PDOA: DỰ BÁO DEADLOCK ĐỂ NÂNG CAO CÂN BẰNG TẢI TRÊN ĐIỆN TOÁN ĐÁM MÂY

Lê Ngọc Hiếu[*], Trần Công Hùng[+]
[*]Khoa Công nghệ thông tin, Trường Đại học Mở TP.HCM
[+]Học viện Công nghệ Bưu chính Viễn thông

Tóm tắt: Trong kỷ nguyên công nghệ thông tin, điện toán đám mây đóng một vai trò quan trọng, giúp giải quyết hầu hết các vấn đề trên nền tảng Internet theo nhu cầu của người dùng. Song, ngày nay việc số lượng người sử dụng điện toán đám mây trên thế giới ngày càng tăng sẽ dẫn đến tình trạng tắc nghẽn tại một số nốt mạng do mất cân bằng tải hoặc do hệ thống treo, hiện tượng này thường được gọi là deadlock. Với bài báo này, nhóm nghiên cứu muốn đề xuất một thuật toán dự báo xảy ra deadlock (PDOA - Prediction of Deadlock Occurrence Algorithm) trong bộ cân bằng tải trên môi trường đám mây. Thuật toán này ứng dụng các thuật toán học máy và các kỹ thuật dự báo, đặc biệt là hồi quy tuyến tính, để dự báo khả năng xảy ra deadlock trong máy ảo. Với dự báo khả năng xảy ra deadlock, bộ cân bằng tải sẽ biết khi nào máy ảo có nguy cơ xảy ra deadlock, và từ đó phân bổ các tài nguyên tránh deadlock, để có thể phục vụ tất cả các request tốt nhất. Thuật toán đề xuất PDOA được thực nghiệm và cài đặt trong môi trường mô phỏng CloudSim tích hợp với kỹ thuật học máy trong bộ thư viện Weka. Các kết quả thực nghiệm được đánh giá bằng cách so sánh với các thuật toán phổ biến trong cân bằng tải hiện nay: FCFS, RoundRobin, MaxMin và MinMin. Kết quả thực nghiệm cho thấy PDOA có hiệu suất tốt hơn thông qua thông số thời gian đáp ứng.

Từ khoá: Điện toán đám mây, cân bằng tải, dự báo Deadlock, PDOA, hồi quy tuyến tính

I. GIỚI THIỆU

Điện toán đám mây [1], [2] đang là xu hướng phát triển của công nghệ thông tin, là công cụ rất tiềm năng phát triển với quy mô lớn trong lĩnh vực công nghệ thông tin. Với đám mây, các lập trình viên có thể tạo ra các ý tưởng tốt hơn cho các dịch vụ chạy trên nền tảng Internet mà không cần quan tâm đến đầu tư vào phần cứng hay triển khai dịch vụ hoặc các chi phí nhân lực để vận hành nó. Người dùng internet cũng có thể truy cập và sử dụng các dịch vụ đám mây thông qua các ứng dụng web, ứng dụng di động hoặc máy tính cá nhân một cách dễ dàng và thuận tiện nhất.

Để tối đa hóa lợi ích mà điện toán đám mây đem lại, cân bằng tải là một trong những nhân tố quan trọng luôn được xem xét và cải thiện. Cân bằng tải [3] nhằm mục đích giải quyết vấn đề về trạng thái mất cân bằng trên môi trường đám mây, giúp nâng cao các dịch vụ cung cấp cho khách hàng. Nó cũng giúp các máy chủ hoạt động hiệu quả hơn bằng cách phân bổ tài nguyên đồng đều, giảm hoặc tránh tình trạng deadlock. Hầu hết các thuật toán cân bằng tải [3] [4] đều xuất phát từ quan điểm nâng cao và cải thiện trạng thái cân bằng của đám mây. Có thể thấy rằng, hầu hết các nhà nghiên cứu và nhà cung cấp dịch vụ đám mây [3] đều tập trung vào các yếu tố này và các yếu tố đặc trưng của cân bằng tải như: các thông số bộ cân bằng tải trên đám mây (thông số hiệu suất và thông số kinh tế), các thuật toán cân bằng tải (thuật toán lập lịch và thuật toán phân bổ; tải phần cứng và tải đàn hồi), bộ cân bằng tải nâng cao (sử dụng Heuristic và thuật toán tối ưu hóa). Nhưng tất cả những cách tiếp cận này đều xuất phát từ góc nhìn và phân tích bộ cân bằng tải từ bên trong nội tại của nó. Tuy nhiên, vẫn còn tồn tại nhiều yếu tố bên ngoài ảnh hưởng trực tiếp đến hiệu suất cân bằng tải của đám mây như đường truyền mạng, hành vi người dùng, đặc điểm địa lý, v/v. Deadlock trên đám mây cũng có thể được xem xét như một yếu tố bên ngoài bởi nguyên nhân của nó đến từ yếu tố bên ngoài bộ cân bằng tải. Deadlock trên đám mây có thể xảy ra do sự biến thiên của các request bởi sự đa dạng về hành vi của người dùng.

Trong bài báo này, với cách tiếp cận từ bên ngoài bộ cân bằng tải trên đám mây, nhóm nghiên cứu muốn tập trung vào deadlock và hiểu cách nó xảy ra do các yếu tố bên ngoài. Từ đó, đề xuất một phương pháp dự báo deadlock để có thể giúp giảm tình trạng tắc nghẽn và treo máy khi phân bổ tài nguyên giữa các máy ảo (VM). Với sự đa dạng về dữ liệu của các request, bài nghiên cứu này đưa ra mô hình deadlock dưới dạng các hàm phụ thuộc đầu vào các thuộc tính của request cũng như các đặc điểm của đám mây. Thuật toán đề xuất là kết quả của việc sử dụng mô hình hồi quy tuyến tính dựa trên dữ liệu lịch sử lưu lại của các máy ảo để phục vụ các request của người dùng tương ứng. Dữ liệu sử dụng bao gồm mức độ sử dụng CPU, mức độ sử dụng RAM và mức độ sử dụng ổ cứng (lưu trữ) của các request trong quá khứ. Thuật toán đề xuất sẽ so sánh và tính toán trạng thái của tất cả các máy ảo, đồng thời sử dụng hồi quy tuyến tính để dự đoán mức sử dụng của request đang xét để phân bổ, sau đó phân bổ request này tương ứng cho máy ảo phù hợp để tránh xảy ra deadlock. Để tránh deadlock, thuật toán sử dụng các ngưỡng về mức độ sử dụng CPU, bộ nhớ (RAM) và ổ cứng (STORAGE) của máy ảo. Nếu mức độ sử dụng được tài nguyên (CPU, RAM và STORAGE) dự đoán vượt quá ngưỡng cho phép, thì có thể xảy ra deadlock và chúng ta sẽ cần phân bổ request đang xử lý cho một máy ảo khác. Đây là ý tưởng chính mà bài báo muốn đóng góp nhằm cải thiện cho bộ cân bằng tải của điện toán đám mây.

Bài báo này cũng nhằm mục đích tìm hiểu việc ứng dụng một số thuật toán học máy (ML) và thống kê có thể áp dụng vào bộ cân bằng tải. Đặc biệt, bài báo này đã chọn ra

Tác giả liên hệ: Lê Ngọc Hiếu,
Email: hieu.ln@ou.edu.vn
Đến tòa soạn: 11/2022, chỉnh sửa: 12/2022, chấp nhận đăng: 01/2023.

một số thuật toán học máy có thể dự báo deadlock trong các công trình liên quan. Các thuật toán tìm được bao gồm hồi quy tuyến tính, mạng nơron và cây hồi quy. Hướng đến sự phù hợp nhất, bài báo chọn hồi quy tuyến tính, và áp dụng vào đề xuất nhằm dự báo sự xuất hiện của deadlock cho từng máy ảo trên đám mây. Tập dữ liệu xây dựng mô hình hồi quy tuyến tính được thu thập từ các request trong quá khứ và từ các máy ảo phục vụ cho các request đó. Kết quả thực nghiệm cho thấy rằng sai số dự đoán có thể chấp nhận được, từ 1,5 - 10,5% và nó giúp hiệu suất của thuật toán đề xuất (PDOA) trở nên tốt hơn so với các thuật toán được so sánh: FCFS, RoundRobin, MaxMin và MinMin.

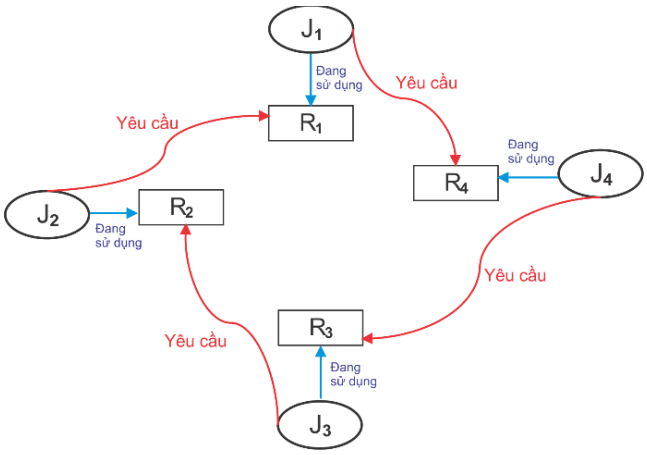
Những đóng góp chính của bài báo này: (i) Cách phân tích từ tiếp cận bên ngoài bộ cân bằng tải để nghiên cứu cân bằng tải trên môi trường điện toán đám mây, yếu tố bên ngoài được quan sát là nguyên nhân gây ra deadlock và deadlock của máy ảo. (ii) Ứng dụng mô hình hồi quy tuyến tính để dự báo deadlock bằng cách sử dụng ngưỡng. (iii) Đánh giá thử nghiệm thuật toán đề xuất, PDOA, với hiệu suất tốt hơn các thuật toán cân bằng tải phổ biến hiện nay.

Để từng bước hiểu rõ hơn về đề xuất của chúng tôi, bài báo gồm 6 phần. Phần đầu tiên là phần giới thiệu. Phần tiếp theo sẽ thảo luận đôi nét về các công trình liên quan. Trong phần 3, chúng tôi sẽ giới thiệu ngắn gọn về cơ sở lý thuyết: điện toán đám mây và cân bằng tải, deadlock và deadlock trên đám mây, máy học và hồi quy tuyến tính. Phần 4 sẽ trình bày và mô tả thuật toán đề xuất PDOA. Trong phần 5, chúng tôi sẽ mô tả về các kết quả mô phỏng thực nghiệm và thảo luận về kết quả này. Và cuối cùng, trong phần 6, chúng tôi sẽ trình bày kết luận của bài nghiên cứu và hướng phát triển trong tương lai.

II. CÁC CÔNG TRÌNH LIÊN QUAN

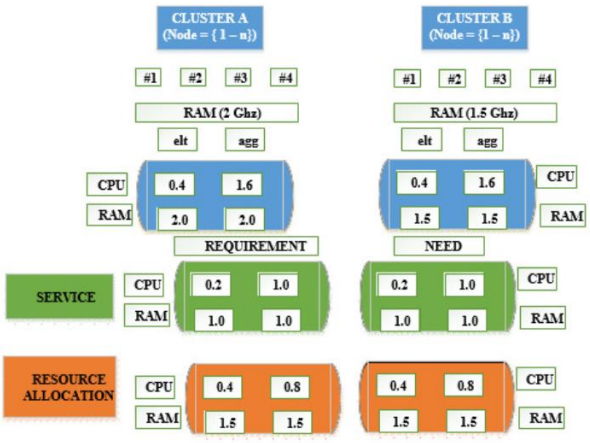
Nghiên cứu về dự báo trên môi trường đám mây và cân bằng tải trên đám mây là một lĩnh vực đã được chú trọng trong nhiều năm gần đây. Mặc dù đã có rất nhiều nghiên cứu được công bố, tuy nhiên có rất ít phương pháp tiếp cận dự báo deadlock xảy ra trên môi trường cloud. Trong phần này, chúng tôi chọn các công trình có liên quan đến deadlock trên đám mây, đồng thời mô tả ưu, nhược điểm của từng công trình cũng như những khó khăn khi áp dụng các phương pháp dự báo và hiệu suất khi chạy trên đám mây. Bài báo này xin trình bày các công trình này theo trình tự thời gian.

Mahitha và cộng sự [5] đã công bố bài nghiên cứu với tiêu đề *"Deadlock Avoidance through Efficient Load Balancing to Control Disaster in Cloud Environment"* vào năm 2013. Các tác giả đã chỉ ra rằng việc quản lý tài nguyên không hiệu quả sẽ dẫn đến deadlock. Bên cạnh đó, bài báo này đã trình bày một kỹ thuật hiệu quả trong cân bằng tải để kiểm soát các vấn đề deadlock trên đám mây. Các tác giả nhận thấy rằng deadlock sẽ xảy ra khi có một vòng lặp bên trong một máy ảo, điều đó có nghĩa là một công việc đang chờ một tài nguyên khác và tài nguyên này cũng chờ công việc đó. Thuật đề xuất của bài báo áp dụng cho cân bằng tải, đánh giá kết quả bằng cách so sánh hiệu suất đám mây khi có và không có bộ cân bằng tải này. Đề xuất sử dụng công cụ CloudAnalyst để cài đặt. Các kết quả mô phỏng cho chúng ta thấy rõ về vấn đề quản lý và sử dụng tài nguyên dẫn đến thời gian phản hồi tối thiểu, thời gian thực thi và thông lượng tốt hơn.



Hình 1 Deadlock trên đám mây [5]

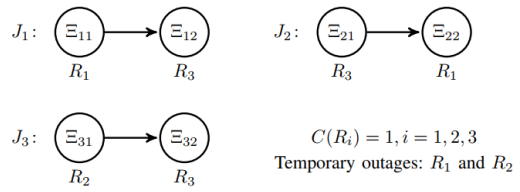
Nguyễn Hà Huy Cường và cộng sự [6] đã cải thiện nghiên cứu của chính nhóm tác giả về deadlock và cách phát hiện nó nhằm phân bổ tốt hơn trong môi trường phân tán không đồng nhất, đặc biệt là môi trường cloud. Bài báo này chú ý nhiều hơn đến việc phân bổ tài nguyên với mức độ sử dụng cơ sở hạ tầng, thay vì nghiên cứu cách phân bổ tài nguyên vật lý thì các tác giả nghiên cứu cách sử dụng tài nguyên ảo để đạt được hiệu quả sử dụng tốt hơn trong môi trường đám mây. Các tác giả đề xuất một thuật toán mới tập trung nhiều vào phân bổ tài nguyên hạ tầng, nó phân bổ động các máy ảo trong đám mây với mục tiêu chính là phát hiện deadlock. Bài báo này đề xuất một thuật toán trên các cụm đám mây, sử dụng đồ thị phân bổ tài nguyên (RAG) có ma trận đại diện cho sự tồn tại của deadlock. Ma trận này có độ phức tạp $O(m \times n)$ cho trường hợp xấu nhất, nhưng nhóm tác giả đã giảm kích thước bằng cách giảm hàng và cột, được ma trận mới có độ phức tạp $O(min\ (m, n))$ cho trường hợp xấu nhất. Các tác giả đã chứng minh và tính toán độ phức tạp của thuật toán đề xuất và họ đã thử nghiệm công trình trên CloudSim. Công trình này khá tốt và có hiệu quả trên hệ thống phân tán như đám mây.



Hình 2 Ví dụ về trường hợp sự cố với hai nốt và một dịch vụ, cho thấy khả năng phân bổ tài nguyên. [6]

Năm 2016, Spyros Reveliotis và cộng sự [7] đã đề cập đến việc tránh tắc nghẽn bằng cách sử dụng hệ thống phân bổ tài nguyên tuần tự (RAS). Họ muốn mở rộng RAS để giải quyết tình trạng thiếu tài nguyên tiềm ẩn, về khối lượng và khả năng kiểm soát của chúng. Bài nghiên cứu này thúc đẩy lý thuyết SC cho các hệ thống sự kiện chuyển mạch rời rạc (s-DES) để cung cấp một phương pháp xử lý có hệ thống mới cho phiên bản phức tạp hơn của vấn đề tránh deadlock

RAS. Theo mô hình mô hình s-DES, hoạt động được xem xét của RAS và chính sách SC cho phép tối đa tương ứng đều được phân tách thành một số chế độ hoạt động được xác định bởi các tập hợp tài nguyên bị lỗi đang chạy. Đặc biệt, trình giám sát mục tiêu phải được phân tách thành một tập hợp các *"vị từ được bản địa hóa"*, trong đó mỗi vị từ được liên kết với một trong các chế độ hoạt động. Phần quan trọng và đóng góp chính của bài báo này liên quan đến sự phát triển của các vị từ bản địa hóa, cho phép mô tả đặc tính chính và tính toán hiệu quả của trình giám sát được tìm kiếm. Với các vị từ có sẵn, một biểu diễn phân tán cho trình giám sát được tìm kiếm thích hợp cho việc triển khai thời gian thực. Cuối cùng, sẽ thu được kết quả thông qua sự điều chỉnh của thuật toán phân tán có liên quan được cung cấp bởi lý thuyết s-DES SC hiện tại.



Hình 3 RAS được xem xét trong ví dụ được cung cấp. Tài nguyên R1 và R2 có thể gặp sự cố tạm thời, trong khi tài nguyên R3 được cho là không bao giờ bị lỗi. [7]

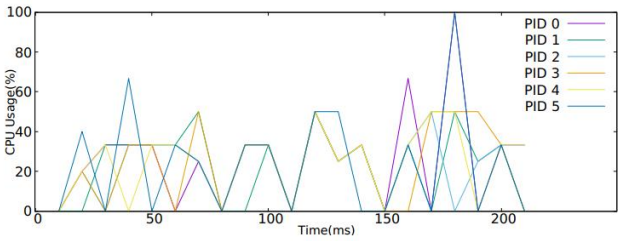
Trong năm 2018, Emeka E. Ugwuanyi [8] đã đề xuất cung cấp tài nguyên bằng cách sử dụng kỹ thuật tránh deadlock trong môi trường giao dịch của ngân hàng. Bài viết này viết về điện toán truy cập biên đa (MEC), một thuật toán cung cấp tài nguyên tránh deadlock được đề xuất cho các thiết bị IoT công nghiệp để đảm bảo độ tin cậy cao hơn của các tương tác trên mạng. Ý tưởng được đề xuất kết hợp thuật toán yêu cầu tài nguyên ngân hàng bằng cách sử dụng SDN để giảm chi phí liên lạc. Kết quả mô phỏng của bài viết đã chỉ ra rằng có thể ngăn chặn hệ thống deadlock bằng cách áp dụng thuật toán được đề xuất, cuối cùng sẽ dẫn đến việc tương tác mạng đáng tin cậy hơn giữa các trạm di động và nền tảng MEC.

Bảng 1. Các chiến lược Deadlock [8]

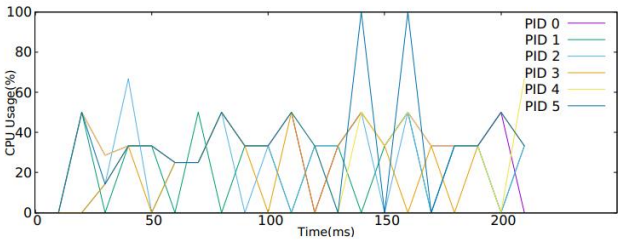
Các thuật toán phát hiện	Thuật toán Lamport
	Thuật toán Chandy-Misra-Haas
	Thuật toán phát hiện Deadlock song song
	Phát hiện trong các hệ thống không đồng nhất
	Phát hiện deadlock không cấu trúc
Các thuật toán ngăn chặn	Các phương pháp cân bằng tải
	Thuật toán ngăn chặn deadlock trong môi trường lưới
Các thuật toán phòng tránh	Thuật toán ngân hàng

Năm 2019, Nguyễn Hà Huy Cường và cộng sự [9] tiếp tục các công trình cải tiến của mình và công bố một mô hình mới về phân bổ tài nguyên nhằm tránh deadlock (ADRA). Với khái niệm kế thừa điện toán lưới, môi trường đám mây cần một chiến lược phân bổ tài nguyên tổng thể và tốt hơn. Bài viết này đã đưa ra một khái niệm mới về việc sử dụng Wait-For-Graph (WFG) để áp dụng cho việc tránh deadlock. Thuật toán mới này chủ yếu nhằm phân bổ nhiều tài nguyên cho các dịch vụ cạnh tranh chạy trong các máy ảo trên các nền tảng phân tán không đồng nhất và được thực hiện trong môi trường mô phỏng CloudSim. ADRA cũng được đánh giá bằng độ phức tạp, với độ phức tạp về thời gian là *O(m*(n - 1)/2 + 2e)*, cải thiện của các bậc độ lớn gần đúng trong các trường hợp thực tế. Việc thực hiện thực nghiệm đề xuất này cho thấy rằng việc quản lý tài nguyên đã hoàn thành tốt công việc và đạt được kết quả tốt.

Trong năm 2019, Sonam Sherpa và cộng sự [10] đã đề xuất một quan điểm khác về deadlock, quan điểm từ nhà phát triển và lập trình viên, những người đã xây dựng các thuật toán dẫn đến deadlock. Trong việc phát triển các thuật toán, các lỗi deadlock rất khó chẩn đoán và tái tạo lại, chi phí thời gian thực hiện cao. Các tác giả trình bày việc xác định các lỗi đồng thời bằng cách sử dụng dấu vết tiêu thụ tài nguyên. Điều này dựa trên quan sát các kiểu truy cập và tiêu thụ tài nguyên là những dấu hiệu quan trọng về hành vi thời gian thực hiện của phần mềm, đồng thời có thể được sử dụng như một cơ chế để hướng dẫn quy trình gỡ lỗi phần mềm. Sau đó, chứng minh rằng việc giám sát dấu vết tài nguyên trong thời gian thực hiện có thể giúp phát hiện lỗi phần mềm một cách hiệu quả. Cụ thể, đối với các chương trình MPI, một bộ phân loại SVM đơn giản có thể phát hiện các deadlock với độ chính xác cao chỉ bằng cách sử dụng các mẫu mức độ sử dụng CPU. Nhìn từ góc độ khác, chúng ta có thể hiểu thêm về deadlock, cách nó xảy ra cũng như cách chúng ta có thể sử dụng một số kỹ thuật máy học để phát hiện và tránh nó.



(a) Normal Execution



(b) Execution with Deadlock

Hình 4 Dấu vết tiêu thụ tài nguyên [10]

Năm 2022, nghiên cứu của Yuandao Cai và cộng sự [11] giới thiệu "Peahen", một phương pháp mới phát hiện deadlock tĩnh nhằm nâng cao độ chính xác và khả năng mở rộng trong các hệ thống phần mềm. Peahen đạt được hiệu quả bằng cách sử dụng hai giai đoạn phân tích hợp tác: giai đoạn xây dựng biểu đồ khóa không nhạy cảm với ngữ cảnh mã hóa thông tin thu thập khóa cần thiết và ba tinh chỉnh chính xác nhưng lười biếng giúp tinh chỉnh dần chu kỳ deadlock cho các ngữ cảnh cần thiết. Các thử nghiệm cho thấy Peahen vượt trội so với các công cụ hiện đại về độ chính xác trong khi kiểm tra hiệu quả các hệ thống hàng triệu dòng với tỷ lệ dương tính giả thấp (false positive), xác định thành công deadlock đã được xác nhận trong nhiều hệ thống mã nguồn mở.

Trong bài báo 2022 của Jinpeng Zhou và cộng sự [12] giải quyết thách thức phát hiện các deadlock liên quan đến các biến điều kiện trong các chương trình đa luồng, rất khó xác định bằng các công cụ hiện có chủ yếu tập trung vào khóa (lock). Công cụ được đề xuất, UnHang, mở rộng sự

phụ thuộc khóa cổ điển sang sự phụ thuộc tổng quát, cho phép nó mô hình hóa các deadlock giao tiếp dưới dạng các chu kỳ giữ và chờ. Kết quả thử nghiệm trên các ứng dụng chứng minh rằng UnHang tìm thấy thành công các deadlock đã biết trước, và phát hiện ra những deadlock mới chỉ với khoảng 3% chi phí hoạt động và 8% chi phí bộ nhớ, khiến nó trở thành một công cụ thiết thực và hiệu quả cho các môi trường triển khai.

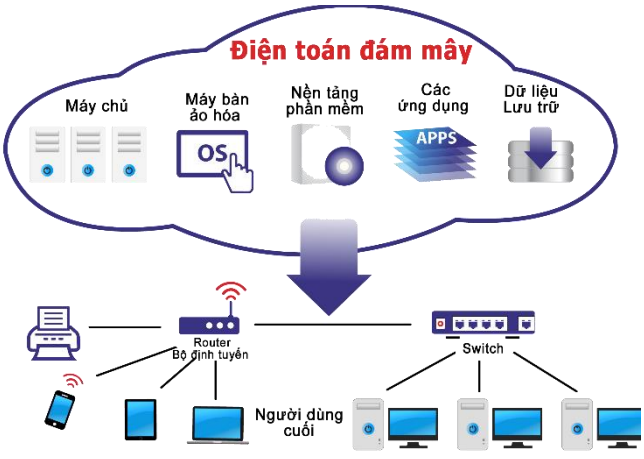
Sau vài đánh giá ngắn gọn ở trên, chúng ta có thể hiểu thêm về deadlock và tình hình nghiên cứu deadlock trên môi trường đám mây. Từ các điểm mới mà chưa được nghiên cứu, là nguồn cảm hứng trong nghiên cứu của bài báo này, nhóm nghiên cứu xin đề xuất một phương pháp có thể dự báo deadlock xảy ra trên máy ảo và mức độ sử dụng các tài nguyên của máy ảo. Với dự báo này, chúng tôi tích hợp vào cân bằng tải, phân bổ các request đến đúng máy ảo có hiệu suất tốt hơn để xử lý. Phương pháp của bài báo hoàn toàn thực hiện trên môi trường viết code và chạy tự động, không có sự tham gia của chuyên gia phân tích.

III. CƠ SỞ LÝ THUYẾT

Trong phần này, bài viết sẽ đề cập đến các vấn đề nền tảng lý thuyết cần thiết cho đề xuất của chúng tôi. Đầu tiên, giới thiệu một số vấn đề nền tảng về điện toán đám mây và cân bằng tải trên đám mây. Sau đó là một số vấn đề cơ bản về deadlock và deadlock trên đám mây. Cuối cùng, phần này sẽ giới thiệu một số kỹ thuật máy học tiềm năng có thể áp dụng được trong bộ cân bằng tải, phương pháp tiếp cận của nhóm nghiên cứu.

A. Điện toán đám mây

Theo [13] [14], điện toán đám mây hay còn được gọi là máy chủ ảo trả tiền cho mỗi lần sử dụng, là một mô hình điện toán sử dụng các công nghệ máy tính và phát triển dựa trên môi trường Internet. Cụ thể hơn, trong mô hình điện toán đám mây, tất cả tài nguyên, thông tin và phần mềm được chia sẻ và cung cấp cho máy tính, thiết bị và người dùng dưới dạng dịch vụ trên nền tảng cơ sở hạ tầng qua mạng công cộng (thường là Internet).

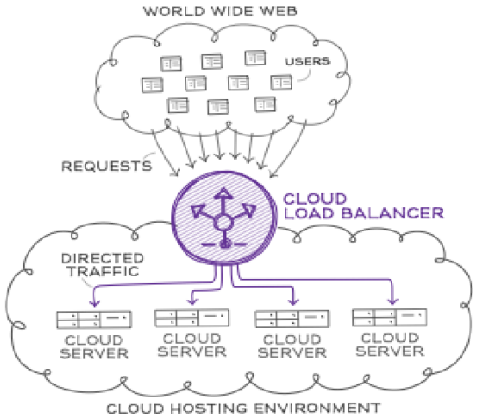


Hình 5 Mô hình điện toán đám mây (Nguồn: https://informationq.com/)

Thuật ngữ Điện toán đám mây xuất hiện để khái quát các hướng phát triển của cơ sở hạ tầng thông tin đã và đang diễn ra. Thuật ngữ này có thể được hiểu một cách dễ dàng: các tài nguyên khổng lồ như cơ sở hạ tầng, nền tảng, phần mềm làm dịch vụ và các dịch vụ này sẽ có thể là các máy chủ ảo (các đám mây) trên Internet thay vì sử dụng các máy tính cục bộ như gia đình hoặc văn phòng, mục đích để mọi người kết nối và sử dụng bất cứ khi nào họ cần.

B. Cân bằng tải trên điện toán đám mây

Cân bằng tải [15] [16] [17] là một phương pháp luận quan trọng trong điện toán đám mây, giúp máy chủ hoạt động hiệu quả hơn bằng cách phân phối tài nguyên hiệu quả, giảm hoặc tránh tình trạng deadlock. Bộ cân bằng tải có nhiều chức năng tốt nhất nhưng hầu hết lại là các chức năng cơ bản, bao gồm Chặn lưu lượng mạng tới một trang web (hoặc một giao diện người dùng khác) - điều này nhằm đảm bảo rằng không phải tất cả các request đều được xử lý bởi một hoặc nhiều máy chủ cố định. Nó chỉ định tải cho các máy chủ tương ứng cho quá trình xử lý, xử lý và trả về kết quả. Cân bằng tải cũng có thể cung cấp khả năng dự phòng bằng cách sử dụng nhiều hơn một kịch bản dự phòng.



Hình 6 Ví dụ về bộ cân bằng tải trên đám mây [18]

Cân bằng tải trên môi trường đám mây [18] [19] cung cấp giải pháp cho các vấn đề thiết lập và sử dụng tài nguyên khác nhau trong môi trường điện toán đám mây. Cân bằng tải phải đảm bảo đồng thời hai nhiệm vụ chính. *Một* là cung cấp tài nguyên hoặc phân bổ tài nguyên để tối ưu hóa tài nguyên sử dụng, tối đa hóa thông lượng, giảm thiểu thời gian phản hồi và để tránh quá tải. *Hai* là lập lịch công việc tương ứng các request trong môi trường phân tán. Điện toán đám mây có 2 dạng cân bằng tải được sử dụng phổ biến hiện nay, đó là cân bằng tải tĩnh và cân bằng tải động.

Môi trường điện toán đám mây [18] dựa trên các nhà phát triển và cách xác định cấu hình đám mây theo yêu cầu của nhà cung cấp đám mây. Cân bằng tải đám mây bị ảnh hưởng bởi 3 yếu tố chính [20] [21]: môi trường muốn cân bằng tải, bản chất của tải trọng và các công cụ cân bằng tải có sẵn.

C. Deadlock trên môi trường đám mây

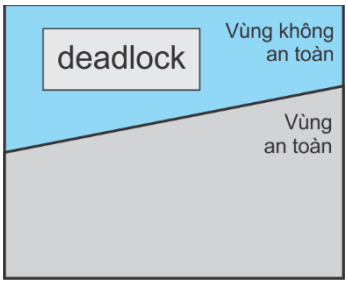
Theo [22], trong bối cảnh điện toán, "deadlock" có thể được coi là một điều kiện khi hai hoặc nhiều tiến trình đang chờ một tài nguyên do một tiến trình khác nắm giữ hoặc khi nhiều hơn hai tiến trình đang chờ tài nguyên trong một vòng lặp. Nói chung, chỉ một tiến trình giữ tài nguyên mới có thể giải phóng tài nguyên. Thông thường, một tiến trình sẽ không giải phóng tài nguyên cho đến khi quá trình xử lý hoàn tất.

Về mặt lý thuyết [23], deadlock có thể được xử lý bằng 3 cách phổ biến. Cách đầu tiên, chúng ta ngăn chặn hoặc phòng tránh deadlock, sau đó chúng ta có thể đảm bảo rằng hệ thống sẽ không bao giờ gặp deadlock. Cách thứ hai, chúng ta chấp nhận sự xuất hiện của deadlock, sau đó chúng ta phát hiện ra nó và khôi phục lại về trạng thái ban đầu.

Giải pháp cuối cùng, chúng ta tuyệt đối bỏ qua, coi như deadlock sẽ không bao giờ đến.

Để hiểu thêm về deadlock, chúng ta nên hiểu trạng thái của hệ thống, chúng ta có trạng thái an toàn và trạng thái không an toàn. *Trạng thái an toàn* [23] là nếu hệ thống có thể phân bổ tài nguyên cho từng tiến trình theo một số trình tự mà vẫn tránh được deadlock. Điều này có nghĩa là một hệ thống chỉ ở trạng thái an toàn nếu tồn tại một *chuỗi an toàn*. Chuỗi các quá trình $<P_1, P_2, \ldots, P_n>$ là một chuỗi an toàn cho trạng thái phát hiện hiện tại nếu đối với mỗi chuỗi P_i, các tài nguyên được yêu cầu bởi P_i vẫn có thể được thỏa mãn bởi các tài nguyên hiện có cộng với các tài nguyên do tất cả P_j nắm giữ, trong khi $j<i$. Trong tình huống này, nếu tài nguyên mà P_i cần không có sẵn ngay lập tức, thì P_i có thể đợi cho đến khi tất cả các P_j kết thúc. Khi chúng hoàn thành, P_i có thể lấy tất cả các tài nguyên cần thiết, hoàn thành nhiệm vụ được chỉ định, trả lại các tài nguyên đã được phân bổ và kết thúc. Khi P_i kết thúc, P_{i+1} có thể nhận được các tài nguyên cần thiết, v.v. Nếu không có trình tự nào như vậy tồn tại, thì trạng thái hệ thống được cho là *không an toàn*.

Trạng thái an toàn không hẳn là trạng thái deadlock. Nhưng trạng thái deadlock là một trạng thái không an toàn. Tuy nhiên, không phải tất cả các trạng thái không an toàn đều là trạng thái dẫn đến bế tắc. Chúng ta nên lưu ý rằng trạng thái không an toàn có thể dẫn đến deadlock. Nếu trạng thái là an toàn, hệ thống có thể tránh được trạng thái không an toàn (và chắc chắn là tránh được deadlock).



Hình 7 Không gian trạng thái an toàn, không an toàn và deadlock [23]

Deadlock là một vấn đề của máy tính và cũng là một vấn đề của hệ thống [22] [23]. Tiếp đó, deadlock trên điện toán đám mây cũng chính là deadlock trong hệ thống hoặc deadlock trong hệ thống phân tán hoặc deadlock trên điện toán lưới. Deadlock là một trạng thái xảy ra đồng thời và theo chương trình khi một tập hợp các tiến trình đi vào điều kiện chờ vô hạn. Với sự phát triển của học máy và phân tích dữ liệu, chúng ta có thể áp dụng loại kỹ thuật này để dự báo deadlock xảy ra trong môi trường đám mây và dự báo này có thể giúp chúng ta phân bổ tài nguyên trên đám mây, đặc biệt là các máy ảo của đám mây.

D. Học máy và mô hình hồi quy tuyến tính

Kỹ thuật học máy (ML) tạo ra mối quan hệ giữa các dữ liệu với nhau. Mối quan hệ này có được từ một quá trình học tập sau khi quan sát tập dữ liệu mà chúng ta biết đầu vào và đầu ra là gì. Với ML, tập dữ liệu bao gồm tập dữ liệu đào tạo và tập dữ liệu thử nghiệm. Tập dữ liệu đào tạo có thể là đầu vào để xây dựng mô hình hoặc mối quan hệ. Còn tập dữ liệu thử nghiệm có thể được sử dụng cho mục đích đánh giá kiểm tra chất lượng của mô hình. ML thường sử dụng dữ liệu lịch sử về các lần thực hiện trong quá khứ và xử lý các nhiệm vụ của yêu cầu như dữ liệu đào tạo. Các thông số tác vụ xử lý bao gồm mức độ sử dụng tài nguyên của đám mây, cụ thể là các máy ảo trên đám mây. Đầu ra cần tìm hiểu cho vấn đề nghiên cứu của bài viết này là mức độ sử dụng tài nguyên của các máy ảo, dẫn đến trạng thái không an toàn để dự báo sự xuất hiện của deadlock.

Dự báo mức sử dụng tài nguyên của request là một cách tiếp cận mới trong điện toán đám mây, nó có thể hiếm khi được tìm thấy bởi các công trình trước đây trong lĩnh vực hệ thống phân tán và học máy (ML). Với ML, chúng ta có thể có nhiều phương pháp và kỹ thuật khác nhau để dự báo mức độ sử dụng tài nguyên, nhưng những cách này có thể có độ chính xác khác nhau [24], [25]. Do tính chất đặc thù của tập dữ liệu, một thuật toán ML cụ thể có thể tốt hơn các thuật toán khác và ngược lại. Trong ML, có một số phương pháp phổ biến mà chúng ta có thể áp dụng để dự báo mức độ sử dụng tài nguyên, đó là Hồi quy tuyến tính [26], Cây hồi quy [27], Đóng gói bằng cách sử dụng cây hồi quy [28] và Mạng thần kinh nhân tạo [29]. Trong bài nghiên cứu này, sau khi đánh giá ngắn gọn các phương pháp ML phổ biến, với mục đích đáp ứng thời gian thực hiện theo thời gian thực, chúng tôi muốn áp dụng kỹ thuật Hồi quy tuyến tính cho dữ liệu lịch sử của quá trình xử lý request của máy ảo. Từ đó, đưa ra dự báo về mức độ sử dụng tài nguyên cho yêu cầu tiếp theo.

Hồi quy tuyến tính giả định sự phụ thuộc tuyến tính giữa đầu vào và đầu ra — trong trường hợp của chúng tôi là mức độ sử dụng tài nguyên với các biến được xem xét (đầu vào của tham số yêu cầu, VM với thời gian thực hiện). Hồi quy tuyến tính mô hình hóa đầu ra bằng cách sử dụng công thức $y = \vec{a}.\vec{x} + b$, trong đó y là đầu ra mong muốn và $\vec{x}$ là vector của các biến độc lập (tức là đầu vào). Hồi quy tuyến tính xác định các giá trị của $\vec{a}$ và b giúp giảm thiểu sai số trên một tập dữ liệu quan sát được. Phương pháp này là hợp lý để dự đoán các tác vụ xử lý, trong đó mức độ sử dụng tài nguyên liên quan tuyến tính đến các biến được xem xét [26]. Trong các tình huống không tồn tại sự tuyến tính giữa đầu vào và đầu ra, độ chính xác của phương pháp có thể không đạt yêu cầu.

Để đánh giá độ chính xác của Hồi quy tuyến tính, chúng ta có thể sử dụng rất nhiều thông số. Các thông số về sai số bao gồm: sai số tuyệt đối trung bình (MAE), sai số thiên vị trung bình (MBE), sai số tuyệt đối tương đối (RAE), sai số tỷ lệ phần trăm tuyệt đối trung bình (MAPE), sai số bình phương trung bình (MSE), sai số bình phương gốc (RMSE), sai số bình phương tương đối (RSE), sai số bình phương gốc trung bình chuẩn hóa (NRMSE), sai số bình phương trung bình gốc tương đối (RRMSE). Trong bài báo này, chúng tôi sử dụng thông số RAE để đo độ chính xác của mô hình hồi quy tuyến tính. RAE càng nhỏ thì độ chính xác của dự đoán càng cao.

Gọi y_i là giá trị sử dụng tài nguyên thực tế trong máy ảo chạy trên đám mây. Gọi $\hat{y}_i$ là giá trị dự đoán tương ứng của việc sử dụng tài nguyên. Để xác nhận tính chính xác của phương pháp tiếp cận của chúng tôi, sai số tuyệt đối tương đối (RAE) [30] được sử dụng làm thước đo để đánh giá. RAE được tính đơn giản bằng cách lấy tổng sai số tuyệt đối và chia cho hiệu số tuyệt đối giữa giá trị trung bình và giá trị thực tế.

$$RAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\sum_{i=1}^{n}|y_i - \bar{y}|}$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Trong đó, n là số lượng các dự đoán được tính toán.

$\bar{y}$ là giá trị trung bình của n giá trị thực.

Giá trị của RAE có thể nằm trong khoảng từ 0 đến 1. Một mô hình tốt sẽ có các giá trị gần bằng 0, với 0 là giá trị tốt nhất. Lỗi này cho thấy phần dư trung bình liên quan như thế nào đến độ lệch trung bình của hàm mục tiêu so với giá trị trung bình của nó.

IV. THUẬT TOÁN ĐỀ XUẤT

A. *Giới thiệu*

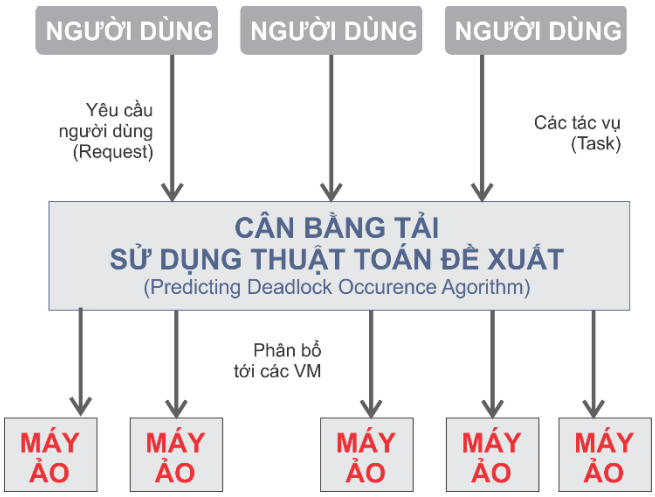
Với ý tưởng về trạng thái an toàn để đảm bảo khả năng đáp ứng tốt trên môi trường đám mây, bài viết đề xuất và áp dụng các thuật toán về dự báo và xác suất của deadlock để phòng tránh deadlock, cung cấp dịch vụ tốt hơn cho người dùng. Trong cách tiếp cận này, chúng tôi tập trung vào đám mây ảo hóa bằng cách sử dụng máy ảo và nhóm máy ảo thông qua máy chủ vật lý và trung tâm dữ liệu. Điều đó là hợp lý để tập trung vào deadlock trong máy ảo do tất cả các request đều được lưu lại và xử lý bởi các máy ảo này.

B. *Mô hình nghiên cứu*

Không thể áp dụng thuật toán phân bổ tài nguyên cho một hệ thống phân bổ tài nguyên có nhiều đặc trưng cho mỗi loại. Thuật toán tránh deadlock mà chúng tôi mô tả dưới đây có thể được áp dụng trong việc cân bằng tải trong môi trường đám mây. Thuật toán này với mục tiêu dự báo deadlock và khả năng xảy ra deadlock, từ việc phát hiện khu vực đang bận và đang hoạt động, sau đó phân bổ request cho các tài nguyên khác có khả năng phục vụ nhiều hơn. Thuật toán đề xuất được đặt tên là PDOA, viết tắt của Thuật toán Dự báo Deadlock xảy ra. Trong thuật toán này, có thể sử dụng các ứng dụng của kỹ thuật dự báo hồi quy tuyến tính, dự báo trạng thái không an toàn của các máy ảo dẫn đến khả năng xảy ra deadlock. Từ đó, thuật toán sẽ biết cách phân bổ yêu cầu cho các tài nguyên có sẵn để tránh deadlock.

Chúng tôi tiến hành PDOA trong mô hình nghiên cứu Đám mây giả định, dựa trên các thuộc tính của deadlock, deadlock xảy ra dưới các điều kiện sau:

- Khi các tác vụ / yêu cầu có thời gian xử lý quá lâu, có thể hết thời gian chờ và bị ngắt kết nối.
- Hoặc các tài nguyên đám mây va chạm nhau. Nói cách khác, có nhiều yêu cầu / tác vụ truy cập cùng một lúc, tạo ra một vòng lặp vô tận và dẫn đến sự cố. Các tài nguyên phổ biến dễ bị deadlock là Bộ nhớ (RAM), CPU và ổ cứng (Storage).

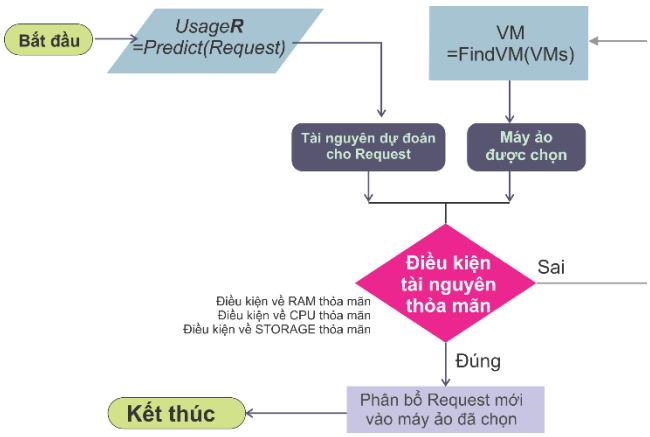


Hình 8 Mô hình nghiên cứu PDOA trên đám mây

C. *Thuật toán đề xuất PDOA*

Khi một request mới (đôi khi được gọi là cloudlet) được đưa vào đám mây, request đó có các thông số như kích thước request, số tệp, kích thước tệp, kiểu trả về, v.v. Tương ứng với các thuộc tính của request, chúng ta cần một lượng tài nguyên tương ứng để xử lý và phục vụ cho request này. Lượng tài nguyên này có thể không vượt quá tổng số tài nguyên của các hệ thống đám mây, và ở đây chúng là tài nguyên của các máy ảo. Khi các request của người dùng cần lượng tài nguyên nhất định, hệ thống phải xác định xem việc phân bổ các tài nguyên này có giữ cho hệ thống ở trạng thái an toàn hay không. Nếu trạng thái hệ thống là an toàn, tài nguyên sẽ được cấp phát. Nếu không, trong trạng thái không an toàn, tiến trình của request đang xử lý phải chờ cho đến khi một số tiến trình khác giải phóng đủ tài nguyên. Hoặc, hệ thống phải cấp phát tiến trình này cho tài nguyên của một máy ảo khác đủ lớn.

Để duy trì trạng thái an toàn và tránh deadlock, bài báo này tiếp cận bằng cách dự báo và tính toán khả năng xảy ra deadlock thông qua thông số của trạng thái không an toàn. Trạng thái không an toàn có thể là trạng thái mà một trong những tài nguyên chủ đạo đang cạn kiệt và có thể không sử dụng được. Dựa trên dữ liệu lịch sử đã xử lý trong quá khứ, chúng tôi sử dụng Hồi quy tuyến tính để xây dựng mô hình dự báo cho việc dự báo này. Tập dữ liệu bao gồm các thuộc tính Request (đầu vào) và mức sử dụng tài nguyên (đầu ra). Request đã chọn có các đầu vào là độ dài request (kích thước), kích thước đầu ra request, kích thước tệp và yêu cầu PES (số bộ xử lý cần thiết để xử lý). Các kết quả đầu ra được quan sát là mức sử dụng BỘ NHỚ (RAM), CPU & Ổ CỨNG, ở dạng phần trăm. Với kết quả đầu ra được tính toán và dự báo, request sẽ được phân bổ cho các máy ảo với VM có mức sử dụng tài nguyên phù hợp. Điều kiện phải được thỏa mãn là tổng mức sử dụng tài nguyên của mỗi máy ảo phải nhỏ hơn ngưỡng cho phép tương ứng. Nếu thỏa mãn 3 điều kiện: mức sử dụng Bộ nhớ, mức sử dụng CPU và mức sử dụng ổ cứng, VM sẽ được chọn để phân bổ yêu cầu.



Hình 9 Sơ đồ hoạt động của thuật toán đề xuất (PDOA)

Như hình 11, chúng tôi đề xuất PDOA với 03 chức năng chính: chức năng dự báo mức độ sử dụng tài nguyên bằng Hồi quy tuyến tính trên tập dữ liệu lịch sử, chức năng lựa chọn VM với trạng thái mức độ sử dụng tài nguyên tối thiểu và chức năng phân bổ.

Mã giả của PDOA (Thuật toán dự đoán deadlock xảy ra)

Đầu vào: Request

Đầu ra: Phân bổ Request vào VM tương ứng

```
1.  For each Request in CloudRequests
2.    isLocated = false;
3.    PredictedUsage = {RAM, CPU, STORAGE}predicted =
            Predict(Request);
            ← Chức năng 1: Dự báo sử dụng hồi quy trên tập
      dữ liệu lịch sử
4.    VM = FindVM(VMList);
            ← Chức năng 2: Chọn máy ảo có mức độ sử dụng
      tài nguyên phù hợp
5.    If isSatisfied(PredictedUsage, VM)
6.        AllocateRequestToVM(VM, Request);
            ← Chức năng 3: Cấp phát Request cho VM
7.        isLocated = true;
8.    End If
9.    If(!isLocated)
10.     VM = VMList.getSelectedVM();
11.     AllocateRequestToVM(VM, Request);
12.   End If
13. End For
```

Độ phức tạp thời gian trong trường hợp xấu nhất của PDOA được đề xuất sẽ bị chi phối bởi độ phức tạp của thuật toán hồi quy (Chức năng 1) nếu nó có độ phức tạp cao hơn so với các hàm khác. Như vậy, độ phức tạp của PDOA được đề xuất có thể được biểu diễn khái quát là $O(m * (f(n) + n))$, trong đó *'m'* là số lượng yêu cầu, *'n'* là số lượng máy ảo và *'f (n)'* là độ phức tạp của thuật toán hồi quy. Biết rằng độ phức tạp thực tế có thể khác nhau tùy thuộc vào các chi tiết cụ thể của thuật toán hồi quy và cấu trúc dữ liệu được sử dụng.

Thuật toán PDOA được đề xuất có thể được mô tả chi tiết hơn với các bước như sau:

Bước 1: Thuật toán dự đoán deadlock xảy ra (PDOA) sẽ thực hiện cân bằng tải bằng cách duy trì và cập nhật liên tục thông tin về trạng thái mức độ sử dụng tài nguyên của tất cả các VM tại bất kỳ thời điểm nào.

- Bộ cân bằng tải (LB) chứa thông tin về các máy ảo (VM) và Mức độ sử dụng của mỗi máy ảo. Mức độ sử dụng máy ảo được tính riêng cho BỘ NHỚ (RAM), CPU & Ổ CỨNG (bộ lưu trữ), tất cả đều được tính trên phần trăm sử dụng.

$$Usage = \{RAM, CPU, STORAGE\}$$

- Lúc đầu, tất cả các máy ảo (VM) được chú ý trong bảng "VMUsageList" với tất cả các số liệu Sử dụng bằng 0 *"VMUsageList"* với tất cả các số liệu Mức độ sử dụng bằng 0.

$$Usage = \{0, 0, 0\}$$

Bước 2: Bộ cân bằng tải nhận được một Request mới và cần xử lý tác vụ tương ứng. Tại đây, có thể dự đoán tài nguyên tương ứng sẽ được sử dụng cho request hiện tại. Thuật toán sử dụng *Mô hình hồi quy tuyến tính* được xây dựng dựa trên dữ liệu của các request trước đó (nếu khởi tạo, chúng tôi sử dụng tập dữ liệu gốc được chọn từ cân bằng tải round-robin trong đám mây) để dự đoán mức sử dụng tài nguyên tương ứng.

$$PredictedUsage = \{RAM, CPU, STORAGE\}_{predicted} = Predict(Request)$$

Bước 3: Bộ điều khiển lệnh trung tâm (CCC) truy vấn bộ cân bằng tải PDOA để phân bổ tiếp theo. Bằng cách biết mức sử dụng tài nguyên tương ứng cho yêu cầu hiện tại, nó sẽ được cấp phát cho VM nếu 3 điều kiện sau được đáp ứng đồng thời:

- $RAM: PredRAM\_UsageRequest + RAMUsageVM < ThresholdRAM$
- $CPU: PredCPU\_UsageRequest + CPUUsageVM < ThresholdCPU$
- $STORAGE: PredSTOR\_UsageRequest + STORUsageVM < ThresholdSTOR$

Ngưỡng RAM được đề xuất từ 90% đến 95% tổng mức sử dụng RAM trong VM. Con số cụ thể sẽ được chọn bởi sức mạnh và cấu hình của VM cùng trung tâm dữ liệu của nó. Đối với ngưỡng CPU, nó cao hơn một chút so với ngưỡng RAM. Chúng tôi sử dụng phạm vi từ 97% đến 98% cho ngưỡng CPU. Ngưỡng cuối cùng, ngưỡng LƯU TRỮ sẽ từ 96% đến 99%. Nếu vi phạm một trong 03 điều kiện, LB sẽ từ chối VM này và cần tìm VM khác để cấp phát.

Bước 4: Bộ cân bằng tải PDOA sẽ phát hiện và gửi ID máy ảo (VM) từ trên xuống dưới trong bảng máy ảo đang sử dụng. Sau đó, tìm ra máy ảo có Mức sử dụng nhỏ nhất và đủ điều kiện để trả lại ID máy ảo cho Bộ điều khiển lệnh trung tâm (CCC):

- Bộ điều khiển lệnh trung tâm (CCC) sẽ gửi yêu cầu đến máy ảo (VM) được xác định bởi ID đó để xử lý và thông báo cho bộ cân bằng tải PDOA về việc phân bổ đó. Bộ cân bằng tải PDOA sẽ cập nhật ID máy ảo (VM) mới được gửi và chờ yêu cầu mới từ Bộ điều khiển lệnh trung tâm (CCC).
- Trong trường hợp, nếu bảng trống (chưa có máy ảo nào được khởi tạo). Bộ cân bằng tải PDOA sẽ trả về -1 cho Bộ điều khiển lệnh trung tâm (CCC).
- Bộ điều khiển lệnh trung tâm (CCC) xếp hàng yêu cầu cho lần cấp phát tiếp theo.

Bước 5: Trên máy ảo (VM), sau khi yêu cầu đã được xử lý, bộ điều khiển trung tâm (CCC) nhận được phản hồi. Nó sẽ thông báo cho bộ cân bằng tải PDOA và bộ cân bằng tải

sẽ cập nhật lại bảng "*vmUsageList*" cũng như danh sách mức độ sử dụng của các máy ảo.

Bước 6: Nếu có nhiều request, bộ điều khiển trung tâm lặp lại Bước 3 và quá trình này sẽ được lặp lại cho đến khi tất cả các request được xử lý ổn thoả.

V. KẾT QUẢ VÀ BÀN LUẬN

Trong bài báo này, chúng tôi mô phỏng môi trường đám mây bằng cách sử dụng thư viện CloudSim [30] được phát triển bằng ngôn ngữ lập trình JAVA. Đồng thời, chúng tôi tích hợp thư viện Weka [31] sử dụng JAVA trong môi trường mô phỏng này để sử dụng chức năng LinearRegression đã được xây dựng.

A. Môi trường mô phỏng

Môi trường đám mây bao gồm 1 Trung tâm dữ liệu với 5 máy chủ chạy 5 máy ảo và chúng tôi tạo các yêu cầu ngẫu nhiên với các yêu cầu khác nhau để thực nghiệm thuật toán đề xuất PDOA.

Bảng 2. Cấu hình môi trường đám mây

Cấu hình trung tâm dữ liệu	Máy chủ và máy ảo
- Số lượng máy chủ trong trung tâm dữ liệu: 5 - kiến trúc: x86 - Hệ điều hành: Linux - VMM: Xen - Múi giờ: +7 GMT - Chi phí: 3.0 - Chi phí mỗi bộ nhớ: 0.05 - Chi phí mỗi bộ lưu trữ: 0.1 - Chi phí mỗi băng thông: 0.1	Mỗi máy chủ trong Trung tâm dữ liệu có cấu hình như sau: - CPU có 4 lõi, mỗi lõi là 1000 mips - RAM: 16384 (MB) - Bộ lưu trữ: 1000000 MB - Băng thông: 10000 MBps Mỗi máy ảo có cấu hình như sau: - Kích thước: 10000 MB - RAM: 512MB - Mips: 250 - Băng thông: 1000 Mbps - Pes số 1 - VMM: Xen

Bảng 3. Biến thiên của các request

Chiều dài (Kb)	Kích thước tập tin (Kb)	Kích thước đầu ra (Kb)	PEs
3000 ~ 1700	5000 ~ 45000	450 ~ 750	1

Trong mã nguồn mở CloudSim, thuật toán đề xuất được xây dựng bằng cách tạo lớp *PDOASchedulingAlgorithm*, lớp này kế thừa từ đối tượng *BaseSchedulingAlgorithm*, cập nhật một số phương thức và thuộc tính liên quan đến *getVMUsage* để tính toán mức sử dụng máy ảo và điều chỉnh các hàm tích hợp cho phù hợp với thuật toán được đề xuất. Hàm *predictRequestUsage* dùng để dự báo mức độ sử dụng tài nguyên của yêu cầu. Ngoài ra, hàm *getMostSuitableVM* được sử dụng để có được một máy ảo đáp ứng 3 điều kiện ngưỡng để phân bổ yêu cầu:

```
@Override
public void run()
public CondorVM getMostSuitableVM (double usagePercentage)
public double getVMUsage (CondorVM vm, int type)
public double predictRequestUsage (Cloudlet req, int type)
```

Tiêu chí đánh giá:

Thử nghiệm mô phỏng đám mây với các thông số trên và chạy thuật toán cân bằng tải có sẵn của CloudSim: Round Robin, MaxMin, MinMin và FCFS. Cài đặt cho chúng với cùng đầu vào và so sánh các đầu ra. Sử dụng các thông số về Thời gian phản hồi (trung bình, tối đa và tối thiểu), tổng thời gian xử lý (Makespan) để đánh giá hiệu năng của các thuật toán. Việc sử dụng các thuật toán Round Robin, MaxMin, MinMin và FCFS (đến trước phục vụ trước), để so sánh là rất quan trọng để đánh giá hiệu quả và hiệu suất của Thuật toán PDOA trong điện toán đám mây. Mỗi thuật toán này đại diện cho các chiến lược cân bằng tải khác nhau và bằng cách so sánh, chúng ta có thể hiểu rõ hơn về hiệu suất của PDOA trong các tình huống khác nhau.

Ngoài 4 thuật toán được chọn để so sánh với PDOA, còn có rất nhiều thuật toán cân bằng tải khác được sử dụng trong điện toán đám mây, chẳng hạn như Weighted Round Robin, Genetic Algorithm, Ant Colony Optimization, v.v. Chú ta có thể sử dụng một nhóm thuật toán đa dạng để so sánh giúp đảm bảo đánh giá toàn diện hơn về hiệu suất của PDOA. Việc so sánh PDOA với các thuật toán khác nhau này sẽ giúp xác định điểm mạnh và điểm yếu của PDOA, thể hiện khả năng tối ưu hóa thời gian phản hồi, thời gian xử lý và sử dụng tài nguyên, đồng thời làm nổi bật hiệu quả chi phí tiềm năng và độ chính xác của mô hình hồi quy tuyến tính được sử dụng để dự đoán. Bằng cách xem xét các yếu tố này, các nhà cung cấp dịch vụ đám mây có thể đưa ra quyết định sáng suốt về việc áp dụng PDOA hoặc cải tiến nó.

Thời gian phản hồi được dự báo của các máy ảo cũng như thời gian phản hồi dự báo của đám mây với sai số càng nhỏ, hiệu quả của thuật toán càng được đánh giá càng tốt. Ngoài ra, chi phí thấp hơn nên kỹ thuật cũng tốt hơn. Chúng tôi cũng sử dụng RAE (Sai số tuyệt đối tương đối) để quan sát và đánh giá độ chính xác của Mô hình hồi quy tuyến tính.

B. Kết quả mô phỏng và thực nghiệm

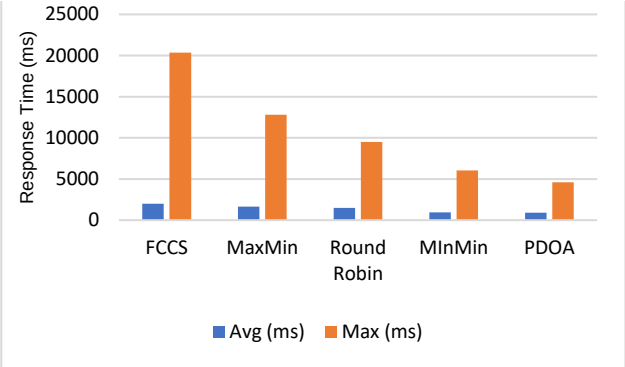
Chúng tôi tiến hành thử nghiệm mô phỏng với 03 trường hợp đầu vào khác nhau, số lượng yêu cầu là 24, 100 và 997. Dữ liệu để tạo yêu cầu chúng tôi sử dụng Epigenomics [32] được đề xuất bởi https://github.com/WorkflowSim/WorkflowSim-1.0.

Trường hợp 1 (*Epigenomics_24*): Thử nghiệm với 24 yêu cầu có sẵn, kết quả nằm trong Bảng 5 và Hình 12.

Trong trường hợp 1, chúng ta có thể thấy rằng thuật toán PDOA có vẻ tốt hơn một chút, có thời gian xử lý trung bình thấp nhất, nhưng do số lượng yêu cầu nhỏ nên sự khác biệt giữa các thuật toán là không đáng kể. Thuật toán FCFS là thuật toán tự nhiên nên thời gian xử lý luôn ở mức cao nhất.

Bảng 4. So sánh thời gian phản hồi của các thuật toán trong trường hợp 1

Trường hợp 1	*Tổng thời gian phản hồi (Response time)*		
	Thời gian trung bình (ms)	***Thời gian tối thiểu (ms)***	***Thời gian tối đa (ms)***
FCFS	1993.833	0.18	20363.49
MaxMin	1632.78	0.11	12825.62
Round Robin	1474.61	0.10	9487.84
MinMin	930.43	0.10	6027.47
PDOA	**915.34**	0.11	**4582.72**



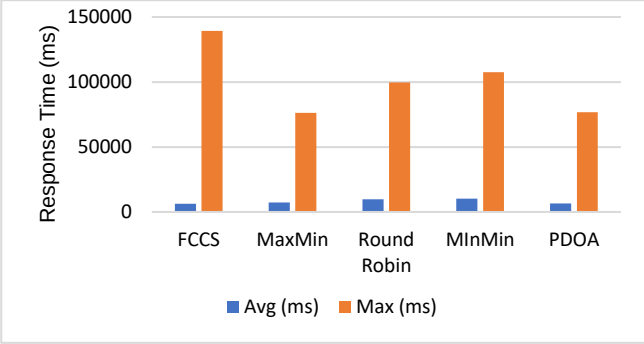
Hình 10 So sánh tổng thời gian phản hồi của các thuật toán trong trường hợp 1

Trường hợp 2 (*Epigenomics_100*): Thử nghiệm với 100 yêu cầu có sẵn, kết quả nằm trong Bảng 6 và Hình 13.

Trong trường hợp 2, chúng ta có thể thấy rằng thuật toán PDOA vượt trội hơn về thời gian xử lý trung bình, nhưng MaxMin lại có thời gian xử lý tối đa thấp nhất. Tuy nhiên, do số lượng yêu cầu ít nên sự khác biệt giữa các thuật toán là không đáng kể. Thuật toán FCFS là thuật toán tự nhiên nên thời gian xử lý luôn ở mức cao nhất.

Bảng 5. So sánh thời gian phản hồi của các thuật toán trong trường hợp 2

Trường hợp 2	*Tổng thời gian phản hồi (Response time)*		
	Thời gian trung bình (ms)	***Thời gian tối thiểu (ms)***	***Thời gian tối đa (ms)***
FCFS	6,352.1466	0.11	139,362.03
MaxMin	7,312.98	0.10	76,327.82
Round Robin	9,766.91	0.25	99,698.84
MinMin	10,158.29	0.11	407,527.17
PDOA	6,659.82	0.12	76,659.82



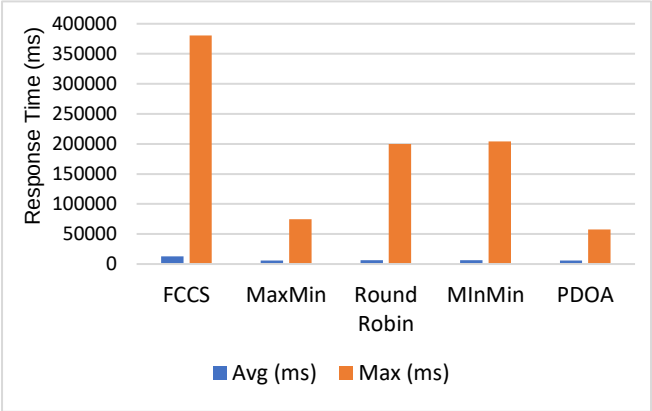
Hình 11 Biểu đồ so sánh tổng thời gian phản hồi của các thuật toán trong trường hợp 2

Trường hợp 3 (*Epigenomics_997*): Thử nghiệm với 997 dữ liệu yêu cầu có sẵn của CloudSim và kết quả như Hình 10 và Bảng 3:

Trong trường hợp 3, chúng tôi thấy rằng PDOA vượt trội hơn về thời gian xử lý trung bình cũng như thời gian xử lý tối đa là thấp nhất. Với số lượng yêu cầu tăng lên, chúng tôi thấy được tính ưu việt của dự đoán và sức mạnh xử lý của thuật toán dự đoán. Thuật toán FCFS là thuật toán tự nhiên nên thời gian xử lý luôn ở mức cao nhất.

Bảng 6. So sánh thời gian phản hồi của các thuật toán trong trường hợp 3

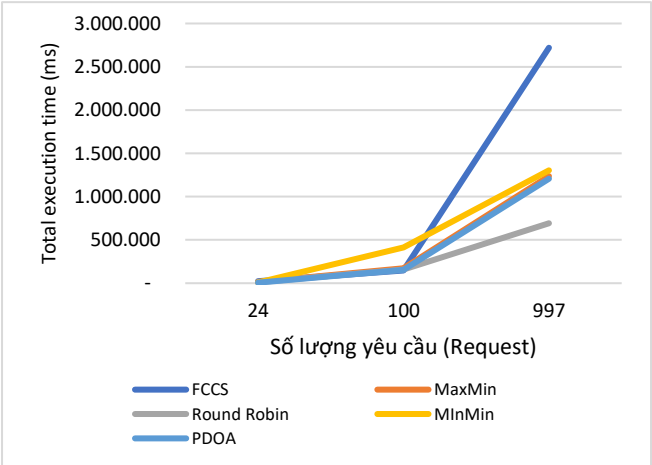
Trường hợp 1	*Tổng thời gian phản hồi (Response time)*		
	Thời gian trung bình (ms)	***Thời gian tối thiểu (ms)***	***Thời gian tối đa (ms)***
FCFS	12,927.32	0.13	380,441.42
MaxMin	6,087.25	0.10	74,719.00
Round Robin	6,476.19	0.10	200,066.70
MinMin	6,455.54	0.11	204,196.72
PDOA	5,940.80	0.11	57,559.75



Hình 12 Biểu đồ so sánh tổng thời gian phản hồi của các thuật toán trong trường hợp 3

Bảng 7. So sánh thời gian phản hồi của các thuật toán trong cả 3 trường hợp

	Tổng thời gian xử lý (Execution time in ms)		
	Epigenomics 24	***Epigenomics 100***	***Epigenomics 997***
FCFS	23,155	146,129	2,719,720
MaxMin	14,546	170,876	1,236,725
Round Robin	11,055	157,786	692,308
MinMin	7,593	414,381	1,303,421
PDOA	6,297	152,507	1,206,652



Hình 13 Biểu đồ so sánh tổng thời gian thực hiện của các thuật toán trong 3 trường hợp

Qua các thí nghiệm mô phỏng, thuật toán PDOA có kết quả tốt hơn một chút so với các thuật toán điển hình trong một số trường hợp dữ liệu đầu vào. Nó không hề thua kém so với các thuật toán có sẵn về thời gian phản hồi, tổng thời gian xử lý cũng luôn ít hơn các kỹ thuật khác.

Đánh giá Mô hình hồi quy tuyến tính trong PDOA

Bảng 8. So sánh RAE của PDOA trong cả 3 trường hợp

	Sai số RAE		
	Trường hợp 1 (24 request)	**Trường hợp 2 (100 request)**	**Trường hợp 3 (997 request)**
CPU	0.565217	0.489868	0.324261
BỘ NHỚ (RAM)	0.148717	0.286780	0.310741
BỘ LƯU TRỮ	0.002531	0.006759	0.495361

Để đánh giá độ chính xác của Mô hình hồi quy tuyến tính sử dụng trong PDOA, chúng tôi sử dụng sai số tuyệt đối tương đối (RAE) để xem xét mức độ chính xác cho bộ cân bằng tải sử dụng PDOA. Bảng 9 cho thấy, RAE xấu nhất và tốt nhất trong dự báo mức độ sử dụng CPU đều xảy ra ở trường hợp 1. Chúng ta có thể thấy rằng, RAE có thể chấp nhận đối với thực nghiệm này, nhưng trong các trường hợp nhìn chung chưa tốt do sự biến thiên của các request. Chúng ta có thể cập nhật và thay đổi bộ dữ liệu lịch sử cho phù hợp, để dự báo chính xác hơn.

VI. KẾT LUẬN

Trong bài báo này, chúng tôi tập trung vào nghiên cứu về deadlock và tránh deadlock bằng cách sử dụng các thuật toán dự báo để ngăn chặn sự xuất hiện của deadlock, nhằm đảm bảo cloud luôn ở trạng thái an toàn. Từ đó, bộ cân bằng tải có thể phân bổ các tác vụ cho máy ảo phù hợp mà không bị deadlock trong môi trường điện toán đám mây. Nhờ vào các thuật toán hiện nay và các nghiên cứu trước đây, bài nghiên cứu này đã có thể phân tích vấn đề về deadlock tốt hơn. Chúng tôi cũng có thể nhận ra những nhược điểm và ưu điểm của từng thuật toán, từ những vấn đề còn tồn tại trong nghiên cứu đã phân tích, đề xuất một thuật toán để cải thiện cân bằng tải trên đám mây PDOA. Quá trình nghiên cứu đã đạt được các mục tiêu sau:

a. Nhìn chung, chúng tôi đã xem xét vấn đề chính và các vấn đề về deadlock và deadlock trong môi trường đám mây, đặc biệt là cách tiếp cận deadlock trong kỹ thuật Cân bằng tải trong Điện toán đám mây.

b. Đề xuất một cách tiếp cận deadlock trong cân bằng tải trên môi trường đám mây: Thuật toán PDOA. Kết quả của PDOA được đề xuất đáp ứng các mục tiêu mong đợi như cải thiện thời gian phản hồi, tài nguyên sử dụng hiệu quả hơn và các máy ảo mạnh hơn để xử lý nhiều yêu cầu hơn. Ngoài ra, thuật toán còn giúp cải thiện hiệu quả cân bằng tải. Trong thực nghiệm mô phỏng cho thấy PDOA tốt hơn các thuật toán phổ biến hiện nay là Round Robin, MaxMin, MinMin và thuật toán tự nhiên FCFS. Thuật toán PDOA là hướng phát triển rất tiềm năng để cải thiện cân bằng tải trên đám mây, chúng ta có thể áp dụng thuật toán này trong môi trường thực tế và đám mây vật lý.

Tuy nhiên, bài nghiên cứu cũng có một số hạn chế: nghiên cứu vẫn chưa được ứng dụng trên đám mây vật lý. Thời gian phản hồi và thời gian xử lý đã được cải thiện, tuy nhiên chưa nhiều. Để cải thiện nghiên cứu để có thể áp dụng được nhiều kỹ thuật dự báo hơn nữa và tối ưu hóa các thông số, bộ dữ liệu huấn luyện cho thuật toán được đề xuất. Quan trọng nhất, phải áp dụng tất cả các kỹ thuật đó trong ứng dụng thực tế mà không chỉ là môi trường mô phỏng.

LỜI CẢM ƠN

Chúng tôi xin gửi lời cảm ơn sâu sắc tới các cơ sở: Học viện Công nghệ Bưu chính Viễn thông cơ sở Thành phố Hồ Chí Minh, Việt Nam và Trường Đại học Mở Thành phố Hồ Chí Minh, Thành phố Hồ Chí Minh, Việt Nam.

TÀI LIỆU THAM KHẢO

[1] Y.-F. Wen and C.-L. Chang, "Load balancing job assignment for cluster-based cloud computing," in *2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2014.

[2] G. Shao and J. Chen, "A load balancing strategy based on data correlation in cloud computing," in *Proceedings of the 9th International Conference on Utility and Cloud Computing*, 2016.

[3] S. Afzal and G. Kavitha, "Load balancing in cloud computing – A hierarchical taxonomical classification," *Journal of Cloud Computing Advanced System Application*., vol. 8, no. 1, 2019.

[4] A. A. A. Alkhatib, A. Alsabbagh, R. Maraqa, and S. Alzubi, "Load balancing techniques in cloud computing: Extensive review," *Advances in Science Technology and Engineering Systems Journal*, vol. 6, no. 2, pp. 860–870, 2021, doi: 10.25046/aj060299

[5] O. Mahitha and V. Suma, "Deadlock avoidance through efficient load balancing to control disaster in cloud environment," in *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, 2013.

[6] 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 *Advances in Intelligent Systems and Computing*, Cham: Springer International Publishing, 2015, pp. 285–295.

[7] S. Reveliotis and Z. Fei, "Robust deadlock avoidance for sequential resource allocation systems with resource outages," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, 2016.

[8] E. E. Ugwuanyi, S. Ghosh, M. Iqbal, and T. Dagiuklas, "Reliable resource provisioning using bankers' deadlock avoidance algorithm in MEC for industrial IoT," *IEEE Access*, vol. 6, pp. 43327–43335, 2018.

[9] H. H. C. Nguyen and V. T. Doan, "Avoid deadlock resource allocation (ADRA) model V VM-out-of-N PM," , *International Journal of Innovative Technology and Interdisciplinary Sciences*, Volume 2, Issue 1, pp. 98-107, 2019.

[10] S. Sherpa, A. Vicenciodelmoral, and X. Zhao, "Deadlock detection for concurrent programs using resource footprints," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion - UCC '19 Companion*, 2019.

[11] Y. Cai, C. Ye, Q. Shi, and C. Zhang, "Peahen: fast and precise static deadlock detection via context reduction," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022.

[12] J. Zhou, H. Yang, J. Lange, and T. Liu, "Deadlock prediction via generalized dependency," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2022.

[13] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: A big picture," *Journal of King Saud University – Computer and Information Sciences*, 2018.

[14] M. A. Shahid, N. Islam, M. M. Alam, M. M. Su'ud, and S. Musa, "A comprehensive study of load balancing approaches in the cloud computing environment and a novel fault tolerance approach," *IEEE Access*, vol. 8, pp. 130500–130526, 2020.

[15] I. C. Nidhi, Jain Kansal, "Cloud Load Balancing Techniques: A Step Towards Green Computing," *International Journal of Computer Science*, vol. 9, no. 1, 2012.

[16] S. Iqbal, M. L. M. Kiah, N. B. Anuar, B. Daghighi, A. W. A. Wahab, and S. Khan, "Service delivery models of cloud computing: security issues and open challenges: Cloud computing security,"

Security and communication networks, vol. 9, no. 17, pp. 4726–4750, 2016.

[17] "Load balancing in cloud computing," *International journal of engineering and advanced technology*, vol. 8, no. 6S3, pp. 2164–2166, 2019.

[18] N. Shah and M. Farik, "Static Load Balancing Algorithms In Cloud Computing: Challenges & Solutions," *International Journal of Scientific & Technology Research*, vol. 4, no. 10, pp. 365–367, 2015.

[19] M. F. Ali, "The study on load balancing strategies in distributed computing system," *International journal of computer science and engineering survey*, vol. 3, no. 2, pp. 19–30, 2012.

[20] M. Katyal and A. Mishra, "A comparative study of load balancing algorithms in cloud computing environment," 2014.

[21] B. S. Ahmed, K. Samsudin, and A. R. Ramli, "Architectural review of load balancing single system image," *Journal of computer science*, vol. 4, no. 9, pp. 752–761, 2008.

[22] A. Anand and M. Sethi, "Prevention of deadlock in a distributed computing environment," 8261280, 04-Sep-2012.

[23] M. S. Almhanna and R. M. Almuttairi, "Chapter 6 Methods for Handling Deadlocks," in *Operation System*, University of Babylon, 2019.

[24] R. F. da Silva, G. Juve, M. Rynge, E. Deelman, and M. Livny, "Online task resource consumption prediction for scientific workflows," *Parallel processing letters.*, vol. 25, no. 03, p. 1541003, 2015.

[25] A. Matsunaga and J. A. B. Fortes, "On the use of machine learning to predict the time and resources consumed by applications," *in 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010.

[26] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[27] S. L. Salzberg, "C4.5: Programs for machine learning by J. ross Quinlan. Morgan Kaufmann publishers, inc., 1993," *Machine learning*, vol. 16, no. 3, pp. 235–240, 1994.

[28] D. A. Monge, M. Holec, F. Železný, and C. G. Garino, "Ensemble learning of runtime prediction models for gene-expression analysis workflows," *Cluster Computing*, vol. 18, no. 4, pp. 1317–1329, 2015.

[29] S. Walczak and N. Cerpa, "Artificial Neural Networks," in *Encyclopedia of Physical Science and Technology*, Elsevier, 2003, pp. 631–645.

[30] D. Abdulkareem, Noor, Z. Jhanjhi, and A. Abdullah, "CloudSim 3.0.3 simulator step by step." Unpublished, 2021.

[31] "Weka 3 - data mining with open source machine learning software in java," *Waikato.ac.nz.* [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/. [Accessed: 27-Apr-2022].

[32] WorkflowSim-1.0: Wiki pages.

PDOA: PREDICTING DEADLOCK TO IMPROVE LOAD BALANCING ON CLOUD COMPUTING

Abstract: In the realm of Information Technology, cloud computing has become an indispensable solution for numerous internet-related challenges, catering to user demands effectively. However, the escalating global adoption of Cloud Computing, coupled with an increasing user base, can result in congestion at specific nodes, leading to imbalanced loads and potential system deadlocks. Addressing this issue, our paper introduces a novel algorithm, PDOA (Prediction of Deadlock Occurrence Algorithm), designed for the cloud environment's load balancer. PDOA harnesses the power of Machine Learning and prediction techniques, with a particular focus on the efficacy of the Linear Regression Model. By leveraging predictive capabilities, our algorithm can anticipate deadlock occurrences in Virtual Machines (VMs). This predictive insight allows us to strategically allocate available resources to fulfill all requests efficiently. To evaluate the effectiveness of PDOA, we implemented and tested the algorithm within the CloudSim simulation environment, incorporating the Weka library for Machine Learning techniques. Through comparative analysis with well-known algorithms like FCFS, RoundRobin, MaxMin, and MinMin, we found that PDOA outperforms these widely used alternatives, demonstrating its superiority in handling the challenges faced in modern cloud computing scenarios.

Keywords: Cloud Computing, Load Balancing, Deadlock Prediction, PDOA, Linear Regression

TIỂU SỬ CỦA CÁC TÁC GIẢ

Lê Ngọc Hiếu làm việc trong lĩnh vực CNTT với vai trò Kiến trúc sư Hệ thống CNTT từ năm 2010. Hiện anh đang là nghiên cứu sinh Tiến sĩ tại Học viện Công nghệ Bưu chính Viễn thông. Hiện tại, anh đang làm giảng viên CNTT tại trường Đại học Mở TP.HCM. Nghiên cứu chính của anh ấy là về điện toán đám mây và hiệu năng trên môi trường đám mây nhằm cung cấp các dịch vụ tốt hơn; bên cạnh đó, anh ấy còn nghiên cứu về giáo dục và kinh tế, đặc biệt là giáo dục trong lĩnh vực CNTT.

Email: hieu.ln@ou.edu.vn

Trần Công Hùng sinh ra tại Việt Nam năm 1961. Ông nhận bằng kỹ sư kỹ thuật điện tử viễn thông hạng nhất của Trường Đại học Bách khoa TP.HCM, Việt Nam năm 1987. Ông nhận bằng kỹ sư Tin học và Kỹ thuật máy tính của Trường Đại học Bách khoa TP.HCM năm 1995. Ông nhận bằng Thạc sĩ Kỹ thuật Viễn thông tại Trường Đại học Bách Khoa Hà Nội năm 1998. Ông nhận bằng Tiến sĩ tại Đại học Bách Khoa Hà Nội, Việt Nam, 2004. Lĩnh vực nghiên cứu chính của ông là các tham số và phương pháp đo hiệu suất B - ISDN, QoS trong mạng tốc độ cao, MPLS. Ông hiện là Phó Giáo sư Tiến sĩ, thuộc Khoa Công nghệ Thông tin II, Học viện Công nghệ Bưu chính Viễn thông tại TP.HCM, Việt Nam.

Email: conghung@ptithcm.edu.vn