Song song hóa thuật toán Lamport trong loại trừ tương hỗ phân tán

Đặng Hùng Vĩ[1], Lê Văn Sơn[1], Nguyễn Xuân Huy[2]
[1] Trường Đại học Sư phạm, Đại học Đà Nẵng
[2] Viện Công nghệ Thông tin, Viện Hàn lâm Khoa học và Công nghệ Việt Nam

Tác giả liên hệ: Đặng Hùng Vĩ, dhungvi@ued.udn.vn
Ngày nhận bài: 05/04/2019, ngày sửa chữa: 04/12/2019, ngày duyệt đăng: 04/12/2019
Định danh DOI: 10.32913/mic-ict-research-vn.v2019.n2.848
Biên tập lĩnh vực điều phối phản biện và quyết định nhận đăng: PGS.TS. Trần Minh Quang

Tóm tắt: Hệ phân tán là hệ thống cung cấp tài nguyên dùng chung với quy mô lớn. Hệ phân tán sử dụng cơ chế truyền thông điệp để hợp lực trong môi trường truyền thông. Trong hợp lực, nhiều tiến trình cùng tương tranh tài nguyên dùng chung dễ dẫn đến bế tắc trong cung cấp tài nguyên. Loại trừ tương hỗ phân tán cho phép chỉ có một tiến trình duy nhất được thực thi trong miền găng tại một thời điểm đối với một tài nguyên để giải quyết bế tắc. Để đạt được loại trừ tương hỗ phân tán, các tiến trình phải được gắn dấu đồng hồ lô-gic để xác lập trật tự và loại trừ các tiến trình gây ra bế tắc. Bài báo trình bày giải pháp song song hóa thuật toán Lamport trong loại trừ tương hỗ phân tán. Kết quả giải pháp là xác lập giá trị đồng hồ lô-gic dựa trên song song hóa thuật toán Lamport và xác định các tiến trình thực thi trong đảm bảo tính nhất quán và gắn bó trong hệ phân tán.

Từ khóa: *Hệ phân tán, đồng hồ lô-gic, thuật toán Lamport, loại trừ tương hỗ phân tán, truyền thông điệp.*

Title: **A Parallelization of the Lamport Algorithm for Distributed Mutual Exclusion**

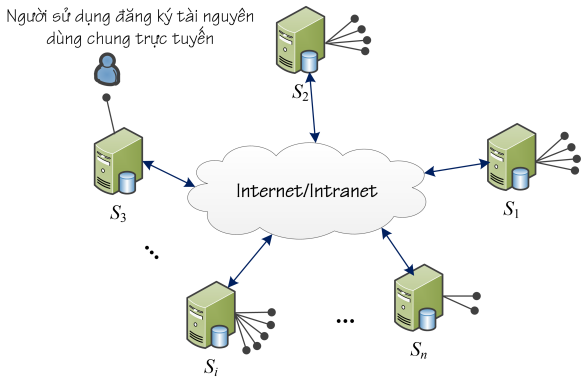
Abstract: A distributed system is a complex system in which the shared resources are allocated at a large scale. Such a system uses the message passing mechanism over the communication environment to coordinate the system's entities. During coordination, multiple concurrent processes might request the same resources, leading to deadlock in resource allocation. In order to resolve this deadlock, distributed mutual exclusion allows only one process to be executed in the critical section at a time for each shared resource. To this end, each process is assigned a timestamp to establish an order, and the processes that cause deadlock are eliminated. There are several proposed distributed mutual exclusion algorithms such as by Lamport, Ricart–Agrawala, Raymond, and Suzuki–Kasami. In this paper, we develop a parallelization of Lamport algorithm for distributed mutual exclusion. Our solution establishes a global state and determines the implementation process in the critical section to ensure consistency and coherence in discrete systems.

Keywords: *Distributed system, logical clock, Lamport algorithm, mutual exclusion distributed, message passing.*

I. GIỚI THIỆU

Hiện nay, các ứng dụng lớn trên môi trường điện toán đám mây được triển khai trong hệ phân tán để đáp ứng số lượng người dùng cực đại. Theo nghiên cứu trong [1], đám mây là một hệ thống song song và hệ phân tán bao gồm một tập hợp các máy chủ được kết nối và ảo hóa, được cung cấp động và được xử lý dưới dạng một hoặc nhiều tài nguyên tính toán hợp nhất dựa trên các thỏa thuận cấp dịch vụ được thiết lập thông qua thỏa thuận giữa nhà cung cấp dịch vụ và người dùng. Điện toán đám mây là một giải pháp toàn diện cung cấp hạ tầng, dịch vụ công nghệ thông tin. Đây là một giải pháp điện toán dựa trên internet cung cấp tài nguyên dùng chung thông qua hệ phân tán.

Hệ phân tán, được biểu diễn trên hình 1, là một tập hợp các máy chủ kết nối qua môi trường truyền thông trong cung cấp tài nguyên dùng chung. Nếu xét hoạt động mỗi máy chủ một cách độc lập, không có sự phối hợp để chia sẻ tài nguyên dùng chung thì đây là hệ tập trung. Nếu xét các máy chủ hợp lực để chia sẻ tài nguyên dùng chung thì đây là hệ phân tán. Sự hợp lực các máy chủ là sự phối hợp giữa các máy chủ với nhau thông qua môi trường truyền thông để cung cấp tài nguyên dùng chung cho người sử dụng. Khác biệt giữa hệ tập trung và hệ phân tán là các đặc tính như: tính gắn bó, khả năng chịu lỗi, sự mở rộng, cân bằng tải, v.v. Các nghiên cứu, triển khai hệ phân tán để cung cấp tài nguyên dùng chung tập trung vào giải pháp đảm bảo gắn bó dữ liệu [2]. Giải pháp gắn bó dựa trên



Hình 1. Mô hình kết nối trong hệ phân tán cung cấp tài nguyên dùng chung.

sự hợp lực của các máy chủ trong hệ phân tán thông qua cơ chế truyền thông điệp [3]. Hệ phân tán không có đồng hồ dùng chung, do đó, giải pháp của bài báo cải tiến thuật toán Lamport để giải quyết loại trừ tương hỗ phân tán trong cung cấp tài nguyên dùng chung.

Nội dung chính của bài báo được tổ chức như sau. Phần II trình bày các nghiên cứu liên quan. Phần III đề xuất giải pháp song song hóa thuật toán Lamport trong loại trừ tương hỗ phân tán. Phần IV trình bày hiệu năng thực thi song song hóa thuật toán Lamport. Phần V đưa ra kết luận.

Trong toàn bộ bài báo, chúng tôi ký hiệu N là số máy chủ, T là độ trễ của quá trình đồng bộ hóa và E là thời gian thực thi miền găng.

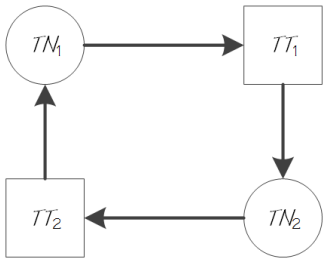
II. CÁC NGHIÊN CỨU LIÊN QUAN

Các nghiên cứu trong [4, 5] trình bày về truyền thông trong hệ phân tán đề cập đến cơ chế truyền multicast. Trong cơ chế này, gói tin vào và ra một máy chủ không tuân thủ nguyên tắc về lưu lượng như truyền unicast. Truyền multicast là sự kết hợp đặc biệt từ các máy chủ kết nối với máy chủ phát thông tin truyền.

Cơ chế hợp lực sử dụng truyền multicast cho phép các máy chủ kết nối với nhau thông qua môi trường truyền thông truyền thông điệp qua lại nhằm xác định các tiến trình di chuyển trong hệ phân tán [3]. Trong quá trình hợp lực, nhiều tiến trình cùng tương tranh tài nguyên dùng chung dễ dẫn đến bế tắc trong cung cấp tài nguyên [6–9].

Theo nghiên cứu của Singhal trong [10], quá trình bế tắc diễn ra khi hai hay nhiều tiến trình chiếm giữ tài nguyên dùng chung được giới hạn và đồng thời tiếp tục phát đi yêu cầu tài nguyên khác đang bị chiếm giữ. Các quá trình này tạo ra một vòng tròn khép kín làm cho các tiến trình kẹt chéo lẫn nhau dẫn đến bế tắc trong cung cấp tài nguyên theo mô tả trong hình 2.

Nhiều giải pháp xử lý phân tán liên quan đến việc chia sẻ tài nguyên dùng chung giữa các tiến trình khác nhau



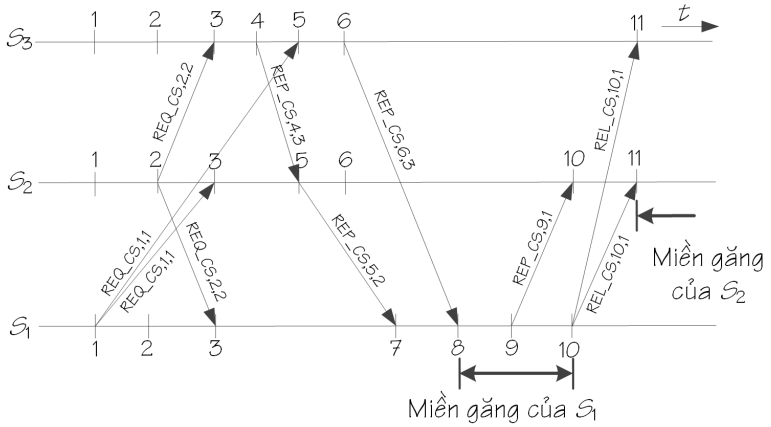
Hình 2. Đồ thị cung cấp tài nguyên.

đòi hỏi tài nguyên phải được cung cấp duy nhất cho một tiến trình tại một thời điểm. Do đó, loại trừ tương hỗ là giải pháp trong các hệ phân tán cung cấp tài nguyên dùng chung [3, 4, 11]. Giải pháp loại trừ tương hỗ được giải quyết dựa trên đồng bộ hóa tiến trình truy cập vào các tài nguyên dùng chung để đảm bảo tính nhất quán và gắn bó trong hệ phân tán. Quá trình đồng bộ hóa bằng cách truyền thông điệp giữa các máy chủ dựa vào môi trường truyền thông. Loại trừ tương hỗ phân tán tuân thủ các yêu cầu sau:

1) Cho phép một tiến trình duy nhất được thực thi trong miền găng tại một thời điểm đối với một tài nguyên;
2) Nếu không có tiến trình nào trong miền găng, tiến trình yêu cầu vào miền găng phải được phép vào và thực thi trong khoảng thời gian cho phép;
3) Khi có nhiều tiến trình yêu cầu vào miền găng, việc cho phép có thể bị trì hoãn đến khi được cấp phép;
4) Tiến trình xử lý trong miền găng trong không bị chặn bởi các tiến trình khác.

Theo thuật toán loại trừ tương hỗ phân tán, Kshemkalyani và Singhal trình bày quá trình một máy chủ vào và ra khỏi miền găng, được mô tả như trong hình 3 [4, Mục 9.3, 9.4]. Thông điệp có cấu trúc và chứa một trong ba giá trị: *REQ-CS* (yêu cầu vào miền găng), *REP-CS* (phản hồi chấp nhận của máy chủ cho phép vào miền găng), và *REL-CS* (giải phóng khỏi miền găng). Một máy chủ được phép vào miền găng khi máy chủ đó tiếp nhận đủ thông điệp *REP-CS* sau thông điệp *REQ-CS*. Máy chủ S_1 phát thông điệp yêu cầu *REQ-CS* vào miền găng tại thời điểm 1. Đến thời điểm 8, S_1 nhận đầy đủ thông điệp phản hồi *REP-CS* chấp nhận và vào miền găng cho đến thời điểm 10 phát thông điệp *REL-CS* rời khỏi miền găng sau khi xử lý xong. Trật tự từng phần trên các máy chủ thể hiện qua bảng I.

Xét truyền thông điệp theo hình 3, nếu một máy chủ bị sự cố trong quá trình truyền thông điệp, ví dụ đối với trường hợp truyền thông điệp *REP-CS* từ máy chủ S_2 đến máy chủ S_1 tại thời điểm 6, S_1 phải chờ đợi vào miền găng với khoảng thời gian không xác định. Ngoài ra, thông điệp trong quá trình truyền có thể bị phân mảnh, thất lạc, nghẽn, v.v. Các vấn đề này dẫn đến hiệu năng của hệ phân tán giảm trong quá trình hợp lực. Cụ thể, [4] trình bày hiệu



Hình 3. Quá trình máy chủ vào và ra miền găng [4, Mục 9.3, 9.4].

Bảng I
HOẠT ĐỘNG DIỄN RA TRÊN CÁC MÁY CHỦ TRONG TRẬT TỰ TỪNG PHẦN

Đồng hồ	Máy chủ 1	Máy chủ 2	Máy chủ 3
1	$S_1 \rightarrow S_2$: REQ-CS,1,1 $S_1 \rightarrow S_3$: REQ-CS,1,1		
2		$S_2 \rightarrow S_1$: REQ-CS,2,2 $S_2 \rightarrow S_3$: REQ-CS,2,2	
3	$S_2 \rightarrow S_1$: REQ-CS,2,2	$S_1 \rightarrow S_2$: REQ-CS,1,1	$S_2 \rightarrow S_3$: REQ-CS,2,2
4			$S_3 \rightarrow S_2$: REP-CS,4,3
5		$S_3 \rightarrow S_2$: REP-CS,4,3	$S_1 \rightarrow S_3$: REQ-CS,1,1
6			$S_3 \rightarrow S_1$: REP-CS,6,3
7	$S_2 \rightarrow S_1$: REP-CS,5,2		
8	**Máy chủ S_1 vào miền găng**		
9	$S_1 \rightarrow S_2$: REP-CS,9,1		
10	$S_1 \rightarrow S_2$: REL-CS,10,1 $S_1 \rightarrow S_3$: REL-CS,10,1	$S_1 \rightarrow S_2$: REP-CS,9,1	
11	**Máy chủ S_2 vào miền găng**		

năng loại trừ tương hỗ được xác định dựa trên các tham số: độ phức tạp thông điệp, độ trễ quá trình đồng bộ hóa, thời gian hồi đáp và thông lượng hệ thống. Bên cạnh đó, hiệu năng loại trừ tương hỗ dựa trên điều kiện của tải trong hệ thống. Trong đó tải được xác định bởi tỷ lệ thông điệp đến yêu cầu thực thi miền găng. Đối với tải thấp, số lượng tiến trình phải đợi chờ để vào miền găng là rất thấp. Đối với tải cao, luôn có tiến trình yêu cầu thực thi miền găng phải chờ trong hàng đợi.

Có hai nhóm giải pháp chính trong loại trừ tương hỗ. Nhóm thứ nhất là phương pháp tiếp cận dựa trên token: Ricart-Agrawala [12], Suzuki-Kasami [13], Mizuno-Neilsen-Rao [14], Neilsen-Mizuno [14], Helary-Plouzeau-Raynal [15], Raymond [16], Singhal [17], Naimi-Trehel [18], Mishra-Srimani [19], và Nishio [20]. Nhóm thứ hai là phương pháp tiếp cận dựa trên quyền: Lamport [21], Ricart-Agrawala [22], Carvalho-Roucairol [23], Raynal [24], Maekawa [25], Sanders [26], Agrawal-El Abbadi [27], và Singhal [28]. Hiệu năng của các thuật toán trong nhóm thứ hai, tính theo số thông điệp cần được truyền, được khảo sát bởi Velazquez [29] và tóm tắt trong bảng II.

Một so sánh khác được trình bày bởi Yadav và cộng sự trong [30]. Trong đó, hai giải pháp tiếp cận cho loại trừ tương hỗ phân tán là giải pháp dựa trên nội dung và giải pháp dựa trên điều khiển. Giải pháp dựa trên nội dung sử dụng thuật toán trật tự nhãn thời gian lô-gic và thuật toán bầu chọn. Giải pháp dựa trên điều khiển sử dụng cấu trúc cây, cấu trúc truyền broadcast và cấu trúc mạng vòng [31]. Yadav và cộng sự phân tích, so sánh hiệu năng của các thuật toán loại trừ tương hỗ phân tán. Kết quả được thể hiện trong bảng III.

Bảng II
HIỆU NĂNG CỦA THUẬT TOÁN DỰA TRÊN QUYỀN [29]

Thuật toán	Tổng số thông điệp
Lamport [21]	$3(N-1)$
Ricart-Agrawala [22]	$2(N-1)$
Carvalho-Roucairol [23]	0 đến $2(N-1)$
Raynal [24]	$2(N-1)^2$
Maekawa [25]	$3\sqrt{N}$ đến $5\sqrt{N}$
Sanders [26]	$\|Ii-\{i\}\|+2\|Ri-\{i\}\|$
Agrawal-El Abbadi [27]	$O(\log N)$
Singhal [28]	$(N-1)$ đến $3(N-1)/2$

Bảng III
PHÂN TÍCH, SO SÁNH HIỆU NĂNG CỦA CÁC THUẬT TOÁN LOẠI TRỪ TƯƠNG HỖ [30]

Thuật toán	Thời gian hồi đáp	Độ trễ đồng bộ	Th. điệp tải thấp	Th. điệp tải cao
[21]	$2T+E$	T	$3(N-1)$	$3(N-1)$
[22]	$2T+E$	T	$2(N-1)$	$2(N-1)$
[13]	$2T+E$	T	N	N
[16]	$T\log N+E$	$(T\log N)/2$	$\log N$	4

Hướng nghiên cứu của bài báo là song song hóa thuật toán Lamport trong loại trừ tương hỗ phân tán để giảm độ phức tạp thông điệp, thời gian hồi đáp và độ trễ đồng bộ. Cụ thể là đồng bộ hóa các tiến trình dựa trên song song hóa thuật toán Lamport đạt được trật tự tổng quát chặt chẽ với độ phức tạp thông điệp $3(N-1)$. Sau đó, các máy chủ hợp lực để vào miền găng thời điểm sớm hơn trong thuật toán loại trừ tương hỗ phân tán với thời gian hồi đáp là $N-1+E$, độ phức tạp thông điệp là $N-1$ và độ trễ đồng bộ $T=0$.

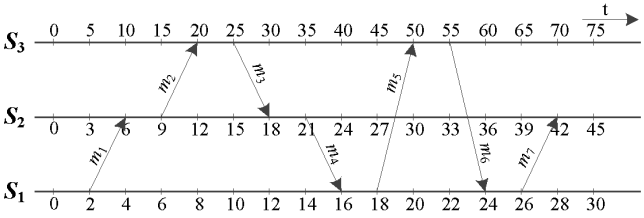
III. SONG SONG HÓA THUẬT TOÁN LAMPORT

1. Trật tự tổng quát chặt chẽ dựa trên song song hóa thuật toán Lamport

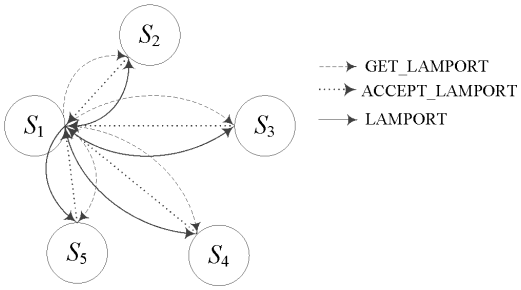
Nhãn thời gian lô-gíc được xây dựng dựa trên thuật toán Lamport trình bày trong [32]. Thuật toán Lamport cho phép ghi lại các sự kiện của hệ phân tán. Thuật toán tập trung vào nguyên lý sau: mỗi máy chủ S đều có trang bị công tơ với các giá trị nguyên gọi là H_{S_i}. Đó chính là đồng hồ lô-gíc tăng lên giữa hai sự kiện kế tiếp. Máy chủ e phát thông điệp ghi dấu của mình dựa trên giá trị hiện hành của H_{S_e}. Khi nhận được thông điệp, máy chủ nhận r cập nhật đồng hồ H_{S_r} riêng của mình bằng giải thuật rút gọn [32]:

$$\textbf{If } H_{S_r} \le E \textbf{ then}$$
$$H_{S_r} := E+1; \qquad (1)$$
$$\textbf{EndIf}.$$

Một sự kiện a (sk_a) sinh ra trong máy chủ i (S_i) và được đánh dấu bởi đồng hồ cục bộ gọi là $H_{S_i}(a)$. Nếu sk_a và



Hình 4. Nhãn thời gian của thông điệp không theo trật tự.



Hình 5. Hoạt động cấp giá trị đồng hồ lô-gic theo thuật toán 1.

sk_b là hai sự kiện được gửi từ cùng máy chủ S_i đến S_j, ta luôn luôn có quan hệ xác định như sau:

$$sk_a \rightarrow sk_b \Leftrightarrow H_{S_i}(a) < H_{S_j}(b), \qquad (2)$$

trong đó $sk_a \rightarrow sk_b$ thể hiện sự kiện a gửi cho sự kiện b, $H_{S_i}(a) < H_{S_j}(b)$ thể hiện giá trị đồng hồ máy chủ a nhỏ hơn giá trị đồng hồ cục bộ máy chủ b, ký hiệu $\rightarrow$ biểu thị phép kéo theo và ký hiệu $\Leftrightarrow$ biểu thị phép tương đương.

Tuy nhiên, các nhãn đồng hồ này phải được cập nhật và nhất quán trên tất cả các máy chủ. Nếu giá trị không được cập nhật thì việc xử lý thông điệp sẽ bị sai và hoạt động của hệ sai trật tự theo lý thuyết trật tự như hình 4.

Khi các thông điệp di chuyển qua các máy chủ, giá trị gửi và nhận có giá trị khác nhau. Chính vì giá trị đồng hồ sai lệch, khi một máy chủ phát lệnh xử lý đồng thời trên các máy chủ sẽ dẫn đến sai lệch về các tiến trình được triệu gọi để xử lý. Do đó, dữ liệu không nhất quán trên tất cả các máy chủ.

Các máy chủ hoạt động nhận và gửi thông điệp dựa trên đồng hồ cục bộ của mình theo cơ chế truyền unicast. Do đó, các máy chủ chỉ biết được *trật tự từng phần* trên máy chủ của mình và không nhận biết được các hoạt động trên máy chủ khác. *Trật tự từng phần* ảnh hưởng đến hoạt động tổng quát trong hệ phân tán. Hai vấn đề cơ bản là: (i) giá trị đồng hồ lô-gic trên các máy chủ không nhất quán; (ii) tiến trình yêu cầu vào đoạn găng phải chờ đợi cho đến khi nhận đủ thông điệp có thể gây ảnh hưởng đến các máy chủ khác hoặc sai lệch khi tiến hành cập nhật dữ liệu. Để giải quyết bài toán trật tự từng phần, nghiên cứu của bài báo là song song hóa thuật toán Lamport để xây dựng *trật tự tổng quát chặt chẽ* trên các máy chủ theo thuật toán 1.

Thuật toán 1: Song song hóa thuật toán Lamport

Dữ liệu vào:
- Máy chủ S_i;
- Giá trị đồng hồ lô-gic H_{S_i};
- Hành động *act*; và
- Sự kiện *sk*.

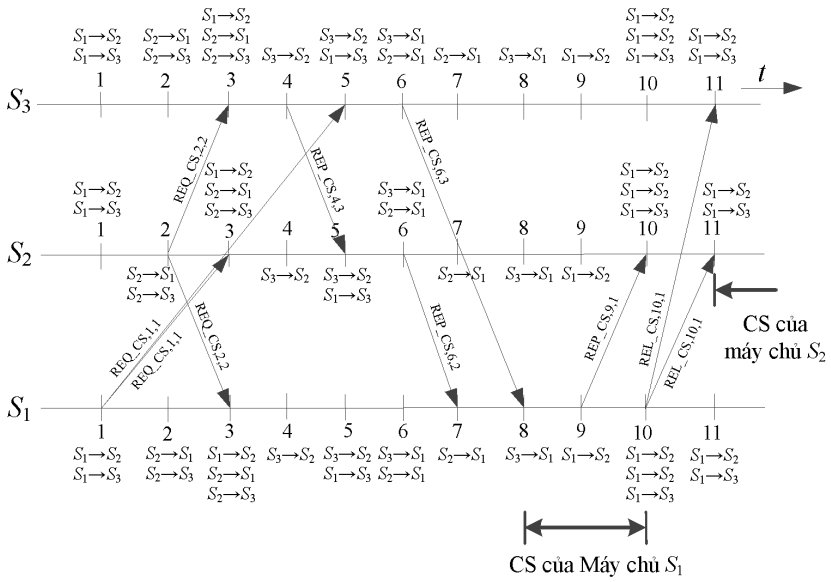
Dữ liệu ra: Giá trị đồng hồ lô-gic H_{S_i} đã cấp.

```
1  Khởi tạo hoạt động H_{S_local} = 0;
2  Biến đếm count = 0;
3  countCS[S_i, sk] = 0;
4  Số lượng máy chủ S = N;
5  Lắng nghe sự kiện act;
6  if act = REQ-LAMPORT then
7  |   H_{S_i} = H_{S_local} + 1;
8  |   act = REQ;
9  |   Thiết lập thông điệp yêu cầu cung cấp giá trị đồng hồ lô-
   |     gic: GetLamport(S_local, H_{S_local}, act, sk);
10 |   return multicast(GetLamport(S_i, H_{S_i}, act, sk));
11 else if act = LAMPORT then
12 |   return H_{S_i};
13 end
14 GetLamport (S_i, H_{S_i}, act, sk)
15 if act = REQ then
16 |   if H_{S_i} ≤ H_{S_local} then
17 |   |   Xác định H_{S_i} bị sai, H_{S_i} đã được gán cho sự kiện
   |   |     khác;
18 |   |   act = REP;
19 |   |   return multicast(AcceptLamport(S_local, S_i, H_{S_i}, act,
   |   |     sk, false));
20 |   else if H_{S_i} = H_{S_local} + 1 then
21 |   |   Xác định H_{S_i} đúng;
22 |   |   act = REP;
23 |   |   return multicast(AcceptLamport(S_local,
   |   |     S_i, H_{S_i}, act, sk, true));
24 |   end
25 end
26 AcceptLamport (S_local, S_i, H_{S_i}, act, sk, boolean)
27 if act = REP then
28 |   if S_i = S_local&&true then
29 |   |   count = count + 1;
30 |   |   if count = N − 1 then
31 |   |   |   Xác nhận đủ số lượng thông điệp phản hồi REP;
32 |   |   |   act = ACC;
33 |   |   |   count = 0;
34 |   |   |   return multicast(UpdateLamport(S_local,
   |   |   |     H_{S_local}, act, sk));
35 |   |   end
36 |   else if S_i = S_local&&false then
37 |   |   H_{S_local} = max(H_{S_i} + 1);
38 |   |   act = REQ;
39 |   |   Thiết lập lại thông điệp yêu cầu cung cấp với giá trị
   |   |     đồng hồ lô-gic mới với sự kiện đang yêu cầu;
40 |   |   GetLamport(S_local, H_{S_local}, act, sk);
41 |   end
42 end
43 UpdateLamport (S_i, H_{S_i}, act, sk)
44 if act = ACC then
45 |   H_{S_local} = H_{S_i};
46 end
47 if sk = REP-CS then
48 |   count[S_i, sk] = count[S_i, sk] + 1;
49 |   if count[S_i, sk] = N − 1 then
50 |   |   CriticalSection(sk, S_i);
51 |   |   count[S_i, sk] = 0;
52 |   end
53 end
54 multicast ((S_local, H_{S_local}, act, sk))
55 S_n = tập tất cả máy chủ ngoại trừ S_local;
56 for j = 1 to S_n do
57 |   kn = connect(IP(S_n), port(S_n));
58 |   if kn then
59 |   |   sendUDP(GetLamport((S_local, H_{S_local}, act, sk),
   |   |     IP(S_n), port(S_n));
60 |   else if kn then
61 |   |   log-err(kn);
62 |   end
63 end
64 return H_{S_i};
```

Để đạt được trật tự tổng quát chặt chẽ, thuật toán Lamport được song song hóa nhằm đồng bộ hóa các tiến trình di chuyển trong hệ phân tán. Mỗi sự kiện diễn ra trên bất kỳ máy chủ nào đều phải yêu cầu giá trị đồng hồ và giá trị này được nhận biết và nhất quán trên tất cả các máy chủ.

Hình 5 mô tả hoạt động theo thuật toán 1, máy chủ S_1 yêu cầu cung cấp giá trị đồng hồ lô-gic, thông điệp mang giá trị *GetLamport* được truyền multicast đến các máy chủ trong hệ thống (dòng 10). Các thủ tục trong thuật toán 1 được giải thích như sau. Thủ tục *multicast* (dòng 54 đến 64) là quá trình xử lý truyền song song các thông điệp đến tập máy chủ trong hệ thống. Thủ tục *GetLamport* (dòng 14 đến 25) thực hiện tính toán và xử lý giá trị đồng hồ so với máy chủ cục bộ nhận được thông điệp, nếu giá trị đồng hồ đúng trả về giá trị *true* (dòng 23), nếu sai trả về giá trị *false* (dòng 19). Thủ tục *AcceptLamport* (dòng 26 đến 42) cho phép giá trị đồng hồ được xác lập dựa trên giá trị true của tất cả các máy chủ. Ngược lại nếu một máy chủ bất kỳ trả về giá trị *false*, thủ tục này xác lập lại giá trị đồng hồ lô-gic dựa trên lấy giá trị cực đại và phát đi thông điệp *GetLamport* (dòng 37). Thủ tục *UpdateLamport* (dòng 43 đến 53) nhằm khẳng định cho máy chủ có sự kiện được yêu cầu gán giá trị đồng hồ lô-gic và giá trị này cập nhật trên tất cả các máy chủ.

Hình 6 mô tả kết quả thực hiện song song hóa thuật toán Lamport so với thuật toán nguyên thủy của Lamport theo hình 3. Theo thuật toán nguyên thủy của Lamport, giá trị đồng hồ lô-gic sinh ra khi nó lấy giá trị cực đại của thông điệp nhận và tăng lên mỗi khi có sự kiện mới phát sinh bên trong máy chủ. Thuật toán này tuân thủ luật *happened-*



Hình 6. Hoạt động song song hóa Thuật toán Lamport.

before. Tuy nhiên, đây là giải pháp thụ động. Thông qua hình 6, mỗi một giá trị đồng hồ lô-gic được gán cho sự kiện đều gửi đến tất cả các trạm. Vì vậy, mặc dù không có hoạt động trên trạm nhưng trạm nhận biết được hoạt động trên các trạm còn lại. Đây là giải pháp chủ động trong việc giám sát và điều khiển sự kiện. Giải pháp này tạo điều kiện thuận lợi cho các sự kiện yêu cầu vào miền găng để giải quyết loại trừ tương hỗ phân tán.

2. Áp dụng song song hóa thuật toán Lamport để giải quyết loại trừ tương hỗ phân tán

Mô hình hệ phân tán theo hình 1 được mô tả là hệ thống bao gồm N máy chủ S_i $(i = 1, \ldots, N)$ kết nối với nhau qua môi trường truyền thông. Một tiến trình p_j $(j = 1, \ldots, M)$ thực thi trên một máy chủ S_i.

Nghiên cứu trong [33] trình bày giải pháp gắn bó dựa trên thuật toán 4PCoDT. Giải pháp thực hiện cơ chế truyền thông điệp trong vòng tròn ảo, mỗi pha di chuyển qua từng máy chủ theo vòng tròn định trước. Pha thứ nhất thực hiện xử lý tiến trình yêu cầu đi vào miền găng để tránh tương tranh tài nguyên. Yêu cầu thông điệp trong tiến trình thực thi miền găng trên máy chủ có chứa một trong ba giá trị: *REQ-CS*, *REP-CS* và *REL-CS*. Khi tiến trình yêu cầu *REQ-CS* đã nhận đầy đủ phản hồi *REP-CS* mới được phép vào miền găng. Tiến trình vào miền găng tuân thủ nguyên tắc loại trừ tương hỗ phân tán trình bày trong phần II. Sau khi tiến trình xử lý xong và rời khỏi miền găng, máy chủ phát đi thông điệp có chứa giá trị *REL-CS* để tiến trình khác được phép vào miền găng.

Trên cơ sở song song hóa thuật toán Lamport trình bày trong mục III.1, thuật toán loại trừ tương hỗ phân tán thực hiện dựa trên cải tiến thuật toán Lamport được trình bày trong thuật toán 2. Hàm *RequestCriticalSection* (dòng 10 đến 24) thể hiện các tiến trình yêu cầu vào miền găng. Hàm *RequestCriticalSection* xác định giá trị đồng hồ lô-gic đã gán cho tiến trình, nếu tiến trình nào có giá trị nhỏ hơn có quyền ưu tiên cao hơn để vào miền găng. Hàm *CriticalSection* (dòng 25 đến 33) thực hiện xử lý tiến trình trong miền găng. Quá trình xử lý tiến trình đảm bảo các trường nội dung tác động lên cơ sở dữ liệu là duy nhất trên tập máy chủ, đảm bảo tính gắn bó trong hệ phân tán. Sau khi kết thúc quá trình xử lý trong miền găng lập tức yêu cầu xóa tiến trình khỏi hàng đợi, phát thông điệp giải phóng miền găng để tiến trình tiếp theo được phép vòa miền găng. Hàm *NextCriticalSection* (dòng 34 đến 38) xử lý tiến trình tiếp theo trong hàng đợi. Nếu tiến trình tiếp theo này đã tiếp nhận đầy đủ phản hồi theo thuật toán 1 thì lập tức vào miền găng mà không cần phải đợi bất kỳ xác nhận nào.

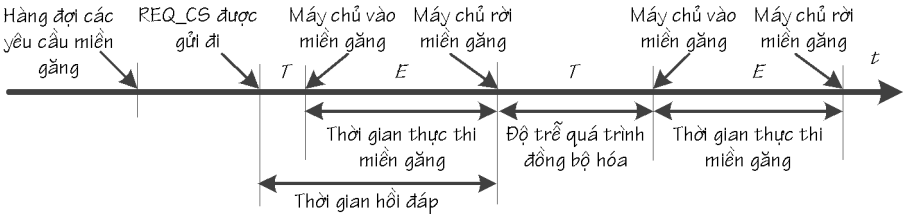
Theo thuật toán Lamport nguyên thủy trong bảng I, máy chủ S_1 vào miền găng tại thời điểm 8 khi đã nhận đủ thông điệp phản hồi *REP-CS*. Sau khi áp dụng song song hóa, thuật toán Lamport đạt được một trật tự tổng quát chặt chẽ trên các máy chủ theo bảng IV. Kết quả bảng IV cho thấy máy chủ vào miền găng thời điểm 6, sớm hơn so với trật tự từng phần ở bảng I.

IV. HIỆU NĂNG THỰC THI SONG SONG HÓA THUẬT TOÁN LAMPORT

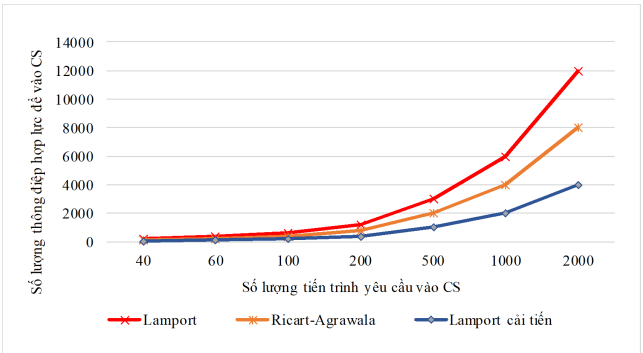
Dựa vào mô tả các tiến trình hoạt động trong miền găng và các tham số trình bày trong phần I, chúng tôi đánh giá hiệu năng loại trừ tương hỗ phân tán theo Hình 7. Tham số độ phức tạp thông điệp của song song hóa thuật toán

Bảng IV
HOẠT ĐỘNG DIỄN RA TRÊN CÁC MÁY CHỦ TRONG TRẬT TỰ TỔNG QUÁT CHẶT CHẼ

Đồng hồ	**Máy chủ 1**	**Máy chủ 2**	**Máy chủ 3**
1	$S_1 \rightarrow S_2$: REQ-CS,1,1 $S_1 \rightarrow S_3$: REQ-CS,1,1	$S_1 \rightarrow S_2$: REQ-CS,1,1 $S_1 \rightarrow S_3$: REQ-CS,1,1	$S_1 \rightarrow S_2$: REQ-CS,1,1 $S_1 \rightarrow S_3$: REQ-CS,1,1
2	$S_2 \rightarrow S_1$: REQ-CS,2,2 $S_2 \rightarrow S_3$: REQ-CS,2,2	$S_2 \rightarrow S_1$: REQ-CS,2,2 $S_2 \rightarrow S_3$: REQ-CS,2,2	$S_2 \rightarrow S_1$: REQ-CS,2,2 $S_2 \rightarrow S_3$: REQ-CS,2,2
3	$S_2 \rightarrow S_1$: REQ-CS,2,2 $S_1 \rightarrow S_2$: REQ-CS,1,1 $S_2 \rightarrow S_3$: REQ-CS,2,2	$S_1 \rightarrow S_2$: REQ-CS,1,1 $S_2 \rightarrow S_3$: REQ-CS,2,2 $S_2 \rightarrow S_1$: REQ-CS,2,2	$S_2 \rightarrow S_3$: REQ-CS,2,2 $S_2 \rightarrow S_1$: REQ-CS,2,2 $S_1 \rightarrow S_2$: REQ-CS,1,1
4	$S_3 \rightarrow S_2$: REP-CS,4,3	$S_3 \rightarrow S_2$: REP-CS,4,3	$S_3 \rightarrow S_2$: REP-CS,4,3
5	$S_3 \rightarrow S_2$: REP-CS,4,3 $S_1 \rightarrow S_3$: REQ-CS,1,1	$S_3 \rightarrow S_2$: REP-CS,4,3 $S_1 \rightarrow S_3$: REQ-CS,1,1	$S_1 \rightarrow S_3$: REQ-CS,1,1 $S_3 \rightarrow S_2$: REP-CS,4,3
6	$S_3 \rightarrow S_1$: REP-CS,6,3	$S_3 \rightarrow S_1$: REP-CS,6,3	$S_3 \rightarrow S_1$: REP-CS,6,3
	S_1 vào miền găng		
7	$S_1 \rightarrow S_2$: REP-CS,7,1	$S_1 \rightarrow S_2$: REP-CS,7,1	$S_1 \rightarrow S_2$: REP-CS,7,1
	S_2 đã nhận đủ REP-CS, chờ lượt tiếp theo vào miền găng		
8	$S_1 \rightarrow S_2$: REL-CS,8,1 $S_1 \rightarrow S_3$: REL-CS,8,1 $S_1 \rightarrow S_2$: REP-CS,7,1	$S_1 \rightarrow S_2$: REP-CS,7,1 $S_1 \rightarrow S_2$: REL-CS,8,1 $S_1 \rightarrow S_3$: REL-CS,8,1	$S_1 \rightarrow S_2$: REL-CS,8,1 $S_1 \rightarrow S_3$: REL-CS,8,1 $S_1 \rightarrow S_2$: REP-CS,7,1
	S_2 vào miền găng		
9	$S_1 \rightarrow S_2$: REL-CS,8,1 $S_1 \rightarrow S_3$: REL-CS,8,1	$S_1 \rightarrow S_2$: REL-CS,8,1 $S_1 \rightarrow S_3$: REL-CS,8,1	$S_1 \rightarrow S_2$: REL-CS,8,1 $S_1 \rightarrow S_3$: REL-CS,8,1



Hình 7. Mô tả các tiến trình hoạt động trong miền găng.



Hình 8. Số lượng thông điệp hợp lực đáp ứng các tiến trình để vào miền găng.

Lamport được xác định dựa trên số lượng thông điệp yêu cầu trên một máy chủ cho mỗi thực thi miền găng. Đối với song song hóa thuật toán Lamport yêu cầu $N-1$ thông điệp *REQ*, $N-1$ thông điệp *REP*, $N-1$ thông điệp *ACC*, do đó, thuật toán yêu cầu $3(N-1)$ thông điệp. Đối với thuật toán cải tiến loại trừ tương hỗ của bài báo yêu cầu $N-1$ thông điệp *REQ-CS* và không xét thông điệp *REP-CS* và *REL-CS* trong quá trình nhận, do đó, thuật toán yêu cầu $N-1$ thông điệp khi vào miền găng.

Nguyên nhân độ phức tạp thông điệp thuật toán loại trừ tương hỗ thấp là khi áp dụng song song hóa thuật toán Lamport, các thông điệp *REP-CS* và *REL-CS* đã được nhận biết và đánh dấu khi yêu cầu giá trị đồng hồ lô-gic trên các máy chủ. Do đó, máy chủ yêu cầu vào miền găng không

cần phải đợi tiếp nhận đủ thông điệp *REP-CS* và *REL-CS*. Ngoài ra, trong quá trình truyền thông điệp trong hệ thống, thông điệp *REP-CS* và *REL-CS* bị phân mảnh hoặc thất lạc không ảnh hưởng đến quá trình vào miền găng của máy chủ. Theo kết quả hình 8, đối với số lượng tiến trình yêu cầu vào miền găng lớn thì số lượng thông điệp hợp lực để tiến trình vào miền găng càng lớn. Do đó, nếu giảm được số lượng thông điệp hợp lực thì hiệu năng tiến trình vào miền găng tăng.

Tham số độ trễ quá trình đồng bộ hóa T được xác định dựa trên khoảng thời gian yêu cầu sau khi máy chủ bắt đầu phát thông điệp yêu cầu *REQ-CS* cho đến khi vào miền găng hoặc máy chủ rời miền găng để nhường cho máy chủ tiếp theo vào miền găng. Trong trường hợp tải cao, nghĩa là các máy chủ yêu cầu vào miền găng đã nhận đủ thông điệp phản hồi theo song song hóa thuật toán Lamport (dòng 52 của thuật toán 1) và trên hàng đợi của các máy chủ luôn có tiến trình sẵn sàng vào miền găng. Máy chủ tiếp theo được thực hiện tức thì trong miền găng trong trường hợp tải cao sau khi máy chủ trước vừa rời khỏi miền găng, tham số độ trễ quá trình đồng bộ hóa được xác định là $T = 0$.

Thuật toán 2: Cải tiến thuật toán loại trừ tương hỗ Lamport

Dữ liệu vào: Tiến trình *tt(start,jeton, lamport$_1$,lamport$_2$, S_act,type,action,circle,content)*
Dữ liệu ra: Trật tự tổng quát chặt chẽ tiến trình, tiến trình vào miền găng và loại trừ tương hỗ nhờ dấu

1 *action* = *tt.action*;
2 **if** *action* = 1 **then**
3 | *sk* = *REQ-CS*;
4 | *act* = *REQ-LAMPORT*;
5 | $H_{S_{local}}$ = *LAMPORT*(S_{local}, *act*, *sk*);
6 | *tt(start,jeton,*$H_{S_{local}}$*,*$lamport_2$*,*S_{local}*,type,action, circle,content)*;
7 | *req_queue(tt)*;
8 | *multicast*(*RequestCriticalSection*(*sk*,*tt*));
9 **end**
10 `RequestCriticalSection(`*sk*, *tt*`)`
11 $H_{S_{local}}$ = *tt.*$lamport_1$;
12 S_i = *tt.*S_{act};
13 **if** *sk* = *REQ-CS*, $H_{S_i} < H_{S_{local}}$ **then**
14 | *sk* = *REP-CS*;
15 | *act* = *REQ-LAMPORT*;
16 | $H_{S_{local}}$ = *LAMPORT*(S_{local}, *act*, *sk*);
17 | *tt(start,jeton,*H_{S_i}*,*$H_{S_{local}}$*,*S_i*,type,action,circle,content)*;
18 **else if** *sk* = *REP-CS*, $H_{S_i} < H_{S_{local}}$ **then**
19 | *sk* = *REP-CS*;
20 | *act* = *REQ-LAMPORT*;
21 | $H_{S_{local}}$ = *LAMPORT*(S_{local}, *act*, *sk*);
22 | *writelog*(*tt*(*start*, *jeton*, H_{S_i}, $H_{S_{local}}$, S_{local}, *type*, *action*, *circle*, *content*));
23 **end**
24 **return** *RequestCriticalSection*(*sk*,*tt*);
25 `CriticalSection(`*sk*, *tt*`)`
26 *content* = *tt.content*;
27 *process*(*content*);
28 *remove_req_queue(tt)*;
29 *sk* = *REL-CS*;
30 *act* = *REQ-LAMPORT*;
31 $H_{S_{local}}$ = *LAMPORT*(S_{local}, *act*, *sk*);
32 *tt*(*start*, *jeton*, *lamport*$_1$, $H_{S_{local}}$, S_i, *type*, *action*, *circle*, *content*);
33 **return** *multicast*(*NextCriticalSection*());
34 `NextCriticalSection()`
35 **if** *req_queue* $\neq \emptyset$ **then**
36 | *pop_req_queue*(*tt*);
37 | **return** *CriticalSection*(*sk*, *tt*);
38 **end**

Tham số thời gian hồi đáp H được xác định là khoảng thời gian từ khi gửi yêu cầu vào miền găng cho đến khi ra khỏi miền găng. Theo mô tả trong hình 7, H được tính bằng công thức sau [4]:

$$H = T + E, \tag{3}$$

trong đó T là độ trễ quá trình đồng bộ hóa và E là thời gian thực thi miền găng. Đối với trường hợp áp dụng song song hóa thuật toán Lamport, máy chủ được phép vào miền găng ngay tại thời điểm máy chủ cuối cùng bắt đầu phản hồi thông điệp *REP-CS* sau thông điệp *REQ-CS*. Như vậy, thời gian hồi đáp được xác định như sau. Đối với trường hợp chờ tiếp nhận máy chủ cuối cùng bắt đầu phát thông điệp *REP-CS*, chúng ta có $T = N - 1$, do đó $H = N - 1 + E$. Đối với trường hợp đã tiếp nhận đủ thông điệp *REP-CS* và đang nằm trong hàng đợi vào miền găng trong lượt tiếp theo, chúng ta có $T = 0$, do đó $H = E$.

Tham số thông lượng hệ thống ký hiệu là A được xác định dựa trên tỷ lệ mà hệ thống thực thi các yêu cầu trong miền găng. A được tính bằng công thức [4]:

$$A = \frac{1}{H}, \tag{4}$$

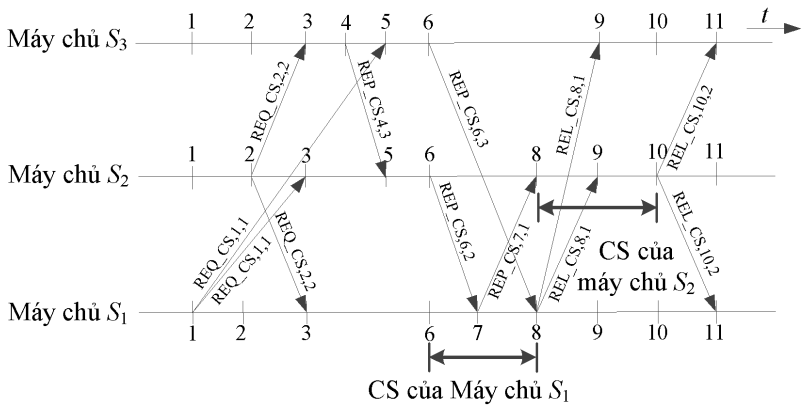
trong đó H là thời gian hồi đáp. Đối với trường hợp tải thấp $T = N - 1$ thì $A = 1/(N-1) + E$. Đối với trường hợp tải cao $T = 0$ thì $A = 1/E$.

Ký hiệu $\overline{X}$ là số lượng trung bình thông điệp vào miền găng trên các máy chủ. Đối với trường hợp tải cao, trên mỗi máy chủ đều có ít nhất một tiến trình yêu cầu vào miền găng. Một máy chủ cần phải phát $N - 1$ thông điệp để yêu cầu vào miền găng. Vì vậy, $\overline{X}$ được xác định như sau:

$$\overline{X} = \frac{N-1}{N} + \frac{N + (N-1)}{N} = \frac{3N-2}{N}. \tag{5}$$

Theo công thức (5), nếu số lượng các các chủ lớn ($N \to \infty$), số lượng trung bình thông điệp trên các máy chủ xấp xỉ 3. Trong khi đó thông số này cho thuật toán Lamport nguyên thủy là xấp xỉ 7 và cho thuật toán Ricart–Agrawala là xấp xỉ 5 với số lượng máy chủ tương tự.

Đối với việc áp dụng song song hóa thuật toán Lamport trong thực thi miền găng, thời gian hồi đáp của tiến trình yêu cầu miền găng và độ trễ quá trình đồng bộ hóa được rút ngắn. Kết quả so sánh các thuật toán thể hiện trong



Hình 9. Hiệu năng thực thi song song hóa thuật toán.

Bảng V
SO SÁNH HIỆU NĂNG CỦA THUẬT TOÁN LAMPORT CẢI TIẾN TRONG LOẠI TRỪ TƯƠNG HỖ PHÂN TÁN

Thuật toán	Thời gian hồi đáp	Độ trễ đồng bộ	Th. điệp tải thấp	Th. điệp tải cao
[21]	$2T + E$	T	$3(N-1)$	$3(N-1)$
[22]	$2T + E$	T	$2(N-1)$	$2(N-1)$
Cải tiến	$T + E$	T	$N-1$	$N-1$

bảng V. Thuật toán Lamport cải tiến khi áp dụng song song hóa thuật toán Lamport đạt được hiệu năng loại trừ tương hỗ cao so với thuật toán Lamport và Ricart-Agrawala.

Theo kết quả thực hiện trong hình 9, áp dụng song song hóa thuật toán Lamport, tiến trình trên máy chủ S_1 vào miền găng tại thời điểm 6 và trên máy chủ S_2 vào miền găng tại thời điểm 8.

Tiến trình yêu cầu miền găng trên máy chủ S_1 mô tả cho trường hợp tải thấp: thời gian hồi đáp $D = 8$ được xác định từ thời điểm giá trị đồng hồ là 1 cho đến lúc phát thông điệp rời khỏi miền găng tại thời điểm 8. So với hình 3, S_1 phải đợi đủ thông điệp phản hồi từ các máy chủ còn lại tại thời điểm 8 mới bắt đầu vào miền găng, do đó S_1 có $D = 10$.

Tiến trình yêu cầu miền găng trên máy chủ S_2 mô tả cho trường hợp tải cao: thời gian hồi đáp $D = 8$ được xác định từ thời điểm giá trị đồng hồ là 2 cho đến lúc phát thông điệp rời khỏi miền găng tại thời điểm 10. Trong trường hợp tải cao, tiến trình trên S_2 đã nhận đủ thông điệp phản hồi và chờ vào miền găng thì tại thời điểm 8 S_2 lập tức vào miền găng. Đối với trường hợp này, S_2 không phải chờ đợi tiếp nhận thông điệp giải phóng khỏi miền găng từ S_1 như mô tả trong hình 3. Bên cạnh đó, theo mô tả trong hình 3 nếu thông điệp giải phóng từ S_1 bị thất lạc hoặc phân mảnh trong quá trình truyền thì S_2 phải chờ đợi cho đến khi tiếp nhận đầy đủ, điều này dẫn đến hiệu năng thực hiện bị giảm.

V. KẾT LUẬN

Trong bài báo này, nghiên cứu giải pháp song song hóa thuật toán Lamport nhằm cải tiến thuật toán loại trừ tương hỗ phân tán. Song song hóa thuật toán Lamport cho phép thiết lập một trật tự tổng quát chặt chẽ và ghi dấu các sự kiện diễn ra trên các máy chủ. Thuật toán cải tiến gán dấu cho sự kiện yêu cầu $3(N-1)$ thông điệp. Khi áp dụng song song hóa thuật toán Lamport trong thuật toán loại trừ tương hỗ, tiến trình đi vào miền găng yêu cầu $N-1$ thông điệp. Do đó, giải pháp cải tiến của bài báo đạt hiệu năng cao trong cải tiến thuật toán loại trừ tương hỗ phân tán thể hiện trong bảng V. Tuy nhiên, để đạt được hiệu năng cao, độ phức tạp thông điệp song song hóa thuật toán Lamport để gắn dấu cho tiến trình yêu cầu $3(N-1)$ thông điệp và độ dài thông điệp lớn hơn so với các thuật toán gắn dấu khác. Do đó, đối với hệ thống tải cao thì song song hóa thuật toán Lamport phải xử lý với số lượng lớn các yêu cầu giá trị đồng hồ lô-gic trong quá trình hợp lực. Vì vậy, giải pháp để giải quyết tối ưu hệ thống trong hợp lực và truyền thông cần được tiếp tục nghiên cứu.

TÀI LIỆU THAM KHẢO

[1] R. Buyya, "Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility," in *2009 Fourth ChinaGrid Annual Conf.*, 2009, pp. xii–xv.
[2] G. F. Coulouris and J. Dollimore, *Distributed systems: Concepts and design*, 4th ed. Addison-Wesley, 2005.
[3] M. Raynal, *Distributed Algorithms for Message-Passing Systems*. Springer, 2013.
[4] A. D. Kshemkalyani and M. Singhal, *Distributed Computing: Principles, Algorithms, and Systems*. Cambridge University Press, 2008.
[5] M. Raynal, A. Schiper, and S. Toueg, "The causal ordering abstraction and a simple way to implement it," *Information Processing Letters*, vol. 39, no. 6, pp. 343–350, 1991.
[6] N. Sharma and A. Parikh, "Deadlock detection and removal in distributed systems," *International Jour. Engineering and Computer Science*, vol. 2, no. 10, pp. 2900–2902, Oct. 2013.
[7] H. H. C. Nguyen, H. V. Dang, N. M. N. Pham, V. S. Le, and T. T. Nguyen, "Deadlock detection for resource

allocation in heterogeneous distributed platforms," in *Recent Advances in Information and Communication Technology 2015*. Springer, 2015, pp. 285–295.
[8] H. H. C. Nguyen, V. S. Le, and T. T. Nguyen, "Algorithmic approach to deadlock detection for resource allocation in heterogeneous platforms," in *International Conference on Smart Computing*, 2014, pp. 97–103.
[9] H. H. C. Nguyen, H. D. Tran, V. T. Doan, and V. T. P. Anh, "Deadlock avoidance for resource allocation model V VM-out-of-N PM," in *Context-Aware Systems and Applications*. Springer, 2017, pp. 172–182.
[10] M. Singhal, "Deadlock detection in distributed systems," *Computer*, vol. 22, no. 11, pp. 37–48, 1989.
[11] K. Erciyes and V. Adve, *Distributed Graph Algorithms for Computer Networks*. Springer, 2013.
[12] G. Ricart and A. K. Agrawala, "Author's response to 'On mutual exclusion in computer networks' by Carvalho and Roucairol," *Commun. of the ACM*, pp. 147–148, 1983.
[13] I. Suzuki and T. Kasami, "A distributed mutual exclusion algorithm," *ACM Transactions on Computer Systems*, vol. 3, no. 4, pp. 344–349, Nov. 1985.
[14] M. Mizuno, M. L. Neilsen, and R. Rao, "A token based distributed mutual exclusion algorithm based on quorum agreements," in *11th International Conference on Distributed Computing Systems*, 1991, pp. 361–368.
[15] J.-M. Helary, N. Plouzeau, and M. Raynal, "A distributed algorithm for mutual exclusion in an arbitrary network," *The Computer Journal*, vol. 31, no. 4, pp. 289–295, 1988.
[16] K. Raymond, "A tree-based algorithm for distributed mutual exclusion," *ACM Transactions on Computer Systems*, vol. 7, no. 1, pp. 61–77, 1989.
[17] M. Singhal, "A heuristically-aided algorithm for mutual exclusion in distributed systems," *IEEE Transactions on Computers*, vol. 38, no. 5, pp. 651–662, 1989.
[18] N. Mohamed and T. Michel, "How to detect a failure and regenerate the token in the Log(n) distributed algorithm for mutual exclusion," in *Distributed Algorithms*. Springer Berlin Heidelberg, 1988, pp. 155–166.
[19] S. Mishra and P. K. Srimani, "Fault-tolerant mutual exclusion algorithms," *Journal of Systems and Software*, vol. 11, no. 2, pp. 111–129, 1990.
[20] S. Nishio, K. F. Li, and E. G. Manning, "A resilient mutual exclusion algorithm for computer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 1, no. 3, pp. 344–356, 1990.
[21] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.
[22] G. Ricart and A. K. Agrawala, "An optimal algorithm for mutual exclusion in computer networks," *Communications of the ACM*, vol. 24, no. 1, pp. 9–17, 1981.
[23] O. Carvalho and G. Roucairol, "On mutual exclusion in computer networks," *Communications of the ACM*, vol. 26, no. 2, pp. 146–147, Feb. 1983.
[24] M. Raynal, "Prime numbers as a tool to design distributed algorithms," *Information Processing Letters*, vol. 33, no. 1, pp. 53–58, 1989.
[25] M. Maekawa, "A Sqrt(N) algorithm for mutual exclusion in decentralized systems," *ACM Transactions on Computer Systems*, vol. 3, no. 2, pp. 145–159, 1985.
[26] B. A. Sanders, "The information structure of distributed mutual exclusion algorithms," *ACM Transactions on Computer Systems*, vol. 5, no. 3, pp. 284–299, 1987.
[27] D. Agrawal and A. E. Abbadi, "An efficient and fault-tolerant solution for distributed mutual exclusion," *ACM Transactions on Computer Systems*, vol. 9, no. 1, pp. 1–20, 1991.
[28] M. Singhal, "A dynamic information-structure mutual exclusion algorithm for distributed systems," in *9th International Conference on Distributed Computing Systems*, 1989, pp. 70–78.
[29] M. G. Velazquez, "A survey of distributed mutual exclusion algorithms," Colorado State University, Tech. Rep., 1993.
[30] N. Yadav, S. Yadav, and S. Mandiratta, "A review of various mutual exclusion algorithms in distributed environment," *International Journal of Computer Applications*, vol. 129, no. 14, pp. 11–16, 2015.
[31] S. Naseera, "A distributed ring algorithm for coordinator election in distributed systems," *ICTACT Journal on Communication Technology*, vol. 7, no. 3, pp. 1341–1344, 2016.
[32] L. V. Sơn, *Hệ tin học phân tán*. Nhà xuất bản Đại học Quốc gia Tp. Hồ Chí Minh, 2002.
[33] D. H. Vĩ and L. V. Sơn, "Cung cấp tài nguyên truyền thông cho hệ phân tán trong máy ảo," *Journal of Science and Technology, The University of Danang*, vol. 11, no. 108, pp. 90–94, 2016.

Đặng Hùng Vĩ sinh năm 1980 tại Đà Nẵng. Tốt nghiệp đại học tại Trường Đại học Bách khoa, Đại học Đà Nẵng năm 2003 chuyên ngành Công nghệ Thông tin. Nhận bằng Thạc sỹ Khoa học Máy tính năm 2008 tại Đại học Đà Nẵng. Hiện đang là nghiên cứu sinh chuyên ngành Khoa học máy tính tại Đại học Đà Nẵng. Lĩnh vực nghiên cứu: Mạng máy tính, mã mạng, hệ phân tán, điện toán đám mây.

Lê Văn Sơn sinh năm 1948 tại Điện Bàn, Quảng Nam. Tốt nghiệp Đại học năm 1978, bảo vệ luận án Tiến sĩ năm 1997 tại Trường Đại học Tổng hợp Donesk, Ukraina. Công nhận Phó Giáo sư năm 2004. Hiện công tác tại Hội tin học Đà Nẵng. Lĩnh vực nghiên cứu: Hệ điều hành, mạng máy tính, hệ phân tán, tính toán đám mây.

Nguyễn Xuân Huy sinh năm 1944 tại Hải Phòng. Bảo vệ luận án Tiến sĩ năm 1978 chuyên ngành Toán-Tin tại Liên Xô. Bảo vệ Tiến sĩ Khoa học năm 1988 thuộc chuyên ngành Công nghệ Thông tin tại Liên Xô. Công nhận Phó Giáo sư năm 1992. Nguyên Chủ tịch Hội đồng Tư vấn Giáo dục Microsoft Việt Nam, Nghiên cứu viên cao cấp Viện Công nghệ thông tin, Viện Khoa học và Công nghệ Việt Nam. Lĩnh vực nghiên cứu: Công nghệ phần mềm, cơ sở dữ liệu.