được lặp lại cho đến khi các Node lá nhận được bản sao cập nhật. Chú ý rằng, các bản cập nhật mới tới Node gốc sẽ bị loại bỏ khi bản sao trước đó chưa được gửi tới tất cả các Node. Độ trễ là thời gian trung bình mỗi bản sao lan truyền tới tất cả các Node, được tính toán bằng biểu thức sau:

$$\begin{aligned} \text{Latency}(average) &= \frac{d^1 cost_1 + d^2 cost_2 + \cdots + d^L cost_L}{d^1 + d^2 + \cdots + d^L} \\ &= \frac{\sum_{l=1}^{l=L} d^l (l * \log N - \frac{b * l * (l-1)}{2})}{N-1} \end{aligned} \quad (3)$$

IV. THỰC NGHIỆM

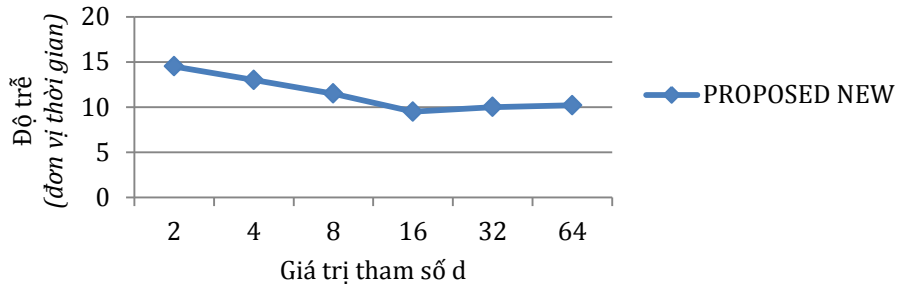
A. Thiết lập các tham số mô phỏng

Nhóm tác giả sử dụng Oversim để tiến hành thực nghiệm giải pháp mới trên nền mạng phủ Pastry nhằm đánh giá sự ảnh hưởng tốc độ vào/ra của Node đối với Độ trễ lan truyền cập nhật. Từ đó so sánh kết quả với các giải pháp của Nakashima và Li. Đây là hai giải pháp sử dụng Cây cập nhật có hiệu quả cao và tương đồng với giải pháp đề xuất về phương thức cập nhật. Trong đó bản sao mới sẽ cập nhật cho tất cả các Node, không giống như phương thức cập nhật của Chen [6] là chỉ lan truyền cho các Node có yêu cầu cập nhật hay của Hu [7] sẽ lan truyền tất cả các cập nhật được tạo ra.

Tham số cấu hình mạng phủ Pastry gồm: mạng phủ với 5000 Node và số lượng đối tượng dữ liệu chia sẻ từ 10^2 tới 10^4 theo phân phối Zipf [12]. Mỗi Node có bảng định tuyến gồm 40 hàng, mỗi hàng có 15 thực thể, 32 Node lá. Node được sinh bởi phân phối Pareto [11] với thiết lập a = 1 và b = 5000 để có 5000 Node có khả năng khác nhau.

Tham số của ứng dụng và mô phỏng: Độ dài mỗi mô phỏng là 1000 đơn vị thời gian. Số lượng Node chia sẻ của mỗi đối tượng dữ liệu theo phân phối Zipf với s = 1, N = 5000. Tốc độ cập nhật trên bản sao của các Node theo phân phối Poisson với $\lambda = 0{,}05$. Kết quả chỉ ra trong các biểu đồ là trung bình của 10 lần thử.

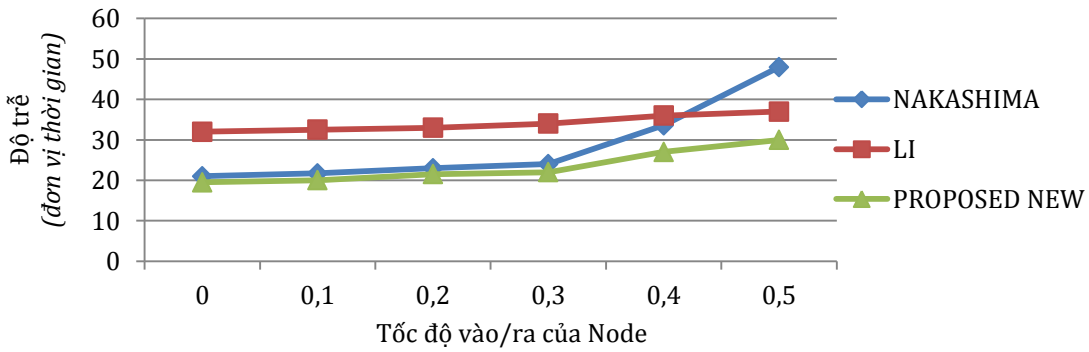
B. Ảnh hưởng của tham số d



Hình 3. Ảnh hưởng của tham số *d* đối với độ trễ

Kết quả trong Hình 3 chỉ ra ảnh hưởng của tham số d đối với độ trễ lan truyền cập nhật (số lượng Node sao là 100). Thời gian lan truyền cập nhật như biểu diễn trong biểu thức (3) phụ thuộc vào chiều cao của Cây và Bậc của Node (giá trị $d = 2^b$). Khi d nhận giá trị nhỏ thì Cây sẽ có chiều cao lớn nên tốn chi phí lan truyền qua nhiều mức. Ngược lại khi d nhận giá trị lớn thì Cây có chiều cao nhỏ giúp giảm chi phí lan truyền giữa các mức nhưng tăng chi phí lan truyền trong cùng một mức (do có nhiều Node hơn trong cùng một mức). Vì vậy, chúng ta thấy rằng khi $\mathrm{d} = 16$ thì độ trễ là nhỏ nhất. Đó là lý do trong thực nghiệm, Nhóm tác giả sử dụng giá trị $\mathrm{d} = 16$.

C. Ảnh hưởng do tốc độ vào/ra của Node



Hình 4. Ảnh hưởng do tốc độ vào/ra của Node

Tốc độ vào/ra của Node là đại lượng theo phân phối Poisson, có thể xác định bằng tỷ lệ thời gian Node tham gia chia sẻ dữ liệu trên tổng thời gian tiến hành thực nghiệm. Kết quả trong hình 4 chỉ ra, giải pháp mới có độ trễ nhỏ hơn so với các giải pháp của Nakashima và Li. Hơn thế nữa, khi tốc độ vào/ra của Node cao thì độ trễ không tăng nhanh (như Nakashima). Điều này là do những nguyên nhân: thứ nhất khi Node liên kết gần nhau về logic trong Cây cập nhật nên giảm thời gian truyền thông điệp giữa các Node để duy trì cấu trúc khi Node vào/ra, cũng như giảm thời gian lan truyền cập nhật. Chẳng hạn chi phí truyền thông từ Node mức l tới Node mức $l+1$ chỉ là $\log\frac{\mathrm{N}}{\mathrm{d}^{\mathrm{l}}}$, trong khi đối với giải pháp của Nakashima chi phí luôn là $\log \mathrm{N}$ (giữa hai Node bất kì). Thứ hai, đối với giải pháp đề xuất chỉ cần thay đổi rất ít cấu trúc của Cây cập nhật nhờ việc quản lý thông tin vùng không gian định danh chịu trách nhiệm của mỗi Node. Do vậy Node đơn hoặc Node đại diện cho Cây con dễ dàng tìm được Node cha mới để liên kết trở lại Cây cập nhật. Ngược lại giải pháp của Li luôn có chi phí cao để xác định khả năng, khoảng cách của Node mới hoặc Node vào lại nhằm xây dựng Cây cập nhật động (tốc độ cập nhật $\lambda = 0{,}05$ trong thực nghiệm) và giải pháp của Nakashima thì do kém hiệu quả trong việc xây dựng, duy trì cấu trúc Cây cập nhật theo thứ tự thời gian vào hệ thống và cân bằng số lượng Node trong mỗi Cây con khi tốc độ vào/ra của Node cao (> 0,3).

V. KẾT LUẬN

Bài báo đã trình bày một giải pháp mới hiệu quả để xây dựng và duy trì cấu trúc Cây cập nhật – vấn đề khó khăn, phức tạp của lược đồ đảm bảo nhất quán dữ liệu trong các ứng dụng phân tán xây dựng trên mạng phủ P2P có cấu trúc. Kết quả của các thực nghiệm chỉ ra rằng, giải pháp có hiệu quả cao đối với độ trễ lan truyền cập nhật (giảm được độ trễ và độ trễ không tăng nhanh khi Node có tốc độ vào/ra cao) so với các giải pháp đã đề xuất.

TÀI LIỆU THAM KHẢO

[1] Rowstron, Antony, and Peter 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, Germany, pp.329-350, 2001.

[2] Zhao, Ben Y, et al, "Tapestry: A resilient global-scale overlay for service deployment", IEEE Journal on selected areas in communications, pp.41-53, 2004.

[3] Cook, J. A, and J. S. Freudenberg, "Controller Area Network (CAN)", 2007.

[4] Bermbach, David, and Jörn Kuhlenkamp, "Consistency in distributed storage systems", Networked Systems, Springer Berlin Heidelberg, pp.175-189, 2013.

[5] Nakashima, Taishi, and Satoshi Fujita, "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.

[6] Chen, Xin, Shansi Ren, and Haining Wang, "SCOPE: Scalable consistency maintenance in structured P2P systems", in 24th Annual Joint Conference of the IEEE Computer and Communications Societies, In INFOCOM, pp.1502-1513, 2005.

[7] Hu, Yi, Laxmi N. Bhuyan, and Min Feng, "Maintaining data consistency in structured P2P systems", IEEE Transactions on Parallel and Distributed Systems, pp.2125-2137, 2012.

[8] Shen, Haiying, Guoxin Liu, and Harrison Chandler, "Swarm intelligence based file replication and consistency maintenance in structured P2P file sharing systems", IEEE Transactions on Computers, pp.2953-2967, 2015.

[9] Li, Zhenyu, 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.

[10] Baumgart, Ingmar, Bernhard Heep, and Stephan Krause, "OverSim: A flexible overlay network simulation framework", IEEE Global Internet Symposium, pp.79-84, 2007.

[11] Steindl, Josef, "The Pareto Distribution", Economic Papers 1941-88, Palgrave Macmillan UK, pp.321-327, 1990.

[12] Adamic, Lada A, and Bernardo A. Huberman, "Zipf's law and the Internet", Glottometrics, vol. 3, no 1, pp.143-150, 2002.

AN EFFECTIVE SOLUTION IN CONSTRUCTING AND MAINTAINING THE DISSEMINATION UPDATE TREE FOR THE DATA CONSISTENCY

Nguyen Hong Minh, Le Van Son

ABSTRACT: *Utilizing a tree structure to disseminate the updates in order to ensure the consistency of shared data in distributed applications, which is built on the structured P2P substrate, is a suitable and effective solution. In this paper, the authors propose a new solution to construct and maintain a tree structure to reduce the latency when the node's join/ leave hight rate and unpredictable. The experiment conducted by using the OverSim simulating application proves its effectiveness in comparison to existing solutions.*

Keyword*: structured p2p, data consistency, dissemination tree, replica.*