Giải Thuật Phòng Tránh Tình Trạng Quá Tải Trong Điện Toán Đám Mây

Nguyễn Xuân Phi, Trần Công Hùng
Học Viện Công Nghệ Bưu Chính Viễn Thông
Email: nguyenxuanphi@gmail.com; conghung@ptithcm.edu.vn

Abstract - **Trong bài báo này, chúng tôi đề xuất một thuật toán ngăn chặn tình trạng quá tải cho các máy chủ ảo trong điện toán đám mây thông qua cơ chế phân phối tải thích hợp. Đặc trưng của điện toán đám mây là tính toán phân tán và sử dụng công nghệ ảo hóa nên hiện tượng quá tải có thể xảy ra. Do đó, phân phối tải là vấn đề rất quan trọng để tránh các hiện tượng quá tải, qua đó nâng cao chất lượng dịch vụ, giảm thiểu chi phí để nâng cấp hạ tầng phần cứng công nghệ thông tin, nâng cao hiệu suất của hệ thống và tăng doanh thu của các nhà cung cấp dịch vụ. Ý tưởng chính của thuật toán là tích hợp hai hàng đợi cho các yêu cầu đến và mức độ sử dụng của các máy chủ phục vụ cho bộ cân bằng tải (Load Balancer) điều phối nguồn lực phục vụ khi có yêu cầu đến. Các hàng đợi này được giải thuật sắp xếp trước khi bộ cân bằng tải sử dụng các thông tin trong đó để ra quyết định chọn máy chủ tương ứng. Kết quả mô phỏng của thuật toán chứng mình rằng giải thuật đã làm giảm được thời gian đáp ứng của hệ thống, qua đó phòng tránh được hiện tượng quá tải cho điện toán đám mây.**

Keywords: thuật toán, phân phối tải, ngăn chặn tình trạng quá tải, cân bằng tải, điện toán đám mây.

I. GIỚI THIỆU

Ngày nay, công nghệ điện toán đám mây mang lại những lợi ích to lớn cho nhân loại, đặc biệt là các doanh nghiệp, tổ chức trong lĩnh vực công nghệ thông tin. Công nghệ này cho phép người dùng truy cập đến nguồn tài nguyên mạng được phân tán khắp nơi, đặc biệt là truy cập tài nguyên theo yêu cầu với tính sẵn sàng cao. Chính vì vậy, việc quản lý tài nguyên mạng trong môi trường này trở thành một nhiệm vụ quan trọng đối với các nhà cung cấp dịch vụ điện toán đám mây. Hơn nữa, số lượng các máy chủ ảo trong các trung tâm dữ liệu của nhà cung cấp dịch vụ là hữu hạn so với các công việc cần xử lý khi có các yêu cầu truy xuất xảy ra đồng loạt. Trong trường hợp này các yêu cầu truy xuất sẽ cạnh tranh nhau để có được sự phục vụ của các máy ảo dẫn đến tắc nghẽn. Hơn nữa, sự tắc nghẽn sẽ làm giảm hiệu suất của hệ thống và doanh thu của các nhà cung cấp dịch vụ. Các kỹ thuật Load Balancing hiện có tồn tại một số vấn đề khác nhau như: chỉ cân bằng tải sau khi một máy chủ đã bị quá tải. Trong môi trường điện toán đám mây thì tải được xem như là các yêu cầu đến để các máy chủ ảo phục vụ [1]. Trong công trình [1] tác giả đề xuất một giải thuật là chuyển các yêu cầu của người sử dụng sang một máy một máy ảo khác có mức độ sử dụng thấp hơn khi xả ra bế tắc trên một VM nào đó. Vì thế, trong bài bái này chúng tôi đề xuất một giải thuật nhằm phòng tránh sự quá tải của các máy ảo. Giải thuật đề xuất nhằm tránh hiện tượng quá tải bằng cơ chế sắp xếp phần trăm mức độ sử dụng của các VMs theo thứ tự nhất định để phục vụ cho các yêu cầu đến theo nguyên tắc: chọn ra máy chủ nào có mức độ sử dụng thấp sẽ đáp ứng cho các yêu cầu đến có dung lượng cao và ngược lại (giả sử đơn vị đo của các yêu cầu đến là dung lượng). Nếu có 2 máy chủ có mức độ sử dụng như nhau thì máy nào có trọng số nhỏ hơn sẽ được chọn.

Bài báo này được tổ chức như sau. Phần I: giới thiệu. Phần II là các nghiên cứu liên quan. Phần III là đề xuất giải thuật phòng tránh hiện tượng quá tải cho các máy chủ ảo. Phần IV, mô phỏng thuật toán và kết quả thu được. Phần V kết luận.

II. CÔNG TRÌNH LIÊN QUAN

Rashmi. K. S [1] và các cộng sự đã thực hiện một giải thuật cân bằng tải nhằm phòng tránh bế tắc của các máy ảo trong môi trường điện toán đám mây. Giải thuật được đưa ra ở đây là chuyển các yêu cầu của người sử dụng sang VM khác mức độ sử dụng thấp hơn nếu xảy ra bế tắc trên một VM nào đó. Nếu như có nhiều máy ảo VM có cùng mức độ sử dụng thì sẽ dựa vào thông số hop time để quyết định chọn máy VM phục vụ, máy nào có hop time thấp sẽ được chọn. Giải thuật này đã làm tăng hiệu quả kinh doanh của các nhà cung cấp dịch vụ đám mây.

P. Srinivasa Rao[2] và các cộng sự đã đề xuất một cơ chế cân bằng tải động với hệ thống giám sát tập trung. Mục đích của hệ thống giám sát tập trung dựa trên ý tưởng rằng, việc tính toán trong một môi trường có thể phân tán, nhưng trạng thái của từng nhiệm vụ, công việc phải được giám sát tại một trung tâm để theo dõi và lập lịch tốt hơn. Điều này cũng cho

ISBN: 978-604-67-0635-9

phép việc phân phối tải hợp lý trên các bộ xử lý công việc, giúp giảm thiểu thời gian xử lý và tryền thông trên mạng. Cơ chế này loại bỏ các giao tiếp không cần thiết của bộ cân bằng tải trước khi phân bổ các máy ảo bằng cách không truy vấn về nguồn lực có sẵn của máy chủ ảo. Mô hình này tính đến thời gian đáp ứng trước khi đi qua và khi đi qua bộ cân bằng tải nên giảm thiểu được thời gian trễ của hệ thống.

Bibhudatta Sahoo [3] và nhóm nghiên cứu của ông đã giới thiệu kỹ thuật cân bằng động trên hệ thống tính toán phân tán không đồng nhất (Heterogeneous Distributed Computing System - HDCS) và phân tích tác động của tính không đồng nhất đến năng lực xử lý nhiệm vụ đến của các node. Kỹ thuật này thực hiện phân bổ các nguồn lực, tài nguyên một cách có hiệu quả và kết quả là đã giảm thiểu được thông số makespan (thời gian tối đa hoàn thành công việc tại mỗi nút).

Nhóm tác giả Javed Ali [4] đã đề xuất một cơ chế phân loại các nhiệm vụ (task) tính toán tại các nút, thực hiện phân vùng nhiệm vụ và cân bằng tải tại các nút đó. Ý tưởng chính của bài báo này là làm sáng tỏ khi nào, lúc nào cần phân loại các chiến lược thích hợp. Bài báo khẳng định, không có chiến lược phân loại lý tưởng mà tùy thuộc vào từng trường hợp và cơ chế mà nhóm tác giả này đưa ra đã thu được kết quả khả quan, đóng góp vào hướng nghiên cứu hệ thống song song phân tán.

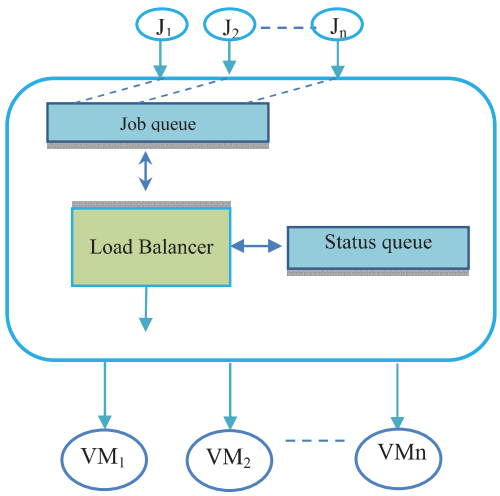
Wenhong Tian [5] và các cộng sự đã giới thiệu thuật toán (Dynamic and Integrated Resource Scheduling Algorithm-DAIRS) tích hợp giữa cân bằng tải động và lập lịch cho trung tâm dữ liệu đám mây, với mục tiêu là tích hợp đo tổng mức độ mất cân bằng của một trung tâm dữ liệu cũng như mức độ mất cân bằng trung bình của mỗi máy chủ. Việc lập lịch cho một hệ thống tính toán phân tán là rất phức tạp, điều khiển các thông số nguồn lực để kiểm soát tình trạng của hệ thống chính xác sẽ nâng cao được hiệu năng hệ thống, giảm được các thông số. Giải thuật LIF [5] đã xem xét tài nguyên một cách đa chiều nên có nhiều thông tin trước khi lựa chọn máy ảo phục vụ.

Agraj Sharma [6] và nhóm tác giả đã đưa ra một giải thuật khắc phục được một số tồn tại trong các giải thuật trước đó: (i) Chỉ thực hiện cân bằng tải sau khi một máy chủ đã quá tải; (ii) liên tục truy vấn về trạng thái nguồn lực máy chủ dẫn đến việc tiêu tốn băng thông, năng lực tính toán. Căn cứ vào thời gian đáp ứng, giải thuật đưa ra cách giải quyết việc phân bổ nguồn lực phù hợp. Ý tưởng chính là xem xét thời gian đáp ứng hiện tại và các thông số liên quan của nó để quyết định phân bổ các yêu cầu mới đến. Mục tiêu của giải thuật là: (i) Loại bỏ sự liên lạc không cần thiết giữa các máy ảo liên quan đến tài nguyên có sẵn với các máy chủ; (ii) Giảm thời gian đáp ứng cho người sử dụng; (iii) Ngăn chặn sự mất cân bằng tải giữa các ảo trước khi đưa vào phục vụ; (iv) Chỉ tiêu thời đáp ứng cho người sử dụng; (v) Dự đoán thời gian đáp ứng trong tương lai từ bất kỳ máy chủ nào cho các yêu cầu hiện tại. Để thực hiện giải thuật này, các tác giả đã giả định: (i) Bộ cân bằng tải biết trước những dịch vụ nào đang chạy trên bất kỳ máy ảo nào, bất cứ lúc nào; (ii) Web Server biết trước thời gian đáp ứng của mỗi dịch vụ trên mỗi máy ảo; (iii) Nếu hai máy ảo có cấu hình tương tự nhau về bộ nhớ RAM, bộ xử lý, và I/O, thì thời gian thực thi không có sự khác biệt đáng kể để có thể thay đổi kết quả thuật toán.

Trong công trình [1] đã thực hiện được việc phòng tránh bế tắc cho các máy chủ trong điện toán đám mây. Tuy nhiên còn một số tồn tại sau: khi có danh sách các yêu cầu đến thì bộ Cloud Manager [1] mới thực hiện phân tích trong cơ sở dữ liệu để tìm ra VM có mức độ sử dụng thấp rồi mới gán yêu cầu cho VM đó và định kỳ giám sát trạng thái của các VMs nhằm phát hiện VMs nào quá tải, do đó xét về khía cạnh thời gian thì hiệu quả chưa cao dẫn đến giảm thời gian đáp ứng cử hệ thống. Vì vậy, trong bài báo này chúng tôi giới thiệu một cách tiếp cận mới trong việc phân bố tải dựa trên việc tích hợp thêm một hàng đợi chứa thông tin máy chủ ảo (mức độ sử dụng) để phục vụ cho bộ xử lý Load Balancer lấy thông tin. Hai hàng đợi sẽ được giải thuật tiền xử lý (sắp xếp theo thứ tự) trước khi đưa thông tin đầu vào cho bộ xử lý Load Balancer. Bộ cân bằng tải chỉ việc đọc thông tin trong hàng đợi và gán các yêu cầu cho máy chủ. Việc này dẫn đến giảm được thời gian ra quyết định chọn máy chủ ảo khi có một danh sách các yêu cầu đến, nên tránh được hiện tượng quá tải, dẫn đến nâng cao hiệu năng hệ thống.

III. ĐỀ XUẤT GIẢI THUẬT

1. Sơ đồ logic



Hình 1. Mô hình kiến trúc đám mây.

Mô hình chúng tôi đề xuất để phòng tránh quá tải trong điện toán đám mây từ đó thực hiện việc phân bổ tải được cân bằng hơn. Hình 1 biểu diễn kiến trúc đám mây theo cách tiếp cận này. Các công việc đến được biểu diễn bằng tập hợp J = {

$J_1, J_2,\ldots,J_n$ }, mức độ sử dụng (%) của các máy ảo được lưu trong hàng đợi S = { $S_1, S_2,\ldots, S_n$} tương ứng với các máy ảo $VM_1, VM_2, \ldots, VM_n$. Cơ chế hoạt động như sau: Khi có một yêu cầu job đến thì Load Balancer sẽ phân tích trong hàng đợi S để chọn ra VM nào có % sử dụng thấp nhất đáp ứng cho yêu cầu vừa đến. Sau khi đã chuyển yêu cầu đến máy phục vụ thì Load Balancertự động cập nhật lại dữ liệu bao gồm: Job ID, VM ID cho S. Ở trong bài báo này chúng tôi đề xuất hàng đợi S theo cơ chế FIFO (Fist in Fist out), theo thứ tự % sử dụng từ thấp đến cao. Khi có yêu cầu đến Load Balancer chỉ việc đọc trạng thái trong hàng đợi và chỉ định máy chủ nào đáp ứng cho yêu cầu đó. Từ đó, giảm thiểu được thời gian thực thi của đám mây. Đây là một ưu điểm của giải thuật đề xuất.

2. Giải thuật trước đây [1]

- Bước 1: Khởi tạo VM Status = 0 cho tất cả các máy ảo. Cloud Manager duy trì một cơ sở dữ liệu bao gồm: Job ID, VM ID and VM Status.
- Bước 2: Khi có một danh sách các yêu cầu đến, Cloud Manager phân tích trong cở sở dữ liệu để phân bổ các yêu cầu cho các VM có mức độ sử dụng thấp. Nếu mức độ sử dụng của các máy ảo VM bằng nhau thì VM nào có hop time nhỏ nhất sẽ được chọn.
- Bước 3: Cloud Manager cập nhật dữ liệu sau khi đã phân bổ các yêu cầu cho VM thỏa mãn.
- Bước 4: Cloud Manager định kỳ giám sát tình trạng của các VM, nếu xảy ra quá tải thì chuyển các yêu cầu công việc đến VM nào có mức độ sử dụng thấp hơn.
- Bước 5: Lựa chọn VM nào để đáp ứng yêu cầu sẽ dựa vào hop time. VM có hop time nhỏ nhất sẽ được chọn.
- Bước 6: Cloud Manager cập nhật cấu trúc dữ liệu bằng cách thay đổi các mục phù hợp
- Bước 7: Lặp lại Bước 2.

Giải thuật này tồn tại một tương tác giữa Cloud Manager [1] khi có yêu cầu đến, lúc này Cloud Manager mới phân tích trong cơ sở dữ liệu để tìm ra VM nào có mức độ sử dụng thấp nhất, do đó không giảm được thời gian delay.

3. Giải thuật đề xuất

Trong giải thuật này chúng tôi sử dụng 2 hàng đợi:

- Hàng đợi J: chứa các yêu cầu đến máy chủ ảo (JobID).
- Hàng đợi S: chứa phần trăm (%) sử dụng của máy ảo (VM Status).

Giải thuật:

Input:
J = { J1, J2,……, Jm); // Tập các Job đến .
S = { S1, S2,….., Sn); // Tập % mức độ đã sử dụng của các VMs
W = [W]n x n: // Ma trận trọng số
Output: Total respontime.

- *Bước 1. Đăng ký ID cho các Job đến {Ji} , $1 \leq i \leq m$;*
- *Bước 2. Sắp xếp lại hàng đợi S theo thứ tự từ nhỏ đến lớn,gán VM tương ứng cho các Si,, $1 \leq i \leq n$;*
- *Bước 3. Khi có Job đến, Load Balancer sẽ lấy thông tin trong S chọn ra VMs nào có mức độ sử dụng nhỏ nhất rồi gán Job vừa đến vào VMs đó, nếu có nhiều VMs có % mức độ sử dụng giống nhau thì máy nào có weight nhỏ sẽ được chọn.*
- *Bước 4. Xóa Ji vừa được phân bổ. Cập nhật lại % sử dụng.*
- *Bước 5. Nếu vẫn còn Job trong hàng đợi J thì lặp lại bước 3 cho đến khi % = 100% và J rỗng.*
- *Bước 6. Xóa VM ID khỏi S nếu % = 100%*
- *Bước 7. Cập nhật lại J, S*

Trong giải thuật đề xuất, nhờ việc sắp xếp các VM Status (% mức độ sử dụng) sẵn sàng cho các Job ID nên Load Balancer chỉ việc gán VM ID cho các Job ID. Việc cập nhật lại queue S một cách tự động và thường xuyên làm cho thời gian delay của hệ thống giảm đi đáng kể.

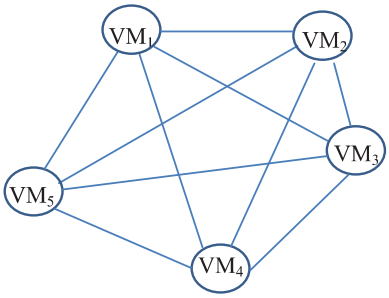
Viết lại giải thuật như sau:
Input:
J = { J1, J2,……, Jm); // Tập các Job đến .
S = { S1, S2,….., Sn); // Tập % mức độ đã sử dụng của các VMs
W = $[W]_{n*n}$: // Ma trận trọng số
Output: Tổng thời gian đáp ứng.

1. *Sắp xếp hàng đợi các công việc đến theo thứ tự từ nhỏ đến lớn Ji và gán ID cho từng công việc , $1 \leq i \leq m$.*
2. *Sắp xếp hàng đợi mức độ sử dụng của máy chủ ảo theo thứ tự từ nhỏ đến lớn Si, $1 \leq i \leq n$;*
   Vòng lặp For: Xét tất cả các node trong đám mây
   Tìm VM có VM Status nhỏ nhất
   gán VM Status : = VM ID
   gán S : ← VM ID;
   Find next hop: next : = weight;
   Kết thúc vòng lặp For.
3. *Vòng lặp For: Xét tất cả các máy ảo*
   Tìm VMs nào có VM Status nhỏ nhất ;
   Gán công việc cho máy ảo tương ứng:
   Nếu có nhiều hơn 2 máy ảo có cùng mức độ sử dụng thì chọn máy có trọng số nhỏ hơn;
   Kết thúc vòng lặp;

4. *Xóa Job ID khỏi J; Cập nhật lại phần trăm sử dụng của VMs.*
5. *Lặp lại bước 3 cho đến khi hết các yêu cầu đến;*
6. *Xóa VM ID khỏi S khi đã sử dụng hết;*
7. *Cập nhật lại J, S;*

IV. MÔ PHỎNG VÀ KẾT QUẢ

1. Sơ đồ thực nghiệm



Hình 2. Sơ đồ kết nối máy chủ thực nghiệm.

Để mô phỏng cho giải thuật đề xuất chúng tôi đưa ra bảng ma trận trọng số đầu vào như Bảng 1. Đây là bảng dữ liệu trọng số giữa các máy chủ ảo trong đám mây với nhau, thể hiện khoảng cách (metric) từ VM này đến VM khác. Ở đây giả sử tất cả các công việc đến cùng một thời điểm. Bảng trọng số này được cố định, trong khi J và S được thay đổi trong quá trình thực nghiệm, nhằm mục đích so sánh kết quả trong 2 trường hợp là J và S chưa được sắp xếp và sau khi đã được sắp xếp cho bộ xử lý Load Balancer sử dụng. Trong khuôn khổ thí nghiệm này chúng tôi sử dụng bộ số liệu J và S dưới dạng mảng một chiều, kích thước 1 x n (với n là số lượng VMs). Mảng J chứa các yêu cầu đến, mảng S chứa phần trăm sử dụng của VMs.

Giả sử ma trận trọng số của các VMs như sau:

Bảng 1. Ma trận trọng số

	VM_1	VM_2	VM_3	VM_4	VM_5
VM_1	0	3	4	6	3
VM_2	3	0	1	5	6
VM_3	4	1	0	3	7
VM_4	6	5	3	0	3
VM_5	3	6	7	3	0

2. Kết quả

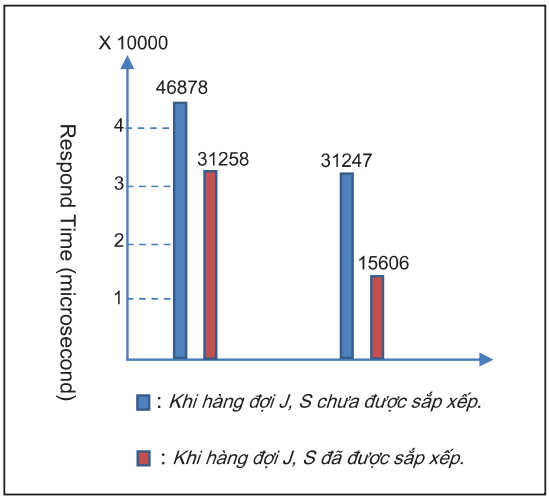
Chương trình mô phỏng được viết bằng ngôn ngữ Python 3.4.1. Ở đây chúng tôi thử nghiệm hai bộ số liệu mô phỏng cho 2 trường hợp:

- Trường hợp 1: Khi J và S chưa được sắp xếp theo thứ tự.
- Trường hợp 2: Khi J và S đã được sắp xếp theo thứ tự.

Chương trình phần mềm sẽ đo tổng thời gian toàn bộ chu trình (Total Respondtime), đơn vị đo là microseconds. Chúng tôi tiến hành thử nghiệm các trường hợp với ma trận trọng số là cố định. So sánh kết quả trong 2 trường hợp là các hàng đợi S và J chưa được sắp xếp và trường hợp đã được sắp xếp trước khi phân bổ máy chủ để đáp ứng các yêu cầu đến.

Trong phần mô phỏng này chúng tôi thực hiện 2 thí nghiệm với ma trận trọng số cố định còn bộ số liệu của J và S được thay đổi. Hình 3 cho thấy, có sự khác biệt về thời gian đáp ứng của hệ thống khi các đầu vào khác nhau. Lần thí nghệm thứ nhất cho các kết quả: khi J và S chưa được sắp xếp để phục vụ cho bộ Load Balancing thì tổng thời gian đáp ứng đo được là 46878 (microsecond), và khi đã sắp xếp thì tổng thời gian này là 31258 (microsecond). Kết quả lần đo thứ 2, tương ứng là 31247 và 15606. Ta thấy rằng, tổng thời gian đáp ứng đã giảm thiểu đi đáng kể khi các hàng đợi được sắp xếp trước khi lựa chọn máy chủ phục vụ cho các yêu cầu mới đến. Kết quả này chỉ ra rằng thuật toán có khả năng giảm được thời gian delay của hệ thống, do đó sẽ cải thiện được hiệu năng và phòng tránh được hiện tượng quá tải trong điện toán đám mây.

Kết quả thực nghiệm được thể hiện ở biểu đồ sau:



Hình 3. So sánh kết quả

V. KẾT LUẬN

Trong tương lai điện toán đám mây là một công nghệ đầy hứa hẹn, là một mô hình cung cấp các nguồn lực cần thiết cho khách hàng theo hướng dịch vụ. Nền tảng áo hóa là một trong những đặc điểm cốt lõi của điện toán đám mây, dùng để ảo hóa các yếu tố ảnh hưởng đến hiệu quả kinh doanh như: nguồn lực công nghệ thông tin, phần cứng, phần mềm, hệ điều hành và dịch vụ. Điện toán đám mây có đặc trưng là tính toán phân tán và đây cũng là nguy cơ có thể xảy ra tắc nghẽn trong quá trình phân bổ nguồn lực để cân bằng tải. Việc phòng tránh được nguy cơ quá tải trong điện toán đám mây sẽ làm cho các nguồn lực ở trạng thái sẵn sàng cao. Các yêu cầu truy cập hệ thống có thể xảy ra đồng thời và sẽ không tránh khỏi sự "cạnh tranh" các nguồn lực nhàn rỗi của đám mây. Chính vì thế thuật toán đề xuất trên đây đã có cơ chế phân phối tải thích hợp và cho kết quả khả quan. Đây cũng là giải pháp mà các nhà cung cấp dịch vụ điện toán đám mây đang cần trong quá trình phát triển kinh doanh của mình, hướng tới nâng cao chất lượng dịch vụ cho khách hàng, thông qua đó cải thiện hiệu quả kinh doanh của mình.

Từ kết quả thực nghiệm có thể khẳng định rằng các hàng đợi nếu được sắp xếp theo thứ tự thì việc giảm thời gian tính toán trong đám mây là hoàn hoàn có thể và điều này chứng tỏ sự đúng đắn của thuật toán đề xuất. Việc giảm thời gian đáp ứng này sẽ làm giảm được đáng kể hiện tượng tắc nghẽn trong đám mây điện toán. Từ đó, nâng cao được hiệu năng cân bằng tải khi phân bổ các nguồn lực trong điện toán đám mây.

TÀI LIỆU THAM KHẢO

[1] Rashmi. K. S, Suma. V, Vaidehi. M, "Enhanced Load Balancing Approach to Avoid Deadlocks in Cloud", Special Issue of International Journal of Computer Applications (0975 – 8887) on Advanced Computing and Communication Technologies for HPC Applications - ACCTHPCA, June 2012.

[2] P. Srinivasa Rao, V.P.C Rao, A.Govardhan, "Dynamic Load Balancing With Central Monitoring of Distributed Job Processing System", International Journal of Computer Applications (0975 – 8887), Volume 65– No.21, March 2013.

[3] Bibhudatta Sahoo, Dilip Kumar, Sanjay Kumar Jena, "Analysing the Impact of Heterogeneity with Greedy Resource Allocation Algorithms for Dynamic Load Balancing in Heterogeneous Distributed Computing System", International Journal of Computer Applications (0975 – 8887), Volume 62– No.19, January 2013.

[4] Javed Ali, Rafiqul Zaman Khan, "Classification of Task Partitioning and Load Balancing Strategies in Distributed Parallel Computing Systems", International Journal of Computer Applications (0975 – 8887) Volume 60– No.17, December 2012.

[5] Wenhong Tian, Xianrong Liu, Chen Jin, Yuanliang Zhong, "LIF: A Dynamic Scheduling Algorithm for Cloud Data Centers Considering Multi-dimensional Resources", Journal of Information & Computational Science, 3925–3937, August 10, 2013.

[6] Agraj Sharma, Sateesh K. Peddoju, "Response Time Based Load Balancing in Cloud Computing", 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014.