HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

HƯỚNG TIẾP CẬN SWOT CHO CÂN BẰNG TẢI TRÊN ĐIỆN TOÁN ĐÁM MÂY

Chuyên ngành: **Hệ thống thông tin**

Mã số: **9.48.01.04**

LUẬN ÁN TIẾN SĨ KỸ THUẬT

HÀ NỘI - 2023

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

LUẬN ÁN TIẾN SĨ KỸ THUẬT

HƯỚNG TIẾP CẬN SWOT CHO CÂN BẰNG TẢI TRÊN ĐIỆN TOÁN ĐÁM MÂY

HÀ NỘI – 2023

LỜI CAM ĐOAN

Tôi cam đoan rằng luận án Tiến sĩ: *"Hướng tiếp cận SWOT cho cân bằng tải trên điện toán đám mây"* là công trình nghiên cứu của riêng tôi dưới sự hướng dẫn của thầy hướng dẫn, trừ những kiến thức, nội dung tham khảo từ các tài liệu đã được chỉ rõ.

Các kết quả, số liệu được trình bày trong luận án là trung thực, một phần đã được công bố trên các Tạp chí và Kỷ yếu Hội thảo khoa học chuyên ngành (danh mục các công trình đã công bố của tác giả được trình bày ở cuối Luận án), phần còn lại chưa từng được công bố trong bất kỳ công trình nào khác.

Không có sản phẩm/nghiên cứu nào của người khác được sử dụng trong luận án này mà không được trích dẫn theo đúng quy định.

TP. Hồ Chí Minh, ngày 30 tháng 06 năm 2023

Tác giả luận án

LỜI CẢM ƠN

Trong suốt quá trình học tập và nghiên cứu thực hiện luận án, ngoài nỗ lực của bản thân, tôi đã nhận được sự hướng dẫn nhiệt tình quý báu của quý Thầy Cô, cùng với sự động viên và ủng hộ của gia đình, bạn bè và đồng nghiệp. Với lòng kính trọng và biết ơn sâu sắc, tôi xin gửi lời cảm ơn chân thành tới:

Thầy PGS.TS Trần Công Hùng và Thầy TS. Lê Xuân Trường, đã tận tâm hướng dẫn và chỉ bảo cho tôi trên con đường học thuật và nghiên cứu, đồng thời quý Thầy cũng tạo điều kiện và giúp đỡ động viên tôi rất nhiều để tôi có thể từng bước hoàn thành được LATS này.

Ban Giám Đốc, Phòng đào tạo sau đại học và quý Thầy Cô đã tạo mọi điều kiện thuận lợi giúp tôi hoàn thành luận án.

Tôi xin chân thành cảm ơn gia đình, bạn bè, đồng nghiệp trong cơ quan đã động viên, hỗ trợ tôi trong lúc khó khăn để tôi có thể học tập và hoàn thành luận án. Mặc dù đã có nhiều cố gắng, nỗ lực, nhưng do thời gian và kinh nghiệm nghiên cứu khoa học còn hạn chế nên không thể tránh khỏi những thiếu sót. Tôi rất mong nhận được sự góp ý của quý Thầy Cô cùng bạn bè đồng nghiệp để kiến thức của tôi ngày một hoàn thiện hơn.

Xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 30 tháng 06 năm 2023

TÓM TẮT

Cân bằng tải trên đám mây là một thách thức cần nghiên cứu và cải tiến, với rất nhiều thuật toán không ngừng ra đời như Max-Min, Min-Min, Round-Robin, CLBDM, Active Clustering nhằm cải thiện hiệu năng của bộ cân bằng tải. Tuy đã có rất nhiều công trình đạt thành tựu đáng kể, nhưng việc sử dụng phương pháp dự đoán kết hợp học máy trên bộ dữ liệu cân bằng tải vẫn còn nhiều thách thức và hướng nghiên cứu. Do đó, luận án này phân tích cân bằng tải trong môi trường đám mây với ý tưởng từ cách tiếp cận SWOT (điểm mạnh, điểm yếu, cơ hội và nguy cơ), từ đó đưa ra đánh giá cân bằng tải với hai hướng tiếp cận: hướng tiếp cận bên trong và hướng tiếp cận bên ngoài. Với hướng tiếp cận bên trong, luận án tập trung phân tích các thuật toán cân bằng tải có liên quan đến các yếu tố bên trong của bộ cân bằng tải như thời gian phản hồi, thông lượng, các tham số khác và các đặc điểm bên trong khác. Với hướng tiếp cận bên ngoài, luận án xem xét các yếu tố bên ngoài bộ cân bằng tải, như hành vi của người dùng đám mây, cấu trúc mạng và môi trường địa lý của Internet, mức độ ưu tiên của các yêu cầu từ phía người dùng, v.v. Với mỗi hướng tiếp cận, luận án nghiên cứu các phương pháp học máy và khai phá dữ liệu tương ứng để cải thiện hiệu năng cân bằng tải trong môi trường điện toán đám mây.

Với ý tưởng trên, luận án đã đề xuất được 4 thuật toán cân bằng tải (MCCVA, APRTA, RCBA và ITA) theo hướng tiếp cận từ bên trong, 2 thuật toán cân bằng tải (PDOA và k-CTPA) theo hướng tiếp cận từ bên ngoài. Các thuật toán được cài đặt triển khai mô phỏng giả lập trên môi trường mô phỏng CloudSim và so sánh với các thuật toán cân bằng tải phổ biến hiện nay (Round Robin, Max Min, Min Min và FCFS). Tương ứng với mỗi thuật toán, xuất phát từ các góc độ phân tích khác nhau của bộ cân bằng tải, mà luận án sử dụng các thông số đo lường khác nhau để đánh giá mô phỏng giả lập (thời gian đáp ứng, thời gian thực hiện, speedup…). Kết quả từ việc mô phỏng đã chứng minh tính vượt trội và khả năng cải thiện hiệu suất của thuật toán học máy dự đoán trong việc tối ưu hóa bộ cân bằng tải trong điện toán đám mây.

ABSTRACT

Cloud load balancing is always a challenge that needs to be researched and improved, with many algorithms constantly emerging such as Max-Min, Min-Min, Round-Robin, CLBDM, Active Clustering to improve the performance of the load balancer. Although there have been many works with remarkable achievements, the use of predictive methods using machine learning on load balancing datasets still has many challenges and research directions. Therefore, this thesis analyzes the load balance in the cloud environment with ideas from the SWOT approach (Strengths, Weaknesses, Opportunities and Threats), thereby making a load balance assessment with two directions: internal approach and external approach. With an internal approach, the thesis focuses on analyzing load balancing algorithms related to the internal factors of the load balancer such as response time, throughput, other parameters and other internal characteristics. With an external approach, the thesis considers factors outside the load balancer, such as the behavior of cloud users, the network structure and geographical environment of the Internet, the priority of requests from user side, etc. With each approach, the thesis studies the corresponding machine learning and data mining methods to improve load balancing performance in the cloud computing environment.

With the above idea, the thesis has proposed 4 load balancing algorithms (MCCVA, APRTA, RCBA and ITA) in the direction of internal approach, 2 load balancing algorithms (PDOA and k-CTPA) in the direction approaching from the outside. These algorithms are installed and deployed experimentally on the CloudSim simulation environment and compared with current popular load balancing algorithms (Round Robin, Max Min, Min Min and FCFS). Corresponding to each algorithm, derived from different analysis angles of the load balancer, the thesis uses different measurement parameters for empirical evaluation (response time, execution time, speedup, etc.). Simulation results show the superiority and efficiency of machine learning prediction algorithms in improving the performance of load balancers in the cloud environment.

DANH MỤC CÁC TỪ VIẾT TẮT

Thuật ngữ	Diễn giải tiếng anh	Diễn giải tiếng việt
CC	Cloud computing	Điện toán đám mây
VM	Virtual Machine	Máy ảo
LB / CBT	Load Balancing	Cân bằng tải
Cloud	Cloud coputing environment	Môi trường điện toán đám mây
AI	Artificial Intelligence	Trí tuệ nhân tạo
ML	Machine Learning	Máy học
PDOA	Prediction Deadlock Occurance Algorithm	Thuật toán dự đoán xảy ra deadlock
ITA	Improved Throttle Algorithm	Thuật toán cải tiến Throttle
RCBA	Response Time Classification with Naïve Bayes Algorithm	Thuật toán phân lớp thời gian đáp ứng sử dụng Naïve Bayes
MCCVA	Makespan Classification & Clustering VM Algorithm	Thuật toán cân bằng tải phân lớp thời gian xử lý và gom cụm máy ảo
k-CTPA	kNN Classification Task Priority Algorithm	
QoS	Quality of Service	Chất lượng dịch vụ
IoT	Internet of things	Internet vạn vật
IP	Internet Protocol	Địa chỉ của các thiết bị trên mạng
SVM	Super Vector Machine	
FCFS	First Come First Serve	Đến trước xử lý trước

DANH MỤC CÁC KÝ HIỆU

Ký hiệu	Diễn giải tiếng việt	Trang
X_i	Thuộc tính của Request	48
T_i	Thời gian xử lý thứ i	49
RT_i	Thời gian đáp ứng thứ i	52
T_{new}	Ngưỡng thời gian mới	52
PRT_i	Thời gian đáp ứng dự báo của máy ảo i	53
AT_i	Chuỗi thời gian xử lý các tải tối đa ghi lại được của cloud	54
IT_i	Chuỗi thời gian xử lý các tải tối thiểu ghi lại được của cloud	55
P_i	Quá trình thứ i	62
Po	Mức tiêu hao năng lượng	74
CPU	Mức độ sử dụng CPU	74
RAM	Mức độ sử dụng RAM	74

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

MỤC LỤC

PHẦN MỞ ĐẦU

1. Giới thiệu

Điện toán đám mây (Cloud computing) [1] [2] [3] [4] ngày nay, đã trở thành một phần không thể thiếu trong cơ sở hạ tầng công nghệ thông tin, đặc biệt là nhờ vào sự linh hoạt cao và khả năng điều chỉnh dịch vụ tùy theo nhu cầu của người dùng. Điện toán đám mây cho phép mở rộng hoặc thu gọn tài nguyên một cách thuận tiện, đồng thời cũng không ngừng đổi mới để không chỉ tăng cường hiệu suất làm việc mà còn tối ưu hóa việc sử dụng tài nguyên. Điều này đảm bảo rằng các dịch vụ không chỉ chất lượng cao mà còn được cung cấp một cách kịp thời. Trong quá trình tối ưu hóa hiệu năng của điện toán đám mây, việc phân phối tải (Load Balancing) [5] là một phần thiết yếu, đóng vai trò trung tâm trong việc đảm bảo các hoạt động trên môi trường đám mây được thực hiện mượt mà và an ninh.

Trong lĩnh vực điện toán đám mây, việc quản lý cân bằng tải là một trong những vấn đề nan giải nhất. Để đám mây hoạt động trơn tru, cân bằng tải (Load Balancing) [6] [7] phải không ngừng đối mặt và giải quyết các yêu cầu từ người dùng một cách nhanh chóng và hiệu quả. Điều này đòi hỏi cơ chế cân bằng tải phải linh hoạt trong việc phân phối các yêu cầu đến các nốt hoạt động trên đám mây, sao cho luôn đảm bảo tải trọng được chia sẻ đều, tránh gây ra tình trạng quá tải cho bất kỳ nốt nào, cũng như tránh việc các tài nguyên không được sử dụng hiệu quả.

2. Lý do chọn đề tài

Khi công nghệ điện toán đám mây tiếp tục mở rộng cả về quy mô lẫn đa dạng các ứng dụng, việc phân phối tải trở nên quan trọng hơn bao giờ hết, việc phân phối này phải không ngừng được cải tiến để đáp ứng sự gia tăng cả về số lượng lẫn sự phức tạp của các yêu cầu do người dùng cloud tạo ra. Chính vì vậy, việc cân bằng tải trở thành một điểm nóng đối với cộng đồng nghiên cứu khoa học, với nhiều công trình đang được thực hiện trong nước và quốc tế nhằm tối ưu hóa hiệu năng của cân bằng tải trong môi trường điện toán đám mây, để nâng cao chất lượng dịch vụ đám mây. Trước đây, các nghiên cứu đã bước đầu đặt nền móng cho công cuộc chinh phục thách thức cân bằng tải như các thuật toán phân bổ yêu cầu người dùng như [6]: Max-Min [8], Min-Min [9], Round-Robin [10] [11], CLBDM [12], Active Clustering … Trong thời gian gần đây, các công trình chủ yếu mang tính riêng lẻ độc lập nhưng tô

điểm cho bức tranh tổng thể Load Balancing ngày một đa dạng và phong phú. Theo tài liệu [13] thì cân bằng tải được chia thành 2 nhóm, nhóm tĩnh (Static) và động (Dynamic). Đối với nhóm thuật toán tĩnh có các thuật toán CLBDM [12] (Central Load Balancing Decision Model) cải tiến từ Round-Robin; thuật toán do Kumar đề xuất là cải tiến từ việc đàn kiến đi qua các nốt mạng của cloud; thuật toán MapReduce… Đối với nhóm thuật toán động: thuật toán INS (Index Name Server) giảm thiểu trùng lắp và dư thừa trên mạng; thuật toán WLC [14] (Weight Least Connection) dựa vào số lượng kết nối qua các nốt mạng; Ngoài ra cải tiến của WLC là ESWLC (Exponential Smooth Forecast based on WLC); đối với dịch vụ FTP thì có DDFTP (Dual Direction Downloading); thuật toán LBMM (Load Balancing Min-Min) là cải tiến từ OLB (Opportunistic Balancing Algorithm). Ngoài ra có nhiều đóng góp cho cân bằng tải trên đám mây, tập trung cải tiến các thuật toán ban đầu, chủ yếu tập trung vào việc lập lịch cho các tác vụ (Task Scheduling), kiểm tra tình trạng của các tài nguyên (ở đây có thể là host, VM), và điều phối đường đi của các Request nhằm đạt được hiệu năng cao nhất. Các nghiên cứu đã đạt được nhiều thành tựu như : *Load Balancing in Cloud Computing using Stochastic Hill Climbing* [15]; *Enhanced Load Balancing Approach to Avoid Deadlocks in Cloud* [16]; User-Priority Guided Min-Min Scheduling Algorithm For Load Balancing in Cloud Computing [17]; Honey bee behavior inspired load balancing [18]; nhóm thuật toán dựa trên SLA (Service Level Agreement) [19]; thuật toán cân bằng tải dựa trên Response Time [20] [21]; cải tiến MaxMin, MinMin [22]; thuật toán cân bằng tải dựa trên Resource Scheduling [23]; thuật toán sủ dụng Meta-Heuristic để phân bổ tài nguyên [24]; Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing [25]. Bên cạnh đó, còn có một số nghiên cứu liên quan trong việc nâng cao hiệu năng của điện toán đám mây và các vấn đề mở rộng của đám mây [26], [27], [28], [29], [30].

Tuy nhiên việc nâng cao hiệu năng cân bằng tải trong điện toán đám mây [31] vẫn luôn là thách thức, là bài toán mà cần có lời giải tốt hơn, hiệu quả hơn, đặc biệt với sự đa dạng và phát triển ngày càng lớn mạnh của cloud (dự đa dạng về dịch vụ, phần mềm, nền tảng chạy trên các máy chủ / máy chủ ảo trên đám mây) cũng như nhu cầu sử dụng mỗi lúc một tăng của người dùng (về cả chất lượng và số lượng). Trong các công trình nghiên cứu về cân bằng tải trên đám mây [32] , chúng ta dễ dàng

nhận thấy việc sử dụng các phương pháp dự đoán kết hợp học máy và dữ liệu chưa được mô tả rõ nét.

Chính vì những lý do trên, luận án này phân tích cân bằng tải trong môi trường đám mây với ý tưởng từ cách tiếp cận SWOT (điểm mạnh, điểm yếu, cơ hội và nguy cơ), từ đó đưa ra đánh giá cân bằng tải với hai hướng tiếp cận: hướng tiếp cận bên trong và hướng tiếp cận bên ngoài. Với hướng tiếp cận bên trong, luận án tập trung phân tích các thuật toán cân bằng tải có liên quan đến các yếu tố bên trong của bộ cân bằng tải như thời gian phản hồi, thông lượng, các tham số khác và các đặc điểm bên trong khác. Với hướng tiếp cận bên ngoài, luận án xem xét các yếu tố bên ngoài bộ cân bằng tải, như hành vi của người dùng đám mây, cấu trúc mạng và môi trường địa lý của Internet, mức độ ưu tiên của các yêu cầu từ phía người dùng, v.v. Với mỗi hướng tiếp cận, luận án tập trung nghiên cứu các phương pháp / thuật toán liên quan đến trí tuệ nhân tạo mà điển hình là học máy (Machine Learning) kết hợp với khai phá dữ liệu, ứng dụng vào cân bằng tải, từ đó cải thiện tốt hiệu năng cân bằng tải trên môi trường điện toán đám mây. Cụ thể, đề tài luận án như sau:

Tên tiếng Việt là: ***"Hướng tiếp cận SWOT cho cân bằng tải trên điện toán đám mây"***

Tên tiếng Anh là: ***"The SWOT Approach for Load Balancing in Cloud Computing."***

3. Mục tiêu nghiên cứu

3.1 Mục tiêu tổng quát

Mục tiêu của luận án là dựa trên ý tưởng hướng tiếp cận SWOT từ đó đề xuất và xây dựng các phương pháp nâng cao hiệu năng cân bằng tải trong điện toán đám mây bằng cách ứng dụng / phát triển các thuật toán trí tuệ nhân tạo mà cụ thể là các thuật toán học máy với việc xử lý và phân tích dữ liệu cân bằng tải.

3.2 Các mục tiêu cụ thể

Phân tích các vấn đề liên quan hiệu năng cân bằng tải trên cloud bằng công cụ SWOT và từ đó đưa ra 2 hướng tiếp cận: tiếp cận từ bên trong và tiếp cận từ bên ngoài, nhằm đề xuất phát triển các thuật toán nâng cao hiệu năng cân bằng tải.

Với hướng tiếp cận từ bên trong, đề xuất xây dựng phương pháp / thuật toán ứng dụng một số thuật toán học máy (Machine Learning) vào bộ cân bằng tải, nhằm nâng cao hiệu năng cân bằng tải trên điện toán đám mây. Xây dựng cải tiến một số thuật toán cân bằng tải phổ biến hiện nay trên cloud. Cụ thể, đề xuất xây dựng bộ cân bằng tải dựa trên phương pháp dự báo các thông số theo thời gian như thời gian đáp ứng (Response Time), thời gian xử lý (Make span) để nâng cao hiệu năng cân bằng tải trên điện toán đám mây.

Với hướng tiếp cận từ bên ngoài, đề xuất xây dựng phương pháp dự báo deadlock hoặc khả năng xảy ra deadlock trên bộ cân bằng tải, đây cũng là yếu tố ảnh hưởng đến khả năng cân bằng tải, từ đó xây dựng bộ cân bằng tải luôn tránh được deadlock và nâng cao hiệu năng cân bằng tải. Đề xuất xây dựng thuật toán cân bằng tải theo góc độ hành vi người dùng bao gồm độ ưu tiên xử lý các tác vụ (task) / các yêu cầu (request) tương ứng của người dùng, từ đó phân bổ hệu quả nhất, nâng cao hiệu năng cân bằng tải trên điện toán đám mây.

4. Đối tượng và phạm vi nghiên cứu

- **Đối tượng nghiên cứu:**

Với mục tiêu nghiên cứu như trên, đối tượng nghiên cứu của đề tài như sau:

- Các yếu tố và tính chất của cân bằng tải trên điện toán đám mây: Cơ chế cân bằng tải, đặc tính hóa các tham số ảnh hưởng hiệu năng cân bằng tải (yêu cầu người dùng, kích thước yêu cầu, thời gian đáp ứng, thời gian tắc nghẽn, đặc tính của tải, xác suất phục vụ của yêu cầu người dùng …)
- Các kỹ thuật và phương pháp xử lý dữ liệu: có thể được áp dụng để tối ưu hóa quá trình cân bằng tải trên điện toán đám mây. Các kỹ thuật này bao gồm các phương pháp dựa trên xác suất, phân tích hồi quy tuyến tính, các phương pháp khác nhau của học máy từ giám sát và không giám sát đến học sâu, cũng như các phương pháp khai thác dữ liệu. Những phương pháp này mang lại tiềm năng lớn trong việc cải thiện hiệu suất của hệ thống cân bằng tải trong môi trường cloud.
- Các yếu tố tác động đến cân bằng tải: người dùng cloud và hành vi người dùng cloud: các dịch vụ và sự đa dạng của dịch vụ cloud, thời gian sử dụng,

nơi sử dụng truy cập cloud… Ở đây cụ thể hơn là các thông số của các yêu cầu (request) và tác vụ (task) dưới góc độ cloud và tài nguyên của cloud.

- Các phương pháp đánh giá hiệu năng và khả năng phục vụ của cân bằng tải và các thuật toán tương ứng, các phương pháp / thuật toán đã đề xuất thông qua: phân tích toán học (mathematical analysis), mô phỏng giả lập môi trường cloud và các phương pháp liên quan khác.

- **Phạm vi nghiên cứu.**

Các kỹ thuật và phương pháp nhằm nâng cao hiệu năng cân bằng tải trên môi trường điện toán đám mây.

Các thuật toán trí tuệ nhân tạo mà cụ thể là các thuật toán học máy và phân tích, khai phá dữ liệu từ cơ bản đến nâng cao, ứng dụng tốt vào cân bằng tải.

Về số quy mô của đám mây, luận án tập trung nghiên cứu môi trường đám mây nhỏ (từ 5 đến 10 máy ảo) và mô phỏng dịch vụ web là chủ yếu, với số lượng Request không quá 5000 trong thời gian 5 phút. Bên cạnh đó, luận án cũng nghiên cứu ảo hóa các máy ảo không bao gồm ảo hóa mức container và ảo hóa mức pooling.

5. Phương pháp nghiên cứu

- *Phương pháp luận:* Khảo sát kết quả nghiên cứu liên quan, phân tích các vấn đề quan trọng ảnh hưởng đến hiệu năng cân bằng tải, đề xuất các phương pháp / thuật toán nhằm nâng cao hiệu năng cân bằng tải toàn diện.
- *Phân tích dự trên tiếp cận SWOT cân bằng tải trên cloud:* Thực hiện việc đánh giá và phân tích SWOT cân bằng tải trong môi trường điện toán đám mây để từ đó phát triển một mô hình tiếp cận có khả năng cải thiện hiệu suất của cân bằng tải trên nền tảng đám mây.
- *Phương pháp đánh giá bằng các phương pháp toán học, mô phỏng, giả lập thực nghiệm:* đề xuất và phát triển thuật toán trên nền tảng toán học, xây dựng mô hình mô phỏng để đánh giá kết quả của phương pháp / thuật toán đã đề xuất, phân tích kết quả mô phỏng.

6. Những đóng góp chính của luận án

Phân tích cân bằng tải bằng công cụ SWOT và từ đó lấy ý tưởng nhằm đưa ra 2 hướng tiếp cận là tiếp cận từ bên trong và tiếp cận từ bên ngoài, để phân tích và đưa ra giải pháp trong việc nâng cao hiệu năng cân bằng tải trên điện toán đám mây:

- Đề xuất xây dựng các kỹ thuật / thuật toán mới trong cân bằng tải trên điện toán đám mây nhằm nâng cao hiệu năng cân bằng tải. Ngoài ra thông qua một số nghiên cứu, vận dụng hiệu quả & phát triển các thuật toán học máy vào cân bằng tải trên điện toán đám mây. Cụ thể thông qua đề xuất thuật toán MCCVA, thuật toán APRTA và RCBA, sử dụng một số kỹ thuật học máy để nâng cao hiệu năng cân bằng tải.
- Đề xuất xây dựng các bộ cân bằng tải mới trên việc xử lý các thông số thời gian, chuỗi thời gian như dự báo thời gian đáp ứng (response time), thời gian xử lý (makespan) nhằm nâng cao hiệu năng cân bằng tải trên điện toán đám mây. Cụ thể là thuật toán đề xuất APRTA sử dụng Response time để nâng cao hiệu năng cân bằng tải, MCCVA và RCBA sử dụng Makespan để nâng cao hiệu năng cân bằng tải.
- Đề xuất nghiên cứu cân bằng tải thông qua việc dự báo Deadlock và khả năng xảy ra deadlock trên môi trường cloud. Với việc dự báo này, giúp cho bộ cân bằng tải kiểm soát tài nguyên trên môi trường cloud tốt hơn. Cụ thể là thuật toán đề xuất PDOA, dự báo khả năng xảy ra deadlock thông qua tài nguyên của các máy ảo trên cloud để cân bằng tải. Đây là thuật toán cân bằng tải xuất phát từ tiếp cận bên ngoài bộ cân bằng tải.
- Đề xuất nghiên cứu cân bằng tải dưới góc độ người dùng trên cloud, khai thác các tính chất người dùng thông qua hành vi người dùng, độ ưu tiên của người dùng. Cụ thể là thuật toán đề xuất k-CTPA, đề xuất thuật toán lấy hành vi người dùng làm trung tâm cân bằng tải, thông qua đô ưu tiên dự báo của tác vụ tương ứng. Với tiếp cận này, thuật toán k-CPTA giúp nâng cao hiệu năng cân bằng tải từ tiếp cận bên ngoài.

7. Cấu trúc của luận án

Bên cạnh phần giới thiệu, mục lục, phần kết luận và phần tài liệu tham khảo, cấu trúc chính của luận án bao gồm bốn phần chính được tổ chức thành bốn chương:

Chương 1 là chương giới thiệu khái quát về điện toán đám mây và vấn đề cân bằng tải trên môi trường cloud. Một số khái niệm về cân bằng tải, các thuật toán điển hình trong cân bằng tải. Đồng thời giới thiệu một số các thuật toán học máy và thuật toán cân bằng tải phổ biến hiện nay. Dựa trên nền tảng đó, luận án đề xuất phát triển các thuật toán dựa trên học máy và phân tích dữ liệu để cải thiện hiệu quả của quá trình cân bằng tải trong môi trường điện toán đám mây..

Chương 2, nội dung được phân tích dựa trên lấy ý tưởng từ phương pháp SWOT về việc tiếp cận phân tích cân bằng tải trong môi trường điện toán đám mây, đánh giá hiệu suất của cân bằng tải, cùng với việc khảo sát các nghiên cứu đã công bố liên quan đến đề tài luận án. Đồng thời, chương này cũng đề cập đến việc tiếp cận nâng cao cân bằng tải từ ý tưởng của SWOT, đó là tiếp cận từ hai hướng khác nhau: một là hướng tiếp cận từ bên trong và hai là hướng tiếp cận từ các yếu tố bên ngoài.

Chương 3 trình bày các thuật toán đề xuất xây dựng bộ cân bằng tải theo hướng tiếp cận từ bên trong, cụ thể là các thuật toán cải tiến từ các thuật toán phổ biến và các thuật toán ứng dụng học máy và phân tích dữ liệu vào cân bằng tải, nhằm đạt hiệu năng cao hơn trên môi trường đám mây. Với chương này, luận án đã đạt được 4 thuật toán CBT mới MCCVA, APRTA, RCBA và ITA tương ứng với các công trình công bố [CT1], [CT2], [CT7] và [CT3].

Chương 4 trình bày các thuật toán đề xuất theo hướng tiếp cận từ bên ngoài. Bao gồm xây dựng bộ cân bằng tải kết hợp với việc kiểm soát deadlock và dự báo deadlock trên môi trường cloud; giải pháp cân bằng tải dưới góc độ hành vi người dùng bao gồm độ ưu tiên của tác vụ (task) / các yêu cầu (request), cụ thể là phân tích các thông số của request dưới góc độ người dung, từ đó ứng dụng các thuật toán phân lớp để đưa ra quyết định phân bổ xử lý, từ đó sử dụng hiệu quả nhất tài nguyên của cloud. Với tiếp cận từ bên ngoài, kết quả đạt được là 02 thuật toán bao gồm PDOA và k-CTPA tương ứng với công trình công bố [CT4 & CT5] và [CT6].

PHẦN NỘI DUNG

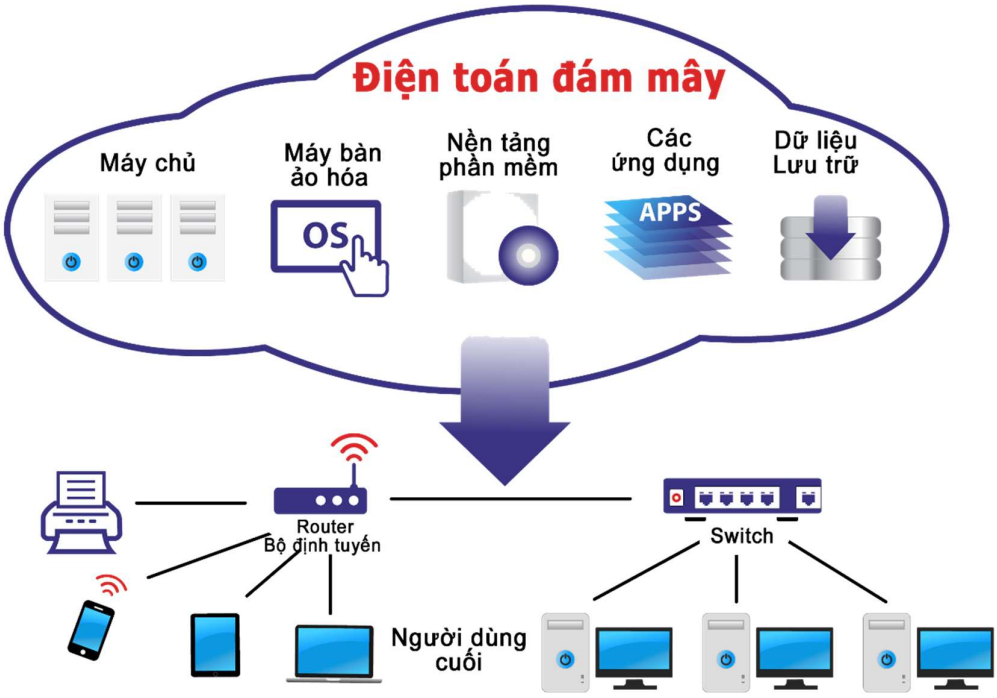
CHƯƠNG 1 – GIỚI THIỆU VỀ ĐỀ TÀI

1.1 TỔNG QUAN VỀ ĐIỆN TOÁN ĐÁM MÂY

Điện toán đám mây (cloud computing) [33] [34] [35] [36], còn được biết đến dưới tên gọi điện toán dựa trên máy chủ ảo, là hệ thống tính toán phục vụ theo yêu cầu và xây dựng trên nền tảng mạng kết nối toàn cầu Internet. Mô hình này cho phép việc chia sẻ tài nguyên, dữ liệu và phần mềm dưới hình thức dịch vụ thông qua một mạng công cộng, thường là Internet, tới các máy tính và thiết bị của người dùng. Trong mô hình điện toán đám mây, người dùng không cần phải biết hoặc quan tâm đến vị trí cụ thể của cơ sở dữ liệu hay trang web mà họ đang truy cập - mọi thứ diễn ra một cách minh bạch và không đòi hỏi hiểu biết kỹ thuật từ phía người dùng. Người dùng có thể truy nhập vào các dịch vụ này qua web browser, ứng dụng di động, hoặc máy tính thông thường. Tính hiệu quả trong sử dụng của người dùng cuối được tăng lên nhờ việc sử dụng phần mềm và cơ sở dữ liệu được lưu trữ trên server ảo trong điện toán đám mây, đặt tại các trung tâm dữ liệu. Các trung tâm dữ liệu là những không gian chứa đựng server, hệ thống lưu trữ, cùng với nguồn điện và các thiết bị liên quan như giá đỡ, dây cáp, v.v., được thiết kế để đảm bảo sự sẵn có và ổn định cao. Chúng còn được đánh giá dựa trên những yếu tố như khả năng mở rộng, tính linh hoạt, hệ thống làm mát, và khả năng hợp nhất server cũng như lưu trữ dữ liệu với mật độ cao.

Trong điện toán đám mây, có ba mô hình triển khai chủ yếu: đám mây công cộng, đám mây riêng tư và đám mây lai. Mô hình đám mây công cộng liên quan đến việc các nhà cung cấp dịch vụ đám mây cung cấp tài nguyên, nền tảng hoặc ứng dụng lưu trữ dưới dạng dịch vụ mở cho công chúng, có thể kèm theo chi phí hoặc không. Trong khi đó, đám mây riêng tư dành riêng các dịch vụ của mình cho một tổ chức cụ thể, thường được bảo vệ bởi tường lửa và dành cho việc sử dụng nội bộ. Đám mây lai là sự pha trộn giữa hai mô hình trên, cung cấp một số dịch vụ mở rộng và một số dành riêng. Bên cạnh đó, còn có mô hình đám mây cộng đồng, nơi mà nhiều tổ chức cùng chia sẻ tài nguyên đám mây. Về phía các mô hình dịch vụ, ba loại phổ biến là IaaS (Cơ sở hạ tầng như một Dịch vụ), PaaS (Nền tảng như một Dịch vụ), và SaaS

(Phần mềm như một Dịch vụ), mỗi loại phục vụ các nhu cầu khác nhau từ cung cấp tài nguyên máy chủ đến nền tảng phát triển ứng dụng và phần mềm sẵn sàng sử dụng.



Hình 1.1:*Mô hình điện toán đám mây* [37]

Trong vài năm trở lại đây, điện toán đám mây đã trở thành một trào lưu công nghệ hàng đầu thế giới, chứng kiến sự tăng trưởng đột phá về cả chất lượng và quy mô, cũng như đa dạng hóa các dịch vụ được cung cấp. Các tập đoàn công nghệ hàng đầu như Google [38], Amazon [39], Microsoft [40], đã dẫn đầu trong việc phát triển lĩnh vực này.

Điện toán đám mây cung cấp một mô hình mà các giải pháp công nghệ thông tin được triển khai như là dịch vụ thông qua Internet, giảm nhu cầu của người dùng về đầu tư vào nhân sự, công nghệ và cơ sở hạ tầng cho việc thiết lập hệ thống. Điều này giúp giảm thiểu chi phí và rút ngắn thời gian cần thiết cho việc triển khai, cho phép người sử dụng có thể chuyên tâm vào những hoạt động chính của mình. Không chỉ mang lại lợi ích cho người sử dụng, nhưng các nhà cung cấp dịch vụ cũng thu được lợi ích từ mô hình điện toán này.

Điện toán đám mây (Cloud Computing) [41] [37], [42] đang là hướng phát triển hàng đầu hiện nay, nối tiếp từ các hệ thống mạng và máy tính phân tán truyền thống bằng cách tích hợp tài nguyên tính toán, lưu trữ, nền tảng và dịch vụ đa dạng theo yêu cầu một cách linh hoạt và nhanh chóng, đồng thời cũng dễ dàng thu hồi tài

nguyên và giảm thiểu sự cần thiết cho sự tương tác trực tiếp với nhà cung cấp. Mô hình này chủ yếu dựa trên việc cung cấp dịch vụ ngay khi có yêu cầu (ondemand service); đảm bảo khả năng truy cập rộng rãi qua mạng từ nhiều loại thiết bị từ máy tính để bàn đến di động (broad network access); chia sẻ tài nguyên tính toán một cách động giữa nhiều khách hàng (resource pooling for multi-tenanci), và khả năng mở rộng linh hoạt năng lực tính toán để đáp ứng nhu cầu thay đổi từ thấp đến cao (rapidelasticity).

Điện toán đám mây, nhờ vào các kỹ thuật ảo hóa [43], sử dụng các dịch vụ qua mạng để cung cấp nguồn lực cơ bản, các nền tảng ứng dụng, phần mềm, và nhiều dịch vụ khác cho người dùng. Trong khuôn khổ của IaaS (Infrastructure as a Service), các nhà cung cấp tạo ra một môi trường ứng dụng phần mềm đầy đủ bằng cách kết hợp phần cứng, phần mềm, và thiết bị liên quan để tuân theo cam kết chất lượng dịch vụ đã thỏa thuận với khách hàng. Công nghệ máy ảo (Virtual Machine) thường xuyên được triển khai trong các data center, máy chủ cụm và cho nhiều ứng dụng khác. Điều này tạo điều kiện để nhiều hệ điều hành có thể hoạt động đồng thời trên một máy chủ đơn lẻ, đảm bảo rằng các dịch vụ được cung cấp một cách độc lập và đáng tin cậy, cũng như tăng cường khả năng tái sử dụng tài nguyên vật lý hiện có.

Điện toán đám mây [44] đang mở ra một lĩnh vực nghiên cứu với nhiều tiềm năng, hứa hẹn mang lại lợi ích về mặt kinh tế cho các doanh nghiệp khắp thế giới. Nó cung cấp một giải pháp hiệu quả cho việc lưu trữ và xử lý dữ liệu, cho phép các công ty truy cập vào cơ sở hạ tầng, các nền tảng và dịch vụ phần mềm theo yêu cầu một cách linh hoạt qua Internet.

Điện toán đám mây [45] đại diện cho một kiểu cung cấp dịch vụ trong lĩnh vực công nghệ thông tin, nối tiếp từ các hệ thống mạng trước đây, cho phép người sử dụng truy nhập vào các nguồn tài nguyên từ việc lưu trữ dữ liệu đến sử dụng phần mềm quản lý và xử lý thông tin phức tạp, đặc biệt là trên các nền tảng lớn như Google hay Facebook. Thay vì quản lý trực tiếp, người dùng chỉ cần kết nối với một thiết bị đầu cuối để sử dụng các nguồn tài nguyên này, trong khi hệ thống điện toán đám mây phụ trách việc lên lịch và xử lý các yêu cầu, từ việc quản lý thời gian chờ đợi đến thời gian xử lý và hoàn thành các tác vụ.

Điện toán đám mây [46] đang thay đổi bản chất của ngành công nghệ thông tin, thúc đẩy sự thay đổi trong việc sử dụng và cung cấp phần mềm và phần cứng. Nó

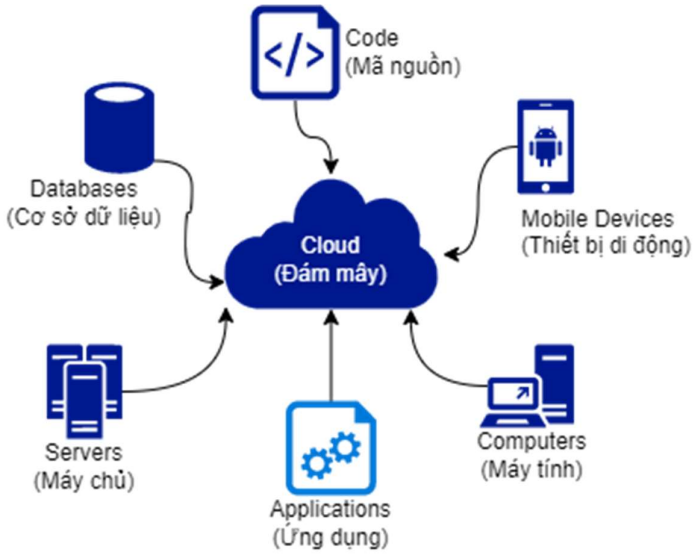
cho phép người dùng chỉ tập trung vào sản phẩm của họ mà không cần lo lắng về việc quản lý tài nguyên máy tính như băng thông, lưu trữ, hoặc ứng dụng phần mềm. Hợp đồng dịch vụ giữa nhà cung cấp và khách hàng xác định rõ các yêu cầu về chất lượng dịch vụ thông qua các thỏa thuận mức dịch vụ (SLA). Bằng việc tuân thủ SLA, nhà cung cấp cam kết đảm bảo chất lượng dịch vụ đã thỏa thuận. Sử dụng máy ảo cho phép tối ưu hóa tài nguyên phần cứng hiện có mà vẫn đảm bảo chất lượng dịch vụ. Để tránh giảm hiệu suất, các thuật toán phải có khả năng phát hiện và phân loại máy chủ quá tải và không sử dụng. Những thuật toán này cung cấp kế hoạch tối ưu cho việc di chuyển và phân phối máy ảo trong thời gian thực.

Là một mô hình tính toán mới, [47] Cloud Computing là một mô hình tính toán hiện đại, ra đời sau sự phát triển của các công nghệ như máy tính phân phối, điện toán lưới, lưu trữ dữ liệu mạng, công nghệ máy chủ cụm và tính toán song song. Trong môi trường Cloud Computing, do có sự đa dạng trong ứng dụng và không đồng nhất trong các nút nguồn, có một số máy tính chịu tải nặng trong khi những máy khác lại ít bị tải khi lưu lượng mạng và dữ liệu phát triển nhanh chóng. Điều này đòi hỏi phải có những phương pháp cân bằng tải hiệu quả để điều phối tải trên các máy chủ, giảm chi phí truyền dữ liệu và tăng cường khả năng sử dụng tài nguyên. Tuy nhiên, trong kỷ nguyên dữ liệu lớn và sự phát triển không ngừng của Cloud Computing, việc xử lý các công việc liên quan đến dữ liệu lớn thông qua máy ảo trong Cloud gặp phải thách thức do chi phí truyền thông tăng cao khi dữ liệu được di chuyển và tính toán giữa các máy chủ, điều này ảnh hưởng đến hiệu suất và sự hiệu quả trong sử dụng tài nguyên của hệ thống.

Cloud Computing [48] đánh dấu một bước tiến nổi bật và là một mô hình mới trong lĩnh vực tính toán. Phương pháp cân bằng tải này phân phối tài nguyên và phối hợp chúng với việc lập kế hoạch các công việc giữa các hệ thống phân tán. Cân bằng tải trong mô hình truyền thống gặp nhiều thách thức liên quan đến quá trình phân bổ tài nguyên trong môi trường Cloud. Nó ảnh hưởng đáng kể đến hiệu suất của hệ thống Cloud cũng như gặp khó khăn trong việc đánh giá hiệu quả do các biến số cân bằng tải đa dạng và đặc thù riêng của Cloud.

Trong thế giới ngày nay [49], điện toán đám mây cho phép chúng ta lưu trữ và truy cập phần cứng cũng như phần mềm từ bất kỳ đâu trên thế giới. Điều này đã tạo ra sự linh động lớn hơn cho yêu cầu về phần cứng. Nhờ đó, người dùng có thể sử

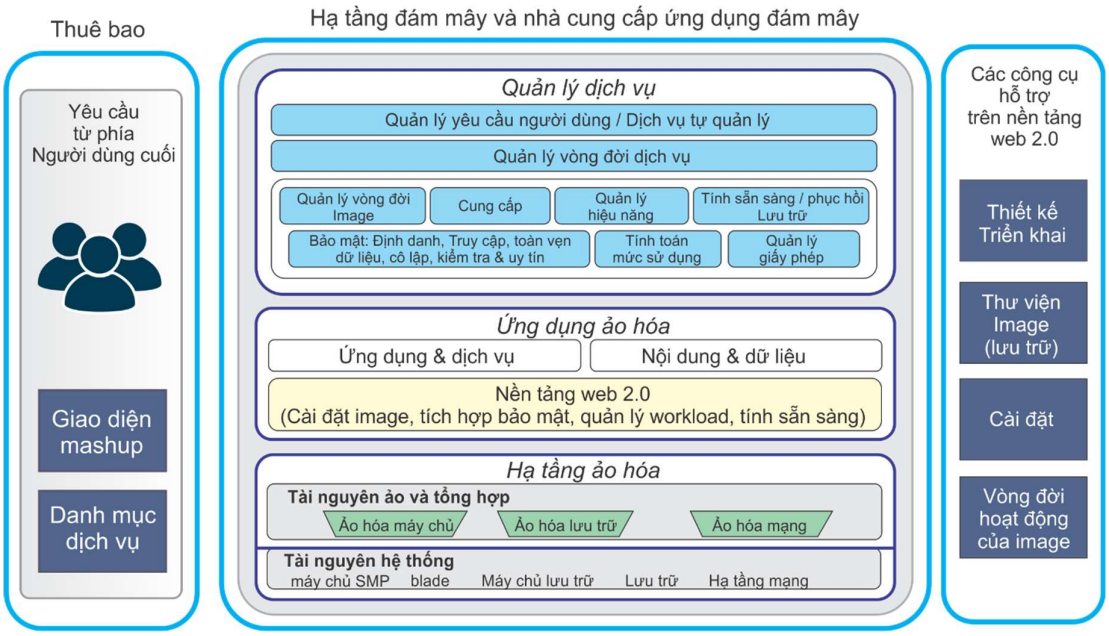
dụng nhiều tài nguyên theo nhu cầu và chỉ chi trả cho thời gian họ sử dụng một lượng tài nguyên nhất định, mô hình này được biết đến là dịch vụ theo mô hình pay-per-use, đẩy mạnh ngành IT hướng tới việc cung cấp dịch vụ Cloud mạnh mẽ hơn. Tương tự như một CPU bị lỗi nhiều lần, các doanh nghiệp với những tập hợp lớn của CPU hoặc máy tính vật lý được gọi là "đám mây". Những tập hợp này chỉ có một không gian và dung lượng bộ nhớ có hạn.



Hình 1. 2 ***Cung cấp tài nguyên đám mây*** **[44]**

Do đó, người dùng của dịch vụ đám mây phải thanh toán theo thời gian họ sử dụng không gian và dung lượng bộ nhớ từ tập hợp tài nguyên cụ thể được cung cấp cho họ. Khi người dùng yêu cầu tài nguyên như bộ nhớ, không gian lưu trữ, và băng thông, các nhà cung cấp cloud thực hiện việc này bằng cách phân bổ máy chủ vào hạ tầng theo nhu cầu của khách hàng. Quá trình cung cấp tài nguyên trên đám mây thường bao gồm việc tổng hợp máy tính vật lý thành máy ảo (VM) để tạo ra không gian lưu trữ ảo.

Cân bằng tải quản lý tài nguyên theo yêu cầu của khách hàng. Các phương pháp cân bằng trước đây đã cải thiện thời gian phản hồi và thời gian phục vụ của đám mây, nhưng chưa đảm bảo chất lượng dịch vụ (QoS) đúng cách. Để cải thiện QoS, các thông số của nó được tích hợp vào quá trình cân bằng tải. Băng thông, ví dụ, được xem xét như một thông số quan trọng, và việc quản lý các vấn đề như suy giảm và hiệu suất giúp định rõ ngưỡng giá trị chính xác hơn, giúp đảm bảo QoS hiệu quả hơn. Điều này giúp giảm thiểu việc cấp phát tài nguyên cho các máy vật lý vượt quá khả năng cung cấp của máy ảo và duy trì tính ổn định trong quá trình cung cấp dịch vụ.



Hình 1.3 ***Kiến trúc của điện toán đám mây*** **[47]**

Trong quá trình sử dụng tính toán tự động, việc kiểm soát chi phí tổng cộng là một thách thức quan trọng và được giải quyết thông qua việc xác định và quản lý tài nguyên thông qua các thuật toán quản lý quy mô. Đồng thời, cân bằng tải tài nguyên là một vấn đề quan trọng, ngay cả trong quá trình phát triển dịch vụ, và điều này được thực hiện thông qua sử dụng các thuật toán khác nhau.

1.2 TỔNG QUAN VỀ CÂN BẰNG TẢI TRONG ĐIỆN TOÁN ĐÁM MÂY

1.2.1 Giới thiệu về cân bằng tải

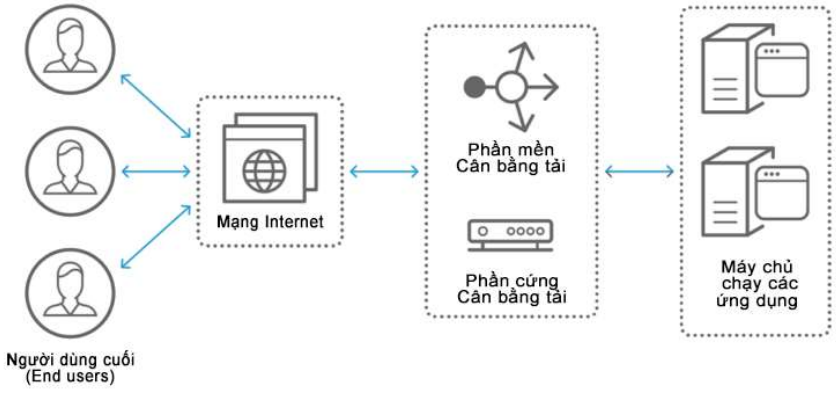
Ngày nay ngành công nghiệp CNTT đang phát triển mỗi ngày, nhu cầu về tài nguyên lưu trữ và tính toán cũng vậy. Một lượng lớn dữ liệu được tạo và trao đổi qua mạng, điều này đòi hỏi thêm nhu cầu về tài nguyên máy tính ngày càng nhiều. Cloud đã giúp các doanh nghiệp tận dụng lợi ích của tài nguyên điện toán được chia sẻ trên môi trường ảo hóa. Rất nhiều doanh nghiệp đã sử dụng các dịch vụ dựa trên đám mây ở dạng này hoặc dạng khác. Điều này đưa chúng ta đến khái niệm cân bằng tải trong điện toán đám mây.

Trong thời đại hiện nay, ngành công nghiệp Công nghệ thông tin (CNTT) đang phát triển rất nhanh, và sự tăng cường về nhu cầu sử dụng tài nguyên lưu trữ và tính toán không ngừng. Dữ liệu ngày càng lớn được tạo ra và trao đổi trên mạng, dẫn đến việc ngày càng cần đến tài nguyên máy tính. Mô hình điện toán đám mây [50] đã giúp

các doanh nghiệp tận dụng tài nguyên máy tính được chia sẻ trong môi trường ảo hóa. Rất nhiều doanh nghiệp [51] đã ứng dụng các dịch vụ của mình dựa trên công nghệ đám mây, với các hình thức khác nhau. Điều này đặt ra khái niệm quan trọng về việc cân bằng tải trong lĩnh vực điện toán đám mây.

Cùng với sự phát triển đáng kể của Internet, các trang web và ứng dụng trực tuyến thu hút một số lượng lớn người truy cập. Trong lúc cao điểm, khi lượng truy cập đột ngột tăng lên, thường xảy ra các vấn đề liên quan đến khả năng xử lý của máy chủ và hạ tầng mạng tại các vị trí tương ứng cụ thể. Vì vậy, việc cân bằng tải luôn đóng một vai trò quan trọng, giúp các máy chủ ảo hoạt động một cách đồng bộ và hiệu quả hơn, bằng cách phân phối tài nguyên một cách đều đặn mà không gặp tình trạng quá tải tại các điểm cụ thể.

Cân bằng tải là một giải pháp tổng thể nhằm phân phối đồng đều và hiệu quả lưu lượng truy cập giữa hai hoặc nhiều máy chủ có chức năng tương tự trong cùng một hệ thống. Điều này giúp hệ thống giảm thiểu tối đa tình trạng quá tải và sự cố của máy chủ. Khi một máy chủ gặp sự cố hoặc bị quá tải, cân bằng tải sẽ tự động phân phối công việc đến các máy chủ khác, đảm bảo thời gian hoạt động của hệ thống luôn ổn định và tối ưu hóa năng suất hoạt động tổng thể.



Hình 1.4 Mô hình Cân bằng tải trong điện toán đám mây theo NGINX [52]

Cân bằng tải là một trong những vấn đề quan trọng nhất trong môi trường phân tán. Với Cloud Computing được xem là một trong những nền tảng tốt nhất cho việc lưu trữ dữ liệu với chi phí thấp và khả năng truy cập dễ dàng qua internet, cân bằng tải trong môi trường điện toán đám mây đã trở thành một lĩnh vực nghiên cứu hấp dẫn và quan trọng. Mục tiêu của cân bằng tải là đảm bảo phân phối tài nguyên một

cách hợp lý để đáp ứng nhu cầu của người dùng và tối ưu hóa sử dụng tài nguyên. Trong môi trường điện toán đám mây hiện đại, có nhiều thách thức như vấn đề bảo mật và khả năng khắc phục lỗi, và nhiều nhà nghiên cứu đã đưa ra các thuật toán và kỹ thuật để nâng cao hiệu suất cân bằng tải, nhằm tìm ra các giải pháp tốt nhất cho hệ thống Cloud Computing.

Cân bằng tải [53] là việc phân phối đều lưu lượng truy cập giữa hai hoặc nhiều máy chủ có chức năng tương tự trong cùng một hệ thống. Bằng cách này, mục tiêu là giảm thiểu tối đa tình trạng quá tải của một máy chủ và ngăn ngừng hoạt động xảy ra. Trong trường hợp máy chủ gặp sự cố hoặc cần bảo trì, cân bằng tải sẽ tự động điều phối công việc của máy chủ đó cho các máy chủ khác, giúp tăng thời gian hoạt động của hệ thống lên mức cao nhất và nâng cao hiệu suất hoạt động tổng thể.

Phân tán dự đoán quá tải trong cân bằng tải [54] đã trở thành một phương pháp hứa hẹn gần đây. Nó bao gồm việc theo dõi tình trạng tắc nghẽn trên mỗi đường dẫn và phân phối dòng chảy trực tiếp đến các đường dẫn không tắc nghẽn. Cách tiếp cận này mang nhiều lợi ích thực tế. Về mặt phân phối, nó có khả năng mở rộng tốt hơn và xử lý lưu lượng truy cập nhanh hơn so với phương pháp tập trung. Về mặt dữ liệu, nó không phụ thuộc vào cấu trúc mạng của máy chủ lưu trữ và cung cấp lợi ích ngay lập tức cho tất cả lưu lượng truy cập khi triển khai. Khả năng hiển thị tắc nghẽn cuối cùng của nó cũng làm cho nó mạnh mẽ hơn mà không cần phải cấu hình lại máy điều khiển. Tuy nhiên, để thiết kế một giao thức cân bằng tải tắc nghẽn, chúng ta cần thông tin về tình trạng tắc nghẽn thời gian thực từ tất cả các đường đi giữa nguồn dòng chảy và điểm đến. Một cách tiếp cận đơn giản là sử dụng thông tin định hướng đường đi cuối cùng: một switch ToR duy trì các chỉ số tắc nghẽn đầu cuối cho tất cả các đường dẫn từ chính nó đến các thiết bị chuyển mạch ToR khác trong mạng. Tuy nhiên, không thể thu thập thông tin tắc nghẽn thời gian thực cho tất cả các đường dẫn này, vì sẽ không có đủ dòng chảy đồng thời xảy ra đi cùng với tất cả chúng cùng một lúc. Trong giai đoạn đầu, chỉ có nguồn và thiết bị chuyển mạch ToR đích tham gia để lựa chọn tốt nhất đường dẫn từ ToR đến tầng tổng hợp. Chuyển đổi nguồn ToR sẽ gửi số liệu tắc nghẽn của nó đến đích ToR, chúng sẽ kết hợp với các chỉ số tắc nghẽn để chọn con đường tốt nhất cho lớp tổng hợp. Trong giai đoạn thứ hai, tập hợp đã chọn sau đó chọn công tắc lõi tốt nhất theo một cách tương tự về tình trạng tắc nghẽn của bước nhảy thứ hai và thứ ba. Con đường quyết định lựa chọn sau đó được duy trì tại ToR

và tập hợp thiết bị chuyển mạch. Về cơ bản, hai giai đoạn lựa chọn đường dẫn sử dụng thông tin một phần đường dẫn để tìm đường tốt nhất cho dòng chảy. Bằng cách khai thác các tính chất cấu trúc của 3 tầng, lựa chọn đường dẫn hai giai đoạn làm giảm đáng kể sự phức tạp mà không gây ra việc sắp xếp lại gói tin cũng như không gây bất kỳ độ trễ nào.

Cân bằng tải là một chủ đề quan trọng được nghiên cứu một cách tích cực trong lĩnh vực trung tâm dữ liệu đám mây, với mục tiêu đảm bảo rằng tất cả tài nguyên máy tính có thể được sử dụng hiệu quả và nhanh chóng. Có nhiều loại thuật toán cân bằng tải, bao gồm cả thuật toán cân bằng tải tĩnh, động và chiến lược lập kế hoạch cân bằng tải. Thuật toán cân bằng tải tĩnh, ví dụ như round robin, thường chỉ sử dụng thông tin tĩnh và không thể thích ứng với tải thay đổi theo thời gian. Chúng thường đơn giản và dễ triển khai, nhưng không phải lúc nào cũng hiệu quả trong các trung tâm dữ liệu đám mây lớn và đa dạng về tài nguyên. Cân bằng tải động, mà được sử dụng trong các hệ thống phân phối máy tính song song, nhằm mục đích phân phối tải một cách thông minh để tránh hiện tượng quá tải và sử dụng không đồng đều tài nguyên trên hệ thống. Tuy nhiên, quá trình cân bằng tải động có thể tạo ra chi phí truyền thông bổ sung và cần được quản lý để không ảnh hưởng đến hiệu suất hệ thống. Một số thuật toán cân bằng tải có thể tối ưu hóa hiệu suất cục bộ, nhưng chúng không phải lúc nào cũng thích hợp cho tất cả trường hợp. Điều quan trọng là đảm bảo cân bằng tải và sử dụng tài nguyên vật lý một cách hiệu quả cho toàn bộ hệ thống đám mây. Tuy nhiên, cân bằng tải chỉ là một phần của việc quản lý trung tâm dữ liệu đám mây. Cần xem xét toàn bộ khía cạnh của hoạt động trung tâm dữ liệu đám mây để đảm bảo hiệu quả và tối ưu.

Cân bằng tải [55] có thể được chia thành 2 thể loại:

- Cân bằng tải cục bộ
- Tải toàn cầu

Cân bằng tải có hai loại chính: cân bằng tải cục bộ và cân bằng tải toàn cục. Cân bằng tải cục bộ được sử dụng để phân phối dự báo tải trong một trung tâm dữ liệu cụ thể. Nó đảm bảo rằng các yêu cầu từ khách hàng được phân phối đều đặn từ máy khách đến máy chủ để đáp ứng nhu cầu. Mặt khác, cân bằng tải toàn cục quản lý và kiểm soát yêu cầu từ khách hàng tự động đến các máy chủ qua nhiều trung tâm dữ liệu khác nhau. Nó xử lý lưu lượng trên cả hai hướng của gói truyền tải. Cân bằng

tải toàn cục thường phức tạp hơn, nhưng có ích trong việc quản lý việc truyền tải gói tin trên mạng trung tâm dữ liệu. Nó cũng đảm bảo tính khả dụng của hệ thống, đồng nghĩa với việc hệ thống vẫn hoạt động bình thường trong trường hợp có sự cố xảy ra.

1.2.2 Mục đích cân bằng tải

Cùng với sự phát triển của điện toán đám mây, việc chia sẻ dữ liệu và cung cấp tài nguyên trở nên dễ dàng hơn. Điều này cho phép người dùng tiếp cận nhiều tài nguyên mà họ cần và chỉ trả tiền cho những gì họ sử dụng. Tuy nhiên, một thách thức quan trọng trong môi trường đám mây là cân bằng tải, đặc biệt khi khối lượng dữ liệu trên nền tảng này tăng nhanh. Cân bằng tải giúp phân phối tải một cách hiệu quả thông qua các nút mạng, đảm bảo rằng không có nút nào bị quá tải. Điều này giúp tối ưu hóa sử dụng tài nguyên và cải thiện hiệu suất của hệ thống. Có nhiều loại tải khác nhau trên nền tảng đám mây, chẳng hạn như tải CPU, tải bộ nhớ và tải mạng. Cân bằng tải là quá trình xác định các nút mạng quá tải và chuyển tải đến các nút khác đang hoạt động ít tải hoặc không tải.

Theo tài liệu [56], trên nền tảng điện toán đám mây, cân bằng tải là quá trình quan trọng để phân phối công việc của các tải công suất lớn sang các tải nhẹ hơn, nhằm tối ưu hóa hiệu suất làm việc và tận dụng tài nguyên của đám mây một cách hiệu quả. Trong môi trường đám mây, cân bằng tải đòi hỏi việc phân phối lại công việc đang hoạt động liên tục giữa tất cả các nút mạng:

- Cân bằng tải đóng vai trò quan trọng trong việc đảm bảo rằng tài nguyên trong môi trường đám mây được phân phối hiệu quả nhất, hỗ trợ tính linh hoạt và khả năng mở rộng cao, đồng thời tránh hiện tượng kẹt cứng.
- Cân bằng tải là một kỹ thuật quản lý phân phối tài nguyên trên mạng, cho phép tối ưu hóa luồng dữ liệu với thời gian phản hồi tối thiểu. Nó giúp chia sẻ thông lượng giữa các máy chủ mà không gây trễ truyền.
- Trong môi trường đám mây, có nhiều thuật toán khác nhau để quản lý tải và phân phối dữ liệu. Chúng có thể được chia thành hai nhóm chính: BMHA (Batch mode Heuristic Allocation - phân bổ theo cơ chế từng đợt) và thuật toán Heuristic theo chế độ trực tuyến. BMHA thường hoạt động theo các đợt cố định để phối hợp công việc khi dữ liệu được gửi đến hệ thống.

- Ví dụ về các thuật toán thuộc nhóm BMHA bao gồm First Come First Served (FCFS), Round Robin (RR), Min Min và Max Min. Đối với nhóm thuật toán Online Mode Heuristic, công việc xử lý dữ liệu thường được thực hiện khi dữ liệu đến hệ thống. Trong môi trường đám mây, hệ thống thường không đồng nhất và hiệu suất xử lý của các máy chủ thay đổi nhanh chóng và đa dạng. Các thuật toán *Online Mode Heuristic* thường phù hợp và cho kết quả tốt hơn trong môi trường đám mây.
- Dự đoán và ước tính tải là yếu tố quan trọng, đòi hỏi so sánh với tất cả các tải, tính đồng đều của các hệ thống, hiệu suất của các hệ thống mục tiêu, và tương tác giữa các nút và công việc trong quá trình phát triển thuật toán cân bằng tải. Việc chọn nút phù hợp cũng quan trọng, bao gồm tải CPU và dung lượng bộ nhớ tổng hợp để định lượng tải của toàn bộ máy.
- Một ví dụ rõ ràng cho tầm quan trọng của cân bằng tải là trong trường hợp của các trang web. Nếu không có cân bằng tải, người dùng của các trang web lớn có thể dễ dàng phát hiện các vấn đề như tải chậm, thời gian đáp ứng dài dòng.

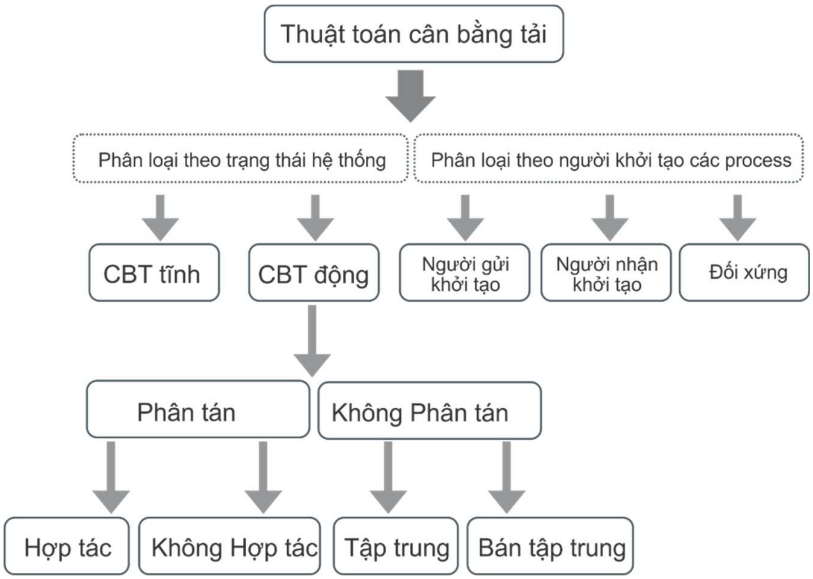
1.2.3 Phân loại cân bằng tải

Theo [21], các thuật toán cân bằng tải được chia thành 02 nhóm chính là :

- Các nhóm thuật toán cân bằng tải có thể được phân loại dựa trên trạng thái hệ thống, gồm hai nhóm chính là cân bằng tải tĩnh (Static) và cân bằng tải động (Dynamic).
- Các nhóm thuật toán cân bằng tải cũng có thể được phân loại dựa trên người khởi tạo quá trình xử lý, bao gồm ba nhóm nhỏ: khởi tạo bởi người gửi (Sender Initiated), khởi tạo bởi người nhận (Receiver Initiated), và khởi tạo bởi cả hai bên (Symmetric).

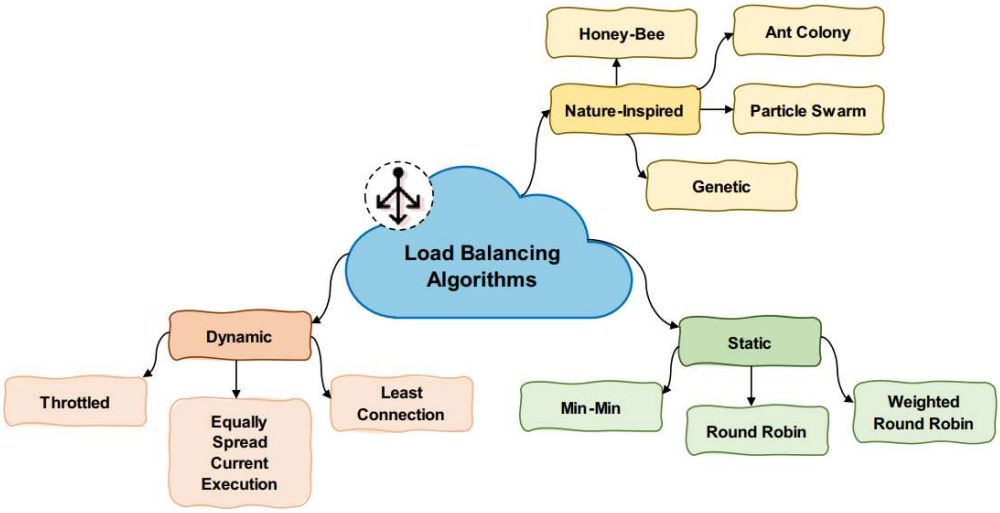
Tuy có nhiều loại thuật toán cân bằng tải, nhưng chúng chủ yếu tập trung vào trạng thái của hệ thống và thường được phân thành hai nhóm chính, bao gồm thuật toán cân bằng tải tĩnh và thuật toán cân bằng tải động:

- Tiếp cận tĩnh (phương pháp tĩnh): Đây là khi sự phân phối của lưu lượng được thực hiện trước và lưu lượng được chia thành các phần bằng nhau giữa các máy chủ.
- Tiếp cận động (phương pháp động): Tập trung vào các quyết định cân bằng tải được thực hiện khi lưu lượng đang diễn ra. Hướng tiếp cận này thường phù hợp cho các hệ thống phân tán như đám mây. Theo hướng này, ta có thể phân loại như sau:
    - Tiếp cận tập trung (Centralized approach): Mô hình này chỉ có một nút duy nhất chịu trách nhiệm quản lý và phân phối lưu lượng cho toàn hệ thống, trong khi các nút còn lại không tham gia vào quá trình quản lý tải.
    - Tiếp cận phân tán (Distributed approach): Ở đây, mỗi nút độc lập xây dựng một vector tải (load vector) riêng. Vector này thu thập thông tin từ các nút khác và dùng để đưa ra quyết định cục bộ. Phương pháp này rất phù hợp với môi trường đám mây.



Hình 1. 5 ***Phân loại thuật toán cân bằng tải theo hệ thống và tài nguyên*** **[21]**

Bên cạnh đó, Dalia Abdulkareem Shafiq và cộng sự [32] đã phân loại các thuật toán cân bằng tải theo tính chất của thuật toán như sau:



Hình 1. 6 ***Phân loại thuật toán cân bằng tải theo tính chất thuật toán*** **[32]**

1.2.4 Đo lường cân bằng tải

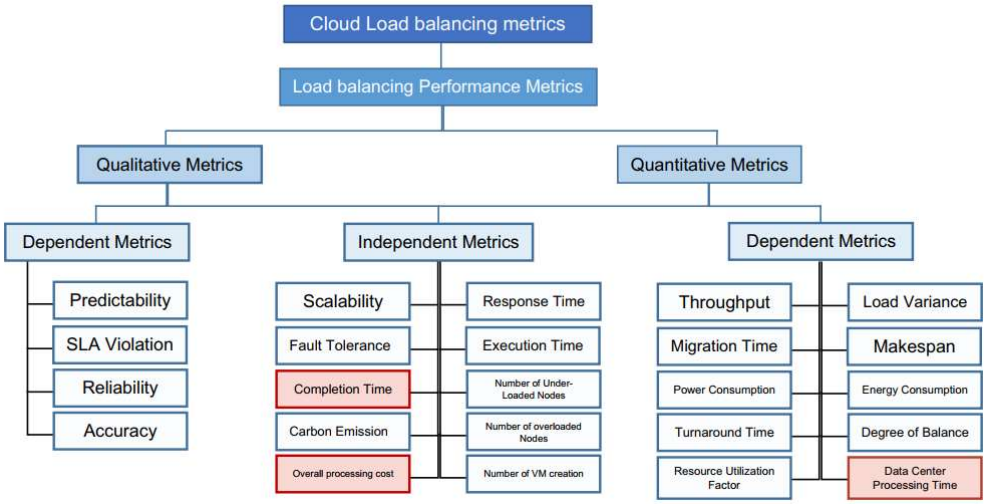
Đo lường cân bằng tải, có rất nhiều tham số để đo lường hiệu năng cân bằng tải trên môi trường điện toán đám mây. Một vài tham số cơ bản bao gồm:

- Thông lượng (Throughput) [57] [58]: Được sử dụng để đo lường số lượng tác vụ đã hoàn thành trong hệ thống. Một thông lượng cao đồng nghĩa với hiệu suất làm việc tốt hơn của hệ thống.
- Dung sai lỗi (Fault Tolerance) [57] [58]: Đề cập đến khả năng của hệ thống khôi phục sau khi xảy ra sự cố. Cân bằng tải cần có khả năng tính toán và xử lý sự cố một cách tốt nhất.
- Thời gian di dời (Migration Time) [57] [58]: Là tổng thời gian cần thiết để di chuyển tác vụ hoặc tài nguyên từ một nút sang nút khác trong hệ thống. Để tối ưu hóa hiệu suất, thời gian di dời cần được giảm thiểu.
- Thời gian đáp ứng (Response Time) [57] [58]: Là thời gian trung bình mà thuật toán cân bằng tải phản hồi một tác vụ trong hệ thống. Để cải thiện hiệu suất, thời gian đáp ứng cần được làm ngắn hơn.
- Thời gian thực hiện (Makespan -MS) [57] [58]: Tổng thời gian cần thiết để hoàn thành một tập hợp các tác vụ và phân bổ tài nguyên cho chúng trong hệ thống. Đây là một yếu tố quan trọng trong việc lập kế hoạch và quản lý tác vụ

trong môi trường đám mây. Thời gian thực hiện cần đạt được mức tối thiểu để đảm bảo hiệu suất tốt nhất.

- Khả năng mở rộng (Scalability) [57] [58]: Là khả năng của thuật toán cân bằng tải để xử lý đồng thời một lượng lớn các nút trong hệ thống. Để cải thiện hiệu suất, khả năng mở rộng cần được cải thiện.

Bên cạnh các tham số trên, cân bằng tải trên môi trường đám mây còn rất nhiều cách đo lường khác thông qua các tham số khác. Theo [32] , tổng hợp các tham số đo lường hiệu năng cân bằng tải như sau:



Hình 1. 7 ***Các tham số đo lường cân bằng tải*** **[32]**

1.2.5 Các chính sách trong cân bằng tải

Có rất nhiều chính sách được sử dụng trong các thuật toán cân bằng tải, bao gồm:

- Chính sách thông tin (Information policy): Định rõ loại thông tin cần thu thập và thời gian thu thập.
- Chính sách kích hoạt (Trigger policy): Xác định thời điểm mà các hành động cân bằng tải được thực hiện và ghi nhận tải.
- Chính sách về loại tài nguyên (Resource type policy): Xác định các loại tài nguyên nào sẽ luôn sẵn sàng để phục vụ trong quá trình cân bằng tải.
- Chính sách về vị trí (Location policy): Sử dụng kết quả từ chính sách về loại tài nguyên để xác định vị trí thích hợp cho máy chủ hoặc người nhận.

- Chính sách lựa chọn (Selection policy): Xác định tác vụ nào sẽ được chuyển từ nút quá tải sang nút trống.

1.2.6 Các mục tiêu chính của thuật toán cân bằng tải

- Hiệu quả chi phí (Cost effectiveness):
- Cân bằng tải hệ thống giúp đảm bảo hoạt động hiệu suất cao với mức chi phí thấp. Chi phí bao gồm chi phí vận hành và chi phí thời gian của cloud, vì vậy luận án tập trung vào chi phí thời gian và chi phí tài nguyên của cloud để nâng cao hiệu năng cân bằng tải.
- Khả năng mở rộng & tính linh hoạt (Scalability and flexibility): Kích thước tổng thể của hệ thống cân bằng tải có thể thay đổi theo thời gian và các thuật toán cân bằng tải sẽ giúp hệ thống quản lý và xử lý các biến đổi này. Chính vì vậy, thuật toán cũng có tính linh hoạt và khả năng mở rộng. Với đặc điểm này của cloud, luận án tập trung vào mô hình ảo hóa, là cân bằng tải động trên các máy ảo, dễ dàng đáp ứng tính mở rộng và tính linh hoạt, hoặc mở pool mới hay cân bằng tại nhóm máy ảo như điện toán cạnh (Edge computing), điện toán sương mù (Fog Computing).
- Tính ưu tiên (Priority): Sắp xếp ưu tiên các tác vụ hoặc các nguồn tài nguyên giúp quyết định xử lý những công việc quan trọng hơn trước. Điều này đồng nghĩa với việc tác vụ có độ ưu tiên cao hơn sẽ được xử lý trước. Với tính ưu tiên, luận án đứng dưới góc độ hành vi người dùng, phân tích và mô tả tính ưu tiên của các tác vụ, từ đó đưa ra tiêu chí cân bằng tải tương ứng.

1.2.7 Một số thuật toán cân bằng tải phổ biến

Có rất nhiều thuật toán cân bằng tải [53], [26] giúp giải quyết thông lượng tốt và giảm thời gian đáp ứng trên môi trường cloud. Mỗi thuật toán đều có những lợi ích riêng:

- *Thuật toán phân bố tác vụ dựa trên LB:* Thuật toán này sử dụng một cơ chế hai mức để phân bổ tác vụ. Ban đầu, nó ánh xạ các tác vụ đầu tiên tới các máy ảo, sau đó ánh xạ tất cả các máy ảo tới các tài nguyên của hệ thống. Thuật toán này tối ưu hóa việc sử dụng tài nguyên và cải thiện thời gian đáp ứng.

- *Thuật toán cân bằng tải cơ hội (Opportunistic Load Balancing - OLB):* OLB giữ cho các nốt mạng trong trạng thái làm việc và giao tác vụ không có thứ tự đến các nốt hữu ích. Nó đơn giản nhưng có thể dẫn đến thời gian hoàn thành lớn.
- *Thuật toán Round Robin (RR):* Thuật toán này chia đều tác vụ cho các bộ xử lý theo thứ tự vòng Robin. Tuy nhiên, có thể dẫn đến tải không cân bằng nếu các bộ xử lý khác nhau có hiệu suất xử lý khác nhau.
- *Thuật toán ngẫu nhiên hóa (Randomized):* Sử dụng xác suất để chọn máy tính xử lý tác vụ. Hoạt động tốt khi tải đồng đều, nhưng không hiệu quả khi tải không đều.
- *Thuật toán Min-Min và Max-Min:* Min-Min bắt đầu bằng việc tìm thời gian nhỏ nhất hoàn thành các tác vụ và sau đó phân tác vụ dựa trên thời gian nhỏ nhất này. Max-Min tương tự, nhưng tìm thời gian tối đa trước. Cả hai thuật toán này được sử dụng để cân bằng tải trong môi trường đám mây.
- *Thuật toán hành vi Tìm kiếm của Ong Mật (Honeybee Foraging Behavior):* Lấy ý tưởng từ hành vi tự tổ chức trong tự nhiên, thuật toán này giúp cải thiện hiệu suất hệ thống thông qua các hành động cục bộ.
- *Thuật toán gom cụm động (Active Clustering):* Gom nhóm các nốt mạng có cùng loại và cho họ làm việc cùng nhau, giúp tối ưu hóa sử dụng tài nguyên và cải thiện thông thường.
- *Thuật toán so sánh và cân bằng (Compare & Balance):* Sử dụng xác suất để chọn máy tính để kiểm tra và so sánh tải, sau đó điều chỉnh để cân bằng tải.
- Thuật toán Lock-free: Tránh sử dụng cùng lúc các vùng nhớ để nâng cao hiệu suất làm việc trong môi trường đám mây với nhiều luồng chạy cùng lúc.
- *Thuật toán đàn kiến (Ant Colony Optimization):* Dựa trên cách tiếp cận đa tác nhân để tối ưu hóa các vấn đề trong hệ thống.
- *Thuật toán thời gian đáp ứng ngắn nhất đầu tiên (Shortest Response Time First):* Ưu tiên thực hiện các tác vụ có thời gian đáp ứng ngắn nhất trước.
- *Thuật toán lấy mẫu ngẫu nhiên (Based Random Sampling):* Xây dựng đồ thị để mô tả các tải và tài nguyên của các nốt, sau đó sử dụng xác suất để phân bổ tác vụ.

- Các thuật toán này có ứng dụng khác nhau và phù hợp với các tình huống cụ thể trong môi trường đám mây để đảm bảo cân bằng tải hiệu quả.

1.3 MỘT SỐ THUẬT TOÁN AI ỨNG DỤNG VÀO CÂN BẰNG TẢI

1.3.1 Tổng quan một số thuật toán AI

Có một cách phân nhóm các thuật toán AI dựa trên chức năng, như được đề cập trong tài liệu [59], bao gồm: *nhóm Regression Algorithms* bao gồm Linear Regression, Logistic Regression và Stepwise Regression; n*hóm Classification Algorithms* bao gồm Linear Classifier, Support Vector Machine (SVM), Kernel SVM, Sparse Representation-based classification (SRC), Instance-based Algorithms như k-Nearest Neighbor (kNN) và Learning Vector Quantization (LVQ); *Nhóm Regularization Algorithms* gồm Ridge Regression, Least Absolute Shrinkage and Selection Operator (LASSO) và Least-Angle Regression (LARS). Các *thuật toán Bayesian Algorithms* bao gồm Naive Bayes và Gaussian Naive Bayes. *Nhóm Clustering Algorithms* bao gồm k-Means clustering, k-Medians và Expectation Maximization (EM). *Nhóm Artificial Neural Network Algorithms* bao gồm Perceptron, Softmax Regression, Multi-layer Perceptron và Back-Propagation. *Nhóm Dimensionality Reduction Algorithms* gồm Principal Component Analysis (PCA) và Linear Discriminant Analysis (LDA). Cuối cùng, *nhóm Ensemble Algorithms* bao gồm Boosting, AdaBoost và Random Forest.

Tác giả luận án đã đánh giá tiềm năng và khả năng phát triển của Machine Learning (ML) và phân tích thông kê dữ liệu trong nghiên cứu ứng dụng vào bộ cân bằng tải trên điện toán đám mây. Dựa trên phân loại thuật toán được đề cập ở trên, tác giả đã chọn một số thuật toán phù hợp và tương thích với cân bằng tải trên môi trường đám mây. *Thứ nhất,* thuật toán xác suất Naïve Bayes được chọn vì khả năng xử lý các bài toán phân loại dựa trên xác suất. Thuật toán này rất hữu ích trong việc phân loại các sự kiện dựa trên dữ liệu có sẵn và mô hình xác suất. *Thứ hai,* thuật toán SVM (Support Vector Machine) được lựa chọn vì khả năng xử lý các bài toán phân loại và hồi quy. SVM có thể xây dựng các siêu phẳng phân cách hiệu quả và làm việc tốt trên dữ liệu không tuyến tính. *Thứ ba,* thuật toán KMeans trong nhóm Clustering Algorithms cũng được tác giả sử dụng. KMeans là một thuật toán phân cụm dựa trên

khoảng cách, có thể phân nhóm các mẫu dữ liệu dựa trên đặc trưng tương tự. *Thứ tư,* tác giả cũng sử dụng thuật toán dự báo ARIMA và thuật toán dự báo Regression trong việc dự báo trong tương lai gần các thông số của yêu cầu (request) trên môi trường đám mây. Cả hai thuật toán này đều có khả năng dự báo xu hướng và biến động trong dữ liệu, giúp tác giả hiểu và đưa ra dự báo về xu hướng tương lai của người dùng đám mây. . *Cuối cùng,* thuật toán phân lớp k-NN (k-Nearest Neighbor) cũng được lựa chọn trong luận án. Thuật toán K-Nearest Neighbors (K-NN) hoạt động bằng cách xác định lớp hoặc giá trị dự đoán cho một điểm dữ liệu mới dựa trên thông tin từ k điểm dữ liệu gần nhất trong tập dữ liệu huấn luyện. Thuật toán này được sử dụng để phân loại và phân lớp dữ liệu dựa trên các điểm gần nhất.

Với lựa chọn trên, dựa trên phân loại thuật toán và các yếu tố cân nhắc, tác giả đã chọn các thuật toán Naïve Bayes, SVM, KMeans, ARIMA, Regression và k-NN trong nghiên cứu ứng dụng vào nâng cao hiệu năng cân bằng tải trên môi trường điện toán đám mây.

1.3.2 Một số thuật toán ML ứng dụng vào CBT

Thuật toán Naïve Bayes: Naive Bayes [59] là một trong các kĩ thuật phân lớp dựa trên định lý về Bayes với các yếu tố dự đoán được xem như độc lập với nhau. Một cách tổng quan, trong thuật toán Naive Bayes, việc xem xét sự xuất hiện hoặc vắng mặt của một đặc trưng cụ thể trong một lớp không bị ảnh hưởng bởi việc xuất hiện hoặc vắng mặt của các đặc trưng khác. Ví dụ như một loại trái cây được xem là quả táo sẽ có các thuộc tính như màu đỏ, hình tròn và có đường kính khoảng 7.6 cm. Các loại đặc trưng này phụ thuộc lẫn nhau hoặc dựa trên sự hiện diện của các đặc trưng khác, tất cả thuộc tính này là độc lập, chúng đóng góp cho khả năng một loại trái cây là quả táo và đó được gọi là Naive.

Mô hình Naive Bayes có thể dễ dàng xây dựng và đặc biệt hữu ích khi làm việc với các tập dữ liệu lớn. Điều đáng chú ý là đơn giản và mạnh mẽ của nó, Naive Bayes đã được công nhận vì khả năng vượt trội so với các phương pháp phân loại khác, bao gồm cả những phương pháp rất phức tạp.

Định lý Bayes cung cấp một cách tính xác suất hậu nghiệm (posterior probability) $P(c|x)$ từ $P(c)$, $P(x)$ và $P(x|c)$. Nhìn vào phương trình dưới đây (Công thức tính xác suất hậu nghiệm – Posterior posibility)

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \quad (1)$$

- *P(c|x)* là xác suất sau của lớp (c, mục tiêu) dự đoán (x, thuộc tính) đã cho (posterior posibility).
- *P(c)* là xác suất trước của lớp (Class prior probability*)*.
- *P(x|c)* là khả năng xảy ra là xác suất của người dự đoán cho lớp đã cho (Likelihood).
- *P(x)* là xác suất trước của yếu tố dự đoán (Predictor prior probability*)*.

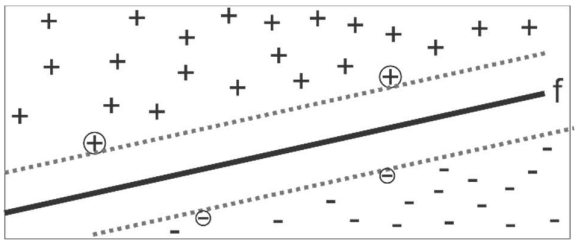
Thuật toán SVM: Bài toán phân lớp và dự đoán là hai bài toán quan trọng và được áp dụng rộng rãi trong nhiều lĩnh vực, bao gồm học máy, nhận dạng, và trí tuệ nhân tạo. SVM (Support Vector Machine) được phát triển bởi các tác giả Vapnik và Chervonenkis vào năm 1995 và được coi là một công cụ mạnh mẽ cho việc giải quyết các bài toán phân lớp phi tuyến tính. SVM [60] hoạt động dựa trên nguyên lý Cực tiểu hóa rủi ro có cấu trúc SRM (Structural Risk Minimization) và được coi là một trong những phương pháp phân lớp giám sát không tham số tinh vi nhất cho đến nay. SVM sử dụng các hàm kernel đa dạng để tạo không gian biến đổi, giúp xây dựng các mặt phân lớp hiệu quả.

Ý tưởng của thuật toán SVM

Máy học SVM (Support Vector Machine) xây dựng một siêu phẳng trong không gian dữ liệu để phân tách các điểm dữ liệu thành hai lớp riêng biệt. SVM sử dụng một kỹ thuật đặc biệt để chuyển đổi tập dữ liệu ban đầu vào không gian có nhiều chiều hơn. Sau khi ánh xạ sang không gian nhiều chiều, SVM sẽ tìm và chọn siêu phẳng phù hợp nhất để phân tách các điểm dữ liệu.

Đối với tập dữ liệu huấn luyện, mỗi điểm dữ liệu được biểu diễn trong không gian vector, và SVM sẽ tìm một siêu phẳng quyết định tốt nhất để chia các điểm này thành hai lớp riêng biệt, thường được gọi là lớp + và lớp -. Chất lượng của siêu phẳng này được đo bằng khoảng cách từ điểm dữ liệu gần nhất của mỗi lớp đến siêu phẳng. Khoảng cách biên lớn hơn đồng nghĩa với việc siêu phẳng quyết định tốt hơn và phân loại chính xác hơn.

Mục đích của phương pháp SVM là tìm được khoảng cách biên lớn nhất, điều này được minh họa như sau:

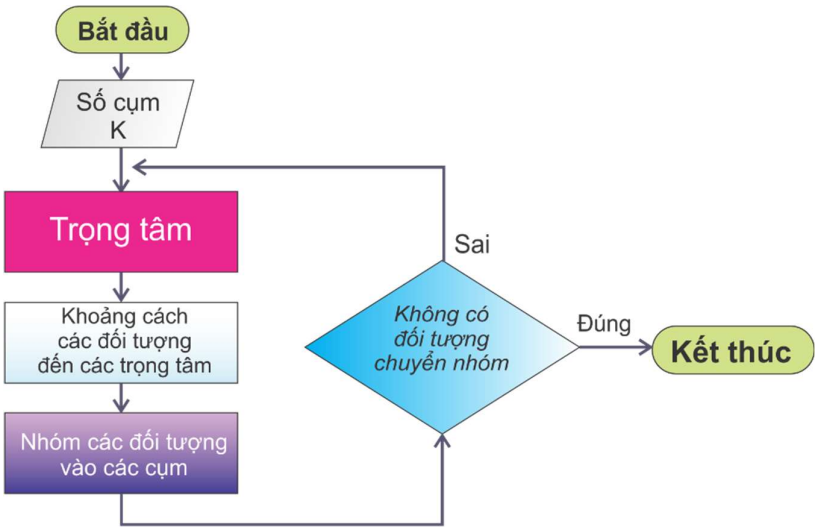


Hình 1. 8 Siêu phẳng phân chia dữ liệu học thành 2 lớp + và - với khoảng cách biên lớn nhất. Các điểm gần nhất là các Support Vector [60]

Thuật toán K-Means

Phân cụm [61] là một kỹ thuật quan trọng trong việc khai phá dữ liệu và thuộc vào loại phương pháp Unsupervised Learning trong Machine Learning. K-Means là một thuật toán quan trọng và phổ biến được sử dụng trong phân cụm. Ý tưởng cơ bản của thuật toán K-Means là chia các đối tượng được cung cấp thành K nhóm (với K là một số nguyên dương đã được xác định trước) sao cho tổng bình phương khoảng cách giữa các đối tượng và tâm của các nhóm là nhỏ nhất.

Thuật toán K-Means được mô tả như sau:



Hình 1. 9 *Sơ đồ thuật toán K – means* [61]

Các bước chính của thuật toán

1. Chọn ngẫu nhiên K tâm (centroid) cho K cụm (cluster). Mỗi cụm được đại diện bằng các tâm của cụm.

2. Tính toán khoảng cách giữa các objects (đối tượng) đến K tâm (thông thường sử dụng khoảng cách Euclidean).
3. Nhóm các đối tượng vào cụm gần nhất.
4. Xác định lại tâm mới cho các cụm bằng cách lấy trung bình cộng tọa độ các điểm đã được gán vào cụm.
5. Lặp lại bước 2 cho đến khi không có sự thay đổi cụm nào của các đối tượng.

Thuật toán dự báo ARIMA

Theo [62], ARIMA (Auto Regression Integrated Moving Average) là một thuật toán dựa trên thống kê, được phát triển từ mô hình hồi quy ARMA (Auto Regression Moving Average). Nó là một mô hình dự đoán dựa trên dữ liệu chuỗi thời gian đã biết để đưa ra dự đoán về dữ liệu trong tương lai.

Nhận dạng mô hình [63]

Mô hình ARIMA (hay còn gọi là phương pháp Box-Jenkin)

Để nhận dạng mô hình trong ARIMA(p,d,q), chúng ta cần lựa chọn giá trị p, d, và q. Trước hết, chúng ta phải đảm bảo rằng chuỗi thời gian đã dừng hoặc đã được sai phân hóa (với mức độ sai phân d được xác định). Sau đó, thông qua việc nghiên cứu hàm tự tương quan ACF và hàm tự tương quan từng phần PACF, chúng ta có thể xác định mô hình phù hợp.

Nếu hàm tự tương quan ACF giảm đột ngột và hàm tự tương quan từng phần PACF giảm mạnh, chúng ta có thể sử dụng mô hình AR (Auto-Regressive). Trong trường hợp cả hai hàm đều giảm đột ngột, chúng ta có thể sử dụng mô hình ARMA (Auto-Regressive Moving Average).

Thỉnh thoảng, có trường hợp cả hai hàm đều giảm đột ngột đồng thời, và để xác định mô hình phù hợp, chúng ta cần thử nghiệm và so sánh một số mô hình khác nhau. Sau đó, chúng ta kiểm tra mô hình nào cho kết quả tốt nhất.

p: dựa vào SPAC

q: dựa vào SAC

d: dựa vào số lần lấy sai phân để làm cho chuỗi dừng

Mô hình ***ARIMA(1, 1, 1)*** :

$$y(t) - y(t\text{-}1) = a_0 + a_1(y(t\text{-}1) - y(t\text{-}2) + e(t) + b_1e(t\text{-}1)) \quad (2.1)$$

Hoặc $z(t) = a_0 + a_1z(t\text{-}1) + e(t) + b_1e(t\text{-}1)$,

Với $z(t) = y(t) - y(t\text{-}1)$ ở sai phân đầu tiên : $d = 1$.

Tương tự ***ARIMA(1,2,1)*** :

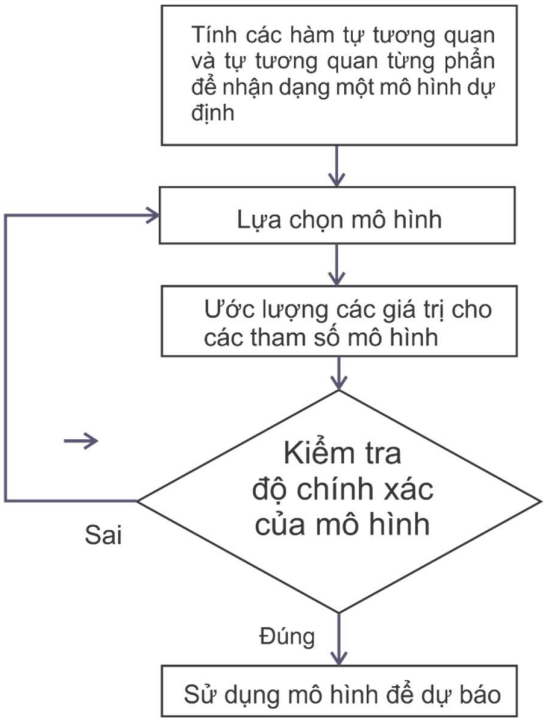
$$h(t) = a_0 + a_1z(t\text{-}1) + e(t) + b_1e(t\text{-}1) \quad (2.2)$$

Với $h(t) = z(t) - z(t\text{-}1)$ ở sai phân thứ hai: $d = 2$. (d lớn hơn 2 rất ít được sử dụng)

Kiểm tra chuẩn đoán mô hình

Mô hình ARIMA tốt có RMSE nhỏ và sai số là nhiễu trắng: Sai số có phân phối chuẩn, và đồ thị SAC giảm nhanh về 0

Tìm kiếm mô hình ARIMA phù hợp là một quá trình thử và sai.



Hình 1. 10 Sơ đồ mô phỏng mô hình Box-Jenkins [63]

Thuật toán Regression

$$f(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_0 \quad (3.1)$$

Dạng của Linear Regression [64]

Trong phương trình (3.1) , nếu chúng ta đặt $w = [w_0, w_1, w_2, w_3]^T$ là vector (cột) hệ số cần phải tối ưu và $\bar{x} = [1, x_1, x_3, x_3]$ (đọc là x bar trong tiếng Anh) là vector (hàng) dữ liệu đầu vào mở rộng. Số 1 ở đầu được thêm vào để phép tính đơn

giản hơn và thuận tiện cho việc tính toán. Khi đó, phương trình (3.1) có thể được viết lại dưới dạng:

$$y \approx \bar{x}w = \hat{y} \tag{3.2}$$

Sai số dự đoán

Chúng ta mong muốn rằng sự sai khác ee giữa giá trị thực *y* và giá trị dự đoán $\hat{y}$ (đọc là *y hat* trong tiếng Anh) là nhỏ nhất. Nói cách khác, chúng ta muốn giá trị sau đây càng nhỏ càng tốt:

$$\frac{1}{2}e^2 = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(y - \bar{x}w)^2 \tag{3.3}$$

trong đó hệ số ½ là để thuận tiện cho việc tính toán (khi tính đạo hàm thì số ½ sẽ bị triệt tiêu). Chúng ta cần e^2 vì $e = y - \hat{y}$ có thể là một số âm, việc nói ee nhỏ nhất sẽ không đúng vì khi $e = -\infty$ là rất nhỏ nhưng sự sai lệch là rất lớn.

Hàm mất mát

Điều tương tự xảy ra với tất cả các cặp (input, outcome) (x_i, y_i) với $i = 1,2, \dots, N$ và N là số lượng dữ liệu quan sát được. Điều chúng ta muốn, tổng sai số là nhỏ nhất, tương đương với việc tìm $\boldsymbol{w}$ để hàm số sau đạt giá trị nhỏ nhất:

$$\mathcal{L}(w) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \bar{x}_i w)^2 \tag{3.4}$$

Hàm số $\boldsymbol{\mathcal{L}(w)}$ được gọi là hàm mất mát (loss function) của bài toán Linear Regression. Chúng ta luôn mong muốn rằng sự mất mát (sai số) là nhỏ nhất, điều đó đồng nghĩa với việc tìm vector hệ số $\boldsymbol{w}$ sao cho giá trị của hàm mất mát này càng nhỏ càng tốt. Giá trị của $\boldsymbol{w}$ làm cho hàm mất mát đạt giá trị nhỏ nhất được gọi là điểm tối ưu (optimal point), ký hiệu:

$$w^* = arg\min_{w} \mathcal{L}(w) \tag{3.5}$$

Trước khi đi tìm lời giải, chúng ta đơn giản hóa phép toán trong phương trình hàm mất mát (3.4). Đặt $y = [y_1; y_2; \dots; y_N;]$ là một vector cột chứa tất cả các output của training data; $\bar{x} = [\bar{x}_1; \bar{x}_2; \dots; \bar{x}_N;]$ là ma trận dữ liệu đầu vào (mở rộng) mà mỗi hàng của nó là một điểm dữ liệu. Khi đó hàm số mất mát $\boldsymbol{\mathcal{L}(w)}$ được viết dưới dạng ma trận đơn giản hơn:

$$\mathcal{L}(w) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \bar{x}_i w)^2 \; = \; \frac{1}{2}\|y - \bar{X}w\|_2^2 \tag{3.6}$$

với $\|z\|_2$ là Euclidean norm (chuẩn Euclid, hay khoảng cách Euclid), nói cách khác $\|z\|_2^2$ là tổng của bình phương mỗi phần tử của vector $\boldsymbol{z}$. Tới đây, ta đã có một dạng đơn giản của hàm mất mát được viết như phương trình (3.6).

Nghiệm cho bài toán Linear Regression

Cách phổ biến nhất để tìm nghiệm cho một bài toán tối ưu là giải phương trình đạo hàm (gradient) bằng 0. Tất nhiên đó là khi việc tính đạo hàm và việc giải phương trình đạo hàm bằng 0 không quá phức tạp. Thật may mắn, với các mô hình tuyến tính, hai việc này là khả thi.

Đạo hàm theo $\boldsymbol{w}$ của hàm mất mát là:

$$\frac{\partial \mathcal{L}(w)}{\partial w} = \bar{X}^T(\bar{X}w - y) \tag{3.7}$$

Sai số dự đoán phía trên về việc tại sao không dùng trị tuyệt đối mà lại dùng bình phương. Câu trả lời là hàm bình phương có đạo hàm tại mọi nơi, trong khi hàm trị tuyệt đối thì không (đạo hàm không xác định tại 0).

Phương trình đạo hàm bằng 0 tương đương với:

$$\bar{X}^T\bar{X}w = \bar{X}^T y \triangleq b \tag{3.8}$$

Trong đó: ký hiệu $\bar{X}^T y \triangleq b$ nghĩa là đặt $\bar{X}^T y$ bằng b.

Nếu ma trận vuông $A \triangleq \bar{X}^T\bar{X}$ khả nghịch (non-singular hay invertible) thì phương trình (3.8) có nghiệm duy nhất: $w = A^{-1}b$

Vậy nếu ma trận A không khả nghịch thì hoặc phương trinh (3.8) vô nghiệm, hoặc là nó có vô số nghiệm. Khi đó, chúng ta sử dụng khái niệm giả nghịch đảo A† (đọc là A dagger trong tiếng Anh). (Giả nghịch đảo (pseudo inverse) là trường hợp tổng quát của nghịch đảo khi ma trận không khả nghịch hoặc thậm chí không vuông)

Với khái niệm giả nghịch đảo, điểm tối ưu của bài toán Linear Regression có dạng:

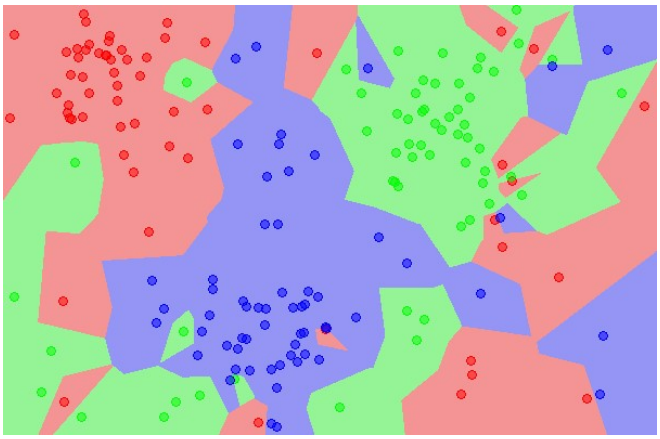
$$w = A^{\dagger}y = (\bar{X}^T\bar{X})^{\dagger}\bar{X}^T y \tag{3.9}$$

Thuật toán K-NN: K-Nearest Neighbor (KNN) là một thuật toán đơn giản và hiệu quả trong Machine Learning, thường được sử dụng trong các tình huống cụ thể. Khi được huấn luyện, thuật toán này không học gì từ dữ liệu huấn luyện, và thực hiện tính toán khi cần dự đoán kết quả cho dữ liệu mới. KNN có thể được áp dụng cho cả hai loại bài toán trong Supervised learning, bao gồm Classification và Regression. Nó còn được gọi là một thuật toán Instance-based hoặc Memory-based learning.

Với thuật toán K-Nearest Neighbor (KNN) trong bài toán Classification, việc xác định nhãn cho một điểm dữ liệu mới (hoặc câu hỏi trong bài thi) được thực hiện dựa trên K điểm dữ liệu gần nhất trong tập huấn luyện. Có hai cách chính để xác định nhãn cho điểm dữ liệu kiểm tra. Một là thông qua bầu chọn dựa trên số phiếu của các điểm gần nhất, và hai là dựa trên việc gán trọng số khác nhau cho mỗi điểm gần nhất và suy ra nhãn từ đó.

Trong bài toán Regression, giá trị đầu ra của một điểm dữ liệu kiểm tra có thể được tính toán dựa trên một số cách khác nhau. Trong trường hợp K=1, giá trị đầu ra của điểm dữ liệu mới sẽ bằng giá trị đầu ra của điểm dữ liệu gần nhất trong tập huấn luyện. Trường hợp khác, nó có thể được tính toán dựa trên trung bình có trọng số của giá trị đầu ra của các điểm gần nhất, hoặc thông qua một mối quan hệ dựa trên khoảng cách đến các điểm gần nhất.

Nói một cách ngắn gọn, KNN là một phương pháp xác định giá trị đầu ra cho một điểm dữ liệu mới bằng cách dựa vào thông tin từ K điểm dữ liệu gần nhất trong tập huấn luyện (K-lân cận), mà không quan tâm đến việc có một số điểm dữ liệu trong những điểm gần nhất này có thể là nhiễu hoặc ngoại lệ.



Hình 1. 11 Bản đồ của 1NN (Nguồn: Wikipedia)

Ví dụ trên đề cập đến một bài toán Classification với 3 lớp: Đỏ, Lam, Lục. Mỗi điểm dữ liệu mới sẽ được gán nhãn dựa vào màu của điểm mà nó gần nhất. Tuy nhiên, trong hình minh họa, có những vùng nhỏ trong đó các lớp màu chồng lên nhau. Ví dụ, có một điểm màu Lục nằm gần góc 11 giờ, nằm giữa hai vùng lớn khác với nhiều điểm màu Đỏ và Lam. Điểm này có thể là nhiễu, và nếu điểm dữ liệu kiểm tra rơi vào vùng này, nó có thể được phân loại sai.

Khoảng cách giữa hai điểm trong không gian đa chiều có thể được tính toán bằng nhiều cách khác nhau, và các chuẩn vector là một nguồn thông tin quan trọng trong Machine Learning. Norms (chuẩn) của vector có thể cung cấp thông tin hữu ích về cách đo khoảng cách giữa các điểm dữ liệu trong không gian nhiều chiều.

1.4 KẾT LUẬN CHƯƠNG

Chương này trình bày tổng quan về công nghệ điện toán đám mây hiện đại, cân bằng tải và các đặc điểm của tính toán hiệu năng cao trên cân bằng tải. Bên cạnh đó, trình bày một số thuật toán trí tuệ nhân tạo hiện nay phổ biến, mà có thể ứng dụng vào cân bằng tải, đặc biệt là nhóm thuật toán phân lớp và dự báo. Chương này cũng đã lựa chọn ra một số thuật toán / kỹ thuật ML về dự báo, phân lớp, gom cụm để làm cơ sở đưa ra giải pháp cho các thuật toán cân bằng tải tương ứng. Song song đó, chương này cũng trình bày một số công trình nghiên cứu liên quan đã được công bố trong giai đoạn gần đây.

CHƯƠNG 2 – **TIẾP CẬN SWOT CHO CÂN BẰNG TẢI TRÊN ĐIỆN TOÁN ĐÁM MÂY**

2.1 GIỚI THIỆU CHUNG

Cân bằng tải là một vấn đề mang tính kỹ thuật và cũng mang tính chất lượng dịch vụ trên môi trường điện toán đám mây, chính vì vậy ta có thể xem xét cân bằng tải như một yếu tố chủ chốt trong điện toán đám mây. Chính vì thế, để hiểu rõ hơn về cân bằng tải, đánh giá tốt hơn về cân bằng tải, luận án này sử dụng công cụ phân tích SWOT, một công cụ hiệu quả, đơn giản và rất dễ dàng sử dụng, để phân tích yếu tố cân bằng tải trên điện toán đám mây. Từ đó thấy được các vấn đề liên quan đến cân bằng tải hiện nay, và đưa ra các hướng tiếp cận để nâng cao hiệu năng cân bằng tải trên điện toán đám mây.

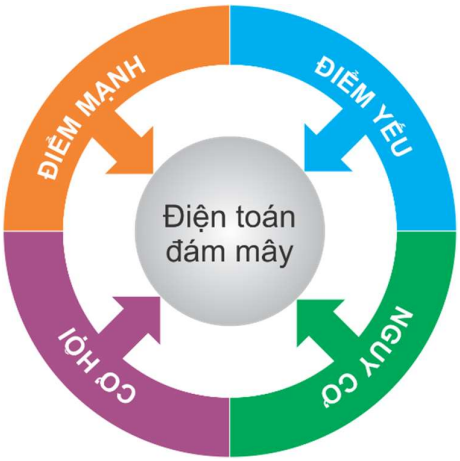
2.2 GIỚI THIỆU VỀ CÔNG CỤ SWOT

SWOT [65] (Strengths - Thế mạnh, Weaknesses - Điểm yếu, Opportunities - Cơ hội, Threats - Thách thức) là một công cụ phân tích phổ biến trong kinh doanh, thường được sử dụng để đánh giá và cải thiện tình hình kinh doanh của các tổ chức. SWOT giúp xác định các yếu tố bên trong và bên ngoài ảnh hưởng đến doanh nghiệp. Các yếu tố "Thế mạnh" và "Điểm yếu" được xem xét như những yếu tố bên trong doanh nghiệp và có thể được cải thiện hoặc thay đổi bởi chính doanh nghiệp. Ví dụ về các yếu tố này có thể bao gồm danh tiếng của doanh nghiệp, đặc điểm sản phẩm hoặc vị trí địa lý. "Cơ hội" và "Thách thức" là những yếu tố bên ngoài doanh nghiệp, và chúng thường không thể kiểm soát hoàn toàn. Ví dụ về các yếu tố này có thể bao gồm thị trường khách hàng và hành vi của họ.

Phân tích SWOT [66] viết tắt của Strengths (Điểm mạnh), Weaknesses (Điểm yếu), Opportunities (Cơ hội), và Threats (Thách thức), đề cập đến việc đánh giá bốn yếu tố quan trọng, bao gồm: Điểm mạnh, Điểm yếu, Cơ hội, và Thách thức. Thông qua việc phân tích 4 yếu tố này, giúp chúng ta xác định rõ hơn các vấn đề tồn tại, từ đó đưa ra mục tiêu chiến lược, hướng đi cho nhằm cải thiện dịch vụ cho doanh nghiệp.

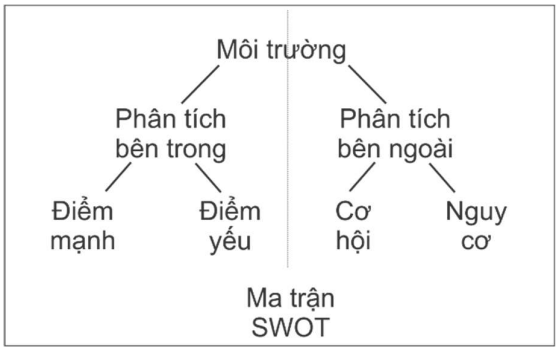
Bởi vì điện toán đám mây, là một dịch vụ mà các nhà cung cấp dịch vụ mạng cung cấp cho khách hàng, là những người sử dụng cloud, nên việc nghiên cứu và phát triển điện toán đám mây cũng gắn liền với việc nghiên cứu và phát triển chiến lược

của các nhà cung cấp dịch vụ, là những doanh nghiệp cung cấp dịch vụ điện toán đám mây. Vì thế, công cụ SWOT rất phù hợp và có thể áp dụng vào phân tích điện toán đám mây để đưa ra các giải pháp tốt nhất cho dịch vụ. Từ đó, luận án này sử dụng công cụ SWOT để tiếp cận phân tích yếu tố cân bằng tải trên điện toán đám mây, sử dụng công cụ này như kim chỉ nam để tìm ra đường đi cho luận án, tìm ra các hướng nâng cao hiệu năng cân bằng tải thông qua phân tích SWOT.



Hình 2. 1 ***Phân tích SWOT*** **[66]**

Phân tích SWOT bao gồm việc xem xét bốn yếu tố quan trọng. Trong đó, Điểm mạnh và Điểm yếu tập trung vào hiện tại hoặc quá khứ, trong khi Cơ hội và Thách thức liên quan đến tương lai. Bằng cách thực hiện phân tích SWOT, nghiên cứu này sẽ tạo một liên kết giữa tình hình cân bằng tải hiện tại và các hướng nghiên cứu và thuật toán mới trong tương lai.



Hình 2. 2 ***Khung phân tích SWOT*** **[67]**

Để phân tích SWOT, ta có thể kết hợp 4 yếu tố của SWOT và tạo ra ma trận SWOT hay ma trận TOWS. Từ đó đưa ra 4 các tiếp cận: S-O, W-O, S-T và W-T.

Bảng 2. 1 Ma trận *SWOT* [67]

	Strength (Điểm mạnh)	Weakness (Điểm yếu)
Opportunities (Cơ hội)	Tiếp cận S-O	Tiếp cận W-O
Threats (Thách thức)	Tiếp cận S-T	Tiếp cận W-T

2.3 PHÂN TÍCH SWOT HIỆU NĂNG CÂN BẰNG TẢI TRÊN CLOUD

2.3.1 Hiệu năng cân bằng tải trên cloud

Khái niệm về hiệu năng

Hiệu năng của một sản phẩm cho thấy tính hiệu quả trong quá trình hoạt động của thiết bị đó. Nó bao gồm mức tiêu thụ năng lượng, khả năng tối ưu phần mềm, công suất làm việc, sức mạnh xử lý và thời gian để hoàn tất các tác vụ. Trên một thiết bị công nghệ, hiệu năng được xem là yếu tố tổng hòa của các đặc điểm kể trên. Chúng ta hoàn toàn có thể đánh giá hiệu năng của một chiếc smartphone, thông qua việc nhận diện cấu hình như chip, RAM, bộ nhớ trong.

Khái niệm về hiệu năng cân bằng tải

Tương tự như một thiết bị, cân bằng tải cũng có thể xem là một thiết bị hoạt động trên môi trường điện toán đám mây, và có hiệu năng cân bằng tải của nó. Theo các nhà phát triển và cung cấp dịch vụ điện toán đám mây lớn như IBM, NGINX,… thì để đo lường hiệu năng cân bằng tải [68], [69], [70] có thể sử dụng các yếu tố chủ yếu như sau:

- *Độ trễ* (Latency)- Lượng thời gian trung bình cần để một yêu cầu đến máy chủ. Bạn muốn giá trị này càng thấp càng tốt, đặc biệt là khi trang web của bạn có nhiều người dùng. Nếu độ trễ trở nên rất cao, người dùng có thể gặp phải thời gian phản hồi chậm hoặc thậm chí hết thời gian chờ. Giá trị độ trễ thấp là mong muốn, nhưng có một số điều bạn cần phải xem xét. Nếu bạn có nhiều máy chủ trong một nhóm và tất cả chúng đang thực hiện các tác vụ đồng thời, có thể có nhiều yêu cầu hơn các máy chủ có sẵn. Điều này có thể khiến hàng đợi yêu cầu tăng lên, dẫn đến độ trễ lâu hơn. Ngoài ra, hãy lưu ý rằng nếu trình cân bằng tải của bạn sử dụng mã hóa SSL với xác thực máy khách,

thời gian phản hồi có thể lâu hơn bình thường do chi phí xử lý bổ sung liên quan đến SSL.

- *Thời gian phản hồi (Response Time)* - Máy chủ cần để trả lời các yêu cầu. Thời gian phản hồi là rất cần thiết, đặc biệt nếu bạn đang lưu trữ các ứng dụng hoặc dịch vụ yêu cầu phản hồi nhanh, chẳng hạn như các trang web thương mại điện tử hoặc hệ thống trò chuyện trực tiếp. Nếu ta sử dụng thông số này, chúng ta cần đo thời gian phản hồi trên các thành phần hoặc phiên bản riêng lẻ của bộ cân bằng tải. Thời gian phản hồi là dấu hiệu cho biết người dùng nhận được nội dung nhanh như thế nào. Thời gian phản hồi lâu hơn dự kiến có thể do tắc nghẽn mạng, phân giải DNS kém hoặc sự cố cơ sở dữ liệu.
- *Khả năng phân bổ tài nguyên (Resource Allocation Capacity)* [71]*:* Năng lực phân bổ tài nguyên trong cân bằng tải trên đám mây đề cập đến lượng tài nguyên có thể được phân bổ cho một dịch vụ hoặc ứng dụng cụ thể trong môi trường đám mây. Đây là một khía cạnh quan trọng của cân bằng tải vì nó đảm bảo rằng các tài nguyên được phân bổ hiệu quả và hiệu quả để đáp ứng nhu cầu của ứng dụng hoặc dịch vụ. Cân bằng tải trên đám mây tối ưu hóa dung lượng ứng dụng toàn cầu của bạn, mang lại trải nghiệm người dùng tốt hơn và chi phí thấp hơn so với hầu hết các triển khai cân bằng tải
- *Mức độ công bằng trong phân bổ tài nguyên (Allocation Fairness)* [71] [72]*:* Phân bổ Công bằng trong cân bằng tải trên đám mây đề cập đến việc phân phối tài nguyên công bằng giữa các dịch vụ hoặc ứng dụng khác nhau trong môi trường đám mây. Đây là một khía cạnh quan trọng của cân bằng tải vì nó đảm bảo rằng các tài nguyên được phân bổ công bằng và hiệu quả để đáp ứng nhu cầu của tất cả các ứng dụng hoặc dịch vụ. Có nhiều kỹ thuật khác nhau để cân bằng tải trong điện toán đám mây như kỹ thuật cân bằng tải chung, cân bằng tải dựa trên hiện tượng thông thường, cân bằng tải hoàn chỉnh dựa trên dự án và cân bằng tải kiểu dựa trên tác nhân.
- *Khả năng tăng tốc (Speedups)* [30]: Các tham số tăng tốc trong cân bằng tải trên đám mây đề cập đến việc tối ưu hóa các tham số bị ràng buộc khác nhau như thời gian phản hồi, thời gian thực thi, độ ổn định của hệ thống, v.v., từ đó cải thiện hiệu suất của đám mây. Cân bằng tải trên đám mây cung cấp hai loại cân bằng tải: Cân bằng tải ứng dụng và Cân bằng tải mạng. Chúng ta sẽ chọn

bộ cân bằng tải ứng dụng khi cần bộ cân bằng tải Lớp 7 cho lưu lượng HTTP(S), SSL(S) hoặc TCP/SSL.

- *Khả năng đồng bộ giữa các tác vụ (Tasks Synchronzition)* [73]: Đồng bộ hóa tác vụ trong cân bằng tải trên đám mây đề cập đến quá trình đồng bộ hóa các tác vụ trên nhiều máy chủ để đảm bảo rằng chúng được thực thi theo đúng thứ tự và không có xung đột giữa chúng. Cân bằng tải là một phương pháp tối ưu hóa để phân phối công việc đồng đều trên nhiều máy chủ, nhằm cải thiện hiệu suất và đáng tin cậy. Đây là một khía cạnh quan trọng trong lĩnh vực điện toán đám mây, giúp đảm bảo tài nguyên được sử dụng một cách hiệu quả và hiệu quả.
- *Khả năng chịu lỗi (Fault Tolerance)* [74]*:* Khả năng chịu lỗi trong cân bằng tải trên điện toán đám mây đề cập đến khả năng hệ thống tiếp tục hoạt động ngay cả khi có lỗi. Việc này là một thách thức quan trọng mà phải được giải quyết để đảm bảo rằng điện toán đám mây luôn đáng tin cậy và sẵn sàng hoạt động. Bằng cách cân bằng hiệu quả tải đến, khả năng chịu lỗi có thể đạt được trong đám mây. Có nhiều kỹ thuật khác nhau để đạt được khả năng chịu lỗi trong điện toán đám mây, chẳng hạn như tạo kế hoạch chi tiết cho công việc đang diễn ra bất cứ khi nào một số bộ phận ngừng hoạt động hoặc không khả dụng, cung cấp dịch vụ trong trường hợp thiết bị tương ứng không khả dụng vì một số lý do và sử dụng các giải pháp cân bằng tải và chuyển đổi dự phòng để đảm bảo tính khả dụng thông qua dự phòng và khắc phục thảm họa nhanh chóng.

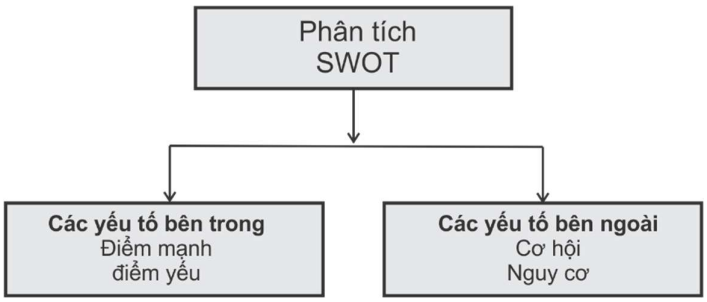
Thông qua các yếu tố trên để đo lường và kiểm soát hiệu năng cân bằng tải trên cloud, cụ thể hóa việc nâng cao hiệu năng cân bằng tải [75] bằng việc đo lường các thông số trên trong các thuật toán. Cũng theo nghiên cứu này, thì các tham số cụ thể để đo lường hiệu năng cân bằng tải của các thuật toán bao gồm: Thông lượng (Throughput), Overhead, Khả năng chịu lỗi (Fault Tolerance), thời gian di dời (Migration Time), thời gian đáp ứng (Response Time), tối ưu hóa tài nguyên (Resouce Utilization), khả năng co giãn (Scalability), hiệu quả hoạt động (efficiency).

2.3.2 Phân tích SWOT cân bằng tải

SWOT [76] là công cụ hữu ích trong việc phân tích các vấn đề kỹ thuật và công nghệ cũng như ứng dụng của các công nghệ kỹ thuật này trên thị trường. Đặc biệt trong công nghệ thông tin SWOT [77] hỗ trợ phân tích rõ nét các vấn đề đang tồn tại cũng như thách thức và cơ hội. SWOT là công cụ phân tích hệ thống thông qua việc liệt kê ra các yếu tố tốt và xấu để đánh giá. Đã có nhiều nghiên cứu về điện toán đám mây sử dụng SWOT để phân tích, và đánh giá. Năm 2015, Sonal Dubey và các cộng sự [67] đã sử dụng SWOT và phân tích đánh giá môi trường điện toán đám mây, tìm ra các vấn đề tồn tại, đưa ra các giải pháp cũng như hướng phát triển cần có cho môi trường cloud. Năm 217, Sugandhi Midha [78] cũng sử dụng SWOT để phân tích sâu hơn các vấn đề của cloud. Năm 2020, Ify Evangel và cộng sự [66], cũng sử dụng SWOT để phân tích công nghệ điện toán đám mây đối với dịch vụ thư viện tại trường đại học Nigeria Nsukka.

Trong điện toán đám mây, cân bằng tải là một bộ phận không thể tách rời, và đóng vai trò quan trọng đối với môi trường điện toán đám mây. Chính vì thế, luận án này sử dụng công cụ SWOT dùng để phân tích yếu tố cân bằng tải trên cloud, từ đó đưa ra hướng tiếp cận để nâng cao hiệu năng cân bằng tải trên môi trường điện toán đám mây.

Theo công cụ SWOT, ta cần phải xác định mục tiêu cần đạt được, đó là nâng cao hiệu năng cân bằng tải. Từ đó, đưa ra các mục tiêu cụ thể, phân tích theo hướng tiếp cận bên trong (điểm mạnh và điểm yếu) và tiếp cận bên ngoài (cơ hội và thách thức / khó khăn), từ đó đưa ra các giải pháp tương ứng với nó, để đáp ứng được mục tiêu là nâng cao hiệu năng cân bằng tải.

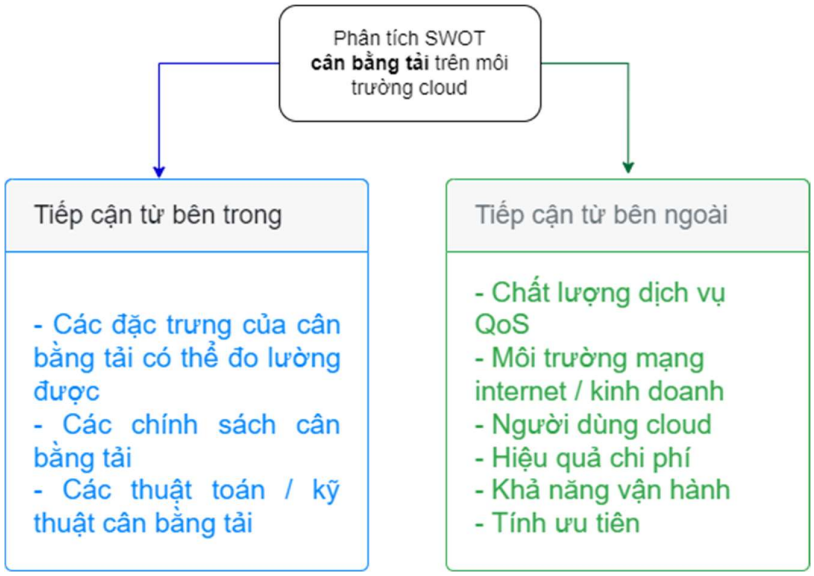


Hình 2. 3 ***Tiếp cận phân tích SWOT*** **[67]**

Như vậy, các yếu tố bên trong cân bằng tải của môi trường đám mây chính là các thuộc tính, tính chất mà chúng ta có thê đo lường được, cụ thể hơn là các thuộc tính đặc trưng tải [21] của cân bằng tải: thông lượng (Throughput), dung sai lỗi (Fault

Tolerance), Thời gian di dời (Migration Time), Thời gian đáp ứng (Response Time), khả năng mở rộng (Scalability). Ngoài ra, yếu tố bên trong của cân bằng tải còn bao gồm chính sách cân bằng tải, cách hoạt động của cân bằng tải hay cụ thể hơn là các thuật toán cân bằng tải.

Cũng theo tài liệu [21], các yếu tố bên ngoài cân bằng tải chính là chất lượng dịch vụ, môi trường kinh doanh, môi trường mạng internet, người dùng cloud và các mục tiêu mà nhà cung cấp cloud hướng tới: Hiệu quả chi phí (Cost effectiveness), Tính ưu tiên (Priority), Khả năng mở rộng & tính linh hoạt (Scalability and flexibility).



Hình 2. 4 Đề xuất 2 hướng tiếp cận nâng cao hiệu năng cân bằng tải

Như vậy, để nâng cao hiệu năng cân bằng tải, theo tiếp cận SWOT, luận án này đề xuất 2 hướng tiếp cận nghiên cứu cân bằng tải trên cloud, đó là hướng tiếp cận từ bên ngoài và hướng tiếp cận từ bên trong.

Cụ thể đối với hướng tiếp cận từ bên trong, chúng ta nghiên cứu các đặc trưng của cân bằng tải, các thông số có thể đo lường được, từ đó cải tiến, ứng dụng các thuật toán mới vào, ví dụ như thời gian đáp ứng. Bên cạnh các thông số của cân bằng tải, chúng ta cũng có thể bắt đầu từ các chính sách cân bằng tải, hoặc cơ chế cân bằng tải, mà cụ thể hơn là các thuật toán cân bằng tải hiện có, từ đó cải tiến hoặc nâng cấp cho phù hợp, nâng cao hiệu năng làm việc của cân bằng tải.

Đối với hướng tiếp cận từ bên ngoài, chính là việc nghiên cứu môi trường xung quanh của cân bằng tải trên môi trường đám mây. Cụ thể đó là yêu cầu chất lượng

dịch vụ, là một yếu tố tiêu chuẩn chất lượng đưa ra do người dùng và nhà cung cấp dịch vụ đưa ra. Ngoại ra, yếu tố môi trường mạng, mạng internet cũng là các vấn đề nằm ngoài cân bằng tải. Người dùng cloud, và hành vi người dùng cloud, cũng như độ ưu tiên của những người dùng này v/v. Tất cả đều nằm bên ngoài bộ cân bằng tải, nhưng nó quyết định đến khả năng vận hành tốt hay hiệu năng làm việc của cân bằng tải trên cloud. Chính vì thế, chúng ta có thể bắt đầu từ các yếu tố này, nghiên cứu đề xuất cải tiến các thuật toán nâng cao hiệu năng cân bằng tải.

2.4 CÁC CÔNG TRÌNH LIÊN QUAN

2.4.1 CÔNG TRÌNH LIÊN QUAN CLOUD VÀ XU HƯỚNG PHÁT TRIỂN CBT TRÊN MÔI TRƯỜNG ĐÁM MÂY

Trong bài nghiên cứu tổng hợp của Atharva Agashe và cộng sự [79], các tác giả đã thảo luận về các mô hình dịch vụ khác nhau (IaaS, PaaS, SaaS) cũng như các mô hình triển khai (Công cộng, Riêng tư, Hybrid) của đám mây trong những nghiên cứu gần đây. Các tác giả cũng đã cũng phân tích các giai đoạn di chuyển (migration) để chuyển ứng dụng hiện có lên đám mây. Tất cả các giai đoạn này đóng vai trò quan trọng trong quá trình di chuyển và cần được thực hiện một cách cẩn thận và cẩn trọng. 6 chiến lược chính (Rehosting, Replatforming, repurchasing, refactoring, retaining, retiring) trong quá trình di chuyển đưa ra một cái nhìn tổng quan về cách di chuyển thực tế sẽ diễn ra và những thay đổi cần thiết. Nhóm tác giả cũng phân tích các kỹ thuật mà các nhà cung cấp dịch vụ đám mây (CSPs) như AWS, Azure và GCP cung cấp.

Một bài nghiên cứu dưới góc độ kỹ thuật về an toàn và tính riêng tư của cloud, nhóm tác giả Yunas Simpa [80] đã tổng hợp những tài liệu và xu hướng của các công nghệ mới nổi, thách thức trong các hệ thống cloud phổ biến, điển hình là cơ sở hạ tầng với khả năng tự bảo vệ của hệ thống liên quan đến àn toàn thông tin và tính riêng tư. Các kỹ thuật thích ứng an toàn được sử dụng phổ biến và có thể được áp dụng ở bất kỳ giai đoạn nào trong bất kỳ công nghệ cơ bản nào của cloud, từ phần cứng và phần mềm đến cơ sở hạ tầng tính toán cốt lõi. Sự thích ứng an toàn được hiểu rằng hệ thống có thể tự bảo vệ trong quá trình chống lại nhiều cuộc tấn công hoặc một người dùng không thân thiện khám phá ra các lỗ hổng. Điện toán đám mây vẫn sẽ dễ bị tổn thương về mặt an toàn thông tin và riêng tư nếu không áp dụng các cơ chế thích

ứng để cải thiện trải nghiệm của khách hàng và người dùng. Bên cạnh đó, bài đánh giá này cũng làm nổi bật những lỗ hổng ảnh hưởng đến các thành phần khác nhau của đám mây tính thông qua phân tích STRIDE. Nghiên cứu cung cấp các hạn chế của các công trình khác nhau trong tài liệu, bao gồm phân loại các vấn đề an ninh và riêng tư dựa trên khắc phục cuộc tấn công. Đánh giá cũng cung cấp một phương pháp kỹ thuật và miêu tả nhu cầu về các kỹ thuật thích ứng tốt hơn để đáp ứng các mối đe dọa và lỗ hổng liên quan đến cloud. Qua tổng hợp các nghiên cứu và các công, không có sự nhất quán trong việc thiết kế và triển khai các biện pháp bảo mật đám mây hiệu quả, điều này có nghĩa là việc triển khai an toàn và riêng tư chưa thật sự đầy đủ. Hơn nữa, các mô hình đám mây cho bảo vệ quyền riêng tư không tập trung vào người dùng, không tạo ra tính linh hoạt và quản lý kiểm soát đối với các giao thức an toàn mạng hoặc riêng tư trên mạng, duy trì dữ liệu nhạy cảm của người dùng.

Một nghiên cứu tổng hợp khác của nhóm tác giả Belen Bermejo [81], đã tóm tắt các nghiên cứu về sự bền vững của hạ tầng đám mây, cạnh (edge), sương mù (fog) và IoT sử dụng trí tuệ nhân tạo. Nghiên cứu xác định rằng học máy và học sâu là các kỹ thuật phổ biến nhất trong các ứng dụng sử dụng đám mây (cloud) /cạnh (edge) /IoT, tập trung vào tiêu thụ năng lượng và hiệu suất. Tuy nhiên, các giải pháp cải thiện sự bền vững chưa tổng quát và có sự thiếu cân bằng giữa tính toàn vẹn, trách nhiệm và quyền riêng tư. Bài viết có thể giúp các nhà nghiên cứu tạo ra kiến thức mới và cung cấp căn cứ cho nghiên cứu tiếp theo trong lĩnh vực AI và sự bền vững trong đám mây (cloud) /cạnh (edge)/IoT. Công việc tương lai có thể tập trung vào việc áp dụng trí tuệ nhân tạo để cải thiện sự bền vững của môi trường đám mây (cloud) /cạnh (edge)/IoT. Cần tiếp tục đánh giá các kỳ vọng và tiến bộ của trí tuệ nhân tạo trong việc nghiên cứu sự bền vững của các hệ sinh thái này.

Gần đây, tác giả Hazzaa N. Alshareef [82] đã công bố nghiên cứu khảo sát về sự phát triển hiện tại, thách thức và xu hướng tương lai trong điện toán đám mây. Bài báo cho rằng điện toán đám mây là một công nghệ phát triển cực kỳ nhanh chóng trong lĩnh vực điện toán. Có một số lợi ích khi áp dụng điện toán đám mây, chẳng hạn như khả năng truy cập mọi lúc mọi nơi, phạm vi địa lý hiệu quả hơn, hiệu quả về thời gian cao hơn và giảm chi phí cơ sở hạ tầng. Tuy nhiên, cũng có những trở ngại khi áp dụng điện toán đám mây, chẳng hạn như thiếu chuyên môn và tài nguyên, quản lý dịch vụ đám mây, quyền riêng tư, và nhu cầu bảo mật dữ liệu. Phần lớn các dịch

vụ liên quan đến cơ sở hạ tầng của đám mây lưu trữ, bao gồm tài nguyên lưu trữ và điện toán, tồn tại trong các trung tâm dữ liệu. Việc lưu trữ các ứng dụng trong đám mây của một nhà cung cấp duy nhất được coi là đơn giản và mang lại nhiều lợi ích khác nhau. Tuy nhiên, có vô số rủi ro và thách thức liên quan đến điện toán đám mây. Quyền riêng tư, bảo mật thông tin và tính toàn vẹn dữ liệu nằm trong số này. Các xu hướng và kết quả từ tài liệu cho thấy rằng điện toán đám mây vẫn đang nổi lên và các công nghệ liên quan mới đang được phát triển để đối phó với những thách thức hiện có. Việc sử dụng AI trong điện toán đám mây có thể giảm thiểu một số rủi ro và cung cấp giải pháp cho các vấn đề chưa được giải quyết trước đây.

Qua dịch bệnh COVID-19, mọi người phụ thuộc vào các ứng dụng điện toán đám mây và công nghệ khác. Do đó, trong nghiên cứu của Shajunyi Zhaovà cộng sự [83], đã nghiên cứu vai trò của trả tiền khi sử dụng (pay-as-go) và điện toán đám mây tại thời điểm xảy ra đại dịch. Nhóm tác giả đã thu thập, tóm tắt và đánh giá bài báo khoa học về điện toán đám mây trong suốt đợt bùng phát bùng phát từ năm 2020 đến tháng 4 năm 2022. Nhóm tác giả cũng đã sử dụng một phương pháp đánh giá được xác định trước để kiểm tra các bộ dữ liệu điện tử thường được biết đến. Các từ khóa đã được sử dụng để tìm kiếm tất cả các ấn phẩm được kết nối với chủ đề này. 18 bài báo đã được chọn cho bài khảo sát này sau khi tuân thủ nghiêm ngặt phương pháp lựa chọn nghiên cứu. Bài báo đánh giá này cung cấp các kết quả và phương pháp luận hiện đại nhất về điện toán đám mây trong suốt thời kỳ dịch bệnh, vạch ra các lỗ hổng nghiên cứu và đưa ra phương hướng cho các nghiên cứu trong tương lai. Kết quả cho thấy điện toán đám mây đóng một vai trò quan trọng trong việc giải quyết và làm giảm các tác dụng phụ trong các tình huống quan trọng như dịch coronavirus.

Điện toán đám mây xanh (Green Cloud Computing)

Điện toán đám mây là một lĩnh vực đang phát triển nhanh chóng với nhiều ứng dụng khác nhau cung cấp khả năng mở rộng, độ tin cậy và hiệu suất cao với chi phí thấp. Nó có khả năng góp phần bảo vệ môi trường bằng cách cải thiện hiệu quả sử dụng năng lượng, giảm lượng khí thải carbon và giảm thiểu chất thải điện tử, do đó chuyển đổi nó thành điện toán đám mây xanh. Khảo sát vào 2017 của Laura-Diana Radu [84] cung cấp cái nhìn tổng quan về điện toán đám mây, nêu bật những tiến bộ tính đến 2017, giải quyết các vấn đề về môi trường và đưa ra các hướng nghiên cứu

trong tương lai, đóng vai trò là tài liệu tham khảo hiện tại cho nghiên cứu điện toán đám mây xanh. Để giải quyết các tác động tiêu cực của điện toán đám mây đối với môi trường, các nhà cung cấp dịch vụ đang khám phá các công nghệ như quản lý năng lượng, ảo hóa và điện toán hiệu năng cao để giảm mức tiêu thụ điện và nước, yêu cầu phần cứng vật lý và lượng khí thải carbon. Báo cáo phân tích của Archarna Patil và cộng sự [85] vào 2019, đã cung cấp một cái nhìn tổng quan chi tiết về điện toán đám mây xanh, bao gồm những thành tựu trong quá khứ, xu hướng hiện tại và những thách thức nghiên cứu trong tương lai, đóng vai trò là nguồn tài nguyên quý giá cho các nhà nghiên cứu quan tâm đến việc hiểu và giải quyết các khía cạnh môi trường của điện toán đám mây.

Năm 2022, theo kết quả nghiên cứu của nhóm tác giả Nesma Abd El-Mawla và cộng sự [86] đã xác định được các yếu tố sau đây mà các nhà cung cấp dịch vụ đám mây và người tiêu dùng nên xem xét nếu điện toán đám mây có tác động tích cực đến môi trường. Một bài viết của J Sylvia Grace1 và G Meeragandhi [87] đã thảo luận về hoạt động của các công ty CNTT xanh hơn, thân thiện với môi trường và giảm lượng khí thải carbon & chi phí vận chuyển. Điện toán đám mây xanh nhằm mục đích giảm tác động môi trường của máy tính bằng cách giảm chất gây ô nhiễm trong khí quyển, nước và đất. Phần này cũng xem xét lượng khí thải carbon của một người cũng như tác động carbon của trung tâm điện toán đám mây.

Trong bài báo khảo sát của Avita Katal [88] và cộng sự vào 2023 , việc sử dụng năng lượng ngày càng tăng của các trung tâm dữ liệu do những tiến bộ trong công nghệ máy chủ và nhu cầu ngày càng tăng về sức mạnh xử lý đòi hỏi phải phát triển các giải pháp phần mềm để giảm mức tiêu thụ điện năng. Bài báo này xem xét các phương pháp khác nhau, bao gồm ảo hóa phần mềm, ảo hóa hệ điều hành, phương pháp luận và các yếu tố môi trường để giảm thiểu mức tiêu thụ điện năng của trung tâm dữ liệu. Tầm quan trọng của container hóa trong việc giảm mức sử dụng năng lượng được nhấn mạnh và các thách thức nghiên cứu trong tương lai, chẳng hạn như kỹ thuật di chuyển và sắp xếp container hiệu quả, được xác định để xây dựng trung tâm dữ liệu bền vững. Từ ý tưởng đó, Jason M. Pittman và Shaho Alaee [89] đã đề xuất thuật toán lập lịch xanh cho honeynet trên môi trường đám mây. Các doanh nghiệp hiện đại tận dụng kiến trúc đám mây để đạt được các dịch vụ công nghệ linh hoạt và tiết kiệm chi phí. Làm như vậy phải trả giá bằng môi trường mặc dù các công

nghệ đám mây tiêu thụ một lượng lớn năng lượng. Tiêu thụ năng lượng đám mây có liên quan đến xu hướng khí hậu toàn cầu và trữ lượng nhiên liệu hóa thạch đang cạn kiệt. Do đó, người ta ngày càng chú ý đến điện toán đám mây xanh và bền vững, nhằm tìm cách tối ưu hóa việc phân bổ tài nguyên máy tính và sử dụng các hệ thống và dịch vụ ảo hóa. Đồng thời, tiến trình hướng tới công nghệ đám mây xanh và bền vững bị cản trở vì càng nhiều doanh nghiệp triển khai dịch vụ vào kiến trúc đám mây, các mối đe dọa an ninh mạng theo sau. Thật không may, các công nghệ an ninh mạng được tối ưu hóa để phục vụ tối đa Overwatch mà không quan tâm đến tài nguyên máy tính và năng lượng. Điều này phủ nhận việc giảm năng lượng đạt được những tiến bộ công nghệ bền vững gần đây. Nhóm tác giả đã đề xuất một thuật toán lập lịch trình honeynet an ninh mạng tổng quát được đề xuất, trong đó chi phí năng lượng, CPU và mạng được vận hành để tăng tính bền vững trong khi cân bằng các cơ chế phòng thủ. Nghiên cứu được mô tả cả dưới dạng toán học cho thuật toán và mã giả.

Điện toán cạnh (Edge) và điện toán sương mù (Fog)

Bài báo của Jorge Pérez và cộng sự [90] làm rõ ý nghĩa của "tính toán cạnh" bằng cách phát triển một lý thuyết toàn diện dựa trên nghiên cứu định tính và phỏng vấn các chuyên gia trong ngành. Lý thuyết xác định các cấu trúc và đề xuất chính xác định các khía cạnh trung tâm của điện toán biên, phù hợp với ngành và tiêu chuẩn ISO/IEC TR 30164. Nghiên cứu nêu bật những lợi ích mong đợi của điện toán biên, chẳng hạn như giảm mức tiêu thụ tài nguyên, cải thiện bảo mật và hiệu suất, như cũng như những thách thức chung, bao gồm xây dựng hệ thống phức tạp, độ tin cậy, hiệu suất tính toán và các vấn đề lưu trữ trong môi trường IoT.

Trong bài nghiên cứu khảo sát của Gustavo Caiza và cộng sự [91], sự gia tăng của các thiết bị được kết nối với IoT tạo ra một lượng lớn dữ liệu, đồng nghĩa với một thách thức đối với cơ sở hạ tầng hiện tại và các ứng dụng công nghiệp vì chúng không thể xử lý dữ liệu một cách hiệu quả. Để giải quyết những vấn đề này, một công nghệ mới có tên Fog Computing (FC) xuất hiện, như một phần mở rộng của điện toán đám mây để cung cấp các nền tảng nhỏ tại các nút sương mù, theo cách này dẫn đến các tài nguyên tính toán và ứng dụng gần hơn với người dùng cuối. FC là một lĩnh vực mở cho sự phát triển của các nghiên cứu và ứng dụng ở cấp độ công nghiệp để cải thiện và giải quyết các vấn đề hiện tại. Một đề xuất cho các nghiên cứu trong tương

lai là sự phát triển của FC theo IEC- 61499 sẽ cung cấp các đặc điểm về tính di động, khả năng tương tác và cấu hình lại các ứng dụng dành riêng cho tiêu chuẩn.

Theo Mohammed Al Masarweh và cộng sự [92], với sự phát triển nhanh chóng của các ứng dụng IoT và những thách thức liên quan của chúng, điện toán sương mù đã nổi lên như một giải pháp bằng cách xử lý và lưu trữ dữ liệu cục bộ giữa các thiết bị IoT. Tuy nhiên, khi dữ liệu nhạy cảm yêu cầu thời gian phản hồi nhanh từ đám mây, bài báo đề xuất một hệ thống có tên DCMB đáp ứng các yêu cầu SLA của IoT, đạt được thời gian xử lý ngắn hơn so với các hệ thống đám mây truyền thống và giảm các công việc bị từ chối, mặc dù thử nghiệm thêm với số lượng công việc lớn hơn là cần thiết để đánh giá khả năng sử dụng và hiệu suất của nó. Một nghiên cứu của Deok-Kee Choi về FC [93], cho rằng các hệ thống sản xuất thông minh dựa trên điện toán đám mây phải đối mặt với những thách thức như độ trễ cao và mức sử dụng băng thông do lượng lớn dữ liệu do các thiết bị IoT tạo ra. Để giải quyết những thách thức này, tác giả đã đề xuất một phương pháp điện toán sương mù sử dụng thuật toán học máy để tạo ra mô hình vật lý không gian mạng của hệ thống quạt, đạt được khả năng giám sát chính xác trạng thái của quạt theo thời gian thực với độ chính xác khoảng 98% bằng thuật toán BOSSVS. Quy trình công việc này có khả năng được áp dụng cho các thiết bị IoT khác nhau trong các hệ thống sản xuất thông minh. Theo Sukhpal Singh Gill [94], việc sử dụng các ứng dụng IoT đang tăng lên từng ngày và tạo ra nhiều dữ liệu trong vài giây. Nền tảng đám mây hiệu quả trong việc quản lý dữ liệu một cách linh hoạt nhưng các ứng dụng IoT mới nhất cần xử lý dữ liệu với độ trễ và thời gian phản hồi tối thiểu. Trong chương này, một bản tuyên ngôn cho các hệ thống điện toán biên và sương mù hiện đại được trình bày để đánh giá nghiên cứu đang diễn ra trong lĩnh vực này. Ngoài ra, loại kiến trúc và ứng dụng cho điện toán biên và sương mù được trình bày. Cuối cùng, các cơ hội nghiên cứu và hướng đi đầy hứa hẹn trong tương lai được nhấn mạnh.

Một nghiên cứu khảo sát của nhóm tác giả Sundas Iftikhar [95] vào 2023, đã tập trung vào việc sử dụng học máy và trí tuệ nhân tạo (AI) để giải quyết các thách thức về quản lý tài nguyên trong môi trường điện toán cạnh (Edge) và điện toán sương mù (Fog). Nghiên cứu nêu bật xu hướng ngày càng tăng của việc sử dụng các phương pháp dựa trên AI để khắc phục những hạn chế của phương pháp tiếp cận phỏng đoán truyền thống, đưa ra các quyết định quản lý tài nguyên chính xác với chi phí thời gian

thấp hơn và chất lượng dịch vụ được cải thiện. Các tình huống và ứng dụng khác nhau, từ chăm sóc sức khỏe đến giao thông thông minh, đã được hưởng lợi từ các kỹ thuật tối ưu hóa dựa trên AI và có tiềm năng tối ưu hóa hơn nữa bằng cách sử dụng các vi dịch vụ và kiến trúc không có máy chủ. Tuy nhiên, đánh giá cũng xác định sự cần thiết của một khung tối ưu hóa toàn diện dựa trên AI để bao quát toàn bộ quy trình quản lý tài nguyên trong điện toán cạnh và điện toán sương mù.

Một nghiên cứu của Pankaj Sharma [96], đã trình bày một thuật toán thông qua sơ đồ hiệu quả để tối ưu hóa đường dẫn chi phí tốt nhất bằng cách sử dụng phương pháp Vị trí nút góc đến cùng với tối ưu hóa ong hạt mới và cũng xử lý Khả năng chịu lỗi (fault Tolerance) của máy chủ trực tiếp bằng cách sao chép các hoạt động của máy chủ và kiểm tra trỏ khi đồng bộ hóa nó với máy chủ proxy. Lược đồ được đề xuất giảm thiểu mức sử dụng mạng là 618.020 để giảm chi phí thực hiện đám mây sương mù 81.900 đô la. Ngoài ra, các thiết bị cho thấy thông lượng cao nhất là 93% cho Devive_ID 3 với thời gian phản hồi 62%, hiệu suất 94%, tính khả dụng 97% và độ tin cậy 85% . Với kết quả thực nghiệm, các sơ đồ được đề xuất cung cấp hiệu suất tốt hơn so với khung lai CRBM. Trong tương lai, các vị trí dịch vụ sương mù có thể được nghiên cứu để nâng cao việc sử dụng tài nguyên và hiệu quả năng lượng của các thiết bị cảm biến sương mù và cũng sẽ được đánh giá dựa trên quan điểm bảo mật và quyền riêng tư.

Mobile Edge Computing - MEC

Theo nhóm tác giả Gwanggil Jeon [97], sự phát triển nhanh chóng của Internet vạn vật (IoT) và khối lượng dữ liệu ngày càng tăng do các thiết bị IoT tạo ra đã tạo ra những thách thức đối với các giải pháp IoT dựa trên đám mây, đặc biệt là về khả năng cung cấp dịch vụ theo thời gian thực, quyền riêng tư và hiệu suất. Để giải quyết những thách thức này, khái niệm điện toán cạnh (edge) đã xuất hiện, cho phép xử lý và lưu trữ dữ liệu được thực hiện gần hơn với các thiết bị của người dùng cuối, giảm thiểu các hạn chế về băng thông mạng và các vấn đề về độ trễ. Điện toán cạnh di động (mobile edge computing - MEC) là một kiến trúc mạng mang khả năng điện toán đám mây đến biên mạng di động, cho phép hiệu suất ứng dụng tốt hơn và giảm tắc nghẽn mạng. MEC được coi là một chủ đề nghiên cứu quan trọng để xử lý các vấn đề dữ liệu lớn phức tạp, vì nó cung cấp các giải pháp hiệu quả cho các thách thức lưu trữ và

xử lý dữ liệu quy mô lớn do các công nghệ như IoT, phương tiện truyền thông xã hội và giao tiếp giữa máy với máy đặt ra.

Theo Kai Peng và cộng sự [98], thành phố thông minh đã thu hút được sự quan tâm ngày càng tăng từ cả giới học thuật và ngành công nghiệp có tiềm năng cải thiện mức sống của con người. Một thành phố thông minh bao gồm một số lượng lớn các thiết bị thông minh đang tạo ra một lượng lớn dữ liệu và các ứng dụng mới nổi mọi lúc. Tuy nhiên, khả năng tính toán của các thiết bị thông minh còn hạn chế. May mắn thay, sự xuất hiện của MEC có thể giải quyết vấn đề trên. Tuy nhiên, tài nguyên của các máy chủ biên trong MEC bị hạn chế và khả năng của các máy chủ biên không đồng nhất. Điều quan trọng là phải cải thiện mức sử dụng tài nguyên trung bình của tất cả các máy chủ cạnh và duy trì đồng thời cân bằng tải của cụm máy chủ biên. Mặt khác, khá nhiều ứng dụng nhạy cảm với độ trễ, cần phải đảm bảo tính bảo mật của các ứng dụng này. Nhóm tác giả xem xét việc tối ưu hóa chung thiết bị di động và máy chủ biên trong thành phố thông minh hỗ trợ MEC, cải thiện hiệu suất tổng thể của hệ thống. Về mặt kỹ thuật, một phương pháp giảm tải tính toán đa mục tiêu mới được triển khai để giảm mức tiêu thụ thời gian, tiêu thụ năng lượng và duy trì cân bằng tải của các máy chủ biên, cũng như tăng mức sử dụng tài nguyên trung bình của các máy chủ biên trong khi đáp ứng hạn chế về thời hạn của các ứng dụng nhạy cảm với độ trễ. Đủ các thí nghiệm đã được tiến hành để chứng minh tính hiệu quả và tính ưu việt của phương pháp đề xuất của chúng tôi trong các tình huống khác nhau. Một nghiên cứu của nhóm tác giả Tarik Taleb [99], giới thiệu một sơ đồ tận dụng điện toán cạnh di động (MEC) để nâng cao Chất lượng dịch vụ (QoS) và trải nghiệm người dùng trong truyền phát video, đặc biệt là trong bối cảnh thành phố thông minh. Phương pháp được đề xuất, được gọi là "Follow-me-Edge", cho phép các ứng dụng và dịch vụ tự động thích ứng với tính di động của người dùng, đảm bảo Chất lượng trải nghiệm (QoE) tối ưu và giảm độ trễ. Bằng cách áp dụng sơ đồ này, bài báo nhấn mạnh tiềm năng của kiến trúc MEC thông minh để giảm đáng kể lưu lượng mạng lõi và đạt được độ trễ cực ngắn, giải quyết các yêu cầu của hệ thống di động 5G sắp tới.

Function as a Service - FaaS

Theo bài viết của Marcello Cinque [100], các công nghệ ảo hóa, thường được sử dụng trong cài đặt đám mây, đang được áp dụng trong các tình huống công nghiệp để hợp

nhất nhiều ứng dụng trên cùng một phần cứng, giảm kích thước (Size), trọng lượng (Weight), công suất (Power) và chi phí (Cost), hay còn gọi là SWAP-C, và cho phép các hệ thống quan trọng hỗn hợp khác. Ngoài ra, ngày càng có nhiều mối quan tâm đến việc áp dụng các giải pháp gốc trên đám mây, chẳng hạn như container hóa và điện toán không có máy chủ, trong môi trường công nghiệp, với tiềm năng cho các ứng dụng Chức năng dưới dạng dịch vụ (FaaS) theo thời gian thực, mặc dù có một số thách thức cần được giải quyết để biến điều này thành hiện thực.

Bài viết của Yasmina Bouizem và cộng sự [101], đã đề xuất tích hợp phương pháp tiếp cận khả năng chịu lỗi sao chép chủ động (RR) vào các nền tảng Chức năng dưới dạng dịch vụ (FaaS) và so sánh nó với sao chép thụ động (AS) và các cơ chế thử lại cơ bản. Kết quả cho thấy phương pháp thử lại không đủ để có tính sẵn sàng cao, trong khi AS giảm thời gian khôi phục bằng cách cung cấp dịch vụ khi bản sao dự phòng phát hiện lỗi. RR đảm bảo tính khả dụng của dịch vụ miễn là có ít nhất một bản sao tiếp tục phản hồi mà không cần thay thế bản sao bị lỗi. Mở ra hướng nghiên cứu trong tương lai bao gồm khám phá các phương pháp tiếp cận khả năng chịu lỗi khác như điểm kiểm tra và thiết kế hệ thống chịu lỗi cho FaaS hỗ trợ nhiều cơ chế dựa trên loại ứng dụng và điều kiện hoạt động trong khi đáp ứng các yêu cầu của người dùng.

Bài viết của Urmil Bhart và cộng sự [102], giới thiệu ReactiveFnJ, một mô hình vũ đạo cho quy trình công việc Fork-Join trong kiến trúc không có máy chủ. Mô hình này hướng đến sự kiện, ST-Safe và tận dụng các tính năng về khả năng mở rộng cũng như khả năng chịu lỗi của môi trường không có máy chủ, giải quyết các thách thức về giới hạn tài nguyên, quản lý trạng thái và khả năng kết hợp. Nghiên cứu chứng minh tính khả thi của việc sử dụng thiết kế không có máy chủ cho các quy trình công việc song song bùng nổ phức tạp mà không cần dựa vào các dịch vụ điều phối bên ngoài hoặc bộ nhớ dùng chung, cung cấp giải pháp nền tảng độc lập với các ứng dụng tiềm năng trong điện toán phân tán quy mô lớn.

2.4.2 CÔNG TRÌNH LIÊN QUAN CBT THEO HƯỚNG TIẾP CẬN BÊN TRONG

"Study the effect of parameters to load balancing in cloud computing." [103]

Các tác giả đã tiến hành nghiên cứu và đề xuất các giải pháp nhằm tăng cường hiệu suất trong lĩnh vực điện toán đám mây, đặc biệt là về cân bằng tải và thời gian đáp ứng. Năm 2016, họ đã công bố một nghiên cứu về các tham số ảnh hưởng đến hiệu quả của quá trình cân bằng tải trong môi trường điện toán đám mây (STUDY THE EFFECT OF PARAMETERS TO LOAD BALANCING IN CLOUD COMPUTING). Nghiên cứu này điểm qua nhiều phương pháp cân bằng tải khác nhau, bao gồm: (i) cân bằng tải sau khi máy chủ quá tải; (ii) cân bằng tải và dự đoán tải tương lai để phân bổ tài nguyên; (iii) cải thiện các tham số ảnh hưởng đến cân bằng tải trên đám mây. Ngoài ra, nghiên cứu cũng đề xuất một số phương pháp để tối ưu hóa cân bằng tải và nâng cao hiệu suất hoạt động của hệ thống đám mây.

"A Load Balancing Game Approach for VM Provision Cloud Computing Based on Ant Colony Optimization" [104]

Một nghiên cứu khác được thực hiện bởi tác giả Trần Công Hùng và đồng nghiệp vào năm 2017 [104], tập trung vào vấn đề cân bằng tải cho các máy ảo trên môi trường đám mây, sử dụng thuật toán tối ưu hóa dựa trên lý thuyết trò chơi và kỹ thuật tối ưu hóa dựa trên quá trình tìm kiếm của kiến (Ant Colony Optimization). Trong nghiên cứu này, tác giả đề xuất một giải pháp nhằm đảm bảo cân bằng các mục tiêu quan trọng mà các bên liên quan cần, bao gồm cả nhà cung cấp dịch vụ và khách hàng của họ, dựa trên lý thuyết trò chơi. Ý tưởng chính là áp dụng thuật toán tối ưu hóa dựa trên quá trình tìm kiếm của kiến (Ant Colony Optimization - ACO) trong ngữ cảnh trạng thái cân bằng Nash. Trong quá trình thực nghiệm, tác giả đã sử dụng các biến thể của thuật toán ACO, bao gồm Ant System, Ma-Min Ant System, và Ant Colony System, để giải quyết bài toán tối ưu hóa dựa trên lý thuyết trò chơi. Kết quả thực nghiệm cho thấy rằng việc sử dụng hệ số tương quan giữa các bên liên quan có thể giúp đạt được cân bằng tải trong việc cung cấp máy ảo.

"Minimum makespan task scheduling algorithm in cloud computing" [105]

Năm 2016, N. Sasikaladevi thuộc trường đại học SASTRA của Ấn Độ đã nghiên cứu thuật toán lập lịch tác vụ tối thiểu trong điện toán đám mây. Ông chỉ ra rằng, điện toán đám mây cung cấp môi trường kinh doanh mạnh mẽ và theo yêu cầu. Nó được xây dựng trên đỉnh của các trung tâm dữ liệu ảo hóa. Ảo hóa cung cấp cơ sở hạ tầng linh hoạt cho đám mây. Bài viết này đề xuất một cơ chế lập lịch tác vụ với

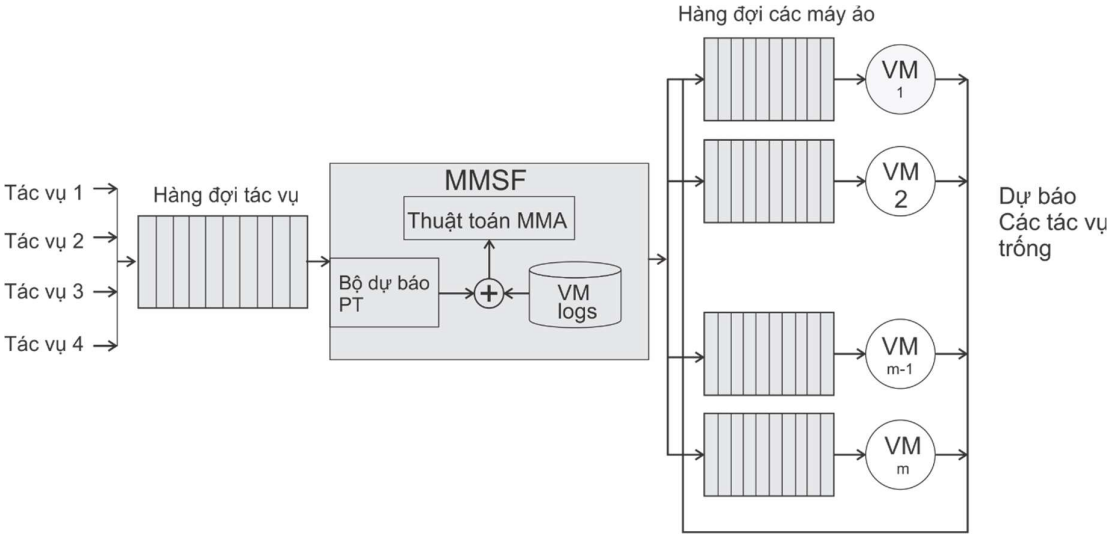
thời gian xử lý tối thiểu (**Makepan)** có tên MMSF và thuật toán lập lịch các tác vụ makepan tối thiểu có tên là MMA. Thuật toán này được phát triển với hai mục tiêu: giảm thiểu tổng số makepan và tối đa hóa việc sử dụng máy ảo. Vấn đề lập lịch tác vụ được coi là vấn đề tối ưu hóa đa mục tiêu. Nó được giải quyết bằng cách sử dụng các kỹ thuật tối ưu hóa. Kết quả thực nghiệm cho thấy MMA vượt trội hơn các thuật toán lập lịch tác vụ truyền thống dựa trên tổng makespan và tổng mức sử dụng máy ảo.

Makespan là tổng thời gian dành cho các tài nguyên để hoàn thành việc thực hiện tất cả các nhiệm vụ. Việc sử dụng VM được định nghĩa là mức độ sử dụng tài nguyên trong đám mây. Thông thường, Makespan là đề xuất ngược với tỷ lệ sử dụng. Thuật toán lập lịch hiệu quả sẽ lên lịch các tác vụ theo cách làm cho makespan đạt mức tối thiểu và mức sử dụng VM là tối đa.

Mỗi VM chỉ có thể xử lý một nhiệm vụ tại một thời điểm. Không có hai VM xử lý cùng một nhiệm vụ tại một thời điểm. Nhiệm vụ được coi là tiền giả. Tất cả các nhiệm vụ đều có tầm quan trọng như nhau. Mục đích là để giảm thiểu makepan và tăng tỷ lệ sử dụng máy ảo.

Trong bài báo này, tác giả đưa ra khung lập lịch Makespan tối thiểu – Minimum Makespan Scheduling Framework (MMSF) và thuật toán lập lịch tối thiểu – Minimum Makespan Scheduling Algorithm (MMA).

Vấn đề chính trong điện toán đám mây là sử dụng ít tài nguyên. Do đó, tối đa hóa việc sử dụng tài nguyên đồng thời giảm tối thiểu Makespan là nhiệm vụ đầy thách thức. Khung lập lịch tối thiểu trong bài viết này được đề xuất và nó được hiển thị trong hình.



Hình 2. 5 ***Khung lập lịch Makespan tối thiểu – Minimum Makespan Scheduling Framework (MMSF)*** **[105]**

Khi nhận các tác vụ từ người dùng khác nhau, các tác vụ được giữ trong hàng đợi tác vụ. Các nhiệm vụ được coi là một nhiệm vụ độc lập và ưu tiên như nhau. Số lượng tác vụ là n, số lượng máy ảo là m. Các nhiệm vụ của n được lên lịch trên các máy ảo với tốc độ tối thiểu và tốc độ sử dụng tài nguyên tối thiểu.

Máy ảo có sẵn được phân bổ với các nhiệm vụ cho đơn vị thời gian. Một thuật toán MMA được sử dụng để sử dụng tất cả các VM một cách hiệu quả. Tất cả các VM đang hoạt động và các tác vụ được thực thi bởi m VM song song. Thuật toán này làm giảm tổng số makepan bằng cách phân bổ hiệu quả các tác vụ cho tất cả các VM. Độ phức tạp thời gian của thuật toán MMA là (log (n)).

Với các kết quả khả quan mà bài báo đem lại, bài báo giúp cho luận àn này định hướng tốt và mở ra nhiều triển vọng trong việc nâng cao hiệu năng và tính chính xác của thuật toán lấy Makespan làm yếu tố xử lý. Còn rất nhiều hướng đi và cách xử lý để nâng cao chất lượng cân bằng tải trên cloud với yếu tố Makespan.

"Time Efficient Dynamic Threshold-based load balancing technique for cloud computing" [106]

Năm 2017, nhóm tác giả đã chỉ ra rằng, điện toán đám mây hỗ trợ các máy ảo (VM) để lưu trữ nhiều ứng dụng cùng một lúc. Việc cân bằng số lượng lớn các ứng dụng trong môi trường đám mây không đồng nhất trở nên khó khăn khi lập lịch trình ảo hóa điều khiển tất cả các máy ảo. Khi bộ lập lịch phân bổ các tác vụ cho các VM bị quá tải, hiệu năng của hệ thống đám mây sẽ giảm. Trong bài báo này, nhóm tác giả

trình bày một cách tiếp cận cân bằng tải mới để tổ chức các tài nguyên ảo hóa của trung tâm dữ liệu một cách hiệu quả. Theo cách tiếp cận này, tải cho VM tăng lên và giảm theo khả năng tài nguyên của VM. Đề án đề xuất giảm thiểu tối đa hóa hệ thống, tối đa hóa việc sử dụng tài nguyên và giảm mức tiêu thụ năng lượng chung. Cách tiếp cận này đã giảm thời gian chờ đợi so với các phương pháp hiện có và tối ưu hóa makespan của tài nguyên đám mây.

Trên điện toán đám mây thông thường sẽ bao gồm một số trung tâm dữ liệu và mỗi trung tâm dữ liệu này sẽ có số lượng máy chủ khác nhau. Người dùng trên cloud sẽ gửi các yêu cầu thực hiện các tác vụ của họ cho nhà cung cấp dịch vụ đám mây (CSP). Trung tâm dữ liệu (là tài nguyên đám mây) và yêu cầu của người dùng thường sẽ không đồng nhất với nhau. CSP phân bổ các yêu cầu người dùng đã gửi cho một số lượng hữu hạn các VM. Kỹ thuật này trình bày một khía cạnh ảo hóa của tài nguyên vật lý được áp dụng để khởi tạo các máy ảo. Việc phân phối các tải không đồng đều cho các máy ảo không hợp lý sẽ làm giảm hiệu năng của hệ thống đám mây. Nhóm tác giả đã đưa ra giới hạn trong tỷ lệ sử dụng bằng cách sử dụng các sơ đồ cân bằng tải cho các tài nguyên trong hệ thống đám mây. Giới hạn tỷ lệ này chính là ngưỡng cho phép (trên hoặc dưới) của máy để thực hiện một tác vụ hoặc tập hợp các tác vụ. Trong môi trường đám mây, các tính năng cân bằng tải khác nhau đáng kể; mục tiêu chính là phân phối tất cả các yêu cầu dịch vụ người dùng (tác vụ) trong một tập hợp hữu hạn các máy chủ hoặc tài nguyên đám mây. Mục đích cuối cùng của cân bằng tải là tăng tốc độ thực thi các ứng dụng trong khi khối lượng công việc thay đổi linh hoạt.

Trong bài báo này, mục tiêu chính là giảm thiểu makepan. Makespan là thời gian cần thiết để hoàn thành việc thực hiện tất cả các tác vụ đầu vào của hệ thống. Trong quá trình thực thi tác vụ, thuật toán đề xuất của nhóm tác giả chuyển một số tác vụ chọn lọc của VM bị quá tải sang các VM được tải dưới mức thích hợp. Tất cả các VM đang chạy song song và mỗi VM chạy trên các tài nguyên riêng của nó.

Trong thuật toán DLBA, nhóm tác giả ưu tiên cao hơn cho tác vụ có độ dài cao hơn với thời hạn thấp hơn, vì vậy, tác vụ được lưu trữ trong hàng đợi Max-Heap được ký hiệu là T. Tất cả các VM hoạt động đang chạy trong một trung tâm dữ liệu cũng được lưu trữ trong hàng đợi Max-Heap được ký hiệu là V. Các bước của thuật toán:

- Bước 1: phân bổ nhiệm vụ ban đầu cho VM phù hợp có sẵn, ở đây mục đích chính là cân bằng tải. Trong bước này, các tác giả tính tổng tải trong một trung tâm dữ liệu, tính tổng công suất của một trung tâm dữ liệu.

- Bước 2: sử dụng để cân bằng tải, mục đích là phân bổ tài nguyên cho nhiệm vụ theo cách mà không có VM nào bị quá tải. Trước khi giao nhiệm vụ cho VM, chúng tôi đảm bảo do tác vụ này, VM này không thể bị quá tải. Tính tổng số tác vụ có sẵn trong hàng đợi tác vụ cho các dịch vụ. Ngưỡng (θ) của mỗi VM được tính một cách linh hoạt. Nếu VM thứ j có khả năng phù hợp nhất để thực hiện nhiệm vụ đó thì DLBA sẽ giao nhiệm vụ đó cho VM đó với sự trợ giúp của chiến lược phù hợp nhất. Điều này lặp đi lặp lại cho đến khi nhiệm vụ đó sẽ không tìm thấy VM phù hợp. Giả sử một số tác vụ không thể tìm thấy để phù hợp với bất kỳ VM thích hợp nào cho đến thời điểm đó, nó sẽ nằm trong hàng đợi tác vụ. Trong quá trình thực thi tác vụ, thuật toán DLBA có thể di chuyển một số tác vụ chọn lọc để cung cấp dịch vụ cho tác vụ ưu tiên cao.

- Bước 3: sử dụng để di chuyển tác vụ khi tác vụ ưu tiên cao mới đang chờ thực thi. Sau đó, thuật toán DLBA sẽ di chuyển mức ưu tiên thấp của tác vụ từ hàng đợi VM tương ứng sang T và tạo cơ hội cho nhiệm vụ ưu tiên cao. Trong thuật toán này, nhóm tác giả thực hiện di chuyển tác vụ thay vì di chuyển VM. Vì vậy, nó sẽ giảm thời gian chờ trung bình (Average_WTT).

Sau khi thực nghiệm và kết quả thực nghiệm của bài báo đã cho thuật toán DLBA đề xuất là phương pháp cân bằng tải động tự nhiên, xem xét tới các đáp ứng hiện thời và các biến của nó để quyết định phân bổ các yêu cầu mới. Thuật toán đã hạn chế và loại bỏ được những phần thông tin không cần thiết giữa cân bằng tải và các máy ảo, bằng cách không kiểm soát ngưỡng của các nguồn tài nguyên. Tuy nhiên, thuật toán sẽ không hiệu quả khi các ngưỡng không chính xác hoặc không phù hợp. Để nâng cao chất lượng thuật toán, thì việc phát triển các cách tính toán ngưỡng phù hợp và cơ chế thay đổi ngưỡng sao cho hợp lý, mở ra nhiều thách thức và hướng đi mới để nghiên cứu. Đây là nền tảng vững chắc để luận án này hướng tới và phát triển.

"An Efficient Load Balancing Scheme for Cloud Computing" [107]

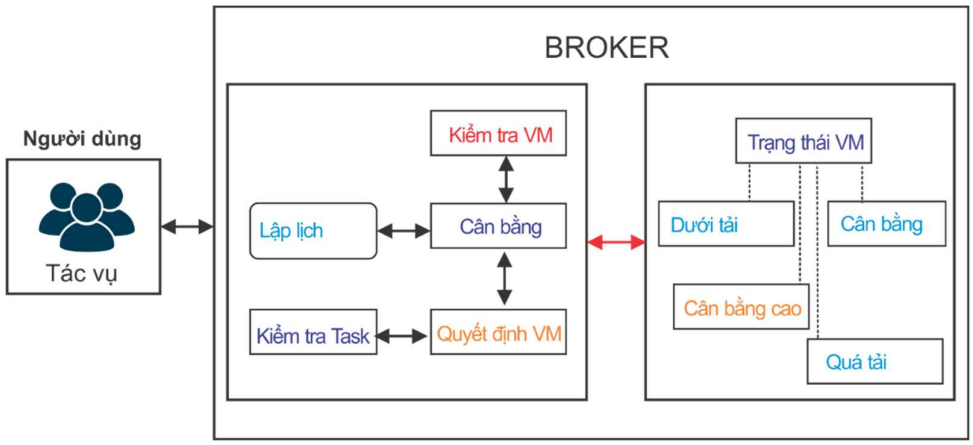
Năm 2017, Atyaf Dhari và Khaldun I.Arif của Ấn Độ đã đề xuất thuật toán quyết định cân bằng tải Load Balancing Decision Algorithm (LBDA) để quản lý và

cân bằng tải giữa các máy ảo trong trung tâm dữ liệu cùng với việc giảm thời gian hoàn thành (Makespan) và thời gian đáp ứng.

Trong môi trường điện toán đám mây có nhiều trung tâm dữ liệu bao gồm các tài nguyên không đồng nhất là các máy chủ và máy ảo (VM). Các máy chủ và VM có thể có các cấu hình khác nhau như kích thước bộ nhớ, băng thông, dung lượng lưu trữ và khả năng xử lý.

Nhóm tác giả đã thiết lập các máy ảo là VM $\{VM_1, VM_2, VM_3, \ldots, VM_m\}$. Mỗi VM có các tham số khác nhau như trạng thái VM và tốc độ tính bằng triệu lệnh mỗi giây (MIPS). Tất cả các máy ảo không bị gián đoạn, không phòng ngừa và độc lập. Các tác vụ là độc lập $\{T_1, T_2, T_3, \ldots, T_n\}$. Mỗi tác vụ có các thuộc tính khác nhau như id, độ dài, thời gian bắt đầu và thời gian kết thúc.

Các tác vụ được gửi cho nhà môi giới, nhà môi giới xác định VM phù hợp dựa trên một số nhiệm vụ được gửi như minh họa trong hình và mỗi VM thay đổi qua các trạng thái: Dưới tải, Cân bằng, Cân bằng cao, Quá tải.



Hình 2. 6 *Sơ đồ đề xuất LBDA* [107]

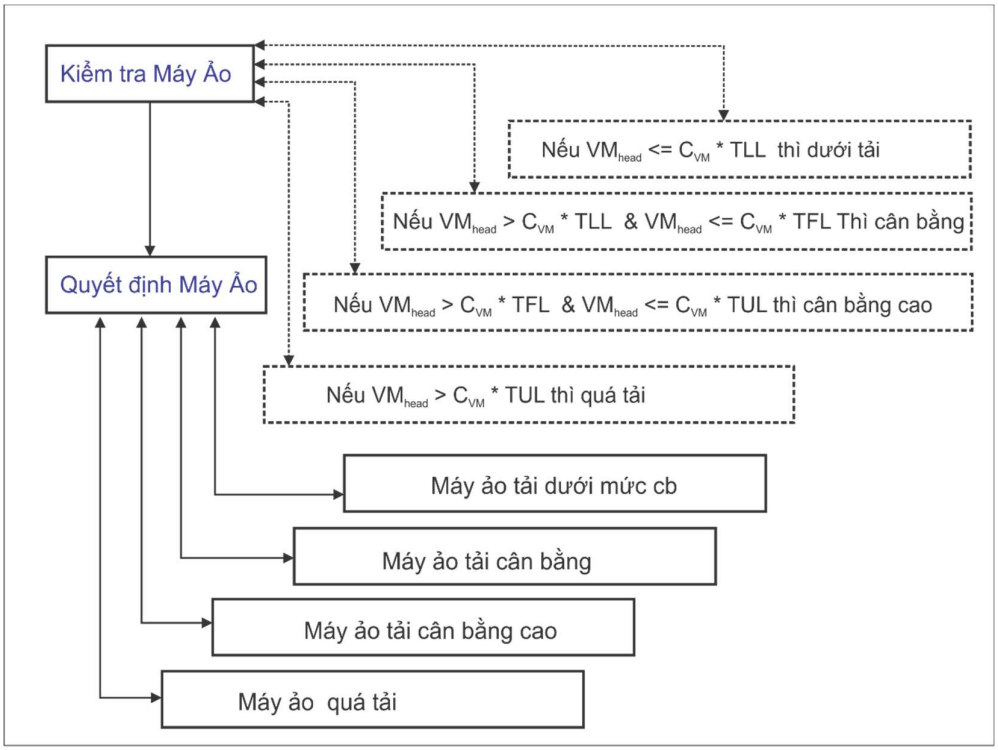
Trình lập lịch: Người dùng sẽ gửi Cloudlet_List, danh sách độc lập của Cloudlet làm đầu vào cho bộ lập lịch nhập yêu cầu để cân bằng. Cân bằng: Nó gán nhiệm vụ cho VM phù hợp được xác định từ đó gán nhiệm vụ cho VM dựa trên thông tin được thu thập. VMCheck: Kiểm tra trạng thái VM bằng cách so sánh dung lượng của VM với tải VM bằng ba loại ngưỡng: ngưỡng trên, ngưỡng cân bằng và ngưỡng dưới.

Dung lượng của VM (C_{VM}) được tính toán dựa trên công thức:

$$C_{VM} = Penum * Pemips$$

Trong đó, Penumis được định nghĩa là một số phần tử xử lý được phân bổ trong VM. Pemips là số lượng triệu hướng dẫn mỗi giây. Tải VM (VM_{load}) mà nó sẽ được tính toán dựa trên công thức: $VM_{Loadi} = \frac{\sum_{j=1}^{n} TL_j}{n}$

Trong đó, TL đề cập đến độ dài nhiệm vụ. Ba ngưỡng được chọn dựa trên quy trình trước đó, TUL giới hạn trên ngưỡng đầu tiên, đó là = 0.9, ngưỡng thứ hai cho Fair Limited (TFL), đó là = 0.8 và ngưỡng thứ ba cho Lower Limited (TLL), đó là = 0.2, như được minh họa trong hình



Hình 2. 7 ***Trạng thái máy ảo*** **[107]**

Giai đoạn 1. $VM_{loadi} <= C_{VMi} * TLL$ sau đó trạng thái VM được gắn nhãn là Dưới tải.

Giai đoạn 2. $VM_{loadi} > C_{VMi} * TLL$ và $VM_{loadi} <= C_{VMi} * TFL$ sau đó trạng thái VM được gắn nhãn là trạng thái Cân bằng.

Giai đoạn 3. $VM_{loadi} > C_{VMi} * TFL$ và $VM_{loadi} <= C_{VMi} * TUL$ sau đó trạng thái VM được gắn nhãn là Cân bằng cao.

Giai đoạn 4. VMloadi > CVMi ∗ TUL sau đó trạng thái VM được gắn nhãn là trạng thái Quá tải.

VM Quyết định: Việc lựa chọn VM phù hợp sẽ được quyết định và các tác giả tính đến lợi ích của người dùng và nhà cung cấp, trong đó việc tăng mức sử dụng tài nguyên, cân bằng tải và thời gian hoàn thành ít hơn.

Một tập hợp các tham số đang xem xét khi xây dựng thuật toán lập lịch tác vụ. Các tham số này đóng vai trò quan trọng để tăng hiệu suất đám mây tổng thể. Makespan: Là tổng thời gian hoàn thành của tất cả các nhiệm vụ hàng đợi VM. Một thuật toán lập lịch tốt luôn cố gắng giảm makepan xuống mức thấp nhất. Thời gian đáp ứng (RT): cho biết thời gian của quá trình tìm kiếm, trong đó có thời gian để thực hiện tác vụ trong hệ thống điện toán đám mây.

Nhóm tác giả đã so sánh thuật toán đề xuất với các thuật toán Round Robin, Max-Min và SJF trong cùng một môi trường cấu hình. Kết quả mô phỏng cho thấy thuật toán được đề xuất đã vượt trội hơn tất cả các trường hợp so với thuật toán Max-Min, SJF và Round Robin bằng cách giảm giá trị trung bình, thời gian phản hồi trung bình và tổng thời gian thực hiện trong tất cả các VM.

"A Modern Approach for Load Balancing Using Forest Optimization Algorithm" [108]

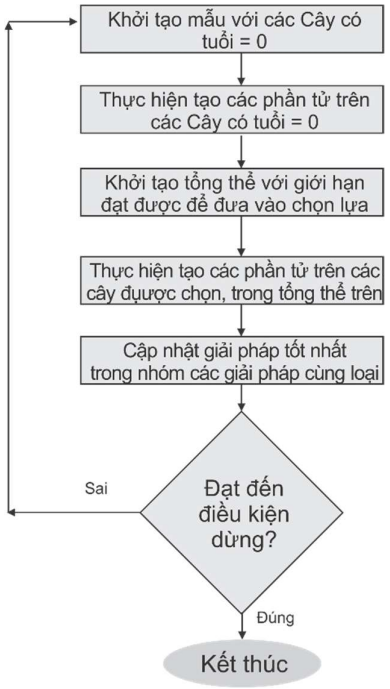
Năm 2018, nhóm tác giả đã trình bày một cách tiếp cận hiện đại để cân bằng tải bằng thuật toán tối ưu hóa rừng cây.

Theo nhóm tác giả, tải trọng cân bằng đóng vai trò năng động trong việc giữ nhịp độ của khung Điện toán đám mây. Bài viết nghiên cứu này đề xuất một thuật toán tối ưu hóa rừng để cân bằng tải trong cấu trúc điện toán phân tán. Điều này phụ thuộc vào hành vi của các cây trong rừng và sử dụng các chiến lược phát tán hạt giống để hợp lý hóa makespan (khoảng thời gian dùng để thực hiện dự án), giúp cải thiện thời gian phản hồi trung bình và tổng thời gian thực hiện. Các kết quả mô phỏng chứng minh rằng thuật toán được đề xuất cho kết quả tốt hơn so với các thuật toán cân bằng tải đối tác của nó.

Phân tán hạt giống là cốt lõi của thuật toán tối ưu hóa này. Một vài hạt rơi xuống gần cây mẹ và phát triển. Điều này được đặt tên là phân tán hạt giống địa phương. Tuy nhiên, phần lớn các hạt rõ ràng được chuyển hướng xa hơn từ cây mẹ

bởi một số toán tử đặc trưng. Theo cách này, cây phát triển ở các khu vực khác nhau của rừng. Kỹ thuật này được đặt tên là " Gieo hạt toàn cầu ".

Nhóm tác giả đã thực hiện phân tích so sánh các thuật toán tiến hóa như GA và PSO với thuật toán tối ưu hóa Forest (FOA) để cân bằng tải. Các kỹ thuật được sử dụng bởi các loại cây và cây được sử dụng để tìm kiếm tối ưu toàn cầu và địa phương của hàm chức năng.



Hình 2. 8 ***Sơ đồ của thuật toán FOA*** **[108]**

Kết quả mô phỏng chứng minh rằng chương trình thuật toán FOA được đề xuất đánh bại GA và PSO bằng cách giảm thiểu thời gian phản hồi trung bình và tổng thời gian thực hiện dưới tải trọng đáng kể. Trong mọi trường hợp, các tham số có thể được thay đổi để hoạt động tốt hơn trong các tải thấp hơn. Trong tương lai, có thể tìm kiếm các ước tính phù hợp của các tham số khung này để thực hiện điều chỉnh dưới tải trọng lớn, nhẹ và vừa phải.

"Proposed Load Balancing Algorithm To Reduce Response Time And Processing Time On Cloud Computing" [109]

Năm 2018, nhóm tác giả đã đề xuất thuật toán Throttled Modified Algorithm nhằm tối ưu hóa thời gian đáp ứng trên các máy ảo trong điện toán đám mây và tối ưu hóa Makespan của trung tâm dữ liệu đám mây.

Trình tự thực hiện của thuật toán:

- Bước 1: Trình cân bằng tải Throttled cải tiến thực hiện cân bằng tải bằng việc cập nhật, duy trì hai bảng chỉ mục:

- Một bảng chứa thông tin các máy chủ ảo (VM) ở trạng thái sẵn sàng ‘0’. (Available Index).
- Một bảng chứa thông tin các máy chủ ảo (VM) ở trạng thái không sẵn sàng ‘1’. (Busy Index).

Tại thời điểm bắt đầu, tất cả các máy chủ ảo (VM) đều được cập nhật trong bảng “Available Index” và bảng “Busy Index” là rỗng.

- Bước 2: Bộ điều khiển trung tâm (Data Center Controller) nhận được một request mới.

- Bước 3: Bộ điều khiển trung tâm (Data Center Controller) truy vấn đến trình cân bằng tải Throttled cải tiến cho phân bổ tiếp theo.

- Bước 4: Trình cân bằng tải Throttled cải tiến dò và gửi ID máy chủ ảo (VM) từ trên xuống trong bảng “Available Index” về bộ điều khiển trung tâm (Data Center Controller):

- Bộ điều khiển trung tâm (Data Center Controller) sẽ gửi yêu cầu tới máy chủ ảo xác định bởi ID đó.
- Bộ điều khiển trung tâm (Data Center Controller) thông báo tới trình cân bằng tải một phân bổ mới.
- Trình cân bằng tải Throttled cải tiến sẽ cập nhật ID máy chủ ảo (VM) vừa được gửi đó vào bảng “Busy Index” và chờ yêu câu mới từ Bộ điều khiển trung tâm (Data Center Controller).

Trường hợp, nếu bảng “Available Index” rỗng (tất cả các VM đang ở trạng thái không sẵn sàng).

- Trình cân bằng tải Throttled cải tiến sẽ trả giá trị về là -1 cho Bộ điều khiển trung tâm (Data Center Controller).
- Bộ điều khiển trung tâm (Data Center Controller) sắp xếp request.

- Bước 5: Về phía máy chủ ảo (VM) sau khi xử lý xong yêu cầu và bộ điều khiển trung tâm (Data Center Controller) nhận được cloudlet phản hồi, nó sẽ thông báo cho trình cân bằng tải Throttled cải tiến để trình cân bằng tải cập nhật lại bảng “Available Index”.

- Bước 6: Nếu có nhiều yêu cầu, bộ điều khiển trung tâm lặp lại Bước 3 và tiến trình được lập lại cho đến khi bảng "Available Index" rỗng.

Với thuật toán đề xuất trên sẽ giúp việc dò tìm máy chủ ảo đang sẵn sàng '0' với kích thước bảng "Available Index" thay đổi linh động hơn so với thuật toán Throttled. Điều này giúp tăng hiệu suất xử lý cho hệ thống.

"MMSIA: Improved Max-Min Scheduling Algorithm for Load Balancing on Cloud Computing"

Năm 2019, nhóm tác giả bài báo này đã đề xuất thuật toán MMSIA nhằm cải tiến thuật toán Max-Min nổi tiếng, nâng cao hiệu năng cân bằng tải trên đám mây. Bài viết này đề xuất một thuật toán MMSIA để cải thiện thuật toán lập lịch Max-Min, giúp cải thiện thời gian hoàn thành các yêu cầu bằng cách sử dụng học máy "đã học", bằng cách phân nhóm các yêu cầu và phân nhóm sử dụng các máy ảo. Sau đó, thuật toán chỉ định các yêu cầu cụm lớn nhất cho máy ảo với phần trăm sử dụng ít nhất, điều này được lặp lại khi danh sách yêu cầu trống. Đặc biệt, thuật toán MMSIA đã cải thiện thời gian hoàn thành. Kết quả mô phỏng cho thấy thuật toán MMSIA đề xuất đã cải thiện thời gian hoàn thành so với 3 thuật toán: Max-Min, Min-Min và Roud Robin.

"Look-ahead energy efficient VM allocation approach for data centers" [110]

Năm 2022, Liksen và cộng sự đã công bố nghiên cứu về vấn đề năng lượng cho các máy ảo. Bài báo tập trung về hiệu quả năng lượng, là một vấn đề quan trọng để giảm phát thải và lãng phí môi trường. Cung cấp tài nguyên hiệu quả năng lượng trong môi trường đám mây là một vấn đề đầy thách thức vì bản chất năng động của nó và các đặc điểm khối lượng công việc ứng dụng đa dạng. Trong tài liệu, việc di chuyển trực tiếp các máy ảo (VM) giữa các máy chủ thường được đề xuất để giảm tiêu thụ năng lượng và tối ưu hóa việc sử dụng tài nguyên, mặc dù nó đi kèm với những nhược điểm cơ bản, chẳng hạn như chi phí di chuyển và giảm hiệu suất. Việc cung cấp năng lượng hiệu quả được đề cập ở cấp độ trung tâm dữ liệu trong nghiên cứu này. Bài báo này đề xuất một thuật toán quản lý tài nguyên hiệu quả mới cho các trung tâm dữ liệu ảo hóa nhằm tối ưu hóa số lượng máy chủ để đáp ứng các yêu cầu của khối lượng công việc động mà không cần di chuyển. Cách tiếp cận được đề xuất, được đặt tên là Phân bổ VM Hiệu quả Năng lượng Nhìn trước (LAA), chứa mô-đun dự đoán dựa trên Holt Winters. Hiệu suất và hiệu suất năng lượng tỷ lệ nghịch. Việc

đánh đổi hiệu suất năng lượng dựa trên việc so sánh định kỳ số lượng máy chủ được dự đoán và đang hoạt động. Để đánh giá thuật toán được đề xuất, nhóm tác giả thử nghiệm với các khối lượng công việc trong thế giới thực từ Google Cluster. LAA được so sánh với cách tiếp cận tốt nhất do CloudSim cung cấp dựa trên việc di chuyển máy ảo được gọi là Thời gian di chuyển hồi quy cục bộ-tối thiểu (LR-MMT). Kết quả thử nghiệm cho thấy thuật toán được đề xuất giúp giảm mức tiêu thụ lên đến 45% để hoàn thành một khối lượng công việc so với LR-MMT.

2.4.3 CÔNG TRÌNH LIÊN QUAN CBT THEO HƯỚNG TIẾP CẬN BÊN NGOÀI

"Enhanced Load Balancing Approach to Avoid Deadlocks in Cloud" [111]

Vào năm 2013, tác giả Rashmi K.S. từ Ấn Độ đã đề xuất một giải pháp để cải thiện cân bằng tải và ngăn chặn tình trạng deadlock trên các môi trường đám mây thông qua nghiên cứu mang tên "Enhanced Load Balancing Approach to Avoid Deadlocks in Cloud". Với tính phân tán và ảo hóa của môi trường đám mây, có khả năng xảy ra tình trạng deadlock. Tác giả đã phát triển một thuật toán để tránh deadlock. Khi người dùng gửi yêu cầu tới đám mây, thuật toán này được kích hoạt để quản lý và xử lý tuần tự các máy ảo (VMs). Điều này được thực hiện bằng cách tăng cường số lượng nhiệm vụ xử lý trên đám mây để cải thiện hiệu suất và ngăn chặn tình trạng deadlock.

Bài báo đã đề xuất thuật toán mới dựa trên thuật toán có sẵn:

- Bước 1: bộ phận xử lý cân bằng tải sẽ tạo và duy trì một bảng chỉ mục (index) của các máy ảo và số lượng yêu cầu (request) đang được chuyển tới các máy ảo. Ban đầu, các máy ảo đều có 0 yêu cầu.
- Bước 2: Khi có một yêu cầu mới cần được phân bổ tới máy ảo từ trung tâm điều khiển (Data Center Control), sẽ đọc và phân tích bảng chỉ mục và tìm ra máy nào đang ít tải nhất. Nếu tìm ra nhiều hơn 1 máy thì sẽ lấy máy đầu tiên.
- Bước 3: Bộ xử lý cân bằng tải sẽ trả ra ID của mác ảo cho trung tâm điều khiển biết.
- Bước 4: Trung tâm điều khiển sẽ gửi yêu cầu tới máy ảo đã được tìm ở trên theo ID

- Bước 5: Trung tâm điều khiển sẽ thông báo cho bộ xử lý cân bằng tải về phân bổ này.
- Bước 6: Bộ xử lý cân bằng tải sẽ cập nhật vào bảng chỉ mục các máy ảo và tăng chỉ số lên của máy ảo.
- Bước 7: Khi máy ảo đã kết thúc thực thi xử lý yêu cầu, và trung tâm điều khiển sẽ nhận được thông tin này từ cloudlet, nó sẽ thông báo cho bộ cân bằng tải về việc chấm dứt phân bổ tới máy ảo.
- Bước 8: Bộ cân bằng tải sẽ cập nhật lại vào bảng chỉ mục và giảm thứ tự cho máy ảo xuống 1 đơn vị.
- Bước 9: lặp lại từ bước 2

Nhận thấy việc thông tin giữa Bộ phận cân bằng tải và trung tâm điều khiển trong việc cập nhật vào bảng chỉ mục sẽ dẫn tới nhiều tốn kém và chi phí. Vì thế, sẽ dẫn tới sự chậm trễ trong việc phản hồi và xử lý các yêu cầu. Từ đó, bài báo này đã đề xuất ra thuật toán nâng cao, sử dụng hệ thống quản lý cloud hiệu quả:

- Bước 1: khởi tạo các máy ảo với trạng thái 0, là sẵn sàng. Bộ quản lý cloud (Cloud Manager) ở trung tâm dữ liệu sẽ thu thập và lưu trữ một cấu trúc dữ liệu bao gồm Job ID, VM ID và trạng thái của VM.
- Bước 2: Khi có hàng đợi các yêu cầu, bộ quản lý cloud sẽ phân tích cấu trúc dữ liệu của phân bổ để xác định máy ảo nào ít sử dụng nhất. Nếu số máy là nhiều, thì máy ảo nào có thời gian chuyển (hop time) ít nhất sẽ được chọn.
- Bước 3: Bộ quản lý cloud sẽ cập nhật cấu trúc dữ liệu tự động ngay sau khi phân bổ các request.
- Bước 4: Bộ quản lý cloud sẽ giám sát trạng thái của các máy ảo đang có tải một cách tuần tự, nếu tìm ra máy ảo nào đang quá tải, thì sẽ chuyển tải của máy đó sang máy không tải.
- Bước 5: Ra quyết định chọn lực máy ảo mà ít tải dựa vào thời gian chuyển. Máy ảo nào có thời gian chuyển nhỏ nhất sẽ được chọn.
- Bước 6: Bộ quản lý cloud sẽ cập nhật cấu trúc dữ liệu bằng cách điều chỉnh toàn bộ theo thứ tự thời gian.
- Bước 7: Lặp lại từ bước 2.

Trong thuật toán được đề xuất thì Bộ quản lý Cloud phân tích khả năng và tính sẵn sàng của các máy ảo trong cùng một thời điểm các tác vụ đưa về, và cập nhật vào cấu trúc dữ liệu nhằm giảm các lãng phí so với thuật toán có sẵn.

"Deadlock Avoidance through Efficient Load Balancing to Control Disaster in Cloud Environment" [112]

Sự tăng trưởng nhanh chóng trong lĩnh vực đám mây điện toán đã tăng cường quản lý tài nguyên nghiêm trọng mối quan tâm. Quản lý tài nguyên vẫn không đổi vấn đề trong các hệ thống máy tính. Tài nguyên không hiệu quả quản lý sẽ dẫn đến Dead Lock và từ đó trở đi dẫn đến thảm họa. Một nghiên cứu gần đây chỉ ra rằng hiệu quả kinh doanh thấp do thiếu thảm họa quản lý trong mô hình điện toán. Bài viết này trình bày một kỹ thuật cân bằng tải hiệu quả để kiểm soát Thảm họa. Thuật toán điều chỉnh được áp dụng để cân bằng tải trong đó việc so sánh hiệu năng hệ thống được thực hiện với và không có bộ cân bằng tải. Thuật toán điều chỉnh là được triển khai bằng cách sử dụng công cụ Cloud Analyst. Các kết quả mô phỏng thu được mô tả một nguồn lực tốt hơn chia sẻ và quản lý với phản ứng tổng thể tối thiểu thời gian, thời gian xử lý và thông lượng tốt hơn.

Deadlock Detection for Resource Allocation in Heterogeneous Distributed Platforms [113]

Trong bài báo này, chúng tôi nghiên cứu phân bổ tài nguyên ở cấp cơ sở hạ tầng, thay vì nghiên cứu cách ánh xạ tài nguyên vật lý sang tài nguyên ảo để sử dụng tài nguyên tốt hơn trong môi trường điện toán đám mây. Chúng tôi đề xuất một thuật toán mới để phân bổ tài nguyên cho cơ sở hạ tầng, phân bổ động các máy ảo trong số các ứng dụng điện toán đám mây dựa trên phát hiện khóa chết thuật toán tiếp cận và có thể sử dụng phương pháp ngưỡng để tối ưu hóa quyết định phân bổ lại tài nguyên. Chúng tôi đã triển khai và thực hiện thuật toán của mình được đề xuất bằng cách sử dụng trình giả lập CloudSim. Kết quả thử nghiệm cho thấy thuật toán của chúng tôi có thể nhanh chóng phát hiện ra bế tắc và sau đó giải quyết tình huống xấp xỉ các đơn đặt hàng cường độ trong các trường hợp thực tế.

Detection and Avoidance Deadlock for Resource Allocation in Heterogeneous Distributed Platforms [114]

Trong một nền tảng phân tán không đồng nhất, vấn đề lớn được phân phối giữa các bộ xử lý (chẳng hạn như giữa các máy tính của hệ thống máy tính phân tán hoặc giữa các máy tính của hệ thống máy tính song song, v.v.) để làm cho nó hiệu quả và tốn ít thời gian hơn thay vì hệ thống máy tính có bộ xử lý đơn . Có nhiều loại vấn đề phân bổ tài nguyên chung hơn những vấn đề chúng tôi xem xét ở đây. Trong bài báo này, chúng tôi trình bày một cách tiếp cận để cải thiện thuật toán phát hiện và tránh, để lên lịch cho các chính sách cung cấp tài nguyên để phân bổ tài nguyên không đồng nhất. Chúng tôi đề xuất một thuật toán để phân bổ nhiều tài nguyên cho các dịch vụ cạnh tranh đang chạy trong nền tảng máy ảo.

Avoid Deadlock Resource Allocation (ADRA) Model V VM-out-of-N PM [115]

Bài viết này trình bày một phân bổ tài nguyên bế tắc (ADRA) cho mô hình V VMout-of-N PM vì điện toán đám mây là một mô hình điện toán mới bao gồm các khái niệm điện toán, điện toán phân tán và các khái niệm tiện ích. Điện toán đám mây trình bày một mô hình phân bổ tài nguyên khác với lưới hoặc hệ thống phân tán. Các nhà cung cấp dịch vụ đám mây tự động mở rộng quy mô tài nguyên máy tính ảo hóa như một dịch vụ qua internet. Do số lượng người dùng thay đổi và tài nguyên hạn chế, đám mây dễ bị bế tắc ở quy mô rất lớn. Phân bổ nguồn lực và tránh sự bế tắc liên quan là vấn đề bắt nguồn từ thiết kế và việc thực hiện điện toán phân tán, điện toán lưới. Trong bài báo này, một khái niệm mới về đám mây không gian trống được đề xuất để tránh bế tắc bằng cách thu thập tài nguyên miễn phí có sẵn từ tất cả người dùng được phân bổ. Các thuật toán mới được phát triển để phân bổ nhiều tài nguyên cho các dịch vụ cạnh tranh chạy trong các máy ảo trên nền tảng phân tán không đồng nhất. Một thử nghiệm được thử nghiệm trong CloudSim. Hiệu suất của trình quản lý nhóm tài nguyên được đánh giá bằng cách sử dụng CloudSim và sử dụng tài nguyên và cho biết kết quả tốt.

"User-Priority Guided Min-Min Scheduling Algorithm for Load Balancing in Cloud Computing" [17]

Vào năm 2013, tác giả Huankai Chen và đồng nghiệp đã giới thiệu một sự cải tiến cho thuật toán "Min-Min Scheduling" với sự tập trung vào ưu tiên của người dùng để cân bằng tải trên môi trường đám mây. Nghiên cứu này tập trung vào vấn đề

quan trọng của đám mây - tải không cân bằng. Tác giả đã giới thiệu thuật toán Min-Min để giảm thiểu khoảng trống và tăng khả năng sử dụng tài nguyên (gọi là LBIMM), cùng với phiên bản cải tiến của nó dựa trên ưu tiên của người dùng, được gọi là thuật toán PA-LBIMM. Nghiên cứu này đã được triển khai và kiểm tra trên môi trường giả lập MathLab, và kết quả thử nghiệm đã cho thấy sự cải thiện đáng kể, với sự tăng hiệu suất lên đến 20% đối với người dùng đặc biệt (VIP) và tạo ra sự tăng cường rõ rệt trong làm việc trên môi trường đám mây.

Trong bài báo này, đề cập tới thuật toán Min-Min truyền thống: là một trong những thuật toán cơ bản trong việc phân bổ nguồn tài nguyên trên đám mây. Nó bắt đầu bằng một tập S bao gồm các tác vụ chưa được ánh xạ. Từ đó tìm ra tài nguyên R mà tốn ít thời gian nhất để thực hiện tất cả các tác vụ trên. Sau đó, tác vụ T mới kích thước nhỏ nhất sẽ được chọn ra và được phân bổ theo nguồn tài nguyên R. Cuối cùng, tác vụ T sẽ bị xóa khỏi tập các tác vụ S, và sẽ lặp đi lặp lại cho đến khi hết tác vụ. Sơ đồ mã giả của thuật toán Min-Min được thể hiện ở hình sau:

```
1.  For all submitted tasks in the set; Ti
2.  For all resources; Rj
3.  Ctij=Etij+rtj; End For; End For;
4.  Do while tasks set is not empty
5.  Find task Tk that cost minimum execution time.
6.  Assign Tk to the resource Rj which gives
       minimum expected complete time
7.  Remove Tk from the tasks set
8.  Update ready time rtj for select Rj
9.  Update Cij for all Ti
10. End Do
```

Hình 2. 9 Sơ đồ mã giả thuật toán Min-Min [17]

Nghiên cứu chỉ ra rằng, từ kết quả thực nghiệm thì thuật toán Min-Min phải đối mặt với nhiều thách thức, và không đạt được hiệu quả khi sử dụng tối đa nguồn tài nguyên, và dẫn tới việc cân bằng tải không tốt. Do không có độ ưu tiên trong quá trình phân bổ, nên một số người dùng đặc biệt (VIP users) sẽ không được đảm bảo cung cấp dịch vụ tốt hơn, dẫn tới người dùng đặc biệt sẽ cảm thấy không thoải mái. Thuật toán LBIMM được đề xuất với mục tiêu tối ưu hóa cân bằng tải của Min-Min, nhằm tăng khả năng sử dụng nguồn tài nguyên hiệu quả, với các nguồn tài nguyên có tải nhẹ hay trạng thái chờ thì chia sẻ ra và giải phóng các nguồn lực

có tải nặng. Song song đó, độ ưu tiên của người dùng cũng được đề xuất trong thuật toán PA-LBIMM nhằm theo dõi và đảm bảo cho các khách hàng quan trọng có thể sử dụng dịch vụ tốt hơn người dùng thông thường. Thuật toán LBIMM (Load Balance Improved Min-Min) dựa trên tính chất của thuật toán Min-Min. Do điểm yếu nhất của Min-Min là không quan tâm tới các tải của các tài nguyên. Do vậy, sẽ có một số tài nguyên luôn bận và một số luôn rảnh. LBIMM sẽ cải tiến việc cân bằng tải của Min-Min và giảm thời gian thực thi ở các tài nguyên. Bắt đầu giống như Min-Min, nhưng ở bước thứ 2, thuật toán này sẽ chọn những tác vụ có kích thước nhỏ nhất trong nhóm tài nguyên có tải lớn nhất và tính toán ra thời gian hoàn tất các tác vụ đó. Và thời gian hoàn tất nhỏ nhất sẽ được so sánh với tính toán ở thuật toán Min-Min. Nếu nhỏ hơn Min-Min, thì tác vụ sẽ được giao lại cho nguồn tài nguyên mà nó sinh ra, và thời gian sẵn sàng cho các tài nguyên sẽ được cập nhật. Quá trình sẽ được lặp lại cho đến khi không còn tài nguyên nào mà có thời gian thực hiện nhỏ nhất mà tài nguyên đó có tải lớn. Sơ đồ mã giả như sau:

```
1. For all submitted tasks in the set; Ti
2. For all resources; Rj
3. Ctij=Etij+rtj; End For; End For;
4. Do while tasks set is not empty
5. Find task Tk that cost minimum execution time.
6. Assign Tk to the resource Rj which gives minimum expected
   complete time
7. Remove Tk from the tasks set
8. Update ready time rj for select Rj
9. Update Cij for all Ti
10.     End Do
11.  //Rescheduling to balance the load
12.       Do while the most heavy load resource is considered
   no need for rescheduling
13.       Find task Ti that cost minimum execution time on
   the heavy load resource Rj
14.       Find the minimum completion time of Ti produced by
   resource Rk
15.       If such minimum completion time < makespan
16.       Reassign Task Ti to Resource Rk
17.       Update the ready time of both Rj and Rk
18.       End If
19.  End Do
20. //where Makespan represents maximumcompletion time of all
   tasks which equals to the completion time of the most
   heavy load resource
```

Hình 2. 10 Sơ đồ mã giả thuật toán LBMin-Min [17]

"Generic SDE and GA-based workload modeling for cloud systems" [116]

Năm 2021, nhóm tác giả St-Onge đưa ra nghiên cứu theo hướng tiếp cận bên ngoài. Các mô hình khối lượng công việc thường được xây dựng dựa trên hành vi của người dùng và ứng dụng trong hệ thống, giới hạn chúng trong các miền cụ thể. Không nghi ngờ gì nữa, cách làm như vậy tạo ra tình thế tiến thoái lưỡng nan trong môi trường điện toán đám mây (đám mây), nơi một loạt các ứng dụng không đồng nhất đang chạy và nhiều người dùng có quyền truy cập vào các tài nguyên này. Mô hình khối lượng công việc trong một cơ sở hạ tầng như vậy phải thích ứng với sự phát triển của các tham số cấu hình hệ thống, chẳng hạn như sự dao động tải công việc. Mục đích của công việc này là đề xuất một cách tiếp cận tạo ra các mô hình khối lượng công việc chung (1) độc lập với hành vi của người dùng và các ứng dụng đang chạy trong hệ thống, đồng thời có thể phù hợp với bất kỳ miền và loại khối lượng công việc nào, (2) các biến thể khối lượng công việc rõ ràng có nhiều khả năng xuất hiện trong môi trường đám mây và (3) với mức độ trung thực cao đối với dữ liệu được quan sát, trong thời gian thực thi ngắn. Các tác giả đề xuất hai cách tiếp cận để ước tính khối lượng công việc, cách thứ nhất là sự kết hợp Hull-White và Giải thuật di truyền (GA), trong khi cách thứ hai là sự kết hợp Hỗ trợ Vector hồi quy (SVR) và Kalman-filter. Các thí nghiệm kỹ lưỡng được tiến hành trên CPU thực và bộ dữ liệu thông lượng từ Hệ thống con đa phương tiện IP (IMS), Web và môi trường đám mây được ảo hóa để nghiên cứu hiệu quả của cả hai đề xuất. Kết quả cho thấy độ chính xác cao hơn đối với phương pháp Hull-White-GA với chi phí biên trên kết hợp SVRKalman-Filter.

"Predicting Workflow Task Execution Time in the Cloud Using A Two-Stage Machine Learning Approach" [117]

Nhiều kỹ thuật như lập lịch và cung cấp tài nguyên dựa vào dự đoán hiệu suất của các tác vụ quy trình làm việc cho các dữ liệu đầu vào khác nhau. Tuy nhiên, những ước tính như vậy rất khó tạo ra trên đám mây. Bài báo này giới thiệu một phương pháp học máy hai giai đoạn mới để dự đoán thời gian thực hiện nhiệm vụ quy trình làm việc cho các dữ liệu đầu vào khác nhau trong đám mây. Để đạt được các dự đoán có độ chính xác cao, cách tiếp cận của nhóm tác giả dựa vào các tham số phản ánh thông tin thời gian chạy và hai giai đoạn dự đoán. Kết quả thực nghiệm

cho bốn ứng dụng quy trình làm việc trong thế giới thực và một số nhà cung cấp đám mây thương mại chứng minh rằng phương pháp tiếp cận của nhóm tác giả vượt trội hơn các phương pháp dự đoán hiện có. Trong các thử nghiệm của nhóm tác giả, phương pháp lần lượt đạt được sai số ước tính trong trường hợp tốt nhất và trường hợp xấu nhất là 1,6 và 12,2 phần trăm, trong khi các phương pháp hiện có đạt được lỗi vượt quá 20 phần trăm (đối với một số trường hợp thậm chí hơn 50 phần trăm) trong hơn 75 phần trăm quy trình công việc được đánh giá các nhiệm vụ. Ngoài ra, các tác giả cho thấy rằng các mô hình được dự đoán bởi phương pháp tiếp cận của họ cho một đám mây cụ thể có thể được chuyển với nỗ lực thấp sang các đám mây mới với sai số thấp bằng cách chỉ yêu cầu một số lượng nhỏ thực thi.

2.4.4 CLOUDSIM – CÔNG CỤ MÔ PHỎNG ĐÁM MÂY

Vào 2009, theo nghiên cứu tại phòng lab về Grid Computing của đại học Melbourne, Úc, nhóm tác giả Rodrigo N. Calheiros và cộng sự [118], họ đã công bố về CloudSim như là một khung mô phỏng tổng quát và có thể mở rộng mới cho phép lập mô hình, mô phỏng và thử nghiệm liền mạch các dịch vụ quản lý và cơ sở hạ tầng điện toán đám mây. Khung mô phỏng có các tính năng mới sau: (i) hỗ trợ mô hình hóa và khởi tạo cơ sở hạ tầng điện toán đám mây quy mô lớn, bao gồm các trung tâm dữ liệu trên một nút điện toán vật lý duy nhất và máy ảo; (ii) một nền tảng khép kín để lập mô hình trung tâm dữ liệu, môi giới dịch vụ, lập lịch trình và chính sách phân bổ; (iii) tính khả dụng của công cụ ảo hóa, hỗ trợ tạo và quản lý nhiều dịch vụ ảo hóa độc lập và đồng lưu trữ trên một nút trung tâm dữ liệu; và (iv) tính linh hoạt để chuyển đổi giữa phân bổ chia sẻ không gian và chia sẻ thời gian của các lõi xử lý sang ảo hóa dịch vụ.

Cũng theo công bố của Rodrigo N. Calheiros và cộng sự năm 2011 [119], thì CloudSim được xem là một bộ công cụ mô phỏng có thể mở rộng cho phép lập mô hình và mô phỏng các hệ thống điện toán đám mây và môi trường cung cấp ứng dụng. Bộ công cụ CloudSim hỗ trợ cả hệ thống và mô hình hóa hành vi của các thành phần Hệ thống đám mây như trung tâm dữ liệu, máy ảo (VM) và các chính sách cung cấp tài nguyên. Hiện tại, nó hỗ trợ mô hình hóa và mô phỏng môi trường điện toán Đám mây bao gồm cả đám mây đơn và liên mạng (liên kết các đám mây). Hơn nữa, nó hiển thị các giao diện tùy chỉnh để triển khai các chính sách và kỹ thuật cung cấp để

phân bổ máy ảo trong các kịch bản Điện toán đám mây liên mạng. Một số nhà nghiên cứu từ các tổ chức, chẳng hạn như HP Labs ở Hoa Kỳ, đang sử dụng CloudSim trong cuộc điều tra của họ về việc cung cấp tài nguyên Đám mây và quản lý hiệu quả năng lượng đối với tài nguyên trung tâm dữ liệu. Tính hữu ích của CloudSim được chứng minh bằng một nghiên cứu điển hình liên quan đến việc cung cấp động các dịch vụ ứng dụng trong môi trường đám mây liên kết lai. Kết quả của nghiên cứu điển hình này chứng minh rằng mô hình Điện toán đám mây liên kết cải thiện đáng kể các yêu cầu về QoS của ứng dụng trong điều kiện các mẫu nhu cầu dịch vụ và tài nguyên luôn biến động.

Theo Pravesh Humane và cộng sự năm 2015 [120], công cụ CloudSim là một dự án nguồn mở, cung cấp các lớp và tính năng chung có thể được sử dụng rộng rãi và người dùng có thể phát triển hành vi cụ thể theo từng trường hợp theo nhu cầu của riêng họ. Cloudim có nhiều lớp cung cấp trừu tượng khác nhau như: chính sách phân bổ máy ảo, trình cung cấp băng thông, trình lập lịch máy ảo (vm), trình lập lịch các request (Cloudlet), Chính sách phân bổ Power cho máy ảo, trình cung cấp bộ nhớ, v.v. Các chính sách này được 'mở rộng' và sau đó chiến lược cải tiến mới có thể được thực hiện; sau đó tích hợp nó vào gói nguồn thecloudsim. Do đó, điều này sẽ ứng biến công nghệ mô phỏng đám mây và phục vụ nhiều người dùng làm việc trong miền trung tâm dữ liệu. Vì vậy, phạm vi tương lai bao gồm sửa đổi các chính sách mặc định trừu tượng; điều này chắc chắn sẽ dẫn đến việc bổ sung các tính năng mới cho dự án cloudim mã nguồn mở này.

Trong một nghiên cứu khảo sát của Saydul Akbar Murad [121] và cộng sự năm 2022, đánh giá kỹ thuật lập lịch công việc trong điện toán đám mây và khung thông minh dựa trên quy tắc ưu tiên. Nhóm tác giả đã tổng hợp hơn 150 công trình nghiên cứu của các tác giả trên thế giới từ 2014 đến 2021. Kết quả cho thấy hơn 130 bài nghiên cứu về cân bằng tải trên điện toán đám mây sử dụng cloudSim như một công cụ mô phỏng đáng tin cậy cho nhà nghiên cứu. Bên cạnh đó, có rất ít bài báo sử dụng MatLab, hay Anaconda để mô phỏng. Chủ yếu sử dụng cloudSim và biến thể của CloudSim như CloudAnalyst, WorkflowSim hay tự điều chỉnh mã nguồn mở này. Ngoài ra, nghiên cứu này cung cấp một đánh giá toàn diện về các thuật toán lập lịch tác vụ và cung cấp tài nguyên trong điện toán đám mây, bao gồm các khái niệm, ưu

điểm và phân loại của các phương pháp tiếp cận khác nhau. Nó xác định nhu cầu xem xét các yếu tố, giới hạn và vi phạm SLA quan trọng về chất lượng dịch vụ (QoS) trong các thuật toán hiện có và đề xuất kỹ thuật lập lịch công việc dựa trên mức độ ưu tiên cho điện toán đám mây để cải thiện hiệu suất và tính công bằng. Nghiên cứu nhằm mục đích làm nền tảng cho việc điều tra thêm về lập lịch tác vụ hiệu quả trong điện toán đám mây.

Một nghiên cứu khảo sát khác về *"Lập lịch quy trình công việc trong môi trường đám mây – Thách thức, công cụ, hạn chế & phương pháp luận: Đánh giá*" với 50 công trình công bố từ 2014 đến 2021 [122], cũng cho thấy hơn 20 công trình sử dụng cloudSim và các công cụ biến thể từ cloudsim. Bên cạnh đó, với phân tích của nhóm tác giả M. Menaka, sẽ mở đường cho việc kết hợp các phương pháp học máy để tạo ra quy trình công việc cho doanh nghiệp, chăm sóc sức khỏe, nhận dạng giọng nói, nhận dạng văn bản; phát hiện buồn ngủ, phát hiện biển báo giao thông, nhận dạng bộ phận kim loại, v.v. Phân tích được thực hiện trong bài báo này dẫn đến nhiều thách thức và vấn đề khác nhau được tìm thấy trong các phương pháp lập lịch trình công việc. Bài báo khảo sát được tóm tắt với phần giới thiệu trong chương 1, sau đó là kết luận so sánh của các tài liệu nghiên cứu khác nhau và kết thúc bằng phần tóm tắt các phát hiện dựa trên cuộc khảo sát. Công việc khảo sát này có thể được mở rộng xem xét từng nhiệm vụ trong quy trình công việc và các loại & vấn đề di chuyển dữ liệu phát sinh ngân sách, các vấn đề pháp lý & chính sách và phương pháp kỹ thuật để tìm giải pháp khả thi cho tự động hóa nhiệm vụ.

2.5 KẾT LUẬN CHƯƠNG

Chương 2 đã sử dụng công cụ SWOT để phân tích và tiếp cận hiệu năng cân bằng tải trên cloud, từ đó đưa ra cách tiếp cận để nâng cao hiệu năng cân bằng tải trên cloud bao gồm hướng tiếp cận từ bên trong và hướng tiếp cận từ bên ngoài. Dựa vào 02 hướng tiếp cận đề xuất, chương này cũng trình bày một số công trình nghiên cứu liên quan có sự phân loại theo các tiếp cận từ bên trong hay từ bên ngoài của cân bằng tải. Các công trình nghiên cứu gần đây trong nước và quốc tế, các nghiên cứu về các kỹ thuật cân bằng tải trên môi trường điện toán đám mây, giúp hiểu rõ hơn về các hướng tiếp cận nâng cao hiệu năng cân bằng tải trên môi trường điện toán đám mây.

Hiểu được những ưu nhược điểm của các thuật toán hiện nay, và đưa ra các thuật toán cải tiến với 02 hướng tiếp cận, mục đích là nâng cao hiệu suất cân bằng tải, từ những đề xuất đó giúp làm cơ sở cho việc đề xuất thêm kỹ thuật mới, ứng dụng học máy và phân tích dữ liệu trong cân bằng tải.

CHƯƠNG 3 – **CÂN BẰNG TẢI THEO HƯỚNG TIẾP CẬN BÊN TRONG**

3.1 GIỚI THIỆU CHUNG

Chương này tập trung vào việc phát triển các thuật toán tối ưu cân bằng tải trong môi trường điện toán đám mây, nắm bắt sự phát triển mạnh mẽ của học máy (ML) và phân tích dữ liệu để giúp cân bằng tải đạt hiệu quả cao hơn. Cụ thể, luận án đưa ra một loạt các giải pháp sáng tạo, trong đó bao gồm cả việc tận dụng các thuật toán phân lớp, phân cụm và dự báo. Bộ tham số được đề xuất để đánh giá cân bằng tải bao gồm Thời gian đáp ứng (Response Time) và Thời gian thực thi (Makespan), nhằm định lượng hiệu quả của việc phân phối tải trên hệ thống. Cụ thể, có bốn thuật toán được đề xuất như sau:

- Thuật toán MCCVA [CT1]: Thuật toán này là một sự kết hợp tinh tế giữa SVM (Support Vector Machine), một phương pháp phân lớp mạnh mẽ, và k-Means, một kỹ thuật phân cụm thông dụng. Sự hợp nhất của SVM và k-Means cho phép thuật toán này phân tích dữ liệu một cách linh hoạt và phân phối tải dựa trên tính năng của dữ liệu, nhắm đến việc giảm Thời gian đáp ứng và Makespan.
- Thuật toán APRTA [CT2]: Thuật toán này tận dụng sức mạnh của ARIMA, một mô hình dự báo thống kê, để dự đoán Thời gian đáp ứng. Kỹ thuật này giúp cải thiện việc cân bằng tải bằng cách dự đoán nhu cầu tài nguyên và phân phối tải trước khi có sự gia tăng đột biến, từ đó tối ưu hóa Thời gian xử lý và Makespan.
- Thuật toán RCBA [CT7]: Sự kết hợp của Naïve Bayes, một phương pháp phân lớp xác suất, và k-Means trong thuật toán này đem đến khả năng phân loại và phân cụm dữ liệu hiệu quả. Nó nhằm đến việc phân phối tải trên các nút mạng một cách cân đối, giảm thiểu Thời gian đáp ứng và Makespan thông qua việc tận dụng lợi thế của việc phân tích dữ liệu phân loại.
- Thuật toán ITA [CT3]: Thuật toán này là một bước cải tiến từ Throttle Algorithm, một thuật toán cân bằng tải nổi tiếng. ITA tập trung vào việc hiệu chỉnh việc phân phối tải dựa trên Thời gian đáp ứng và Makespan,

giúp đạt được hiệu quả cao hơn so với phiên bản gốc bằng cách áp dụng các kỹ thuật tiên tiến từ học máy.

Sự liên kết giữa các thuật toán này không chỉ nằm ở việc chia sẻ bộ tham số cân bằng tải mà còn trong việc chúng hỗ trợ lẫn nhau thông qua việc áp dụng các phương pháp tiếp cận dữ liệu đa dạng phong phú. Về cài đặt mô phỏng các thuật toán, luận án giả lập môi trường cloud bằng cách sử dụng bộ công cụ CloudSim [123] và lập trình trên ngôn ngữ JAVA [124].

Môi trường thực nghiệm mô phỏng được tạo ra bằng cách sử dụng từ 5 đến 15 máy ảo trong môi trường giả lập đám mây, và các yêu cầu ngẫu nhiên được tạo ra và gửi đến các dịch vụ trên đám mây. Điều này bao gồm các dịch vụ cung cấp máy ảo và dịch vụ phục vụ người dùng của CloudSim để thực hiện các thử nghiệm.

Các thuật toán được đề xuất đã được triển khai bằng ngôn ngữ Java và được thử nghiệm và hiển thị kết quả thông qua giao diện dòng lệnh sử dụng IDE APACHE NETBEAN [125]. Môi trường giả lập được xây dựng bằng việc sử dụng thư viện mã nguồn mở CloudSim 4.0, được cung cấp tại địa chỉ http://www.cloudbus.org/.

Bên cạnh đó, các thuật toán SVM, K-Means, Naïve Bayes, Linear Regression, ARIMA được cài đặt từ bộ thư viện Weka [126] và Tensorflow Java [127], kết hợp với mã nguồn mở CloudSim (môi trường mô phỏng) để cài đặt các thuật toán đề xuất. Tuy nhiên, với thuật toán ITA thì cấu hình và cài đặt trên CloudAnalyst tool [128], là một phiên bản của CloudSim nhưng có đầy đủ GUI.

3.2 THUẬT TOÁN MCCVA

Dựa vào yếu tố thời gian xử lý (Makespans) của các request và một số thuộc tính khác, ta sử dụng thuật toán SVM để phân lớp các request này, từ đó ta biết cách phân bố tài nguyên cho các request này. Song song, các tài nguyên (máy ảo/ host) được phân cụm theo mức độ sử dụng. Bằng việc kết hợp đánh giá số lần sai và sai số, chúng ta nâng cao hiệu suất của thuật toán thông qua việc áp dụng học máy. Tuy nhiên, việc sử dụng học máy sẽ hạn chế do có một mức sai số cho phép.

Dựa vào các công trình nghiên cứu về Makespan, xin đề xuất thuật toán gồm 3 nhóm module chính:

(1) Module phân lớp các request bằng thuật toán SVM:

Trong module này, thuật toán SVM sẽ dựa vào các thuộc tính của request mà tính toán ra thời gian xử lý của request đó, từ đó phân lớp request này. Các thuộc tính bao gồm: size, Response length, Max Length, …

Nhóm Thời Gian xử lý : $MK_{New} = SVM(X_1, X_2, ..., X_n)$ (5.1)

Trong đó X_i là các thuộc tính của Request khi gửi lên cloud.

Ở đây có thể chia thành nhiều nhóm (từ 4 ~10 nhóm) hoặc hơn nữa dựa vào độ biến thiên của Request.

(2) Module phân cụm các máy ảo / host / tài nguyên:

Trong module này, sẽ áp dụng thuật toán phân cụm K-Means với giá trị k=3 để nhóm các máy ảo lại với nhau dựa trên hoạt động và sử dụng tài nguyên của chúng, bao gồm các nhóm cao, trung bình và thấp. Quá trình phân cụm này sẽ dựa trên thông số hiện tại của các máy ảo:

$Cluster_i = KMeans(cpu\ usage, ram, ...)$ (5.2)

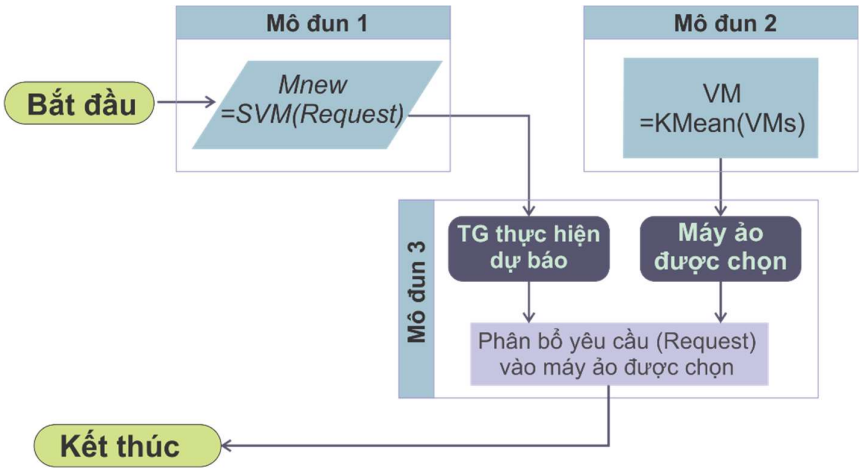
Trong đó: i= 1 là nhóm thấp

i = 2 là nhóm trung bình

i = 3 là nhóm cao

(3) Module phân bổ các dịch vụ (chọn máy ảo)

Module này thực hiện nhiệm vụ phân chia các yêu cầu đến các máy ảo dựa trên loại yêu cầu và cụm máy ảo thích hợp. Cho dù một yêu cầu được gửi, nó sẽ được xử lý bằng Module 1 và sau đó các máy ảo, kể cả những máy ảo không tải, sẽ được phân loại bằng Module 2. Sau đó thuật toán sẽ tính toán ra request nào phù hợp với máy ảo nào nhất thông qua thông số trả về của 2 hàm SVM và K-Means ở trên. Nếu thời gian xử lý tính toán của Request đang xét (được tính toán từ module 1) nhỏ nhất thì yêu cầu này sẽ được xử lý trên VM với mức độ xa means nhất (tức là thuộc nhóm 1, và có mức độ sử dụng thấp nhất). Đối với các request không nhỏ không lớn ta có thể dùng các phương pháp tính toán như loại suy hay sai phân để tính toán việc phân bổ.



Hình 3. 1 ***Sơ đồ thuật toán MCCVA***

Sơ đồ mã giả thuật toán ***MCCVA***

Thuật toán MCCVA

Input: Tập requests

Output: Phân bổ máy ảo cho các request trong tập Request

```
1.  For each Request in CloudRequests
2.     isLocated = true;
3.     MakeSpan_Classnew = SVM(T1,T2…..);  // Module 1
4.     VM_Cluster = kMeans(situation);  //situation: Trạng thái của
các VM Module 2
5.     For each VM in VMList
6.        If isFitSituation(Request.MakeSpan_Class ,
                VM.VM_Cluster)
7.              AllocateRequestToVM(VM, Request); // Module 3
8.              isLocated = true;
9.              break;
10.       End If
11.    End For
12.    If (!isLocated)
13.       VM = VMList.getMinFromMean();  // Module 2
14.       AllocateRequestToVM(VM, Request);
15.    End If
16. End For
```

Theo thuật toán đề xuất MCCVA, đầu ra của phân lớp request được tính toán chính là thời gian xử lý, và không biết được giá trị max hay giá trị min, nên có thể lưu lại 1 số lượng nhất định thời gian xử lý của các request trước nhằm thực hiện tính toán và phân bổ. Do đó, thuật toán này áp dụng phương pháp Newton-Raphson để xác định vị trí cân đối cần thiết, nhưng với việc điều chỉnh các biến đổi nhất định hoặc tích hợp thêm các hệ số và biến số khác, dựa trên dữ liệu thu được từ mô phỏng thực nghiệm.

CÀI ĐẶT THUẬT TOÁN MCCVA

Các tham số của mô hình mạng mô phỏng cài đặt chung:

- Môi trường mô phỏng giả lập gồm các thông số sau:

- 01 Datacenter với thông số như bảng 3.1

Bảng 3. 1 Thông số cấu hình Datacenter thuật toán MCCVA

Thông tin Datacenter	*Thông tin Host trong Datacenter*
- Số lượng máy (host) trong datacenter: 5 - Không sử dụng Storage (các ổ SAN) - Kiến trúc(arch): x86 - Hệ điều hành (OS): Linux - Xử lý (VMM): Xen - TimeZone: +7 GMT - Cost: 3.0 - Cost per Memory: 0.05 - Cost per Storage: 0.1 - Cost per Bandwidth: 0.1	Mỗi host trong Datacenter có cấu hình như sau: - CPU có 4 nhân, mỗi nhân có tốc độ xử lý là 1000 (mips) - Ram: 16384 (MB) - Storage: 1000000 Mb - Bandwidth: 10000 Kbps

- Các máy ảo khi được tạo ra dựa trên thông tin từ bảng 3.2, tất cả các máy ảo đều có cấu hình tương tự nhau như sau:

Bảng 3. 2 Cấu hình máy ảo thuật toán MCCVA

Kích thước (size)	**Ram**	**Mips**	**Bandwidth (Kbps)**	**Số lượng cpu (pes no.)**	**VMM**
10000 MB	512 MB	250	1000	1	Xen

- Trong mô phỏng bằng cloudSim, các yêu cầu (request) chạy trên web được mô phỏng dưới dạng Cloudlet, và kích thước của chúng được xác định một cách ngẫu nhiên sử dụng hàm random của JAVA. Số Cloudlet được tạo ra có số lượng từ 20 cho đến 1000.

Bảng 3. 3 Cấu hình thông số các Request thuật toán MCCVA

Chiều dài (Length) Kb	**Kích thước file (File Size) Kb**	**Kích thước file xuất ra (Output Size)**	**Số CPU xử lý (PEs)**
3000 ~ 1700	5000 ~ 45000	450 ~ 750	1

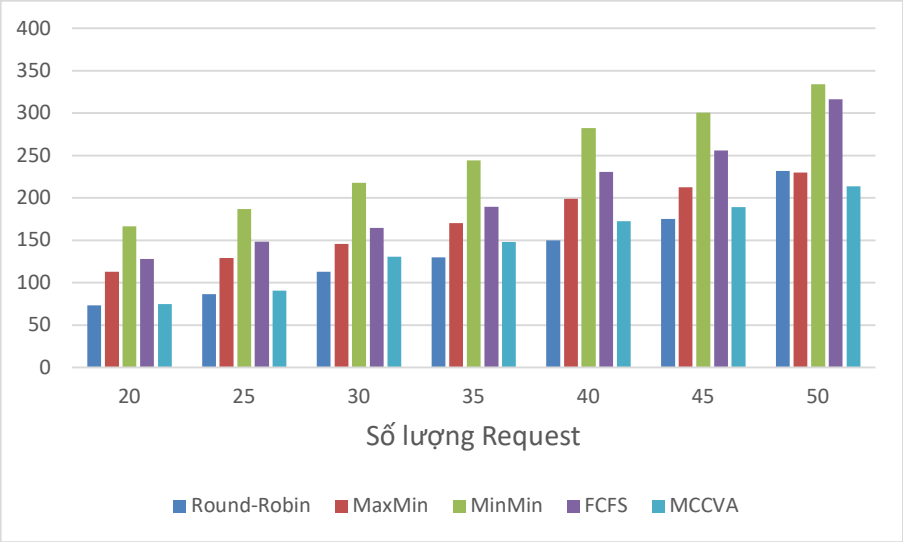
Thuật toán *MCCVASchedulingAlgorithm* được phát triển bằng cách mở rộng lớp *BaseSchedulingAlgorithm*, bổ sung thêm các phương thức và thuộc tính mới cho *predictRequestSVM*, và chỉnh sửa các hàm có sẵn để chúng tương thích với thuật toán được đề xuất:

```
@Override
public void run()  // Module 3
public CondorVM getMostFreeVM(String vmClass)
// Module 2
public String predictRequestSVM(Cloudlet req)
// Module 1
```

Trong một môi trường mô phỏng sử dụng CloudSim, thí nghiệm đã được thực hiện trên 5 máy ảo cấu hình sẵn để xử lý các Request. Các Request này được tạo một cách ngẫu nhiên với kích thước và chiều dài khác nhau, với số lượng dao động từ 20 đến 50. Kết quả sau đó được so sánh với hiệu suất của các thuật toán Round Robin, MaxMin, MinMin và FCFS, dựa trên thời gian hoàn thành công việc (Makespan).

Bảng 3. 4 So sánh thời gian thực hiện các thuật toán với thuật toán MCCVA ở trường hợp 50 Request

Số lần request	Round-Robin	MaxMin	MinMin	FCFS	MCCVA
20	**73,48**	112,81	166,77	127,95	74,70
25	**86,50**	129,10	186,93	148,70	90,67
30	**112,73**	145,79	217,82	164,41	130,82
35	**129,81**	170,22	244,26	189,48	147,89
40	**150,10**	198,91	282,34	230,53	172,85
45	**175,15**	212,54	300,53	256,17	189,16
50	231,83	229,80	334,34	316,47	**213,57**

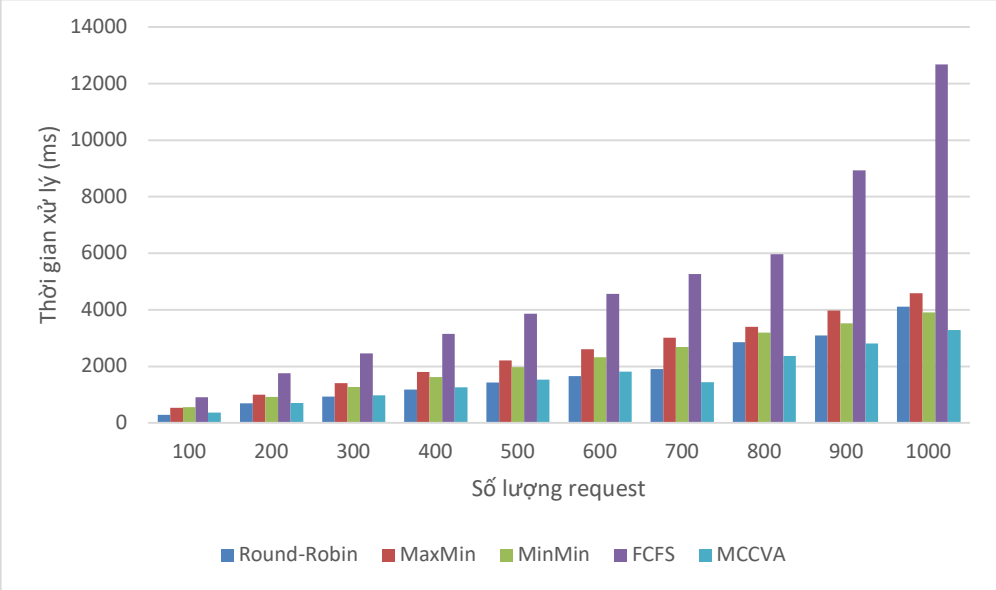


Hình 3. 2 Biểu đồ so sánh thời gian thực hiện các thuật toán với thuật toán MCCVA ở trường hợp 50 Request

Với kết quả thực nghiệm với 50 Request trở lại, *Round-Robin:* Thời gian thực hiện của thuật toán này tăng dần theo số lần request. Tuy nhiên, thời gian thực hiện của nó thường nhanh hơn so với các thuật toán khác. *MaxMin:* Thời gian thực hiện của thuật toán này cũng tăng dần theo số lần request. Tuy nhiên, thời gian thực hiện của nó thường lớn hơn so với Round-Robin. *MinMin:* Thời gian thực hiện của thuật

toán này cũng tăng dần theo số lần request và lớn hơn cả MaxMin. Đây có thể là một điểm yếu của thuật toán trong việc cân bằng tải. *FCFS (First-Come, First-Served):* Thời gian thực hiện của thuật toán này cũng tăng dần theo số lần request và lớn hơn cả MinMin. Thuật toán này có thể gặp khó khăn trong việc xử lý các request theo thứ tự đến trước. *MCCVA:* Thời gian thực hiện của thuật toán này cũng tăng dần theo số lần request, nhưng thời gian thực hiện của nó thường nhanh hơn so với các thuật toán khác, kể cả Round-Robin. Nhìn chung, Round-Robin và MCCVA có thời gian thực hiện tốt hơn so với MaxMin, MinMin và FCFS trong việc cân bằng tải. Bên cạnh đó, thuật toán Round-Robin chiếm ưu thế và xử lý nhanh, thuật toán MaxMin cũng khá ổn định. Thuật toán FCFS thì chưa có thế mạnh. Tuy nhiên thuật toán đề xuất MCCVA cũng khá ổn định, và chứng tỏ dần ổn định và tốt hơn khi xử lý nhiều request hơn.

Trong thực nghiệm sử dụng CloudSim, 5 máy ảo được thiết lập từ trước phục vụ các yêu cầu, mà các yêu cầu này được sinh ra ngẫu nhiên về chiều dài và dung lượng. Số lượng Request được tạo ra có sự biến động từ 100 đến 1000:



Hình 3. 3 Biểu đồ so sánh thời gian thực hiện các thuật toán với thuật toán MCCVA ở trường hợp 1000 Request

Bảng 3. 5 So sánh thời gian thực hiện các thuật toán với thuật toán MCCVA ở trường hợp 1000 Request

Số lần request	Round-Robin	MaxMin	MinMin	FCFS	MCCVA
100	**280,46**	529,01	564,25	915,94	359,47
200	**685,99**	995,76	923,20	1.746,95	693,87
300	**936,88**	1.402,47	1.276,19	2.450,61	972,44
400	**1.175,86**	1.806,76	1.625,49	3.148,05	1.253,25
500	**1.419,01**	2.205,04	1.978,60	3.856,57	1.531,72
600	**1.659,67**	2.605,43	2.323,70	4.565,09	1.810,83
700	1.898,65	3.004,55	2.673,20	5.258,63	**1.444,51**
800	2.848,74	3.400,15	3.184,27	5.967,59	**2.272,32**
900	3.097,55	3.978,18	3.521,57	8.930,12	**2.801,84**
1000	4.114,58	4.579,85	3.899,78	12.682,62	**3.282,37**

Với số lượng Request lần lượt là 100 đến 1000, *Round-Robin:* Thời gian thực hiện của thuật toán tăng dần theo số lần request. Tuy nhiên, thời gian thực hiện của nó thường nhanh hơn so với các thuật toán khác, đặc biệt khi số lần request lớn. *MaxMin:* Thời gian thực hiện của thuật toán này cũng tăng dần theo số lần request. Nó có thời gian thực hiện lớn hơn so với Round-Robin, và khoảng cách này có xu hướng tăng khi số lần request tăng. *MinMin:* Thời gian thực hiện của thuật toán này cũng tăng dần theo số lần request và lớn hơn cả MaxMin. Khoảng cách giữa thời gian thực hiện của MinMin và MaxMin tăng khi số lần request tăng. *FCFS (First-Come, First-Served):* Thời gian thực hiện của thuật toán này cũng tăng dần theo số lần request và lớn hơn cả MinMin. Khoảng cách giữa thời gian thực hiện của FCFS và MinMin tăng khi số lần request tăng. *MCCVA:* Thời gian thực hiện của thuật toán này tăng dần theo số lần request. Nó thường nhanh hơn so với các thuật toán khác, nhưng thời gian thực hiện của nó cũng tăng khi số lần request tăng. Như vậy, có thể thấy thuật toán MCCVA vượt trội hơn hẳn so với MaxMin, MinMin. Tuy nhiên vẫn chưa thấy ưu thế so với RoundRobin. Nhưng với số lượng request càng lớn thì MCCVA càng lợi thế hơn. Và dần dần chiếm ưu thế tuyệt đối so với các thuật toán còn lại. Rõ ràng FCFS thể hiện sự thiếu thông minh và tính tự nhiên của giải thuật.

Thông qua 02 biểu đồ so sánh thời gian xử lý của các thuật toán với điều kiện như nhau ta có thể thấy sự phân bổ khá ổn định và hợp lý của thuật toán đề xuất MCCVA, thời gian xử lý của các máy ảo khả quan so với thời gian xử lý của các thuật toán khác trên cloud (ở trường hợp ít và nhiều request).

Khi số lượng request càng lớn, MCCVA có lợi thế hơn vì khi với số lượng request lớn, SVM có thể xây dựng một mô hình phân lớp chính xác hơn. SVM có khả năng học từ dữ liệu và tìm ra các đặc trưng quan trọng để phân loại các request vào

các nhóm tài nguyên phù hợp. Điều này giúp MCCVA hiểu rõ hơn về yêu cầu và phân bố tài nguyên cho từng request. Ngoài ra số lượng request lớn cũng đồng nghĩa với việc có nhiều tài nguyên (máy ảo/ host) trong hệ thống. Khi áp dụng phương pháp phân cụm kMeans, MCCVA có thể tạo ra các nhóm tài nguyên có sự đa dạng và đại diện cho các mức độ sử dụng khác nhau. Điều này giúp MCCVA phân bố tài nguyên một cách hiệu quả và cân bằng tải tốt hơn khi có số lượng lớn các request cần được xử lý.

Tóm lại, khi số lượng request càng lớn, MCCVA có thể học từ dữ liệu và phân bố tài nguyên dựa trên SVM và phân cụm kMeans một cách chính xác và hiệu quả hơn. Điều này giúp MCCVA đáp ứng yêu cầu cân bằng tải trong hệ thống đám mây một cách tốt nhất khi có số lượng lớn các request.

3.3 THUẬT TOÁN APRTA

Từ dữ liệu chuỗi thời gian hiện có liên quan đến thời gian đáp ứng, luận án này giới thiệu việc áp dụng thuật toán *ARIMA* cho việc dự đoán thời gian đáp ứng của request tiếp theo, giúp việc phân bổ nguồn lực cho request tiếp theo được tối ưu hóa. Thuật toán này được đặt tên là APRTA (Arima Prediction Response Time Algorithm). Để tăng cường hiệu quả của APRTA, luận án cũng xem xét việc sử dụng machine learning dựa trên số lượng lần dự đoán sai và độ lệch sai số, mặc dù sự áp dụng này chỉ diễn ra khi sai số nằm trong ngưỡng cho phép. Dựa trên thông tin từ tài liệu [7], APRTA được cấu trúc thành ba module chính:

(1) Module tính toán ngưỡng bằng thuật toán ARIMA:

Trong module này, ngưỡng chính là thời gian đáp ứng dự báo của Cloud, sẽ được tính toán bằng thời gian đáp ứng dự báo bằng thuật toán ARIMA, và sẽ tăng hoặc giảm tùy theo thời gian đáp ứng theo dữ liệu chuỗi thời gian. Ngưỡng mới chính là thời gian đáp ứng dự đoán xét trong tập các VM đang xét trong vòng 100 request gần nhất.

Ngưỡng mới : $T_{New} = ARIMA(RT_1, RT_2, \ldots, RT_{100})$ (6.1)

Trong đó RT_i là chuỗi thời gian đáp ứng ghi lại được của cloud (chỉ xét trong vòng 100 Request gần nhất)

(2) Module dự báo thời gian đáp ứng tiếp theo cho từng máy ảo:

Module này áp dụng thuật toán ARIMA để dự đoán thời gian phản hồi tiếp theo cho từng máy ảo, sử dụng dữ liệu từ 50 yêu cầu (Request) gần nhất được xử lý bởi máy ảo đó, thực hiện thông qua hàm *getPredictedRT()*. Nó cũng tính toán và cung cấp giá trị dự đoán chính xác nhất cho mỗi máy ảo dựa trên một ngưỡng xác định, qua hàm *AllocateRequestToVM(VM, Request)*;

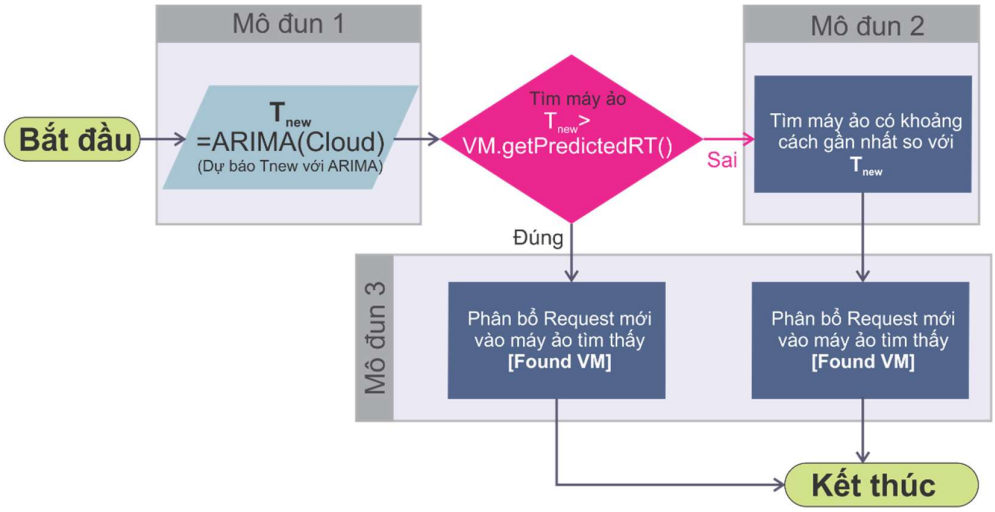
PRT_i = Prediected Response Time = Thời gian đáp ứng dự đoán của máy ảo i.

(3) Module phân bổ các dịch vụ (chọn máy ảo)

Nhiệm vụ của module này là giao các yêu cầu tới máy ảo mà thời gian đáp ứng dự kiến nằm trong giới hạn đã đưa ra. Khi có yêu cầu mới đến và máy ảo được chọn không quá tải, yêu cầu đó sẽ ngay lập tức được chuyển cho máy ảo này xử lý, đồng thời thu thập được thời gian đáp ứng thực tế tương ứng và lưu vào bộ dữ liệu. Nếu máy ảo này có thời gian đáp ứng dự báo (tính toán bởi module 2) ít hơn so với thời gian đáp ứng tiếp theo của toàn bộ hệ thống cloud (xác định bởi module 1), thì máy ảo sẽ được lựa chọn để xử lý yêu cầu. Trong trường hợp không có máy ảo nào đáp ứng được ngưỡng thời gian đáp ứng, yêu cầu sẽ được chuyển tới máy ảo với dự báo thời gian đáp ứng gần với ngưỡng nhất.

Việc khởi tạo ngưỡng: ban đầu do chưa có dữ liệu thời gian, nên ta lấy ngưỡng bằng thời gian đáp ứng của request đầu tiên.

khởi tạo ngưỡng: $T_{initial} = RT_1$



Hình 3. 4 ***Sơ đồ thuật toán APRTA***

Sơ đồ mã giả thuật toán *APRTA*

Thuật toán APRTA
Input: Tập requests
Output: Phân bổ máy ảo cho các request trong tập Request

```
1.   For each Request in CloudRequests
2.      Tnew = ARIMA(RTi);  // Module 1
3.      isLocated = false;
4.      For each VM in VMList
5.         If VM.getPredictedRT()  < Tnew
6.            AllocateRequestToVM(VM, Request); // Module 3
7.            isLocated = true;
8.            break;
9.         End If
10.     End For
11.     If (!isLocated)
12.        VM = VMList.getMinDistance(Tnew);  // Module 2
13.        AllocateRequestToVM(VM, Request);
14.     End If
15.  End For
```

CÀI ĐẶT THUẬT TOÁN APRTA

Thuật toán được đề xuất phát triển thông qua việc thiết kế lớp *ArimaDatacenterBroker*, dựa trên cơ sở kế thừa từ lớp *DatacenterBroker*. Trong quá trình này, lớp mới được bổ sung thêm các phương thức và thuộc tính liên quan đến *PredictedResponseTime*, cũng như điều chỉnh những phương thức hiện có để chúng phù hợp hơn với thuật toán đề xuất:

processResourceCharacteristics(SimEvent ev)

createVmsInDatacenter(int datacenterId)

processVmCreate(SimEvent ev)

processCloudletReturn(SimEvent ev)

Tiêu chí đánh giá:

Trong khuôn khổ của một thực nghiệm mô phỏng môi trường cloud dựa trên các thông số đã được xác định, luận án tiến hành mô phỏng thuật toán cân bằng tải tích hợp sẵn trong CloudSim. Đồng thời, cũng triển khai và vận hành mô phỏng thuật toán APRTA đề xuất. Các thuật toán này đều được chạy với cùng một bộ dữ liệu đầu vào để có thể tiến hành so sánh chính xác và công bằng giữa các kết quả đầu ra của chúng, đặc biệt quan tâm đến thông số thời gian đáp ứng (Response Time). Chú trọng đến việc đánh giá và so sánh thời gian đáp ứng dự đoán của từng máy ảo cũng như

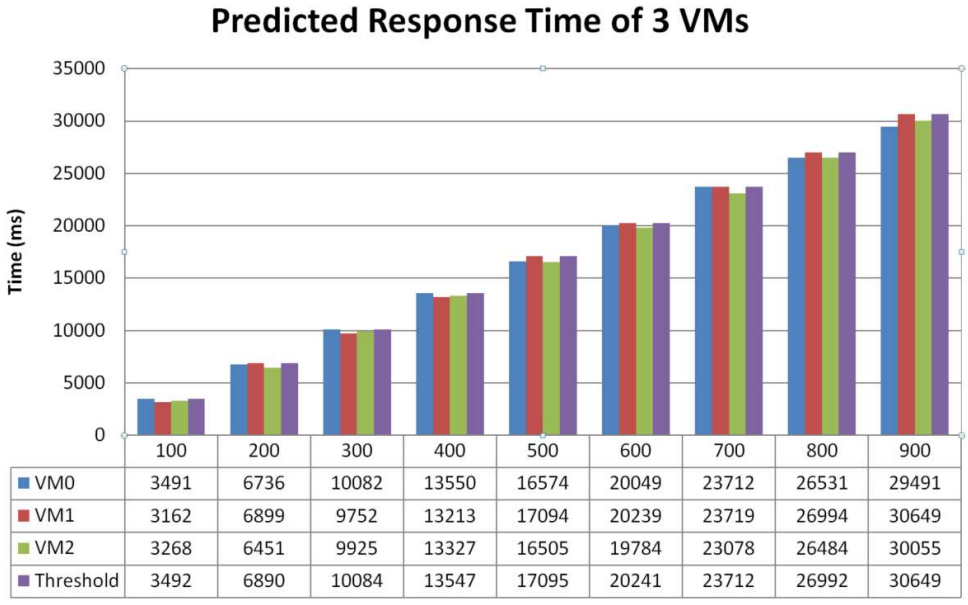
của toàn bộ hệ thống cloud, có nghĩa là mức độ chính xác càng cao, tức là sai số càng thấp, thì chứng tỏ hiệu quả của thuật toán càng được cải thiện. Kết quả này sẽ giúp chúng ta hiểu rõ hơn về mức độ hiệu quả của thuật toán APRTA so với phương pháp cân bằng tải mặc định của CloudSim trong việc mô phỏng quản lý và phân phối các tài nguyên computing.

Thực nghiệm và kết quả thực nghiệm

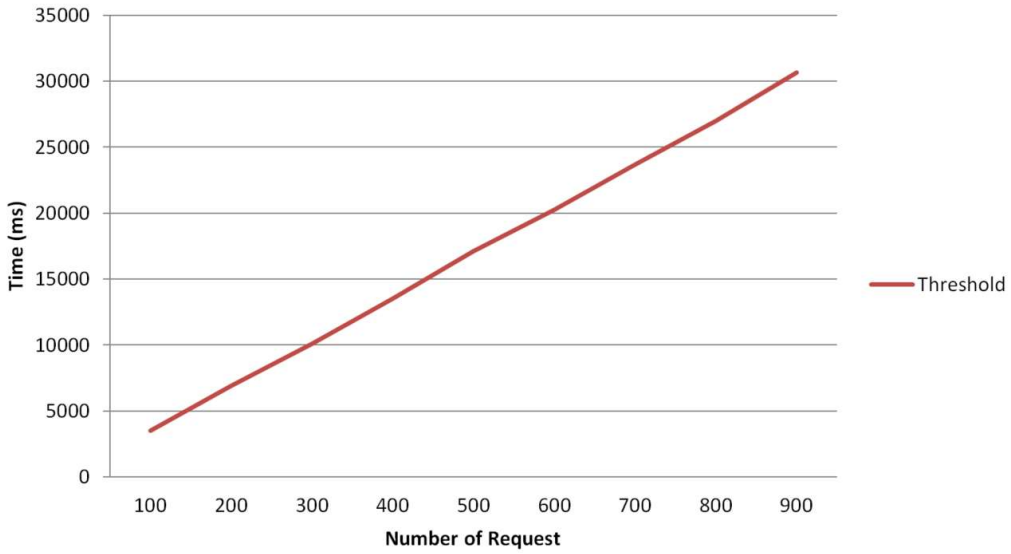
Trong một thử nghiệm thực hiện trên CloudSim, với 3 máy ảo đã được cài đặt trước sẵn sàng xử lý các yêu cầu. Các yêu cầu này được sinh ra một cách ngẫu nhiên về độ dài và dung lượng, với tổng số Request được tạo ra mỗi lần thực nghiệm tăng dần từ 100 đến 900:

Bảng 3. 6 So sánh thời gian đáp ứng dự báo của 3 máy ảo và ngưỡng thuật toán APRTA

Số lần request	VM0	VM1	VM2	Threshold
100	3.491	3.162	3.268	3.492
200	6.736	6.899	6.451	6.890
300	10.082	9.752	9.925	10.084
400	13.550	13.213	13.327	13.547
500	16.574	17.094	16.505	17.095
600	20.049	20.239	19.784	20.241
700	23.712	23.719	23.078	23.712
800	26.531	26.994	26.484	26.992
900	29.491	30.649	30.055	30.649



Hình 3. 5 Biểu đồ so sánh thời gian đáp ứng dự báo của 3 máy ảo và ngưỡng thuật toán APRTA



Hình 3. 6 Biểu đồ ngưỡng thời gian đáp ứng dự báo trong trường hợp 3 máy ảo thuật toán APRTA

Dựa trên bảng 3.6 về thời gian đáp ứng dự báo (predicted response time) của thuật toán APRTA với các máy ảo VM0, VM1, VM2 và ngưỡng Threshold, ta có các nhận xét sau về bộ cân bằng tải với thuật toán APRTA:

- *Thời gian đáp ứng dự báo:* Thời gian đáp ứng dự báo của các máy ảo (VM0, VM1, VM2) thay đổi theo số lần request. Thuật toán APRTA sử dụng thuật toán ARIMA để dự báo thời gian đáp ứng tiếp theo. Các giá trị dự báo này cung cấp thông tin về thời gian đáp ứng được kỳ vọng và được sử dụng để phân bổ request.
- *Threshold:* Ngưỡng Threshold được sử dụng để xác định việc phân bổ request cho các máy ảo. Nếu thời gian đáp ứng dự báo của máy ảo nhỏ hơn ngưỡng Threshold, request sẽ được gửi đến máy ảo đó. Ngưỡng Threshold có vai trò quan trọng trong việc cân bằng tải và đảm bảo rằng các máy ảo không bị quá tải.

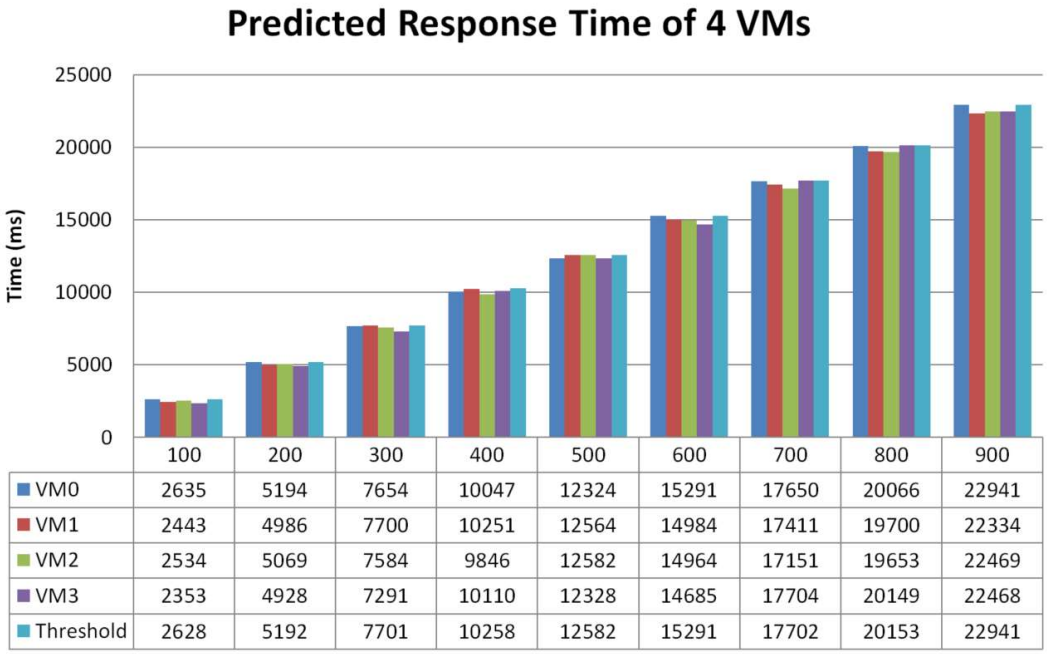
Ở trường hợp 3 máy ảo, thuật toán APRTA sử dụng thuật toán ARIMA để dự báo thời gian đáp ứng và phân bổ request dựa trên ngưỡng Threshold. Thời gian đáp ứng dự báo cho các máy ảo cung cấp thông tin về hiệu suất dự kiến và giúp quyết

định việc phân bổ tài nguyên. Điều này giúp cân bằng tải và đảm bảo rằng các máy ảo không bị quá tải.

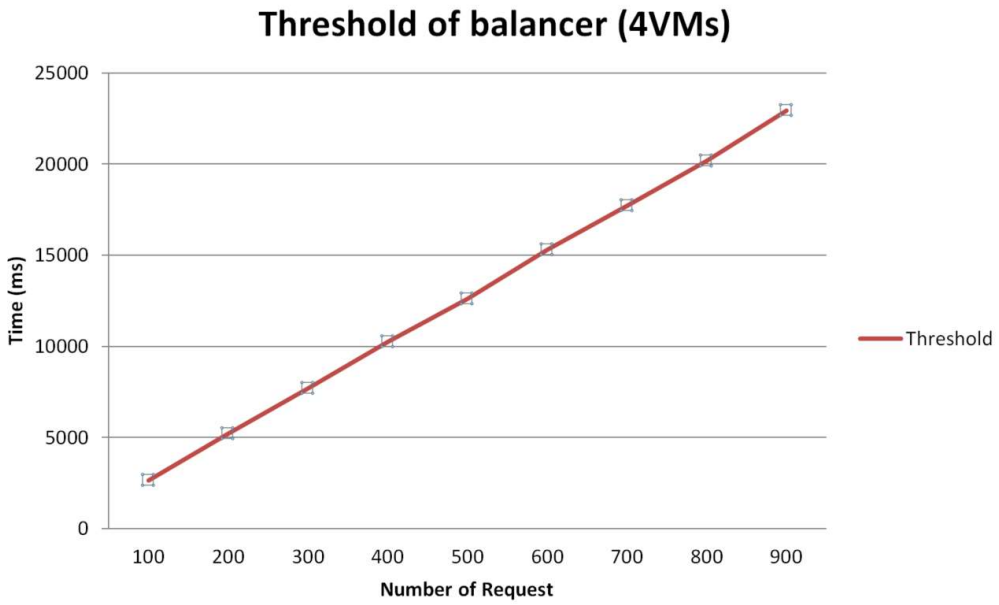
Trong một loạt thí nghiệm mô phỏng trên CloudSim, với 04 máy ảo đã được thiết lập từ trước để xử lý các yêu cầu. Những yêu cầu này được tạo ra một cách ngẫu nhiên với các đặc điểm như kích thước và chiều dài khác nhau, và số lượng Request được phát sinh mỗi lần là 100, tiếp theo là 200, và tiếp tục như vậy cho đến 900:

Bảng 3. 7 So sánh thời gian đáp ứng dự báo của 4 máy ảo và ngưỡng thuật toán APRTA

Số lần request	VM0	VM1	VM2	VM3	Threshold
100	2.635	2.443	2.534	2.353	2.628
200	5.194	4.986	5.069	4.928	5.192
300	7.654	7.700	7.584	7.291	7.701
400	10.047	10.251	9.846	10.110	10.258
500	12.324	12.564	12.582	12.328	12.582
600	15.291	14.984	14.964	14.685	15.291
700	17.650	17.411	17.151	17.704	17.702
800	20.066	19.700	19.653	20.149	20.153
900	22.941	22.334	22.469	22.468	22.941



Hình 3. 7 Biểu đồ so sánh thời gian đáp ứng dự báo của 4 máy ảo và ngưỡng thuật toán APRTA



Hình 3. 8 Biểu đồ ngưỡng thời gian đáp ứng dự báo trong trường hợp 4 máy ảo thuật toán APRTA

Dựa trên bảng 3.7 về thời gian đáp ứng dự báo (predicted response time) của thuật toán APRTA với 4 máy ảo VM0, VM1, VM2, VM3 và ngưỡng Threshold, ta có các nhận xét sau về bộ cân bằng tải với thuật toán APRTA:

- *Thời gian đáp ứng dự báo của máy ảo:* Thời gian đáp ứng dự báo của các máy ảo (VM0, VM1, VM2, VM3) thay đổi theo số lần request. Khi số lần request tăng, thời gian đáp ứng dự báo của các máy ảo cũng tăng. Điều này cho thấy sự phản ánh của việc tăng tải công việc lên hiệu suất của các máy ảo. Tuy nhiên, tỷ lệ tăng không đồng đều giữa các máy ảo, có sự khác biệt trong thời gian đáp ứng dự báo của từng máy ảo.
- *Ngưỡng Threshold:* Ngưỡng Threshold được sử dụng để xác định việc phân bổ request cho các máy ảo. Nếu thời gian đáp ứng dự báo của máy ảo nhỏ hơn ngưỡng Threshold, request sẽ được gửi đến máy ảo đó. Ngưỡng Threshold đóng vai trò quan trọng trong việc cân bằng tải và đảm bảo rằng các máy ảo không bị quá tải.
- *Mức độ tăng so với số lần request tăng:* Thời gian đáp ứng dự báo của các máy ảo tăng theo số lần request tăng, nhưng mức độ tăng không tuyến tính. Có thể quan sát rằng khi số lần request tăng gấp đôi (từ 100 đến 200), thời gian

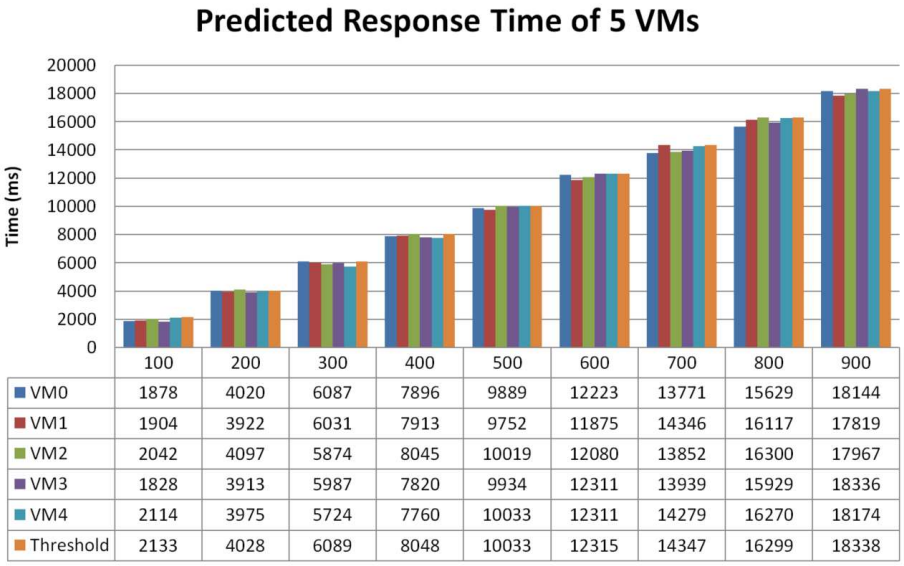
đáp ứng dự báo tăng một khoảng đáng kể. Tuy nhiên, khi số lần request tiếp tục tăng, mức độ tăng của thời gian đáp ứng dự báo giảm dần.

Ở trường hợp 4 máy ảo, thuật toán APRTA dự báo thời gian đáp ứng tiếp theo và phân bổ request dựa trên ngưỡng Threshold. Thời gian đáp ứng dự báo của các máy ảo cung cấp thông tin về hiệu suất dự kiến và giúp quyết định việc phân bổ tài nguyên. Mức độ tăng thời gian đáp ứng dự báo của các máy ảo không tuyến tính khi số lần request tăng, có thể phụ thuộc vào tính chất công việc và khả năng mở rộng của hệ thống đám mây.

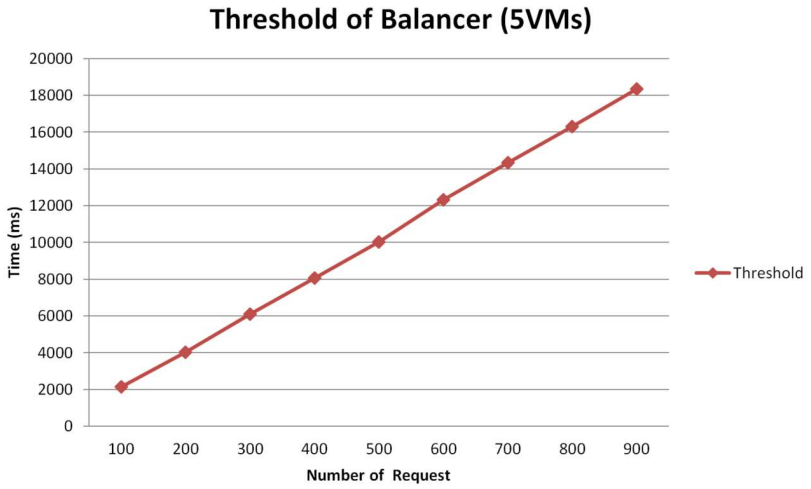
Trong các phiên mô phỏng thực hiện tiếp theo trên CloudSim, với 05 máy ảo đã sẵn sàng để giải quyết các yêu cầu đặt ra. Những yêu cầu này được tạo ra với các thuộc tính như chiều dài và kích thước được xác định một cách ngẫu nhiên, với số lượng Request được phân phối theo từng loạt lần lượt tăng từ 100 đến 900:

Bảng 3. 8 So sánh thời gian đáp ứng dự báo của 5 máy ảo và ngưỡng thuật toán APRTA

Số lần request	VM0	VM1	VM2	VM3	VM4	Threshold
100	1.878	1.904	2.042	1.828	2.114	2.133
200	4.020	3.922	4.097	3.913	3.975	4.028
300	6.087	6.031	5.874	5.987	5.724	6.089
400	7.896	7.913	8.045	7.820	7.760	8.048
500	9.889	9.752	10.019	9.934	10.033	10.033
600	12.223	11.875	12.080	12.311	12.311	12.315
700	13.771	14.346	13.852	13.939	14.279	14.347
800	15.629	16.117	16.300	15.929	16.270	16.299
900	18.144	17.819	17.967	18.336	18.174	18.338



Hình 3. 9 Biểu đồ so sánh thời gian đáp ứng dự báo của 5 máy ảo và ngưỡng thuật toán APRTA



Hình 3. 10 Biểu đồ ngưỡng thời gian đáp ứng dự báo trong trường hợp 5 máy ảo thuật toán APRTA

Dựa vào bảng 3.8 về thời gian đáp ứng dự báo (predicted response time) của thuật toán APRTA với 5 máy ảo VM0, VM1, VM2, VM3, VM4 và ngưỡng Threshold, ta có các nhận xét sau về bộ cân bằng tải với thuật toán APRTA:

- *Thời gian đáp ứng dự báo của các máy ảo:* Thời gian đáp ứng dự báo của các máy ảo (VM0, VM1, VM2, VM3, VM4) thay đổi theo số lần request. Khi số lần request tăng, thời gian đáp ứng dự báo của các máy ảo cũng tăng. Điều này cho thấy sự phản ánh của việc tăng tải công việc lên hiệu suất của các máy ảo. Mức độ tăng thời gian đáp ứng dự báo có sự khác biệt giữa các máy ảo.
- *Mức độ tăng so với số lần request tăng:* Thời gian đáp ứng dự báo của các máy ảo tăng theo số lần request tăng, nhưng mức độ tăng không tuyến tính. Có thể quan sát rằng khi số lần request tăng gấp đôi (từ 100 đến 200), thời gian đáp ứng dự báo tăng một khoảng đáng kể. Tuy nhiên, khi số lần request tiếp tục tăng, mức độ tăng của thời gian đáp ứng dự báo giảm dần.

Ở trường hợp 5 máy ảo, thuật toán APRTA dự báo thời gian đáp ứng tiếp theo và phân bổ request dựa trên ngưỡng Threshold. Thời gian đáp ứng dự báo của các máy ảo cung cấp thông tin về hiệu suất dự kiến và giúp quyết định việc phân bổ tài nguyên. Mức độ tăng thời gian đáp ứng dự báo của các máy ảo không tuyến tính khi

số lần request tăng, có thể phụ thuộc vào tính chất công việc và khả năng mở rộng của hệ thống đám mây.

Qua 3 trường hợp (3 máy ảo, 4 máy ảo và 5 máy ảo) trong thuật toán APRTA về bộ cân bằng tải dựa trên thời gian đáp ứng dự báo và ngưỡng Threshold, chúng ta có thể nhận thấy về số lượng máy ảo, trong trường hợp 3 máy ảo, 4 máy ảo và 5 máy ảo, số lượng máy ảo được sử dụng để phân bổ tài nguyên và xử lý các request tăng dần. Khi có nhiều máy ảo hơn, có thể phân chia công việc một cách hiệu quả hơn, giảm tải công việc trên mỗi máy ảo và tăng khả năng mở rộng của hệ thống. Thời gian đáp ứng dự báo của các máy ảo có xu hướng tăng khi số lượng máy ảo tăng. Điều này có thể cho thấy sự phản ánh của việc tăng cường tài nguyên máy ảo để xử lý các request. Sự khác biệt về thời gian đáp ứng dự báo giữa các máy ảo có thể phụ thuộc vào khả năng xử lý và hiệu suất của từng máy ảo. Nhìn chung, số lượng máy ảo trong thuật toán APRTA ảnh hưởng đến cách phân bổ tài nguyên và hiệu suất của hệ thống. Việc tăng số lượng máy ảo có thể giúp tăng khả năng mở rộng và giảm tải công việc trên mỗi máy ảo, tuy nhiên, cần xem xét cẩn thận việc cân bằng tải và khả năng xử lý của từng máy ảo để đảm bảo hiệu suất cao nhất của hệ thống.

Qua ba đồ thị biểu diễn so sánh, mô tả thời gian đáp ứng dự kiến của từng máy ảo so với ngưỡng thời gian đã được tính toán trước (đối với các kịch bản với 3, 4 và 5 máy ảo), chúng ta có thể quan sát rõ ràng rằng thuật toán đã thực hiện việc phân bổ công việc một cách ổn định và logic. Thời gian đáp ứng mà thuật toán ARIMA dự báo cho từng máy ảo không có sự chênh lệch đáng kể so với thời gian đáp ứng dự báo chung của hệ thống cloud (nghĩa là ngưỡng thời gian đáp ứng). Sự chính xác cao trong dự báo của thuật toán ARIMA trong dự báo chuỗi thời gian ngắn, với sai số thấp, chứng minh rằng thuật toán này rất hiệu quả trong việc phân phối các yêu cầu đến các máy ảo, tối ưu hóa quá trình xử lý và cải thiện hiệu suất tổng thể của môi trường mô phỏng cloud.

3.4 THUẬT TOÁN RCBA

Tương tự như thuật toán MCCVA, sử dụng các dữ liệu có sẵn liên quan đến chuỗi thời gian của thời gian đáp ứng (Response time) từ các yêu cầu (Request) của khách hàng, cũng như phân tích các đặc điểm khác, luận án này đưa ra đề xuất thêm về

sử dụng Machine Learning, áp dụng một phương pháp phối hợp giữa thuật toán Naive Bayes và K-means. Mục đích của sự kết hợp này là để dự đoán thời gian đáp ứng cho các yêu cầu sắp tới một cách chính xác hơn. Bằng cách này, chúng ta có thể hiểu rõ hơn về cách thức phân bổ các nguồn lực một cách hiệu quả cho những yêu cầu sẽ được đưa ra sau này, dựa trên kết quả dự báo của mô hình. Thuật toán đề xuất gồm 3 mô đun (module) chính:

(1) Module 1 : Phân lớp các yêu cầu bằng thuật toán Naive Bayes

Trong Mô-đun này, thuật toán Naive Bayes sẽ dựa vào các thuộc tính của các yêu cầu từ khách hàng(client) để tính ra thời gian đáp ứng của các yêu cầu đó, Naive Bayes sẽ phân lớp các yêu cầu này. Các thuộc tính bao gồm: Response length, Max length, Size, …

Nhóm thời gian đáp ứng: $RT_{new} = NB(X_1, X_2, X_3, ..., X_n)$ (7.1)

Trong đó X_i là các thuộc tính của các yêu cầu khi gởi lên đám mây(cloud)

Ở góc độ này ta có thể chia ra thành nhiều nhóm (3 đến 8 nhóm) hoặc nhiều hơn tùy vào độ biến thiên của các yêu cầu.

(2) Module 2 phân cụm các máy ảo/ các máy chủ / tài nguyên

Tương tự MCCVA, trong mô đun này sẽ sử dụng thuật toán phân cụm K-Means (với k=3) để phân cụm các máy ảo dựa vào mức độ hoạt động, sử dụng tài nguyên của các máy ảo, bao gồm các cụm như: thấp, trung bình, cao.

$Cluster_i = KMeans(ram, usage, cpu, ...)$ (7.2)

Trong đó: i = 1 là nhóm thấp

i = 2 là nhóm trung bình

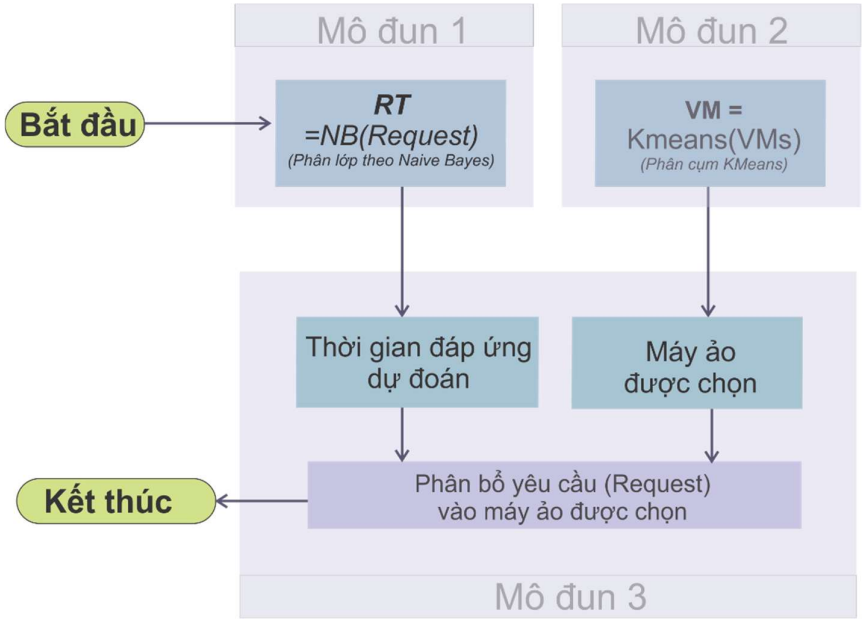
i = 3 là nhóm cao

(3) Module 3 phân bổ các dịch vụ máy ảo

Mô đun này có nhiệm vụ phân bổ các yêu cầu đến các máy ảo thông qua loại yêu cầu và cụm máy ảo phù hợp. Nếu một yêu cầu được gửi tới thì yêu cầu này được phân loại bởi mô đun 1, và các máy chủ ảo đang xét kể cả máy chủ ảo này không tải cũng được phân cụm theo mô đun 2. Sau đó thuật toán sẽ tính toán ra yêu cầu(request) nào phù hợp với máy ảo nào nhất thông qua thông số trả về của 2 hàm Naive Bayes và K-Means ở trên. Nếu thời gian đáp ứng của yêu cầu (request) đang xét (được tính toán từ module 1) nhỏ nhất thì yêu cầu này sẽ được xử lý trên các

chủ ảo với mức độ xa means nhất (tức là thuộc nhóm 1, và có mức độ sử dụng thấp nhất). Đối với các yêu cầu(request) không nhỏ không lớn ta có thể dùng các phương pháp tính toán như loại suy hay sai phân để tính toán việc phân bổ.

Mô đun phân bổ các máy ảo được hiển thị như sau:



Hình 3. 11 ***Sơ đồ của thuật toán RCBA***

Sơ đồ mã giả thuật toán RCBA

Thuật toán RCBA
Input: Tập requests
Output: Phân bổ máy ảo cho các request trong tập Request

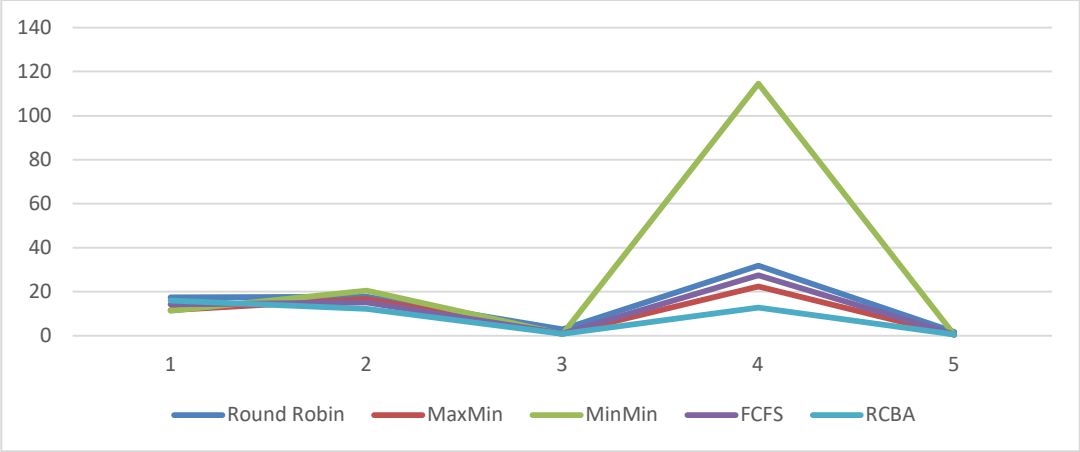
```
1.    For each Request in CloudRequests
2.          isLocated = true;
3.          RT_new = NB(RT1, RT2…..);  // Module 1
4.          VM_Cluster = kMeans(situation);  // situation: Trạng thái
      của các VM Module 2
5.          For each VM in VMList
6.                If  isFitSituation(Request.RT_new , VM.VM_Cluster)
7.                      AllocateRequestToVM(VM, Request); // Module 3
8.                      isLocated = true;
9.                      break;
10.               End If
11.         End For
12.         If (!isLocated)
13.               VM = VMList.getMinFromMean();  // Module 2
14.               AllocateRequestToVM(VM, Request);
15.         End If
16.   End For
```

CÀI ĐẶT THUẬT TOÁN RCBA

Trong khuôn khổ của một thực nghiệm mô phỏng được thực hiện trên nền tảng mã nguồn mở CloudSim, với 05 máy ảo đã được cấu hình trước để xử lý các yêu cầu. Những yêu cầu này, với chiều dài và kích thước được xác định một cách ngẫu nhiên, được tạo ra với số lượng khác nhau, bắt đầu từ 25, sau đó tăng lên 50, tiếp theo là 100 và cuối cùng là 1000. Để đánh giá hiệu quả của việc xử lý các yêu cầu này, thời gian đáp ứng (Response Time) được chọn làm chỉ số chính. Các kết quả thu được từ thực nghiệm sau đó được so sánh với hiệu suất của các thuật toán cân bằng tải khác như Round-Robin, MaxMin, MinMin và FCFS để xác định phương pháp nào cung cấp thời gian đáp ứng tối ưu nhất trong môi trường mô phỏng đám mây.

Bảng 3. 9 So sánh thời gian đáp ứng của các thuật toán với thuật toán RCBA ở trường hợp 25 Request

Thuật toán	Số lần request				
	5	10	15	20	25
Round Robin	17,42	17,64	2,89	31,90	1,81
MaxMin	11,71	16,76	0,79	22,39	0,49
MinMin	11,37	20,54	0,75	114,60	0,48
FCFS	14,27	15,06	0,99	27,41	0,62
RCBA	16,04	12,23	0,83	12,72	0,53



Hình 3. 12 Biểu đồ so sánh thời gian đáp ứng của các thuật toán với thuật toán RCBA ở trường hợp 25 Request

Dựa vào bảng 3.9 và hình 3.12 về thời gian đáp ứng (Response Time) của các thuật toán (Round-Robin, MaxMin, MinMin, FCFS và RCBA) với số lần request tương

ứng (5, 10, 15, 20, 25) trong trường hợp 25 request, ta có thể nhận xét và so sánh hiệu năng của các thuật toán như sau:

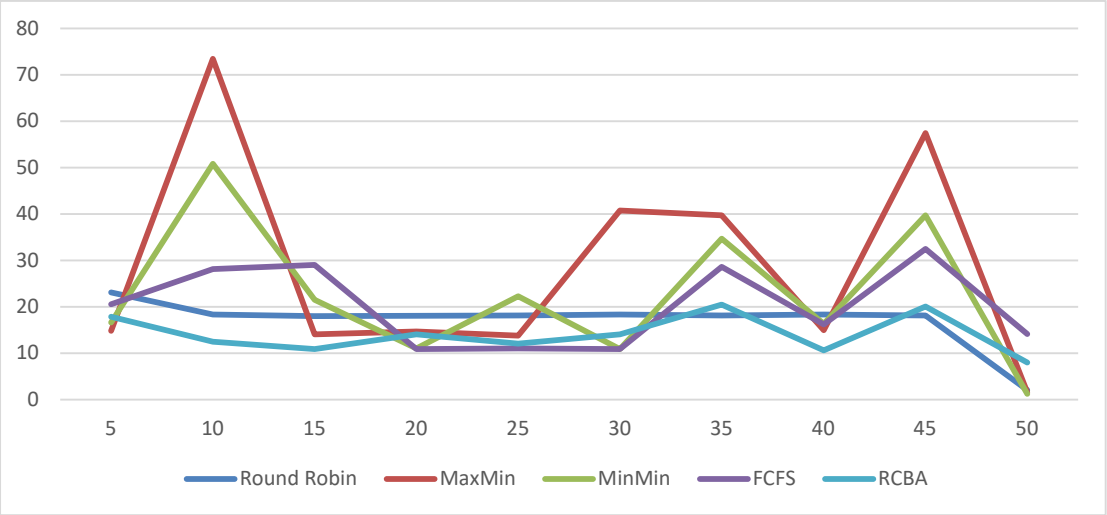
- *Round-Robin:* Thời gian đáp ứng của Round-Robin tăng dần theo số lần request tăng. Tuy nhiên, thuật toán này không cân nhắc hiệu suất hoặc tải công việc của các máy ảo, mà đơn giản là phân chia công việc một cách tuần tự cho từng máy ảo. Do đó, Round-Robin có thời gian đáp ứng tương đối cao và không cải thiện hiệu năng cân bằng tải.
- *MaxMin:* Thời gian đáp ứng của MaxMin giảm theo số lần request tăng. Thuật toán này sử dụng phương pháp tìm kiếm máy ảo có thời gian hoàn thành tốt nhất và phân bổ công việc cho máy ảo đó. MaxMin có hiệu năng cân bằng tải tốt hơn Round-Robin nhưng vẫn còn một số sự chênh lệch giữa các máy ảo.
- *MinMin:* Thời gian đáp ứng của MinMin có sự biến đổi lớn khi số lần request tăng. Thuật toán này tìm kiếm máy ảo có thời gian hoàn thành nhỏ nhất và phân bổ công việc cho máy ảo đó. MinMin có thời gian đáp ứng tốt hơn MaxMin, nhưng cần phải đánh đổi với độ phức tạp tính toán cao hơn.
- *FCFS:* Thời gian đáp ứng của FCFS tăng dần khi số lần request tăng. Thuật toán này đơn giản là xử lý công việc theo thứ tự đến đầu tiên đến đích. Tuy nhiên, FCFS không đảm bảo cân bằng tải và có thời gian đáp ứng cao khi tải công việc không được phân chia đồng đều.
- *RCBA:* Thời gian đáp ứng của RCBA có sự biến đổi nhỏ và thường là thấp nhất trong các thuật toán. RCBA kết hợp giữa việc dự báo thời gian đáp ứng và cân nhắc tải công việc của các máy ảo. Điều này cho phép RCBA phân bổ công việc một cách thông minh và cân bằng tải hiệu quả.

Ở trường hợp 25 request, RCBA có hiệu năng tốt nhất trong các thuật toán được so sánh, với thời gian đáp ứng thấp và khả năng cân bằng tải tốt hơn. MaxMin và MinMin cũng có hiệu năng tốt, nhưng MinMin có độ phức tạp tính toán cao hơn. Round-Robin và FCFS có hiệu năng thấp hơn và không cân bằng tải tốt.

Bảng 3. 10 So sánh thời gian đáp ứng của các thuật toán với thuật toán RCBA ở trường hợp 50 Request

Thuật toán	Số lần request									
	5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
Round Robin	23,13	18,39	18,04	18,09	18,17	18,33	18,17	18,35	18,14	2,10

MaxMin	14,84	73,43	14,10	14,75	13,84	40,79	39,76	14,96	57,43	1,60
MinMin	16,64	50,82	21,53	11,02	22,29	10,91	34,69	16,35	39,75	1,26
FCFS	20,55	28,14	29,04	10,91	11,05	10,92	28,63	16,22	32,50	14,16
RCBA	17,86	12,44	10,91	14,10	12,06	14,07	20,47	10,68	20,08	8,01



Hình 3. 13 Biểu đồ so sánh thời gian đáp ứng của các thuật toán với thuật toán RCBA ở trường hợp 50 Request

Dựa vào bảng 3.10 và hình 3.13 về thời gian đáp ứng (Response Time) của các thuật toán (Round-Robin, MaxMin, MinMin, FCFS và RCBA) với số lần request tương ứng (5, 10, 15, 20, 25, 30, 35, 40, 45, 50) trong trường hợp 50 request, ta có thể nhận xét và so sánh hiệu năng của các thuật toán như sau:

- *Round-Robin:* Thời gian đáp ứng của Round-Robin dao động và không có xu hướng tăng hay giảm rõ rệt khi số lần request tăng. Tuy nhiên, thuật toán này không cân nhắc hiệu suất hay tải công việc của các máy ảo, mà đơn giản là phân chia công việc theo thứ tự tuần tự cho từng máy ảo. Do đó, Round-Robin không cải thiện hiệu năng cân bằng tải và có thời gian đáp ứng tương đối cao.
- *MaxMin:* Thời gian đáp ứng của MaxMin có sự biến đổi lớn khi số lần request tăng. Thuật toán này sử dụng phương pháp tìm kiếm máy ảo có thời gian hoàn thành tốt nhất và phân bổ công việc cho máy ảo đó. Tuy nhiên, MaxMin có thời gian đáp ứng dao động cao và không ổn định, đặc biệt khi số lần request tăng lên.
- *MinMin:* Thời gian đáp ứng của MinMin cũng có sự biến đổi lớn khi số lần request tăng. Thuật toán này tìm kiếm máy ảo có thời gian hoàn thành nhỏ nhất và phân bổ công việc cho máy ảo đó. MinMin có thời gian đáp ứng tốt

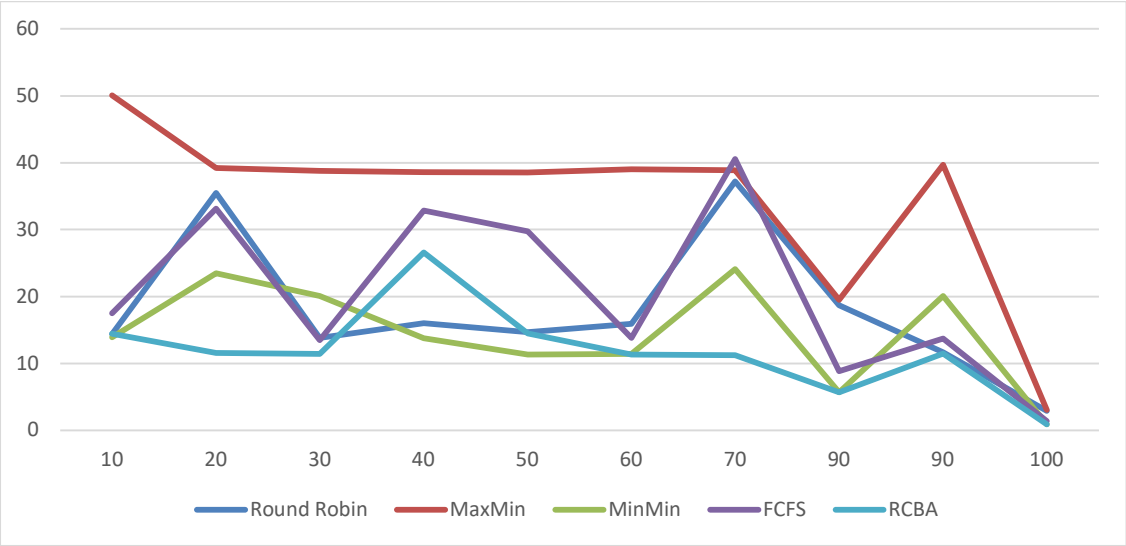
hơn MaxMin và ổn định hơn, nhưng cần phải đánh đổi với độ phức tạp tính toán cao hơn.

- *FCFS:* Thời gian đáp ứng của FCFS có sự biến đổi lớn khi số lần request tăng. Thuật toán này xử lý công việc theo thứ tự đến đầu tiên đến đích. Tuy nhiên, FCFS không đảm bảo cân bằng tải và có thời gian đáp ứng cao khi tải công việc không được phân chia đồng đều.
- *RCBA:* Thời gian đáp ứng của RCBA có sự biến đổi nhỏ và thường là thấp nhất trong các thuật toán. RCBA kết hợp giữa việc dự báo thời gian đáp ứng và cân nhắc tải công việc của các máy ảo. Điều này cho phép RCBA phân bổ công việc một cách thông minh và cân bằng tải hiệu quả.

Ở trường hợp 50 request, RCBA có hiệu năng tốt nhất trong các thuật toán được so sánh, với thời gian đáp ứng thấp và khả năng cân bằng tải tốt hơn. MinMin cũng có hiệu năng tốt, với thời gian đáp ứng ổn định và thấp. MaxMin có hiệu năng thấp hơn, trong khi Round-Robin và FCFS có hiệu năng thấp nhất và không cân bằng tải tốt. Với kết quả thực nghiệm với 50 Request trở lại, ta thấy thuật toán Round-Robin chiếm ưu thế và xử lý nhanh, thuật toán MaxMin cũng khá ổn định. Thuật toán FCFS thì chưa có thế mạnh. Tuy nhiên thuật toán đề xuất RCBA cũng khá ổn định, và chứng tỏ dần ổn định và tốt hơn khi xử lý nhiều request hơn.

Bảng 3. 11 So sánh thời gian đáp ứng của các thuật toán với thuật toán RCBA ở trường hợp 100 Request

Thuật toán	Số lần request									
	10	**20**	**30**	**40**	**50**	**60**	**70**	**90**	**90**	**100**
Round Robin	14,36	35,49	13,84	16,04	14,69	15,90	37,21	18,70	11,69	2,91
MaxMin	50,07	39,25	38,80	38,59	38,56	38,99	38,88	19,51	39,68	3,03
MinMin	14,53	14,76	14,62	12,67	15,28	16,79	11,15	12,75	50,02	14,85
FCFS	17,50	33,10	13,46	32,85	29,76	13,83	40,57	8,86	13,73	1,37
RCBA	14,40	11,58	11,42	26,60	14,50	11,30	11,22	5,68	11,51	0,88



Hình 3. 14 Biểu đồ So sánh thời gian đáp ứng của các thuật toán với thuật toán RCBA ở trường hợp 100 Request

Dựa vào bảng 3.11 và hình 3.14 về thời gian đáp ứng (Response Time) của các thuật toán (Round-Robin, MaxMin, MinMin, FCFS và RCBA) với số lần request tương ứng (10, 20, 30, 40, 50, 60, 70, 80, 90, 100) trong trường hợp 100 request, ta có thể nhận xét và so sánh hiệu năng của các thuật toán như sau:

- *Round-Robin:* Thời gian đáp ứng của Round-Robin có biến động khá lớn khi số lần request tăng. Thuật toán này đơn giản phân chia công việc theo thứ tự tuần tự cho từng máy ảo, không cân nhắc hiệu suất hay tải công việc của các máy ảo. Do đó, Round-Robin có thời gian đáp ứng khá cao và không cân bằng tải tốt.
- *MaxMin:* Thời gian đáp ứng của MaxMin dao động tương đối ổn định khi số lần request tăng. Thuật toán này tìm kiếm máy ảo có thời gian hoàn thành tốt nhất và phân bổ công việc cho máy ảo đó. Tuy nhiên, MaxMin có thời gian đáp ứng cao và không cân bằng tải tốt.
- *MinMin:* Thời gian đáp ứng của MinMin có biến động khá lớn khi số lần request tăng. Thuật toán này tìm kiếm máy ảo có thời gian hoàn thành nhỏ nhất và phân bổ công việc cho máy ảo đó. MinMin có thời gian đáp ứng thấp hơn MaxMin, nhưng không ổn định và không cân bằng tải tốt.
- *FCFS:* Thời gian đáp ứng của FCFS có biến động khá lớn khi số lần request tăng. Thuật toán này xử lý công việc theo thứ tự đến đầu tiên đến đích. FCFS

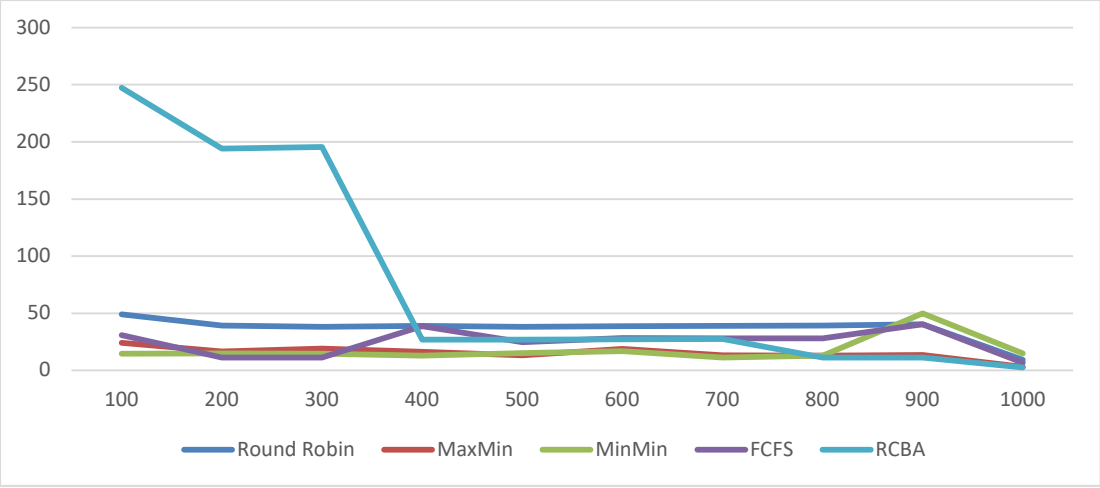
không đảm bảo cân bằng tải và có thời gian đáp ứng cao khi tải công việc không được phân chia đồng đều.

- *RCBA:* Thời gian đáp ứng của RCBA có biến động tương đối ổn định và thường là thấp nhất trong các thuật toán. RCBA kết hợp giữa việc dự báo thời gian đáp ứng và cân nhắc tải công việc của các máy ảo. Điều này cho phép RCBA phân bổ công việc một cách thông minh và cân bằng tải hiệu quả.

Ở trường hợp 100 request, RCBA có hiệu năng tốt nhất trong các thuật toán được so sánh, với thời gian đáp ứng thấp và khả năng cân bằng tải tốt hơn. MinMin có hiệu năng tốt nhưng không ổn định và không cân bằng tải tốt. MaxMin, FCFS và Round-Robin có hiệu năng thấp hơn và không cân bằng tải tốt. Từ request thứ 100 trở đi, thuật toán RCBA vượt trội hơn hẳn so với MaxMin, MinMin. Tuy nhiên vẫn chưa thấy ưu thế so với RoundRobin. Nhưng với số lượng request càng lớn thì RCBA càng lợi thế hơn. Và dần dần chiếm ưu thế tuyệt đối so với các thuật toán còn lại. Rõ ràng FCFS thể hiện sự thiếu thông minh và tính tự nhiên của giải thuật.

Bảng 3. 12 So sánh thời gian đáp ứng của các thuật toán với thuật toán RCBA ở trường hợp 1000 Request

Thuật toán	Số lần request									
	100	200	300	400	500	600	700	800	900	1000
Round Robin	14,53	14,76	14,62	12,67	15,28	16,79	11,15	12,75	50,02	14,85
MaxMin	24,25	16,41	19,17	16,19	13,38	18,94	13,27	13,16	13,63	3,18
MinMin	247,38	194,12	195,41	27,11	27,08	27,21	27,49	11,10	11,17	2,63
FCFS	31,07	11,10	11,10	38,57	24,57	28,32	28,09	28,03	40,34	6,77
RCBA	48,96	39,05	38,18	38,84	38,15	38,62	38,91	39,09	40,44	9,16



Hình 3. 15 Biểu đồ so sánh thời gian đáp ứng của các thuật toán với thuật toán RCBA ở trường hợp 1000 Request

Dựa vào bảng 3.12 và hình 3.15 về thời gian đáp ứng (Response Time) của các thuật toán (Round-Robin, MaxMin, MinMin, FCFS và RCBA) với số lần request tương ứng (100, 200, 300, 400, 500, 600, 700, 800, 900, 1000) trong trường hợp 1000 request, ta có thể nhận xét và so sánh hiệu năng của các thuật toán như sau:

- *Round-Robin:* Thời gian đáp ứng của Round-Robin duy trì ổn định và không có biến động đáng kể khi số lần request tăng. Tuy nhiên, Round-Robin không cân nhắc tải công việc của các máy ảo và có thời gian đáp ứng cao.
- *MaxMin:* Thời gian đáp ứng của MaxMin có sự biến động đáng kể khi số lần request tăng. MaxMin tìm kiếm máy ảo có thời gian hoàn thành tốt nhất và phân bổ công việc cho máy ảo đó. Tuy nhiên, MaxMin không cân bằng tải tốt và có thời gian đáp ứng cao.
- *MinMin:* Thời gian đáp ứng của MinMin có sự biến động đáng kể khi số lần request tăng. MinMin tìm kiếm máy ảo có thời gian hoàn thành nhỏ nhất và phân bổ công việc cho máy ảo đó. MinMin có thời gian đáp ứng thấp hơn MaxMin, nhưng không cân bằng tải tốt.
- *FCFS:* Thời gian đáp ứng của FCFS có sự biến động đáng kể khi số lần request tăng. FCFS xử lý công việc theo thứ tự đến đầu tiên đến đích. FCFS không đảm bảo cân bằng tải và có thời gian đáp ứng cao khi tải công việc không được phân chia đồng đều.
- *RCBA:* Thời gian đáp ứng của RCBA duy trì ổn định và không có biến động đáng kể khi số lần request tăng. RCBA kết hợp giữa việc dự báo thời gian đáp ứng và cân nhắc tải công việc của các máy ảo. Điều này cho phép RCBA phân bổ công việc một cách thông minh và cân bằng tải hiệu quả.

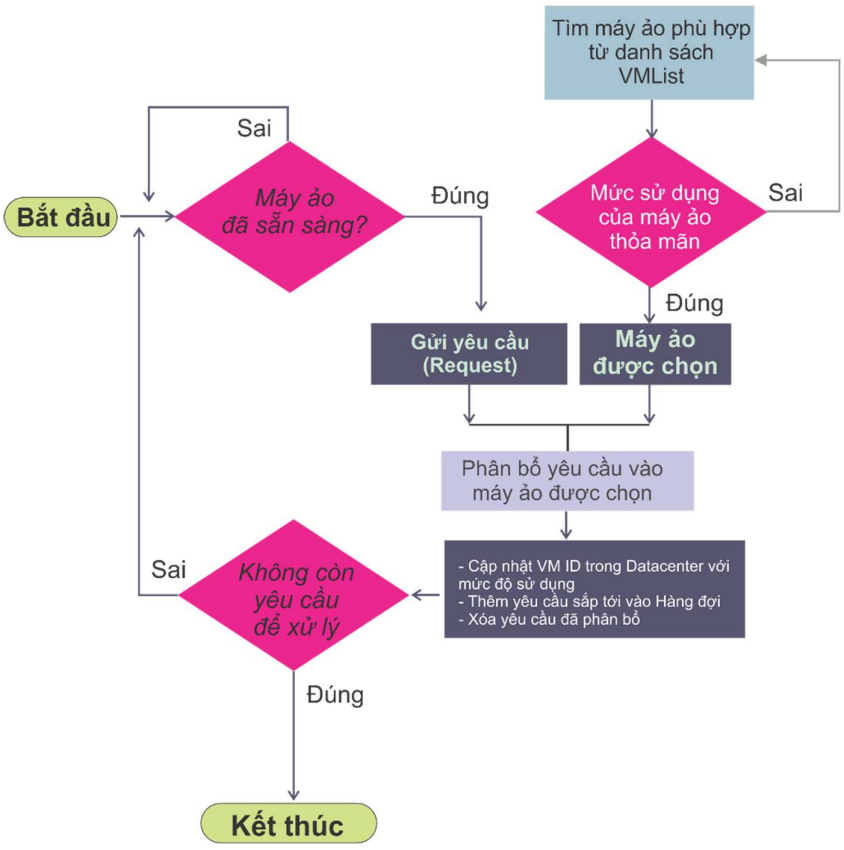
Ở trường hợp 1000 request, RCBA có hiệu năng tương đối tốt trong các thuật toán được so sánh, với thời gian đáp ứng ổn định và khả năng cân bằng tải tốt. MinMin có thời gian đáp ứng tốt hơn MaxMin nhưng không ổn định và không cân bằng tải tốt. FCFS và Round-Robin có hiệu năng thấp hơn và không cân bằng tải tốt.

Qua việc phân tích 04 biểu đồ so sánh, trong cùng một điều kiện được áp dụng cho tất cả các thuật toán, ta có thể nhận thấy rằng thuật toán được đề xuất, RCBA, đã thể hiện khả năng phân bổ tài nguyên một cách ổn định và hợp lý. Điều này được chứng minh qua thời gian đáp ứng tương đối tốt của các máy ảo khi so sánh với các thuật toán khác được triển khai trong môi trường cloud, đặc biệt là trong các trường hợp có số lượng request ít hoặc nhiều.

Ngoài ra, thực nghiệm mô phỏng đã được tiến hành trên một nhóm máy ảo cố định và không tính đến khả năng mở rộng nhóm máy ảo này (VM pool) để giải quyết vấn đề tải lớn khi cần thiết. Giả thiết được đặt ra là nhóm máy ảo hiện tại có khả năng xử lý một lượng tối đa các request nhất định và chỉ khi nào số lượng request vượt quá ngưỡng này, việc mở rộng VM pool mới được xem xét. Tuy nhiên, khi tiến hành thí nghiệm mô phỏng với một lượng lớn request, vượt quá 1000, sẽ đòi hỏi cấu hình máy tính mạnh mẽ hơn và bộ vi xử lý hiệu quả hơn, điều này làm nổi bật một hạn chế cụ thể của thí nghiệm mô phỏng này.

3.5 THUẬT TOÁN ITA

Từ nghiên cứu và phân tích một số đặc điểm thuật toán Throttled, một thuật toán điển hình và nổi tiếng của bộ cân bằng tải trên môi trường đám mây, luận án đề xuất một thuật toán cải tiến từ thuật toán Throttled, là Improved Throttled Algorithm (ITA).



Hình 3. 16 Hình Sơ đồ thuật toán Throttled cải tiến (ITA)

Các bước của thuật toán ITA

Bước 1. Trình cân bằng tải ***Improved Throttled Algorithm*** (ITA) thực hiện cân bằng tải bằng việc duy trì và cập nhật liên tục bảng chỉ mục của danh sách ***Usage Máy ảo***

Bảng chỉ mục chứa thông tin các máy ảo (VM) và mức Usage của mỗi máy ảo.

Tại thời điểm khởi tạo, tất cả các máy ảo (VM) trong danh sách "**vmUsageList**" có mức sử dụng (Usage) bằng 0.

Bước 2. Bộ điều khiển trung tâm (DCC) nhận được một nhiệm vụ cần xử lí mới.

Bước 3: Bộ điều khiển trung tâm (DCC) gửi thông tin truy vấn đến bộ cân bằng tải ITA để hỏi về việc phân bổ tiếp theo cho máy nào.

Bước 4: Bộ cân bằng tải ITA sẽ tìm ra ID máy ảo (VM) tương ứng theo thứ tự từ trên xuống trong danh sách được so sánh theo mức độ sử dụng (***Usage) Máy ảo***

và tìm ra máy ảo có ***Usage nhỏ nhất*** , sau đó trả về ID máy ảo tìm được cho bộ điều khiển trung tâm (DCC).

- Bộ điều khiển trung tâm (DCC) sẽ gửi Request đó tới máy ảo (VM, thông qua ID) để máy ảo đó xử lý và thông báo cho bộ cân bằng tải ITA về việc phân bổ vừa thực hiện. Bộ cân bằng tải ITA sẽ cập nhật ID máy ảo (VM) vừa được gửi đó và chờ Request mới từ Bộ điều khiển trung tâm (DCC).
- Trường hợp, nếu danh sách "*vmUsagelist*" rỗng (chưa có máy ảo nào được khởi tạo), thì bộ cân bằng tải ITA sẽ trả giá trị về là -1 cho Bộ điều khiển trung tâm (DCC).
- Bộ điều khiển trung tâm (DCC) xếp Request đó vào hàng đợi chờ cho lần phân bổ tiếp theo.

Bước 5: Về phía máy ảo (VM) sau khi xử lý xong Request đồng thời bộ điều khiển trung tâm (DCC) cũng nhận được phản hồi, sẽ có 1 thông báo được gửi tới cho bộ cân bằng tải ITA để cập nhật lại danh sách "**vmUsageList**" và cùng lúcs cập nhật lại danh sách ***Usage***.

Bước 6: Nếu có nhiều yêu cầu, bộ điều khiển trung tâm (DCC) lập lại Bước 3 và tiến trình được lập lại cho đến khi hết yêu cầu cần xử ly.

Đánh giá thuật toán ITA

Các kết quả thu được từ thuật toán đề xuất đã đáp ứng các mục tiêu, chẳng hạn như giới hạn số lượng yêu cầu xếp hàng để phân phối, cải thiện thời gian xử lý và thời gian phản hồi của đám mây trung tâm so với bốn thuật toán phổ biến. Điều này cũng có nghĩa là với thuật toán được đề xuất, hiệu suất của điện toán đám mây được cải thiện so với bốn thuật toán Equally Spread Current Execution Load, Round Robin, Throttled và Throttled Modified Algorithm (TMA) Thuật toán đề xuất đã cho thấy hiệu quả khi máy ảo có số lượng cao lên thì ITA đảm bảo thời gian phản hồi và thời gian xử lý tốt, giảm chi phí của các trung tâm dữ liệu đám mây.

Tuy nhiên thuật toán vẫn còn 1 số nhược điểm như:

- Nếu số lượng máy ảo nhiều thì việc tìm ra máy có Usage nhỏ nhất là khó khăn hơn
- Chưa sắp xếp các máy ảo theo danh sách tăng dần.

CÀI ĐẶT THUẬT TOÁN ITA

Dựa vào thuật toán Throttled đã có, ta sử dụng VMUsage để chọn ra máy ảo có mức sử dụng thấp nhất, từ đó ta biết cách phân bố tài nguyên cho cái request tiếp theo. Sau mỗi lần giải quyết xong 1 request ta lại cập nhật thông tin Usage cho máy ảo tương ứng trong danh sách mà thuật toán quản lý.

Trong nghiên cứu giả lập thuật toán ITA, môi trường cloud được mô phỏng thông qua việc sử dụng bộ thư viện mã nguồn mở ***Cloud_Analyst*** (phát triển từ CloudSim), và viết mã bằng ngôn ngữ lập trình *JAVA*. Mô hình môi trường giả lập cloud được cấu hình là một *Datacenter* chứa 5 máy ảo trong trường hợp của một Datacenter đơn lẻ; và được mở rộng ra 50 máy ảo tại *Datacenter1* và 5 máy ảo tại *Datacenter2*, đặt ở các khu vực địa lý khác nhau. Trong môi trường này, các yêu cầu được tạo ra ngẫu nhiên từ các *UserBase* có đặc điểm địa lý đa dạng, hướng tới việc mô phỏng sử dụng các dịch vụ cloud một cách tự nhiên hơn.

Thuật toán ITA đã được triển khai trong môi trường mô phỏng này để thu thập kết quả và đánh giá hiệu năng. Tương tự, thuật toán được phát triển cũng được cài đặt và chạy thử nghiệm, sau đó các kết quả thu được từ thuật toán này sẽ được so sánh một cách cẩn thận với kết quả của bốn thuật toán khác đã được biết đến, bao gồm ***Equally Spread Current Execution Load, Round Robin, Throttled,*** và ***TMA***, để xác định xem thuật toán nào mang lại hiệu suất tốt nhất trong việc phân phối và xử lý tải trong môi trường mô phỏng cloud.

Môi trường mô phỏng:

Thực nghiệm mô phỏng thuật toán ***Improved Throttled Algorithm*** được cài đặt trên ngôn ngữ JAVA với **Cloud Analyst** và sử dụng ECLIPSE IDE [129] để chạy thử và hiển thị kết quả theo tiêu chuẩn của Cloud Anlalyst. Môi trường giả lập với bộ thư viện mã nguồn mở CloudSim 4.0 (được cung cấp bởi http://www.cloudbus.org/).

Thuật toán đề xuất được xây dựng bằng cách tạo ra lớp *ThrottledVmITALoadBalancer*, kế thừa từ đối tượng *VmLoadBalancer*, cập nhật thêm một số phương thức và thuộc tính liên quan tới *getNumber* để tính ra Usage của máy ảo, và điều chỉnh các hàm dựng sẵn để phù hợp với thuật toán đề xuất:

```
@Override
public int getNextAvailableVm() {
        int vmId = -1;
        double min = 0;
```

```
            if (vmUsageList.size() > 0)
            {
                int temp;
                int i =0;
                for (Iterator<Integer> itr = vmUsageList.keySet().iterator(); itr.hasNext();)
                {
                    temp = itr.next();
                    Double[] state = vmUsageList.get(temp);
                    //System.out.println(temp + " state is " + state + " total vms " + vmStatesList.size());
                    double x = getNumber(state);
                    if(i==0)
                    {
                        min = x;
                        vmId = temp;
                    }
                    else {
                        if (min > x){
                                min = x;
                                vmId = temp;
                        }
                        i++;
                    }
                    allocatedVm(vmId);
                    return vmId;
                }
                return i;
            }
            return -1;
    }
    public double getNumber(Double[] nums)
    {
            double res =0;
            int i=0;
            for(i=0; i < nums.length; i++)
            {
                res = res + nums[i];
            }
            res = res/i;
            return res;
    }
    public void cloudSimEventFired(CloudSimEvent e)
```

Tiêu chí đánh giá:

Trong khuôn khổ của thí nghiệm mô phỏng môi trường đám mây sử dụng các thông số đã nêu, cài đặt các thuật toán cân bằng tải mà Cloud Analyst cung cấp sẵn, bao gồm Round Robin, Throttled và Equally Spread Current Execution Load. Ngoài ra, thuật toán TMA [109] của tác giả Nguyễn Xuân Phi và các cộng sự cũng đã được tích hợp vào thực nghiệm so sánh. Các thuật toán và thuật toán ITA được đề xuất cũng đã được phát triển và đánh giá dựa trên cùng một bộ dữ liệu đầu vào. Sự so sánh giữa các kết quả thu được chú trọng vào hai chỉ số chính là thời gian đáp ứng (Response Time) và chi phí dành cho Datacenter (Costing). Một thuật toán có thời

gian đáp ứng dự đoán gần với thực tế hơn và chi phí thấp hơn sẽ được coi là có hiệu suất tối ưu hơn trong việc quản lý tài nguyên đám mây.

Để thực hiện thực nghiệm thuật toán ITA, luận án thử nghiệm với 04 trường hợp với tương ứng cụ thể cấu hình khác nhau: Trường hợp 1 (01 datacenter và 20 máy ảo và 1 userbase), Trường hợp 2 (01 Datacenter với 5 máy ảo 3 userbase), Trường hợp 3 (01 Datacenter với 5 máy ảo 4 userbase), Trường hợp 4 (02 Datacenter, trong đó datacenter 1 gồm 50 máy ảo và Datacenter 2 gồm 5 máy ảo, 5 userbase).

Thực nghiệm và kết quả đạt được thuật toán ITA

Trường hợp 1: 01 Datacenter với 20 máy ảo và 1UB

Các thông số giả lập datacenter, máy ảo, chi tiết cấu hình host, các users base độ trễ mạng lấy từ tài liệu [109]

Môi trường mô phỏng giả lập gồm các thông số sau thông số như sau:

Data Center	# VMs	Image Size	Memory	BW
DC1	20	10000	1024	1000

Hình 3. 17 Thông số cấu hình Datacenter và máy ảo thuật toán ITA ở trường hợp 1

Name	Region	Arch	OS	VMM	Cost per VM $/Hr	Memory Cost $/s	Storage Cost $/s	Data Transfer Cost $/Gb	Physical HW Units
DC1	0	x86	Linux	Xen	0.1	0.05	0.1	0.1	3

Hình 3. 18 Cấu hình và chi phí Datacenter thuật toán ITA ở trường hợp 1

Physical Hardware Details of Data Center : DC1

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	2048	100000	10000	4	100	TIME_SHARED
1	20480	100000	10000	4	100	TIME_SHARED
2	10240	100000	10000	4	100	TIME_SHARED

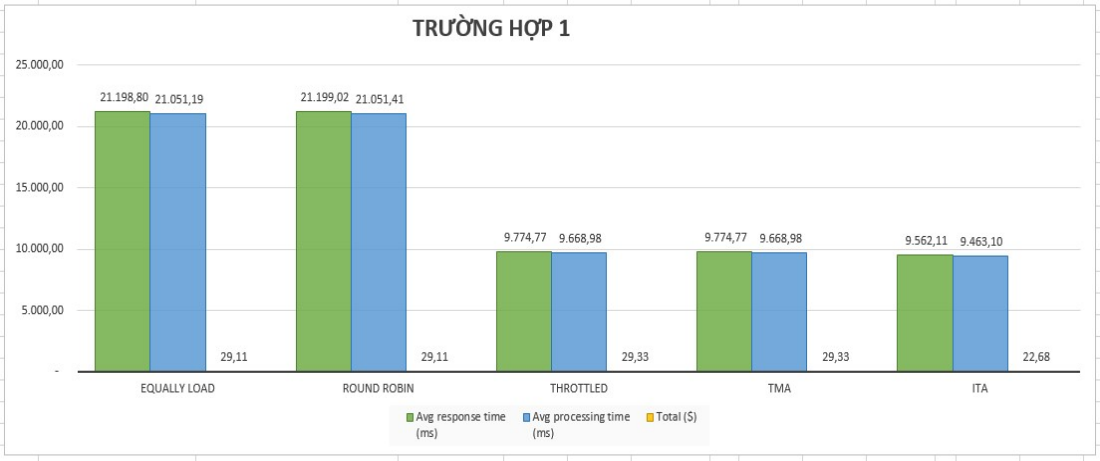
Hình 3. 19 Chi tiết cấu hình vật lý host của Datacenter thuật toán ITA ở trường hợp 1

Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	0	60	100	13	15	400000	40000
UB2	1	60	100	15	17	100000	10000

Hình 3. 20 Thông số cấu hình Cơ sở người dùng (2UB) thuật toán ITA ở trường hợp 1

Bảng 3. 13 Kết quả thực nghiệm trường hợp 1 thuật toán ITA

TH 1	Overall response time			Data Center processing time			DC Request Servicing Times			Data Center Cost		
	Avg (ms)	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)	VM Cost ($)	Data Transfer cost	Total ($)
Equally Load	21.198,80	3.637,19	39.052,12	21.051,19	3.466,10	39.001,62	21.051,19	3.466,10	39.001,62	0,50	28,61	29,11
Round Robin	21.199,02	3.637,19	39.052,12	21.051,41	3.466,10	39.001,62	21.051,41	3.466,10	39.001,62	0,50	28,61	29,11
Throttled	9.774,77	289,48	27.750,66	9.668,98	250,02	27.696,11	9.668,98	250,02	27.696,11	0,50	28,83	29,33
TMA	9.774,77	289,48	27.750,66	9.668,98	250,02	27.696,11	9.668,98	250,02	27.696,11	0,50	28,83	29,33
ITA	**9.562,11**	159,12	396,42	**9.463,10**	1.729,10	562.324,34	9.463,10	1.729,10	562.324,34	-	22,68	**22,68**



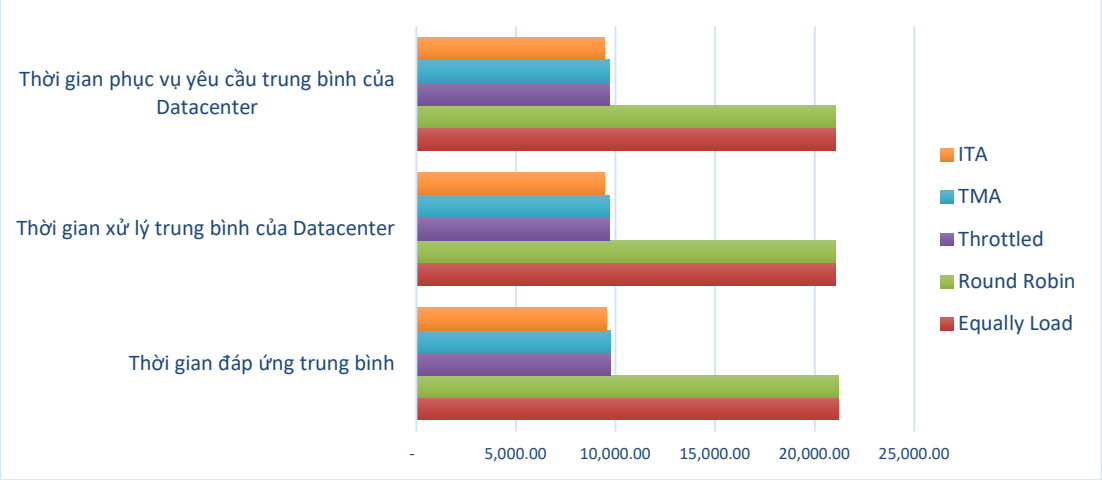
Hình 3. 21 Biểu đồ so sánh ITA với các thuật toán khác ở trường hợp 1

Dựa vào kết quả thực nghiệm về thời gian đáp ứng của các thuật toán trong trường hợp 1 (1 datacenter, 20 máy ảo và 1 userbase), ta có nhận xét và phân tích hiệu năng của các thuật toán như sau:

- *Equally Load:* Thời gian đáp ứng trung bình (Avg) của thuật toán Equally Load là 21,198.80 ms. Thời gian đáp ứng tối thiểu (Min) và tối đa (Max) của thuật toán này cũng tương ứng là 3,637.19 ms và 39,052.12 ms. Equally Load có khả năng phân bổ tải đồng đều cho các máy ảo, dẫn đến thời gian đáp ứng ổn định nhưng không tối ưu.
- *Round Robin:* Thời gian đáp ứng trung bình của thuật toán Round Robin cũng là 21,199.02 ms. Round Robin có các giá trị Min và Max tương tự Equally Load. Thuật toán này phân bổ tải theo cơ chế tuần tự, đảm bảo công bằng nhưng không tối ưu về thời gian đáp ứng.

- *Throttled:* Thời gian đáp ứng trung bình của thuật toán Throttled là 9,774.77 ms. Thời gian đáp ứng tối thiểu và tối đa cũng có giá trị tương ứng là 289.48 ms và 27,750.66 ms. Throttled tập trung vào việc giới hạn số lượng request, giúp cải thiện thời gian đáp ứng nhưng có thể gây ra thiếu công bằng trong phân bổ tải.
- *TMA:* Thời gian đáp ứng trung bình của thuật toán TMA cũng là 9,774.77 ms. Các giá trị Min và Max của TMA tương tự Throttled. TMA sử dụng ngưỡng (Threshold) và tối thiểu (Minimum) để giới hạn tài nguyên, đảm bảo tính sẵn có nhưng có thể gây ra thiếu công bằng.
- *ITA:*Thời gian đáp ứng trung bình của thuật toán ITA là 9,562.11 ms. ITA có giá trị Min là 159.12 ms và Max là 38,642.87 ms. ITA sử dụng kỹ thuật điều chỉnh ngưỡng theo quá trình lặp để cân bằng tải và giảm thời gian đáp ứng.

Ở trường hợp 1, thuật toán ITA cho thấy hiệu năng tốt nhất trong trường hợp này với thời gian đáp ứng trung bình thấp nhất. Equally Load và Round Robin đảm bảo công bằng trong phân bổ tải nhưng có thời gian đáp ứng cao hơn. Throttled và TMA sử dụng ngưỡng để hạn chế tải nhưng có thể gây ra thiếu công bằng.



Hình 3. 22 Biểu đồ so sánh 3 thông số của ITA với các thuật toán khác ở trường hợp 1

Dựa vào hình 3.22, thuật toán Throttled, TMA và ITA đều có hiệu suất tốt hơn so với Equally Load và Round Robin. Trong số đó, ITA cho thấy hiệu suất tốt nhất với thời gian đáp ứng trung bình thấp nhất và khả năng cân bằng tải tốt hơn. Tuy nhiên, cần lưu ý rằng hiệu suất của các thuật toán có thể phụ thuộc vào cấu hình cụ thể của hệ thống và yêu cầu sử dụng.

Trường hợp 2: 01 Datacenter với 5 máy ảo 3UB

Data Center	# VMs	Image Size	Memory	BW
DC1	5	10000	512	1000

Name	Region	Arch	OS	VMM	Cost per VM $/Hr	Memory Cost $/s	Storage Cost $/s	Data Transfer Cost $/Gb	Physical HW Units
DC1	0	x86	Linux	Xen	0.1	0.05	0.1	0.1	2

Physical Hardware Details of Data Center : DC1

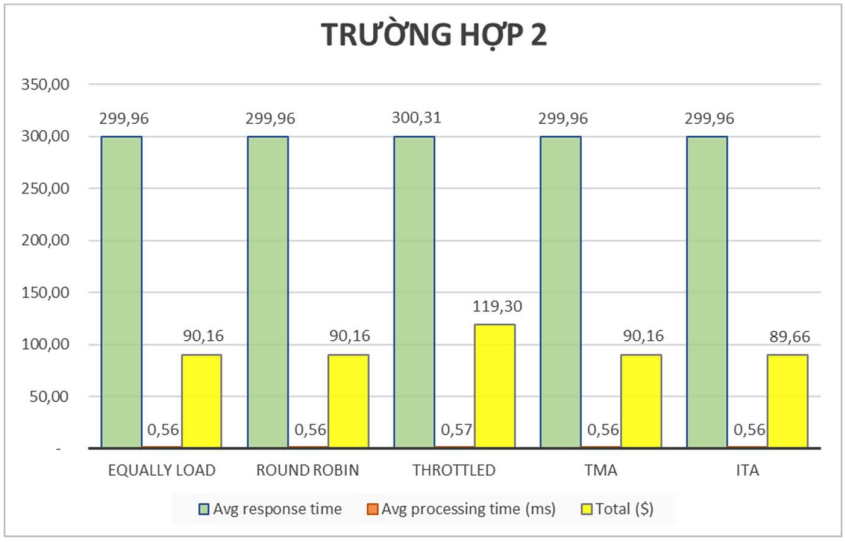
Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	10000	TIME_SHARED
1	204800	100000000	1000000	4	10000	TIME_SHARED

Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	2	100	1000	5	11	1500	200
UB2	1	70	150	3	9	1500	100
UB3	2	1110	1500	1	19	2000	500

Hình 3. 23 Thông số cấu hình trường hợp 2 thuật toán ITA

Bảng 3. 14 Kết quả thực nghiệm trường hợp 2 thuật toán ITA

TH 2	*Overall response time*			*Data Center processing time*			*DC Request Servicing Times*			*Data Center Cost*		
	Avg (ms)	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)	VM Cost ($)	Data Transfer cost	Total ($)
Equally Load	299,96	157,13	397,45	0,56	0,02	1,10	0,56	0,02	1,10	0,50	89,66	90,16
Round Robin	299,96	157,13	397,45	0,56	0,02	1,10	0,56	0,02	1,10	0,50	89,66	90,16
Throttled	300,31	157,13	397,90	0,57	0,02	1,10	0,57	0,02	1,10	0,50	118,80	119,30
TMA	299,96	157,13	397,45	0,56	0,02	1,10	0,56	0,02	1,10	0,50	89,66	90,16
ITA	***299,96***	***157,13***	***397,45***	0,56	0,02	1,10	0,56	0,02	1,10	-	***89,66***	***89,66***



Hình 3. 24 Biểu đồ so sánh ITA với các thuật toán khác ở trường hợp 2

Dựa vào kết quả thực nghiệm về thời gian đáp ứng của các thuật toán trong trường hợp 2 (01 Datacenter với 5 máy ảo 3 userbase), ta có nhận xét và phân tích hiệu năng của các thuật toán như sau:

- *Equally Load:* Thời gian đáp ứng trung bình (Avg) của thuật toán Equally Load là 299.96 ms. Thời gian đáp ứng tối thiểu (Min) và tối đa (Max) của thuật toán này cũng tương ứng là 157.13 ms và 397.45 ms. Equally Load có khả năng phân bổ tải đồng đều cho các máy ảo và userbase, dẫn đến thời gian đáp ứng ổn định nhưng không tối ưu.
- *Round Robin:* Thời gian đáp ứng trung bình của thuật toán Round Robin cũng là 299.96 ms. Round Robin có các giá trị Min và Max tương tự Equally Load. Thuật toán này phân bổ tải theo cơ chế tuần tự, đảm bảo công bằng nhưng không tối ưu về thời gian đáp ứng.
- *Throttled:* Thời gian đáp ứng trung bình của thuật toán Throttled là 300.31 ms.Thời gian đáp ứng tối thiểu và tối đa cũng có giá trị tương ứng là 157.13 ms và 397.90 ms.Throttled tập trung vào việc giới hạn số lượng request, giúp cải thiện thời gian đáp ứng nhưng có thể gây ra thiếu công bằng trong phân bổ tải.
- *TMA:* Thời gian đáp ứng trung bình của thuật toán TMA cũng là 299.96 ms.Các giá trị Min và Max của TMA tương tự Equally Load và Round Robin.
- *ITA:* Thời gian đáp ứng trung bình của thuật toán ITA cũng là 299.96 ms. ITA có giá trị Min và Max tương tự các thuật toán khác.

Ở trường hợp 2, tất cả các thuật toán đều có thời gian đáp ứng trung bình tương đương và không có sự khác biệt đáng kể. Các thuật toán Equally Load, Round Robin, TMA và ITA đều đảm bảo công bằng trong phân bổ tải, trong khi Throttled tập trung vào việc giới hạn số lượng request. Tuy nhiên, vì số lượng máy ảo và userbase trong trường hợp này không lớn, nên hiệu năng của các thuật toán không có sự chênh lệch đáng kể.

Trường hợp 3: 01 Datacenter với 5 máy ảo 4UB

Data Center	# VMs	Image Size	Memory	BW
DC1	5	10000	512	1000

Name	Region	Arch	OS	VMM	Cost per VM $/Hr	Memory Cost $/s	Storage Cost $/s	Data Transfer Cost $/Gb	Physical HW Units
DC1	0	x86	Linux	Xen	0.1	0.05	0.1	0.1	2

Physical Hardware Details of Data Center : DC1

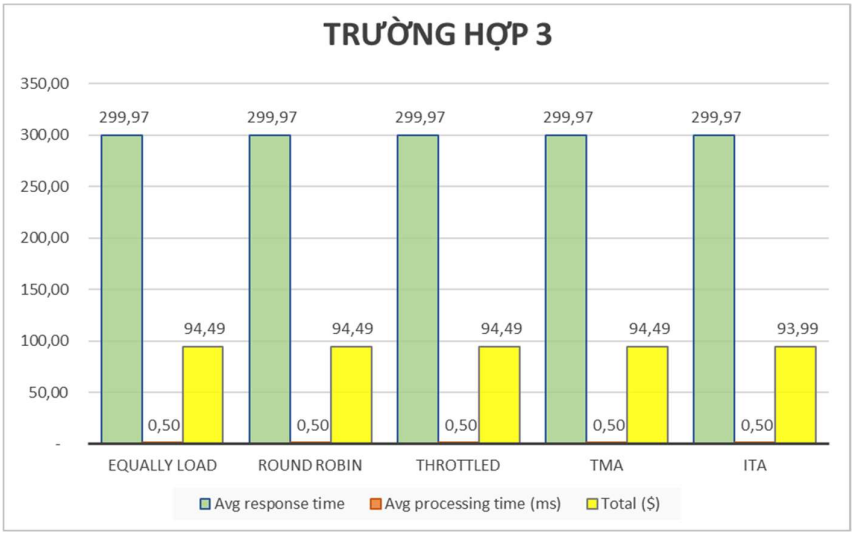
Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	10000	TIME_SHARED
1	204800	100000000	1000000	4	10000	TIME_SHARED

Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	2	100	1000	5	11	1500	200
UB2	1	70	150	3	9	1500	100
UB3	2	1110	1500	1	19	2000	500
UB4	2	800	155	7	22	3000	335

Hình 3. 25 Thông số cấu hình trường hợp 3 thuật toán ITA

Bảng 3. 15 Kết quả sau khi mô phỏng trường hợp 3 thuật toán ITA

TH 3	*Overall response time*			*Data Center processing time*			*DC Request Servicing Times*			*Data Center Cost*		
	Avg (ms)	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)	VM Cost ($)	Data Transfer cost	Total ($)
Equally Load	299,97	164,13	395,74	0,50	0,01	1,10	0,50	0,01	1,10	0,50	93,99	94,49
Round Robin	299,97	164,13	395,74	0,50	0,01	1,10	0,50	0,01	1,10	0,50	93,99	94,49
Throttled	299,97	164,13	395,74	0,50	0,01	1,10	0,50	0,01	1,10	0,50	93,99	94,49
TMA	299,97	164,13	395,74	0,50	0,01	1,10	0,50	0,01	1,10	0,50	93,99	94,49
ITA	299,97	157,12	395,74	0,50	0,01	1,10	0,50	0,01	1,10	-	***93,99***	***93,99***



Hình 3. 26 Biểu đồ so sánh ITA với các thuật toán khác trường hợp 3

Dựa vào kết quả thực nghiệm về thời gian đáp ứng của các thuật toán trong trường hợp 3 (01 Datacenter với 5 máy ảo 4 userbase), ta có nhận xét và phân tích hiệu năng của các thuật toán như sau:

- *Equally Load:* Thời gian đáp ứng trung bình (Avg) của thuật toán Equally Load là 299.97 ms.Thời gian đáp ứng tối thiểu (Min) và tối đa (Max) của thuật toán này tương ứng là 164.13 ms và 395.74 ms.Equally Load tiến hành phân bổ tải một cách đồng đều cho các máy ảo và userbase, giúp đảm bảo công bằng trong việc sử dụng tài nguyên.
- *Round Robin:* Thời gian đáp ứng trung bình của thuật toán Round Robin cũng là 299.97 ms.Round Robin có các giá trị Min và Max tương tự Equally Load.Thuật toán này sử dụng cơ chế phân bổ tải tuần tự, giúp đảm bảo công bằng nhưng không tối ưu về thời gian đáp ứng.
- *Throttled:* Thời gian đáp ứng trung bình của thuật toán Throttled là 299.97 ms.Thời gian đáp ứng tối thiểu và tối đa cũng có giá trị tương ứng là 164.13 ms và 395.74 ms.Throttled tập trung vào việc giới hạn số lượng request để cải thiện hiệu suất hệ thống, nhưng có thể gây ra sự thiếu công bằng trong phân bổ tải.
- *TMA:* Thời gian đáp ứng trung bình của thuật toán TMA cũng là 299.97 ms. Các giá trị Min và Max của TMA tương tự Equally Load và Round Robin.
- *ITA:* Thời gian đáp ứng trung bình của thuật toán ITA cũng là 299.97 ms. ITA có giá trị Min và Max tương tự các thuật toán khác.

Ở trường hợp 3, tất cả các thuật toán có thời gian đáp ứng trung bình như nhau và không có sự chênh lệch đáng kể. Equally Load, Round Robin, Throttled, TMA và ITA đều đảm bảo công bằng trong phân bổ tải và có thời gian đáp ứng ổn định. Tuy nhiên, hiệu năng của các thuật toán có thể bị ảnh hưởng bởi số lượng máy ảo và userbase, do đó, việc điều chỉnh ngưỡng và sử dụng các kỹ thuật tối ưu khác có thể cần thiết để cải thiện hiệu suất hệ thống.

Trường hợp 4: 02 Datacenter 1 gồm 50 máy ảo và Datacenter 2 gồm 5 máy ảo

Data Center	# VMs	Image Size	Memory	BW
DC1	50	10000	512	1000
DC2	5	10000	512	1000

Hình 3. 27 Thông số cấu hình 2 Datacenter và các máy ảo thuật toán ITA ở trường hợp 4

Name	Region	Arch	OS	VMM	Cost per VM $/Hr	Memory Cost $/s	Storage Cost $/s	Data Transfer Cost $/Gb	Physical HW Units
DC1	0	x86	Linux	Xen	0.1	0.05	0.1	0.1	2
DC2	2	x86	Linux	Xen	0.1	0.05	0.1	0.1	1

Hình 3. 28 Cấu hình và chi phí của các Datacenter thuật toán ITA ở trường hợp 4

Physical Hardware Details of Data Center : DC1

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	10000	TIME_SHARED
1	204800	100000000	1000000	4	10000	TIME_SHARED

Hình 3. 29 Chi tiết cấu hình vật lý host của Datacenter 1 thuật toán ITA ở trường hợp 4

Physical Hardware Details of Data Center : DC2

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	102400	10000000	100000	4	10000	TIME_SHARED

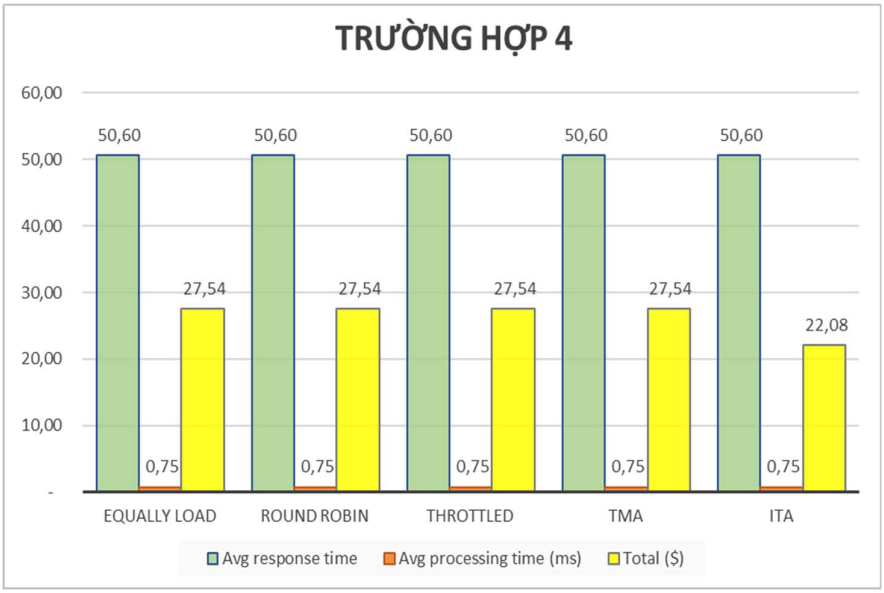
Hình 3. 30 Chi tiết cấu hình vật lý host của Datacenter 2 thuật toán ITA ở trường hợp 4

Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	2	60	100	3	9	1000	100
UB2	0	160	10000	3	19	1000	100
UB3	2	260	100	3	9	1000	100
UB4	0	30	10000	13	21	1000	100
UB5	0	60	2000	13	19	1000	100

Hình 3. 31 Thông số cấu hình Cơ sở người dùng (5 UB) thuật toán ITA ở trường hợp 4

Bảng 3. 16 Kết quả thực nghiệm trường hợp 4 thuật toán ITA

TH 4	Overall response time			Data Center processing time			DC Request Servicing Times			Data Center Cost		
	Avg (ms)	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)	Avg (ms)	Min (ms)	Max (ms)	VM Cost ($)	Data Transfer cost	Total ($)
Equally Load	50,60	38,94	65,13	0,75	0,02	1,62	1,57	1,00	2,51	5,50	22,03	27,54
Round Robin	50,60	38,94	65,13	0,75	0,02	1,62	1,57	0,10	2,50	5,50	22,03	27,54
Throttled	50,60	38,94	65,13	0,75	0,02	1,62	1,57	0,10	2,50	5,50	22,03	27,54
TMA	50,60	38,94	65,13	0,75	0,02	1,62	1,57	0,10	2,50	5,50	22,03	27,54
ITA	50,60	38,94	65,13	0,75	0,02	1,62	1,57	0,10	2,50	*0,05*	*22,03*	*22,08*



Hình 3. 32 Biểu đồ so sánh ITA với các thuật toán khác ở trường hợp 4

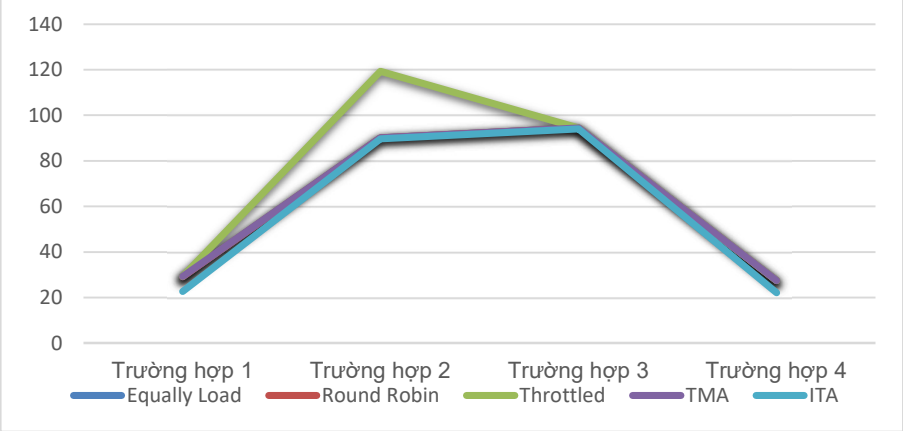
Dựa vào kết quả thực nghiệm về thời gian đáp ứng của các thuật toán trong trường hợp 4 (02 Datacenter: Datacenter 1 với 50 máy ảo và Datacenter 2 với 5 máy ảo, và 5 userbase), ta có nhận xét và phân tích hiệu năng các thuật toán như sau:

- Equally Load:Thời gian đáp ứng trung bình (Avg) của thuật toán Equally Load là 50.60 ms. Thời gian đáp ứng tối thiểu (Min) và tối đa (Max) của thuật toán

này tương ứng là 38.94 ms và 65.13 ms. Equally Load phân bổ tải một cách đồng đều giữa các máy ảo và userbase trong cả hai datacenter.

- Round Robin: Thời gian đáp ứng trung bình của thuật toán Round Robin cũng là 50.60 ms. Round Robin có các giá trị Min và Max tương tự Equally Load. Thuật toán này tuần tự phân bổ tải đến từng máy ảo trong cả hai datacenter.
- Throttled: Thời gian đáp ứng trung bình của thuật toán Throttled là 50.60 ms. Thời gian đáp ứng tối thiểu và tối đa cũng có giá trị tương tự Equally Load và Round Robin. Throttled tập trung vào giới hạn số lượng request để kiểm soát tài nguyên trong cả hai datacenter.
- TMA: Thời gian đáp ứng trung bình của thuật toán TMA cũng là 50.60 ms. Các giá trị Min và Max của TMA tương tự Equally Load và Round Robin.
- ITA : Thời gian đáp ứng trung bình của thuật toán ITA cũng là 50.60 ms. ITA có giá trị Min và Max tương tự các thuật toán khác.

Ở trường hợp 4, tất cả các thuật toán có thời gian đáp ứng trung bình như nhau và không có sự chênh lệch đáng kể. Equally Load, Round Robin, Throttled, TMA và ITA đều cố gắng đảm bảo công bằng trong phân bổ tải và có thời gian đáp ứng ổn định. Tuy nhiên, do sự khác biệt về số lượng máy ảo và userbase giữa hai datacenter, hiệu năng của các thuật toán có thể không được tối ưu. Việc điều chỉnh ngưỡng và sử dụng các kỹ thuật tối ưu khác có thể cần thiết để cải thiện hiệu suất hệ thống. Có thể nói, thuật toán ITA có kết quả tốt hơn Throttled ở một số trường hợp data đầu vào, và không hề thua kém các thuật toán có sẵn về các mặt như thời gian đáp ứng về phần chi phí datacenter thì luôn ít hơn các kỹ thuật khác.



Hình 3. 33 Biểu đồ so sánh tổng chi phí (Total Data Center Cost) của các thuật toán với thuật toán IT ở 4 trường hợp

Dựa vào hình 3.33 biểu đồ đường về tổng chi phí của datacenter cho 4 trường hợp (Equally Load, Round Robin, Throttled, TMA và ITA), ta có các nhận xét sau: *Equally Load và Round Robin*: Cả Equally Load và Round Robin có tổng chi phí datacenter như nhau trong tất cả các trường hợp. Điều này cho thấy cách phân bổ tải và xử lý của hai thuật toán này không ảnh hưởng đến tổng chi phí của datacenter. *Throttled*: Throttled có tổng chi phí datacenter cao hơn so với Equally Load và Round Robin trong trường hợp 2 và trường hợp 3. Tuy nhiên, trong trường hợp 1 và trường hợp 4, tổng chi phí datacenter của Throttled không khác biệt so với hai thuật toán khác. *TMA:* TMA có tổng chi phí datacenter cao hơn so với Equally Load và Round Robin trong trường hợp 2. Tuy nhiên, trong trường hợp 1, trường hợp 3 và trường hợp 4, tổng chi phí datacenter của TMA không khác biệt so với hai thuật toán khác. *ITA:* ITA có tổng chi phí datacenter thấp hơn so với Equally Load, Round Robin, Throttled và TMA trong tất cả các trường hợp. Điều này cho thấy ITA có khả năng tối ưu hóa chi phí của datacenter tốt hơn các thuật toán khác. Như vậy, ITA có xu hướng có tổng chi phí datacenter thấp hơn so với các thuật toán khác, trong khi Throttled có thể có tổng chi phí datacenter cao hơn trong một số trường hợp cụ thể. Equally Load, Round Robin và TMA có tổng chi phí datacenter tương đương và không có sự khác biệt đáng kể.

3.6 TỔNG KẾT CHƯƠNG

Trong chương 3, đã đề xuất được 4 thuật toán, trong đó sự liên kết giữa bốn thuật toán MCCVA [CT1], APRTA [CT2], RCBA [CT7], và ITA [CT3] được xây dựng dựa trên một số nguyên tắc cốt lõi trong việc xử lý và phân tích dữ liệu, nhằm mục tiêu tối ưu hóa cân bằng tải trong điện toán đám mây. Dưới đây là các điểm liên kết chính:

- *Khả năng Học và Dự báo:* APRTA [CT2] và MCCVA [CT1] chia sẻ khả năng học từ dữ liệu lịch sử để cải thiện quyết định trong tương lai. APRTA sử dụng ARIMA để dự báo thời gian đáp ứng, trong khi MCCVA sử dụng SVM để phân lớp và k-Means để phân cụm, cả hai đều học từ dữ liệu và đề xuất cách phân phối tải.
- *Phân loại và Phân cụm:* MCCVA [CT1] và RCBA [CT7] cả hai đều sử dụng kỹ thuật phân cụm k-Means nhưng với mục tiêu khác nhau. MCCVA

dùng nó để phân chia tải làm hai trong khi RCBA kết hợp nó với Naïve Bayes để cải thiện độ chính xác của quá trình phân loại. Cả hai đều tận dụng thông tin từ phân cụm để tối ưu hóa việc phân bố tài nguyên.

- *Tối ưu hóa Thời gian đáp ứng và Makespan:* Tất cả bốn thuật toán đều hướng đến việc cải thiện hai tham số chính là Thời gian đáp ứng và Makespan. Chúng được thiết kế để làm giảm thời gian đáp ứng thông qua dự báo và phân tích, cũng như tối ưu hóa Makespan bằng cách cân nhắc đến tải tổng thể của hệ thống.
- *Cải tiến từ thuật toán Throttle:* ITA [CT3] là thuật toán cải tiến từ Throttle Algorithm, một thuật toán cân bằng tải hiện hành. Sự cải tiến này mang lại cách nhìn mới về việc xử lý tài nguyên đám mây, cung cấp một bước tiến so với các thuật toán truyền thống bằng cách tích hợp các phương pháp tiên tiến của học máy.

Mặc dù mỗi thuật toán có tiếp cận riêng, nhưng khi kết hợp lại, chúng tạo ra một hệ thống toàn diện có khả năng học từ dữ liệu (phân loại, phân cụm, dự báo), đáp ứng nhanh chóng nhu cầu tài nguyên và phân bổ tải một cách thông minh. Sự liên kết giữa chúng tạo nên một cấu trúc đa lớp, nơi mỗi thuật toán đóng góp vào khả năng chung của hệ thống để quản lý và điều phối tài nguyên một cách hiệu quả. Nhìn chung, sự liên kết giữa các thuật toán không chỉ qua việc chia sẻ một số thành phần cốt lõi như phương pháp ML và bộ tham số đánh giá, mà còn thông qua mục tiêu chung là cải thiện tính linh hoạt và hiệu suất của hệ thống điện toán đám mây trong việc cân bằng tải.

Trong chương này thông qua việc nghiên cứu theo hướng tiếp cận từ bên trong, đã thấy được các hướng đi để nâng cao hiệu năng cân bằng tải bằng cách ứng dụng các thuật toán trí tuệ nhân tạo điển hình là học máy và các thuật toán dựa va sác suất thống kê. Theo tiếp cận các tham số của cân bằng tải đề xuất được 3 thuật toán. Theo tiếp cận tính hiệu quả của các thuạt toán đề xuất được 1 thuật toán. Như vậy, luận án đã đề xuất và xây dựng được 04 thuật toán với cách tiếp cận bên trong, nghiên cứu các đặc tính của cân bằng tải kết hợp với ML. Tuy nhiên, cân bằng tải trên cloud cũng cần quan tâm tới các yếu tố bên ngoài chứ không chỉ đặc tính bên trong của cân bằng tải, do đó luận án tiếp tục nghiên cứu các tác động bên ngoài của

cân bằng tải là deadlock và hành vi người dung trên cloud. Các nghiên cứu tiếp theo sẽ được giới thiệu trong chương 4.

CHƯƠNG 4 – **CÂN BẰNG TẢI THEO HƯỚNG TIẾP CẬN BÊN NGOÀI**

4.1 GIỚI THIỆU CHUNG

Để nâng cao hiệu năng cân bằng tải trên cloud từ hướng tiếp cận bên ngoài, ta cần xem xét các yếu tố mang tính thách thức và cơ hội mà các nhà cung cấp dịch vụ cloud không thể kiểm soát hay tác động tới. Trong luận án này, hướng tiếp cận từ bên ngoài sẽ chọn ra 02 yếu tố: yếu tố đường truyền mạng hay mạng internet, và yếu tố người dùng cloud hay hành vi người dùng cloud.

Đối với yếu tố mạng internet, việc bị timeout và hanging là dễ dàng xảy ra nếu cân bằng tải không tốt. Deadlock đại diện cho yếu tố nguy cơ mà trên mạng thường xảy ra. Với mục ý tưởng tiếp cận cân bằng tải từ deadlock, luận án này đề xuất nghiên cứu về deadlock và deadlock trên cloud, từ đó xây dựng thuật toán PDOA [CT4] [CT5] nhằm nâng cao khả năng cân bằng tải thông qua dự báo khả năng xảy ra deadlock của cloud.

Đối với yếu tố người dùng cloud và hành vi người dùng cloud, luận án này tập trung vào tính ưu tiên của người dùng mà điển hình là độ ưu tiên của tác vụ. Trong môi trường cloud, ta phân biệt người dùng thông qua tính chất của các request, từ đó dựa vào các thông số request mà ta tính toán ra độ ưu tiên của tác vụ. Từ đó, luận án đề xuất một thuật toán cân bằng tải k-CTPA [CT6] , dựa vào độ ưu tiên tác vụ để phân bổ các request, giải quyết vấn đề cân bằng tải theo tiếp cận người dùng.

4.2 DEADLOCK VÀ THUẬT TOÁN PDOA

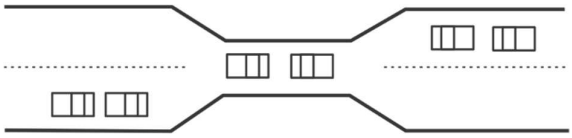
4.2.1 Khái niệm Deadlock

Deadlock [130] là tình trạng xảy ra đồng thời lập trình khi một tập hợp các quy trình đi vào điều kiện chờ vô hạn. Một quá trình hoặc một giao dịch có thể tồn tại ở hai trạng thái:

- Trạng thái đang hoạt động: tất cả các tài nguyên hoặc các mục dữ liệu được yêu cầu bởi một tiến trình đã được phân bổ cho tiến trình đó và tiến trình đó đang được thực thi hoặc sẵn sàng thực thi.
- Trạng thái bị chặn một tiến trình phải chờ các tài nguyên được yêu cầu vì một số tiến trình khác đang giữ tài nguyên đó.

Vì vậy, trong một hệ thống khi một tập hợp các tiến trình bị chặn kết thúc việc thực thi của chúng chờ nhau giải phóng tài nguyên thì hệ thống được cho là đã đi vào trạng thái deadlock.

Deadlock là một vấn đề phổ biến trong môi trường đa chương, tính toán song song, điện toán phân tán, nơi mà khóa phần cứng và phần mềm được sử dụng phân chia nguồn tài nguyên và đồng bộ hóa quy trình. Trong hệ thống thông tin liên lạc, Deadlock thường xảy ra do tín hiệu bị mất hay bị hỏng thay vì tranh chấp nguồn tài nguyên.

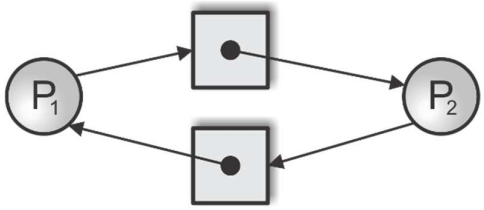


Hình 4. 1 Vấn đề Deadlock trong ví dụ qua cầu [130]

Có 4 điều kiện Deadlock xảy ra:

- *Điều kiện loại trừ lẫn nhau:* Một tài nguyên không thể được sử dụng bởi nhiều tiến trình tại cùng một thời điểm.
- *Điều kiện giữ và chờ:* Các tiến trình vừa giữ tài nguyên và chờ tài nguyên mới.
- *Điều kiện không thể chiếm:* Các tài nguyên không thể bị đòi lại, chúng chỉ có thể được giải phóng bởi chính tiến trình chiếm giữ chúng.
- *Điều kiện chu trình chờ:* Các tiến trình giữ tài nguyên và chờ các tài nguyên bị giữ bởi tiến trình khác, tạo thành một chu trình.

Hầu hết các hệ điều hành hiện nay không thể ngăn cản deadlock. Khi deadlock diễn ra, những hệ điều hành sẽ có những phản ứng khác nhau, không có sự nhất quán.



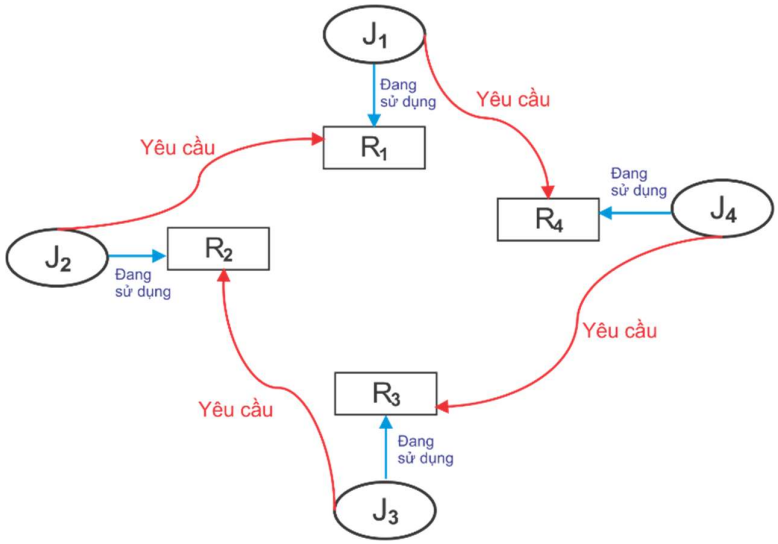
Hình 4. 2 Điều kiện xảy ra Deadlock [130]

4.2.2 Deadlock trên cloud

Các nhà cung cấp dịch vụ điện toán đám mây sẽ cung cấp tài nguyên máy tính ảo hóa một cách linh động và đa dạng dưới dạng dịch vụ thông qua mạng Internet.

Do số lượng người dùng thay đổi và tài nguyên hạn chế, đám mây dễ bị deadlock ở quy mô rất lớn.

Deadlock trong đám mây là một tình huống khi một công việc trong hệ thống đã có được một số tài nguyên và chờ đợi để biết thêm tài nguyên được mua bởi một số người khác công việc đang chờ đợi các tài nguyên có được bởi công việc này. Do đó không có công việc nào trong hệ thống có thể tiến hành thêm [111].

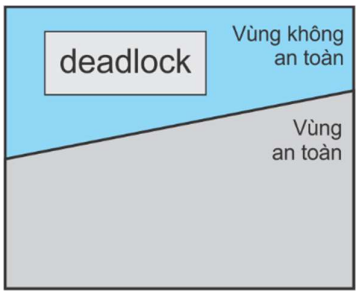


Hình 4. 3 Deadlock trong môi trường đám mây [16]

Một hệ thống được coi là ở trong trạng thái an toàn nếu nó có khả năng phân phối tài nguyên cho mỗi quá trình theo một trật tự nhất định mà không rơi vào tình trạng deadlock. Điều này có nghĩa là, có sự tồn tại của ít nhất một thứ tự an toàn cho các quá trình. Một trật tự an toàn của các quá trình $<P_1, P_2, ..., P_n>$ cho một trạng thái cấp phát nhất định có nghĩa là, cho mỗi quá trình P_i, yêu cầu về tài nguyên của nó có thể được thỏa mãn bằng tài nguyên hiện có cùng với những tài nguyên được phóng thích bởi các quá trình P_j có $j<i$. Trong trường hợp các tài nguyên cần thiết cho P_i không ngay lập tức khả dụng, P_i có thể đợi cho đến khi tất cả các quá trình P_j hoàn thành, thả ra tài nguyên của họ. Sau đó, P_i có thể sử dụng các tài nguyên này để hoàn thành nhiệm vụ của mình, giải phóng các tài nguyên mà nó giữ và sau cùng là kết thúc. Khi P_i hoàn thành, quá trình tiếp theo P_i+1 có thể tiếp tục quy trình này. Nếu không có bất kỳ thứ tự an toàn nào tồn tại, hệ thống được xem là ở trong trạng thái không an toàn.

Một trạng thái an toàn không đồng nghĩa với việc không có deadlock. Điều này có nghĩa là, trong khi mọi trạng thái deadlock chắc chắn là không an toàn, không

phải mọi trạng thái không an toàn đều dẫn đến deadlock. Tuy nhiên, một trạng thái không an toàn có khả năng biến thành deadlock. Nếu hệ thống duy trì được trạng thái an toàn, hệ điều hành có khả năng phòng tránh được trạng thái không an toàn và do đó là tránh được deadlock. Trong một trạng thái không an toàn, hệ điều hành có thể thực hiện các biện pháp để ngăn chặn các quá trình tiếp cận với tài nguyên mà chúng đang yêu cầu nếu việc cấp phát đó có thể dẫn đến deadlock, vì các hành động của các quá trình này có thể tạo ra các trạng thái không an toàn.



Hình 4. 4 Không gian trạng thái an toàn, không an toàn, deadlock

Ví dụ, xét một hệ thống bao gồm 12 ổ băng từ và ba quá trình được gọi là P0, P1 và P2. Quá trình P0 đang yêu cầu sử dụng 10 ổ băng từ, P1 có thể yêu cầu đến 4 ổ, và P2 có nhu cầu sử dụng lên đến 9 ổ băng từ. Tại một thời điểm cụ thể, t0, chúng ta nhận thấy rằng P0 đang chiếm giữ 5 ổ băng từ, P1 đang giữ 2 ổ, và P2 cũng đang sử dụng 2 ổ băng từ. Điều này để lại 3 ổ băng từ không được sử dụng trong hệ thống.

	Nhu cầu tối đa	**Nhu cầu hiện tại**
P0	10	5
P1	4	2
P2	9	2

Ở thời điểm t0, hệ thống đang nằm trong một trạng thái an toàn. Có một thứ tự an toàn có thể được áp dụng là <P1, P0, P2>, theo đó P1 có khả năng được cấp phát ngay lập tức toàn bộ ổ băng từ mà nó yêu cầu và sau khi sử dụng xong sẽ giải phóng chúng, khiến hệ thống có 5 ổ băng từ trống. Tiếp theo, P0 có thể được cấp phát tất cả các ổ băng từ mà nó cần và sau đó giải phóng chúng, để lại 10 ổ băng từ trống cho hệ thống. Cuối cùng, P2 có thể nhận toàn bộ ổ băng từ mà nó cần và cũng trả lại sau khi sử dụng, khiến hệ thống trở lại với tổng số 12 ổ băng từ không bị chiếm dụng.

Một hệ thống có thể chuyển từ một trạng thái an toàn sang trạng thái không an toàn. Ví dụ, tại một thời điểm khác, t1, nếu P2 yêu cầu và nhận thêm một ổ băng từ,

hệ thống sẽ mất trạng thái an toàn của mình. Ở thời điểm này, chỉ có P1 là có thể nhận đủ ổ băng từ mà nó cần và sau khi hoàn tất, nó giải phóng chúng, khiến hệ thống chỉ còn 4 ổ băng từ khả dụng. Do P0 cần 5 ổ băng từ và có thể yêu cầu tới 10, nó sẽ không thể tiếp tục cho tới khi có đủ tài nguyên, nên phải đợi. P2 cũng trong tình trạng tương tự, cần thêm 6 ổ và sẽ không thể tiếp tục nếu không đủ tài nguyên, có nguy cơ dẫn đến tình trạng deadlock.

Sự cố xảy ra khi chúng ta cấp thêm một ổ băng từ cho quá trình P2. Chúng ta có thể ngăn chặn tình trạng deadlock bằng cách yêu cầu P2 đợi cho đến khi các quá trình còn lại hoàn thành và trả lại tài nguyên ban đầu.

Với khái niệm trạng thái an toàn đảm bảo khả năng đáp ứng tốt cho cloud, chúng ta có thể đề xuất và áp dụng các giải thuật dự báo deadlock và khả năng xảy ra deadlock nhằm tránh deadlock. Ý tưởng đơn giản là đảm bảo hệ thống sẽ luôn còn trong trạng thái an toàn. Khởi đầu, hệ thống ở trong trạng thái an toàn. Bất cứ khi nào một quá trình yêu cầu một tài nguyên hiện có, hệ thống phải quyết định tài nguyên có thể được cấp phát tức thì hoặc quá trình phải chờ. Yêu cầu được gán chỉ nếu việc cấp phát để hệ thống trong trạng thái an toàn.

Trong kịch bản này, có khả năng một quá trình yêu cầu tài nguyên hiện hữu vẫn phải đợi. Điều này làm chậm việc sử dụng tài nguyên vì thiếu một phương pháp tiên đoán và ngăn chặn tình trạng deadlock. Do đó, phát triển một thuật toán có khả năng dự đoán và tính toán khả năng xảy ra của deadlock trong môi trường đám mây sẽ giúp đám mây tránh được deadlock, qua đó cải thiện chất lượng dịch vụ cung cấp cho người dùng.

Theo nghiên cứu của Deep Shikha và Lalit Kumar [131], deadlock trên môi trường đám mây xảy ra khi một số lượng lớn các hướng đi của dữ liệu bị chặn do cách mọi phương pháp đều có lợi thế và cố gắng giành lấy lợi thế khác mà một số hệ thống khác có được. Deadlock là khi một số lượng lớn các chiến lược bị cản trở do cách mọi phương pháp nắm giữ tài nguyên và cố gắng giành lấy lợi ích khác do một phương pháp khác nắm lấy. Một kỹ thuật trong hệ thống làm việc sử dụng nhiều tài nguyên khác nhau và sử dụng tài nguyên một cách đồng bộ. Có ba cách tiếp cận để quản lý tình trạng tắc nghẽn. (1) Hoạt động cân bằng deadlock hoặc tránh: Lý do là không cấp cho cấu trúc quyền truy cập vào trạng thái deadlock. Chúng ta chỉ có thể phóng to từng lớp; Việc ngăn chặn được hoàn thành bằng cách vô hiệu hóa một trong

số chúng kể từ các điều kiện đặc biệt được tham chiếu muộn cho tình trạng tắc nghẽn. (2) Né tránh là một kiểu đi đầu trong tự nhiên: Bằng cách sử dụng "Tránh né", chúng ta cần đưa ra một giả định. Chúng ta phải đảm bảo rằng tất cả dữ liệu về các tài nguyên mà hệ thống sẽ yêu cầu đều được chúng tôi biết trước khi thực hiện các chiến lược. Chúng ta có thể sử dụng số đếm của Banker (Đó là một lợi thế có được từ Dijkstra) để duy trì khoảng cách chiến lược khỏi tình trạng tắc nghẽn. (3) Khu vực deadlock và phục hồi: Để xảy ra điểm dừng, sau đó phân bổ để quản lý khi xảy ra. (4) Bỏ qua deadlock hoàn toàn: Nếu tình trạng tắc nghẽn là cực kỳ đặc biệt, thì hãy khởi động lại hệ thống. Đây là cách tiếp cận mà hai hệ điều hành Windows và UNIX sử dụng là *thuật toán xử lý deadlock 3.2.3*. Để xử lý deadlock, chúng ta có thể sử dụng các thuật toán khác nhau. Vì vậy, chúng ta có thể sử dụng thuật toán bắt nạt (bully algorithm) để xử lý deadlock trên máy chủ đám mây.

4.2.3 Thuật toán đề xuất PDOA

Giới thiệu thuật toán PDOA

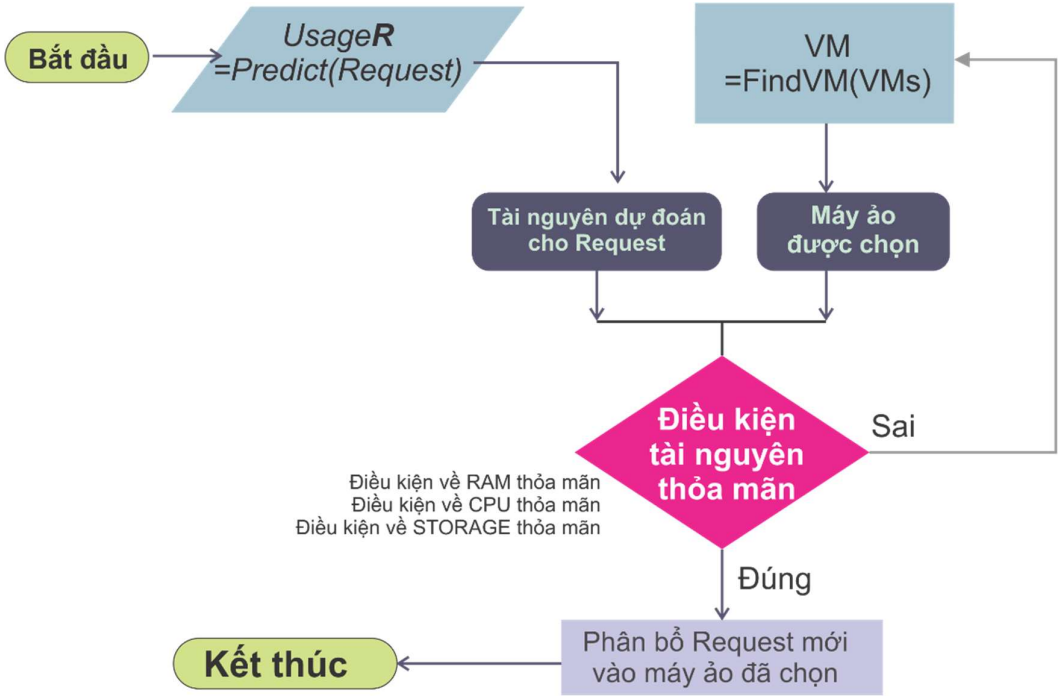
Phương pháp phân phối tài nguyên hiện tại không thích hợp cho các hệ thống phân phối nhiều phiên bản của mỗi loại tài nguyên. Tuy nhiên, thuật toán ngăn chặn deadlock mà chúng ta sẽ trình bày sau đây có thể được triển khai hiệu quả trong môi trường đám mây. Giải thuật này là giải thuật với mục tiêu dự báo deadlock và khả năng xảy ra deadlock, từ đó nhận biết được khu vực bận và hoạt động mạnh thì phân bổ đến các tài nguyên khác rãnh rỗi hơn. Vì thế, giải thuật này tạm gọi là ***PDOA (Predicting Deadlock Occurrence Algorithm).*** Tên được chọn vì giải thuật này có thể sử dụng các ứng dụng của machine learning và kỹ thuật dự báo, dự báo ra khả năng xãy ra deadlock, từ đó biết cách cấp phát tài nguyên đang có của nó khi nó không thể thoả mãn các yêu cầu của tất cả request.

Trong mô hình nghiên cứu cloud ở đây, dựa vào tính chất của Deadlock, thì Deadlock xảy ra theo các điều kiện sau:

- Khi các tác vụ / request có thời gian chờ quá lâu (timeout) và ngắt kết nối.
- Hoặc các tài nguyên của cloud bị xung đột, cùng lúc có nhiều request / tác vụ cùng truy cập tạo ra một chu trình lặp vô tận và dẫn đến treo máy. Ở đâu, tài nguyên dễ bị deadlock là RAM, CPU, và STORAGE.

Khi một request / yêu cầu mới đưa vào cloud, thì request đó mang theo các thông số như độ lớn, số file, độ lớn file, kiểu trả về… và tương ứng với các tính chất đó sẽ cần có một lượng tài nguyên tương ứng để xử lý và phục vụ nhu cầu này. Số lượng tài nguyên yêu cầu phải không vượt quá số lượng tài nguyên tổng có của hệ thống đám mây, trong đó bao gồm tài nguyên từ các máy ảo. Khi có yêu cầu từ người dùng đối với nhóm tài nguyên nào đó, hệ thống cần phải đánh giá xem việc cấp phát những tài nguyên đó có để hệ thống ở trong trạng thái an toàn hay không. Nếu việc cấp phát đảm bảo rằng hệ thống vẫn ở trong trạng thái an toàn, thì tài nguyên sẽ được cấp phát. Nếu không, quá trình yêu cầu tài nguyên sẽ phải đợi cho đến khi các quá trình khác hoàn tất và giải phóng tài nguyên cần thiết, hoặc cho đến khi hệ thống cấp phát thêm tài nguyên từ các máy ảo mới được thêm vào.

Nhằm duy trì trạng thái an toàn, bài toán này tiếp cận bằng cách dự báo deadlock và tính toán khả năng xảy ra deadlock thông qua đầu vào là các request, và tính toán ra mức độ sử dụng của tài nguyên cần có: RAM, CPU & STORAGE. Nếu các thông số tính toán ra, sẽ phân bổ vào các máy ảo có mức độ sử dụng tài nguyên thấp sao cho tổng tài nguyên sử dụng dự đoán là ít hơn ngưỡng cho phép tương ứng.



Hình 4. 5 ***Sơ đồ nguyên lý hoạt động của thuật toán PDOA***

Các bước của thuật toán

Bước 1. Thuật toán ***Predicting Deadlock Occurence Algorithm*** (PDOA) sẽ thực hiện cân bằng tải bằng việc duy trì và cập nhật liên tục bảng chỉ mục của danh sách ***Usage Máy ảo*** tại bất kỳ thời điểm nào đó.

- Bảng chỉ mục chứa thông tin các máy ảo (VM) và mức Usage của mỗi máy ảo. Usage của máy ảo được tính riêng rẽ cho RAM, CPU & STORAGE, tất cả đều được tính trên phần trăm sử dụng.

$$Usage = \{RAM, CPU, STORAGE\} \quad (8.1)$$

- Tại thời điểm bắt đầu, tất cả các máy ảo (VM) đều nằm trong danh sách "**vmUsageList**" với mức độ sử dụng Usage bằng 0.

$$Usage = \{0, 0, 0\} \quad (8.2)$$

Bước 2: Bộ điều khiển trung tâm (DCC) nhận được một nhiệm vụ / request cần xử lí mới. Ở đây, sẽ dự đoán ra được tài nguyên tương ứng cần dùng cho Request hiện tại. Tại đây, thuật toán sử dụng kỹ thuật Regression trên data các Request trước để dự báo ra tài nguyên tương ứng.

Bước 3: Bộ điều khiển trung tâm (DCC) sẽ gửi truy vấn yêu cầu đến bộ cân bằng tải PDOA cho phân bổ tiếp theo. Với việc biết được tài nguyên tương ứng cần dùng cho Request hiện tại, từ đó sẽ phân bổ vào VM sao cho thỏa đồng thời 3 điều kiện sau:

- ***RAM***: $Predicted\_Usage_{Request} + Usage_{VM} < Threshold_{RAM} \sim [90\% - 95\%]$ (8.3)
- ***CPU***: $Predicted\_Usage_{Request} + Usage_{VM} < Threshold_{CPU} \sim [97\% - 98\%]$ (8.4)
- ***STORAGE***: $Predicted\_Usage_{Request} + Usage_{VM} < Threshold_{STORAGE} \sim [96\% - 99\%]$ (8.5)

Bước 4: Bộ cân bằng tải PDOA sẽ tìm ra và gửi ID máy ảo (VM, tìm từ trên xuống trong bảng ***Usage Máy ảo***) và với điệu kiện máy ảo có ***Usage nhỏ nhất và*** đồng thời trả về giá trị là ID máy ảo về cho bộ điều khiển trung tâm (DCC).

- Bộ điều khiển trung tâm (DCC) sẽ gữi yêu cầu đó tới máy ảo (VM, xác định bởi ID đó) để xử lý và thông báo tới bộ cân bằng tải PDOA về phân bổ đó. Bộ cân bằng tải PDOA sẽ cập nhật ID máy ảo (VM) vừa được gửi đến và chờ yêu câu mới từ Bộ điều khiển trung tâm (DCC).

- Trường hợp, nếu bảng "*vmUsagelist*" rỗng (chưa có máy ảo nào được khởi tạo). Bộ cân bằng tải PDOA sẽ trả giá trị về là -1 cho Bộ điều khiển trung tâm (DCC).
- Bộ điều khiển trung tâm (DCC) xếp yêu cầu đó vào hàng đợi chờ cho lần phân bổ tiếp theo.

Bước 5: Về phía máy ảo (VM) sau khi xử lý xong yêu cầu và bộ điều khiển trung tâm (DCC) nhận được phản hồi, nó sẽ thông báo cho bộ cân bằng tải PDOA để bộ cân bằng tải cập nhật lại bảng "**vmUsageList**" đồng thời cập nhật lại danh sách ***Usage* của các VM**.

Bước 6: Nếu có nhiều yêu cầu, bộ điều khiển trung tâm lập lại Bước 3 và tiến trình được lập lại cho đến khi hết yêu cầu xử lý.

Sơ đồ mã giả thuật toán PDOA

Thuật toán PDOA
Input: Tập Requests
Output: Phân bổ máy ảo cho các request trong tập Requests

```
1.  For each Request in CloudRequests
2.    isLocated = false;
3.    PredictedUsage = {RAM, CPU, STORAGE}predicted =
            Predict(Request);
            ← Function 1: regression on historical dataset
4.    VM = FindVM(VMList);
            ← Function 2: select the VM with least resources usage
5.    If isSatisfied(PredictedUsage, VM)
6.        AllocateRequestToVM(VM, Request);
            ← Function 3: allocate the Request to the VM
7.        isLocated = true;
8.    End If
9.    If(!isLocated)
10.     VM = VMList.getSelectedVM();
11.     AllocateRequestToVM(VM, Request);
12.   End If
13. End For
```

Đánh giá thuật toán PDOA

Kết quả đạt được từ thuật toán mới đã thực hiện thành công các mục tiêu đặt ra, bao gồm việc hạn chế số lượng yêu cầu chờ đợi xử lý, tối ưu hóa thời gian xử lý và cải thiện đáng kể thời gian phản hồi khi so sánh với bốn thuật toán phổ biến điển hình trong cân bằng tải. Điều này cũng có nghĩa là với thuật toán được đề xuất, hiệu suất của điện toán đám mây được cải thiện so với bốn thuật toán ***MaxMin, Round Robin, MinMin*** và ***FCFS***. Thuật toán đề xuất đã cho thấy hiệu quả khi máy ảo có số

lượng cao lên thì PDOA đảm bảo thời gian phản hồi và thời gian xử lý tốt, giảm chi phí của các trung tâm dữ liệu đám mây.

Tuy nhiên thuật toán vẫn còn 1 số nhược điểm như:

- Nếu số lượng máy ảo nhiều thì việc tìm ra máy có Usage nhỏ nhất là khó khăn hơn, nên có thể tìm một máy có Usage phù hợp là đạt.
- Chưa sắp xếp các máy ảo theo danh sách tăng dần.
- Thuật toán dự báo càng chính xác thì phân bổ càng hiệu quả.
- Tính toán và xử lý các ngưỡng động, tức là các ngưỡng này thay đổi theo VM, theo request hoặc theo đặc điểm cloud.

CÀI ĐẶT THUẬT TOÁN PDOA

Môi trường mô phỏng:

Để Phát triển thuật toán mới PDOA, luận án đã thiết kế lớp *PDOASchedulingAlgorithm* dựa trên kế thừa lớp cơ sở *BaseSchedulingAlgorithm*, bổ sung thêm các phương thức mới và thuộc tính mới, đặc biệt là phương thức *getVMUsage* nhằm tính toán mức sử dụng của các máy ảo, và điều chỉnh lại các hàm có sẵn để chúng phù hợp với các yêu cầu của thuật toán mới được đề xuất. Hàm ***predictRequestUsage*** nhằm dự đoán khả năng sử dụng tài nguyên của Request. Bên cạnh đó, hàm ***getMostSuitableVM*** dùng để lấy ra VM phù hợp thỏa 3 ngưỡng để phân bổ request.

```
@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:

Mô phỏng giả lập cloud với các tham số như trên, và chạy thuật toán cân bằng tải của ***CloudSim*** có sẵn: ***Round Robin, MaxMin, MinMin và FCFS***, cài đặt các thuật toán cùng đầu vào, so sánh kết quả đầu ra, đặc biệt là thông số thời gian đáp ứng (Response Time), thời gian thực hiện (Makespan). Thời gian đáp ứng dự đoán của các máy ảo cũng như thời gian đáp ứng dự đoán của cloud với sai số càng ít đi thì hiệu quả của thuật toán được đánh giá sẽ càng tốt, chi phí củng càng thấp thì kỹ thuật đó cũng hiểu là tốt hơn.

Các Request được đại diện bởi Cloudlet trong cloudSim và kích thước của các Cloudlet được xây dựng sẵn theo dataset của CloudSim. Số lượng Cloudlet lần lượt là *Epigenomics_24, Epigenomics_100 và Epigenomics_997*.

Bảng 4. 1 Cấu hình thông số các Request thuật toán PDOA

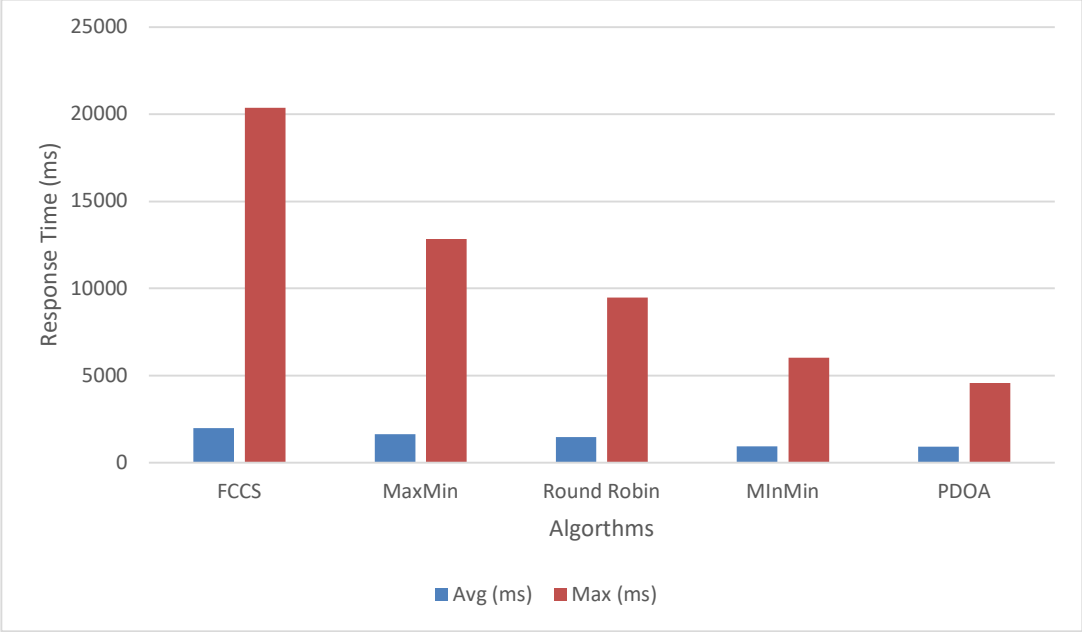
Chiều dài (Length) Kb	**Kích thước file (File Size) Kb**	**Kích thước file đầu ra** (Output Size in Kb)	**Số CPU xử lý (PEs)**
100 ~ 1700	5000 ~ 45000	450 ~ 750	1

Kết quả mô phỏng giả lập

Trường hợp 1 *(Epigenomics_24)*: mô phỏng giả lập với data 24 request có sẵn của CloudSim, và kết quả như sau:

Bảng 4. 2 So sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 1

TH 1	*Thời gian đáp ứng (Overall response time)*		
	Avg (ms)	**Min (ms)**	**Max (ms)**
FCFS	1.993,83	0,18	20.363,49
MaxMin	1.632,78	0,11	12.825,62
Round Robin	1.474,61	0,10	9.487,84
MinMin	930,43	0,10	6.027,47
PDOA	**915,34**	0,11	**4.582,72**



Hình 4. 6 Biểu đồ so sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 1

Theo hình 4.6, biểu đồ cột về thời gian đáp ứng trung bình (Avg) và thời gian đáp ứng lớn nhất trong trường hợp 1 của các thuật toán trong mô phỏng trên môi trường đám mây, ta có thể thấy trong các thuật toán được thử nghiệm, PDOA và MinMin có thời gian đáp ứng trung bình thấp nhất, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. Các thuật toán MaxMin và Round Robin cũng có thời gian đáp ứng trung bình khá tương đồng và không chênh lệch quá nhiều. Tuy nhiên, chúng có hiệu năng thấp hơn so với PDOA và MinMin. FCFS có thời gian đáp ứng trung bình cao nhất, tức là thuật toán này có hiệu năng thấp hơn so với các thuật toán khác trong việc xử lý các request.

Ở trường hợp 1, trong các thuật toán được thử nghiệm, PDOA có thời gian đáp ứng tối thiểu (min) và tối đa (max) thấp nhất, cho thấy khả năng cân bằng tải tốt nhất trong việc phân phối công việc. MinMin cũng có thời gian đáp ứng tối thiểu và tối đa khá thấp, cho thấy khả năng cân bằng tải tốt. Các thuật toán MaxMin và Round Robin cũng có khả năng cân bằng tải khá tốt, nhưng chênh lệch thời gian đáp ứng tối đa giữa các request lớn hơn so với PDOA và MinMin. FCFS không có khả năng cân bằng tải tốt, vì thời gian đáp ứng tối đa của các request chênh lệch rất lớn.

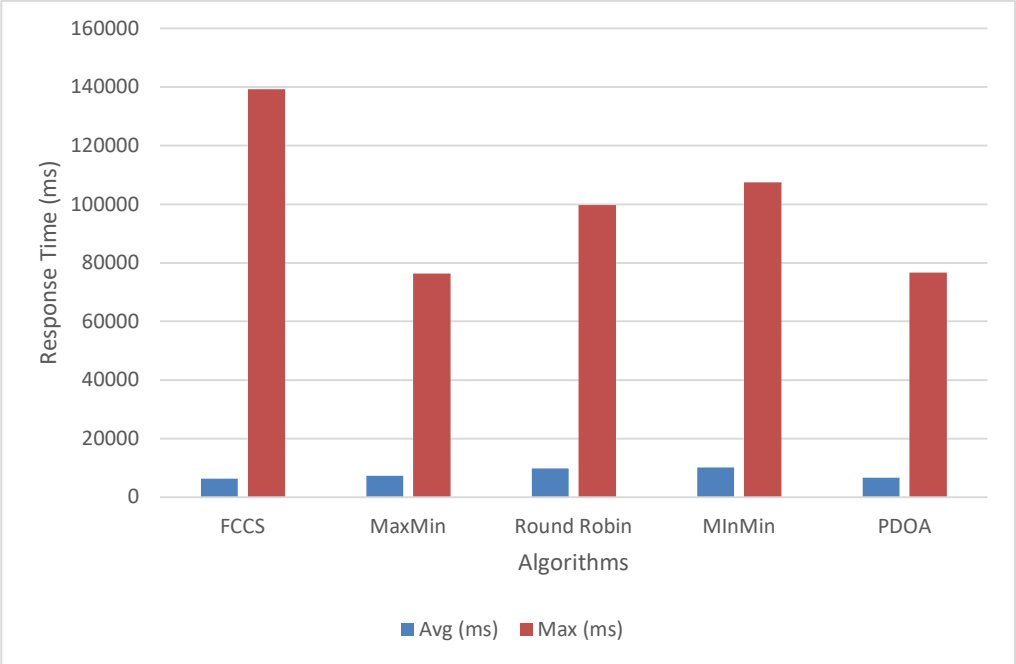
Như vậy, ở trường hợp mô phỏng giả lập này, thuật toán PDOA và MinMin có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. Các thuật toán MaxMin và Round Robin cũng cho thấy khả năng cân bằng tải tốt, nhưng hiệu năng của chúng không tốt bằng PDOA và MinMin. FCFS là thuật toán có hiệu năng thấp nhất và không có khả năng cân bằng tải tốt.

Trường hợp 2 *(Epigenomics_100)***:** mô phỏng giả lập với data 100 request có sẵn của CloudSim, và kết quả như sau:

Bảng 4. 3 So sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 2

TH 2	*Thời gian đáp ứng (Overall response time)*		
	Avg (ms)	**Min (ms)**	**Max (ms)**
FCFS	**6.352,15**	**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 4. 7 Biểu đồ so sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 2

Dựa trên hình 4.7 và bảng 4.3 về thời gian đáp ứng trung bình (ms) và thời gian đáp ứng tối thiểu (min) và tối đa (max) của các thuật toán trong mô phỏng giả lập trên môi trường đám mây, chúng ta có thể thấy rằng trong trường hợp 2, thuật toán PDOA có thời gian đáp ứng trung bình thấp nhất, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. FCFS cũng có thời gian đáp ứng trung bình khá thấp, tuy nhiên, MaxMin, Round Robin và MinMin có thời gian đáp ứng trung bình cao hơn, cho thấy hiệu năng thấp hơn so với FCFS và PDOA.

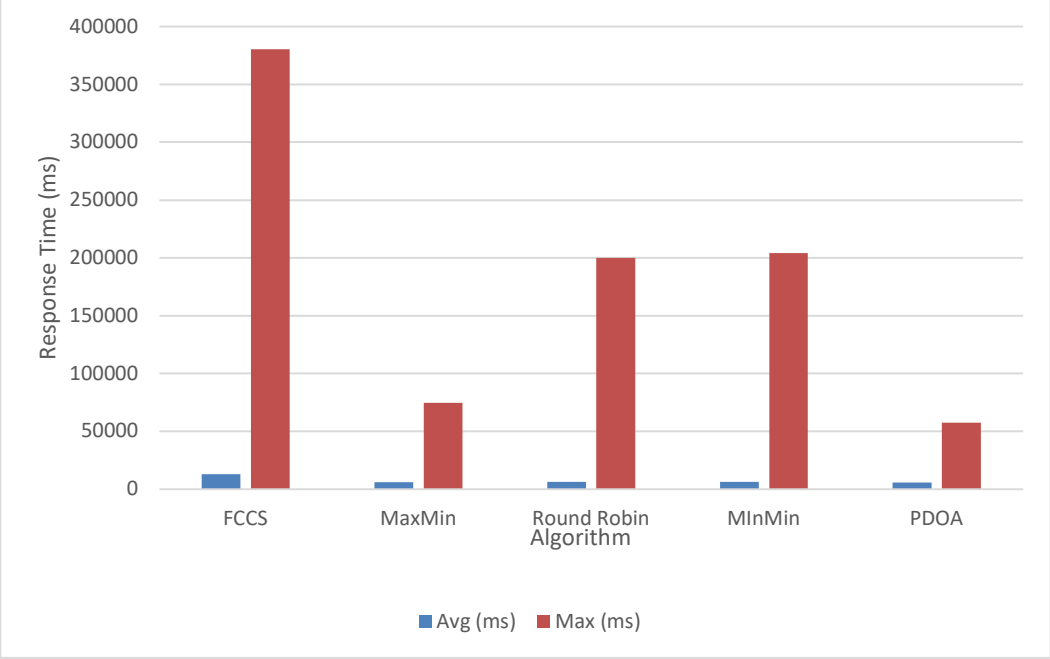
Về khả năng cân bằng tải, thì PDOA và FCFS có thời gian đáp ứng tối thiểu và tối đa khá thấp, cho thấy khả năng cân bằng tải tốt hơn so với các thuật toán khác. MinMin có thời gian đáp ứng tối đa rất cao, cho thấy sự không cân bằng trong việc phân phối công việc. Các thuật toán MaxMin và Round Robin cũng có khả năng cân bằng tải tương đối tốt, nhưng chênh lệch thời gian đáp ứng tối đa giữa các request lớn hơn so với PDOA và FCFS.

Ở trường hợp 2 với 100 request, thuật toán PDOA và FCFS có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. Các thuật toán MaxMin và Round Robin cũng cho thấy khả năng cân bằng tải tốt, nhưng hiệu năng của chúng không tốt bằng PDOA và FCFS. MinMin có hiệu năng thấp và khả năng cân bằng tải không tốt do chênh lệch thời gian đáp ứng tối đa quá cao.

Trường hợp 3 *(Epigenomics_997)***:** mô phỏng giả lập với data 997 request có sẵn của CloudSim, và kết quả như sau:

Bảng 4. 4 So sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 3

TH 3	*Thời gian đáp ứng (Overall response time)*		
	Avg (ms)	**Min (ms)**	**Max (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**



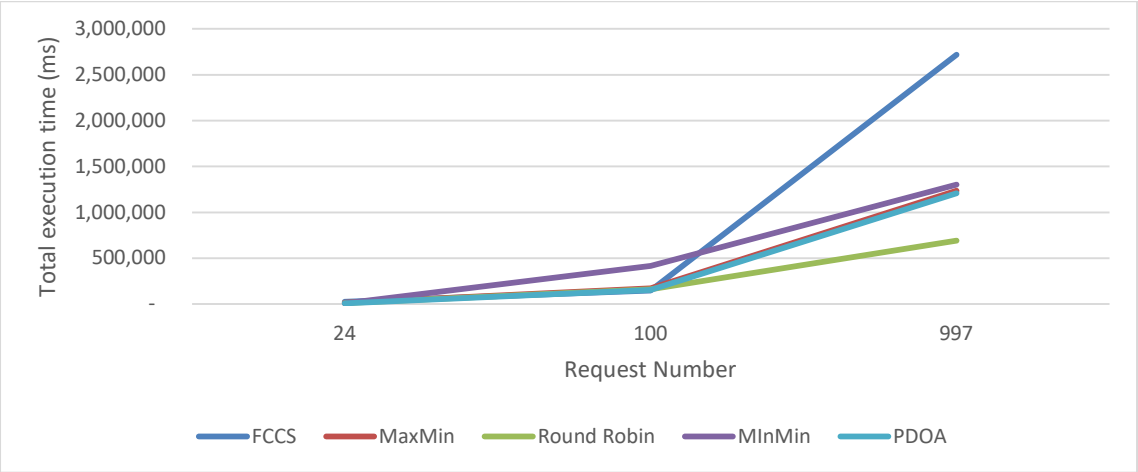
Hình 4. 8 Biểu đồ so sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 3

Trong trường hợp 3, qua bảng 4.4 và hình 4.8 , ta thấy thuật toán PDOA vượt trội về thời gian xử lý trung bình, và cả thời gian xử lý Max là thấp nhất. Với lượng request tang nhiều, thấy được tính ưu việt của dự báo và khả năng xử lý của thuật toán dự báo. Thuật toán FCFS thể hiện sự tự nhiên, nên việc thời gian xử lý luôn là cao nhất. Trong trường hợp này, thuật toán PDOA có thời gian đáp ứng trung bình thấp nhất, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. MaxMin, Round Robin, và MinMin có thời gian đáp ứng trung bình khá tương đồng và thấp hơn so với FCFS, nhưng vẫn cao hơn so với PDOA. FCFS có thời gian đáp ứng trung bình cao nhất, tức là thuật toán này có hiệu năng thấp hơn so với các thuật toán khác trong việc xử lý các request.

Về khả năng cân bằng tải, ta thấy PDOA có thời gian đáp ứng tối thiểu và tối đa thấp nhất, cho thấy khả năng cân bằng tải tốt hơn so với các thuật toán khác. Các thuật toán MaxMin, Round Robin, và MinMin cũng có khả năng cân bằng tải tương đối tốt, với chênh lệch thời gian đáp ứng tối đa không quá lớn. FCFS có khả năng cân bằng tải thấp hơn, vì thời gian đáp ứng tối đa của các request chênh lệch rất lớn. Như vậy, trong mô phỏng giả lập ở trường hợp 3, thuật toán PDOA có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. Các thuật toán MaxMin, Round Robin, và MinMin cũng cho thấy khả năng cân bằng tải tốt, nhưng hiệu năng của chúng không tốt bằng PDOA. FCFS có hiệu năng thấp nhất và khả năng cân bằng tải không tốt do chênh lệch thời gian đáp ứng tối đa quá cao.

Bảng 4. 5 So sánh thời gian thực hiện các thuật toán với thuật toán PDOA ở 3 trường hợp

	Thời gian thực hiện (Makespan - ms)		
	24	**100**	**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 4. 9 Biểu đồ so sánh thời gian thực hiện các thuật toán với thuật toán PDOA ở 3 trường hợp

Dựa vào số liệu trong bảng 4.5 so sánh thời gian thực hiện (makespan) của các thuật toán với thuật toán PDOA ở 3 trường hợp, chúng ta có thể có một vài nhận xét như sau:

- *Trường hợp 24 Request:* PDOA có thời gian thực hiện (makespan) thấp nhất là 6.297 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. MinMin và Round Robin cũng có thời gian thực hiện khá thấp, lần lượt là 7.593 ms và 11.055 ms, cho thấy khả năng cân bằng tải tương đối tốt. MaxMin và FCFS có thời gian thực hiện cao hơn, lần lượt là 14.546 ms và 23.155 ms, cho thấy khả năng cân bằng tải kém hơn so với PDOA, MinMin và Round Robin.
- *Trường hợp 100 Request:* PDOA vẫn có thời gian thực hiện thấp nhất là 152.507 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. MaxMin và Round Robin có thời gian thực hiện tương đối thấp, lần lượt là 170.876 ms và 157.786 ms, cho thấy khả năng cân bằng tải tương đối tốt. MinMin có thời gian thực hiện cao hơn, là 414.381 ms, cho thấy khả năng cân bằng tải không tốt bằng PDOA, MaxMin và Round Robin. FCFS có thời gian thực hiện cao nhất, là 146.129 ms, cho thấy khả năng cân bằng tải kém hơn so với các thuật toán khác.
- *Trường hợp 997 Request:* PDOA vẫn tiếp tục có thời gian thực hiện thấp nhất là 1.206.652 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. Round Robin và MaxMin có thời gian thực hiện tương đối thấp, lần lượt

là 692.308 ms và 1.236.725 ms, cho thấy khả năng cân bằng tải tương đối tốt. MinMin có thời gian thực hiện cao hơn, là 1.303.421 ms, cho thấy khả năng cân bằng tải không tốt bằng PDOA, Round Robin và MaxMin. FCFS có thời gian thực hiện cao nhất, là 2.719.720 ms, cho thấy khả năng cân bằng tải kém hơn so với các thuật toán khác.

Đánh giá chung

PDOA thường có hiệu năng cân bằng tải tốt hơn so với các thuật toán khác, như được thể hiện qua thời gian thực hiện (makespan) thấp hơn trong các trường hợp thử nghiệm. Các thuật toán MaxMin và Round Robin cũng cho thấy khả năng cân bằng tải tốt, trong khi MinMin có khả năng cân bằng tải kém hơn. FCFS thường có hiệu năng cân bằng tải kém nhất trong các trường hợp được đề cập. Thuật toán PDOA có kết quả nhỉnh hơn so với các thuật toán điển hình ở một số trường hợp data đầu vào, và không hề thua kém các thuật toán có sẵn về các mặt như thời gian đáp ứng về tổng thời gian xử lý thì luôn ít hơn các kỹ thuật khác.

Đánh giá Mô hình hồi quy tuyến tính trong PDOA

Bảng 4. 6 So sánh RAE của PDOA trong cả 3 trường hợp mô phỏng giả lập

	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

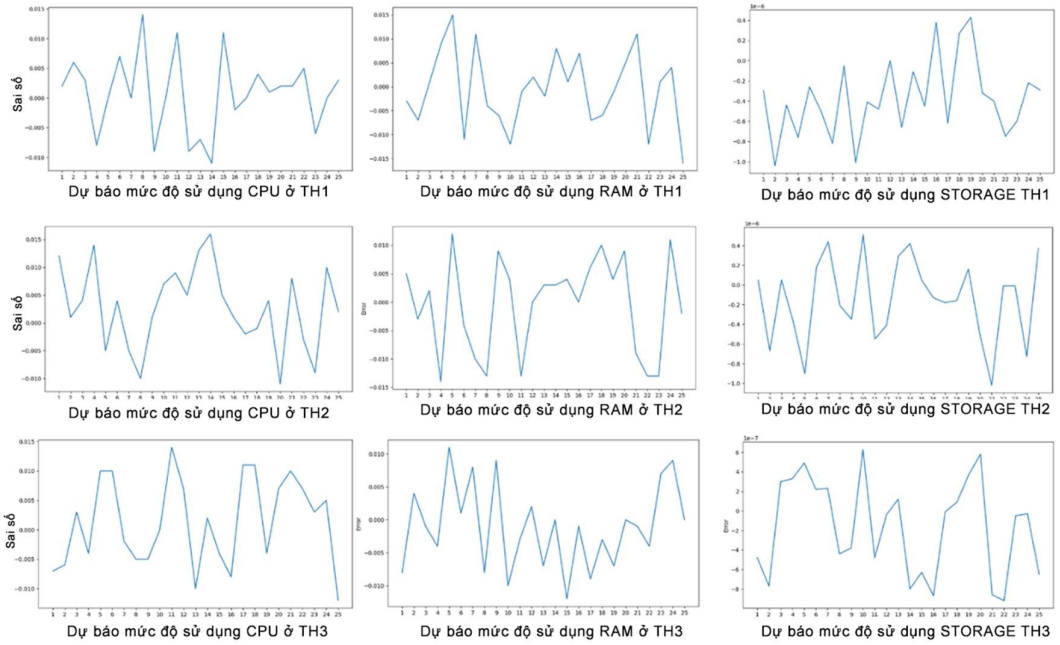
Để đánh giá độ chính xác của Mô hình hồi quy tuyến tính sử dụng trong PDOA, chúng tôi sử dụng sai số tuyệt đối tương đối (RAE) để xem xét mức độ chính xác cho bộ cân bằng tải sử dụng PDOA. Bảng 4.6 cho thấy, RAE xấu nhất và tốt nhất trong dự báo mức độ sử dụng CPU đều xảy ra ở trường hợp 1. Chúng ta có thể thấy rằng, RAE có thể chấp nhận đối với mô phỏng giả lập 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.

- *Trường hợp 24 Request:* Mức độ sử dụng CPU có sai số RAE là 0,565217; mức độ sử dụng BỘ NHỚ (RAM) có sai số RAE là 0,148717; và mức độ sử dụng BỘ LƯU TRỮ (STORAGE) có sai số RAE là 0,002531. Sai số RAE

thấp cho thấy Regression trong PDOA có khả năng dự đoán chính xác mức độ sử dụng CPU và BỘ NHỚ (RAM), và khả năng dự đoán rất chính xác mức độ sử dụng BỘ LƯU TRỮ (STORAGE).

- *Trường hợp 100 Request:* Mức độ sử dụng CPU có sai số RAE là 0,489868; mức độ sử dụng BỘ NHỚ (RAM) có sai số RAE là 0,286780; và mức độ sử dụng BỘ LƯU TRỮ (STORAGE) có sai số RAE là 0,006759. Regression trong PDOA có khả năng dự đoán chính xác mức độ sử dụng CPU và BỘ NHỚ (RAM) với sai số RAE khá thấp. Tuy nhiên, khả năng dự đoán mức độ sử dụng BỘ LƯU TRỮ (STORAGE) có sai số RAE thấp nhất, cho thấy khả năng dự đoán rất chính xác.
- *Trường hợp 997 Request:* Mức độ sử dụng CPU có sai số RAE là 0,324261; mức độ sử dụng BỘ NHỚ (RAM) có sai số RAE là 0,310741; và mức độ sử dụng BỘ LƯU TRỮ (STORAGE) có sai số RAE là 0,495361. Regression trong PDOA có khả năng dự đoán chính xác mức độ sử dụng CPU và BỘ NHỚ (RAM) với sai số RAE khá thấp. Tuy nhiên, khả năng dự đoán mức độ sử dụng BỘ LƯU TRỮ (STORAGE) có sai số RAE cao nhất, cho thấy khả năng dự đoán không chính xác như trường hợp trước.

Như vậy, Regression trong PDOA có khả năng dự đoán chính xác mức độ sử dụng CPU và BỘ NHỚ (RAM) tốt trong các trường hợp mô phỏng giả lập. Tuy nhiên, khả năng dự đoán mức độ sử dụng BỘ LƯU TRỮ (STORAGE) có thể không chính xác trong một số trường hợp, như trường hợp 997 Request.



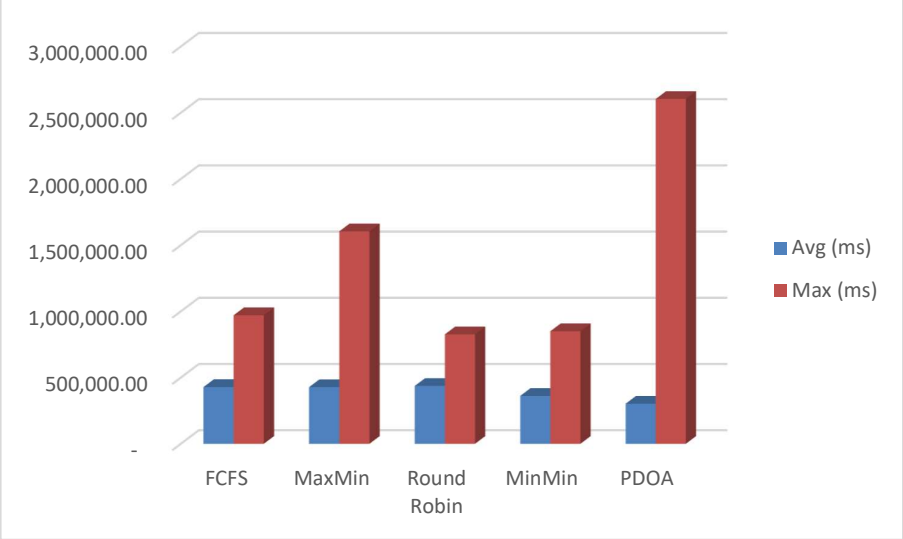
Hình 4. 10 Sai số Regression của thuật toán PDOA trong 3 trường hợp mô phỏng giả lập (bao gồm CPU, RAM, Storage)

Với cấu hình tương tự, ta mô phỏng giả lập ***với 10 máy ảo***, và tăng số request lên lần lượt là ***2000, 3000, 4000 và 5000***, ta thu được các kết quả khá khả quan. Bên cạnh đó, với số lượng request lớn, luận án bổ sung tham số speedups để tính toán và đánh giá.

Trường hợp 2000 Request *(Epigenomics_2000)* với kết quả như sau:

Bảng 4. 7 So sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 2000 request

Trường hợp 2000 Request	*Thời gian đáp ứng (Overall response time)*	
	Avg (ms)	**Max (ms)**
FCFS	428.274,37	970.917,64
MaxMin	428.107,28	1.604.589,61
Round Robin	437.059,82	**826.906,57**
MinMin	361.671,47	850.168,74
PDOA	**302.741,18**	2.604.299,18



Hình 4. 11 Biểu đồ so sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 2000 request

Dựa trên dữ liệu trên biểu đồ cột về thời gian đáp ứng trung bình (ms) và thời gian đáp ứng tối đa (max) của các thuật toán trong trường hợp 2000 Request (Epigenomics_2000), chúng ta có thể thấy PDOA có thời gian đáp ứng trung bình thấp nhất, chỉ là 302,741.18 ms, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. MinMin có thời gian đáp ứng trung bình thấp hơn so với các thuật toán FCFS, MaxMin và Round Robin, nhưng vẫn cao hơn so với PDOA. FCFS, MaxMin và Round Robin có thời gian đáp ứng trung bình cao hơn nhiều so với PDOA và MinMin, cho thấy hiệu năng thấp hơn trong việc xử lý các request.

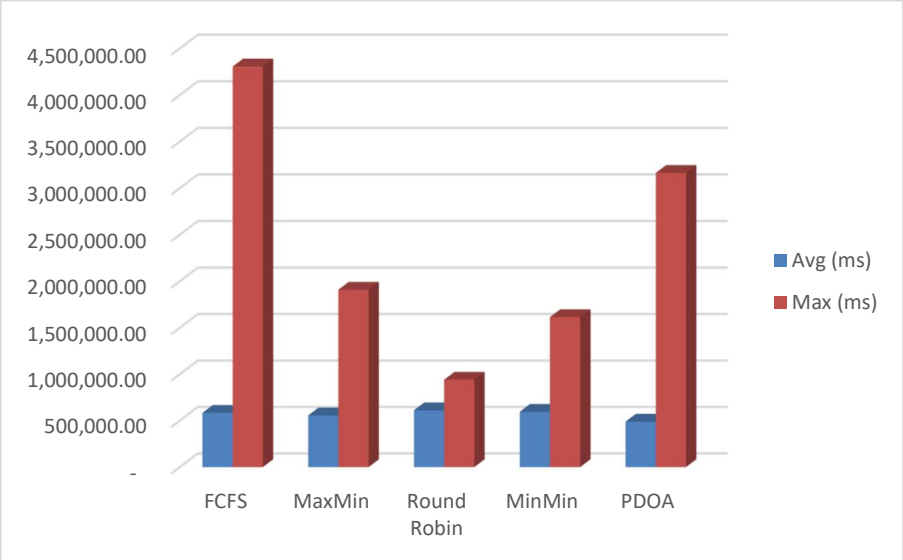
Về khả năng cân bằng tải, ta thấy PDOA có thời gian đáp ứng tối đa cao nhất là 2,604,299.18 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. MinMin và Round Robin cũng có thời gian đáp ứng tối đa không quá cao, nhưng vẫn cao hơn so với PDOA, cho thấy khả năng cân bằng tải tốt hơn so với FCFS và MaxMin. FCFS và MaxMin có thời gian đáp ứng tối đa rất cao, cho thấy khả năng cân bằng tải kém hơn so với PDOA, MinMin và Round Robin.

Nhìn chung trong trường hợp 2000 Request (Epigenomics_2000), PDOA có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. MinMin và Round Robin cũng cho thấy khả năng cân bằng tải tốt, trong khi FCFS và MaxMin có hiệu năng thấp hơn và khả năng cân bằng tải kém.

Trường hợp 3000 Request *(Epigenomics_3000)* với kết quả như sau:

Bảng 4. 8 So sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 3000 request

Trường hợp 3000 Request	*Thời gian đáp ứng (Overall response time)*	
	Avg (ms)	**Max (ms)**
FCFS	583.853,17	4.305.517,51
MaxMin	556.379,84	1.909.108,88
Round Robin	611.704,79	**940.533,31**
MinMin	595.498,99	1.614.385,78
PDOA	**489.072,31**	3.163.941,97



Hình 4. 12 Biểu đồ so sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 3000 request

Theo bảng 4.8 và hình 4.12 về thời gian đáp ứng trung bình (ms) và thời gian đáp ứng tối đa (max) của các thuật toán trong trường hợp 3000 Request (Epigenomics_3000), chúng ta có thể thấy rằng PDOA có thời gian đáp ứng trung bình thấp nhất là 489,072.31 ms, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. MinMin, MaxMin và FCFS có thời gian đáp ứng trung bình tương đối gần nhau, nhưng đều cao hơn so với PDOA. Round Robin có thời gian đáp ứng trung bình cao nhất, là 611,704.79 ms, cho thấy hiệu năng thấp hơn so với các thuật toán khác.

Về khả năng cân bằng tải thì PDOA có thời gian đáp ứng tối đa cao nhất là 3,163,941.97 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc.

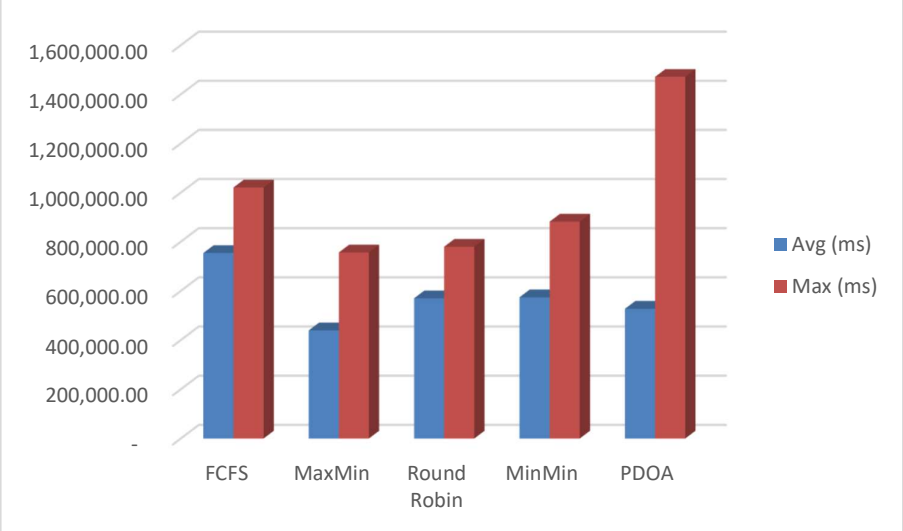
MinMin và MaxMin cũng có thời gian đáp ứng tối đa không quá cao, nhưng vẫn cao hơn so với PDOA, cho thấy khả năng cân bằng tải tốt hơn so với Round Robin và FCFS. Round Robin có thời gian đáp ứng tối đa cao hơn so với các thuật toán khác, cho thấy khả năng cân bằng tải kém hơn.

Ở trường hợp 3000 Request (Epigenomics_3000), PDOA có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. MinMin và MaxMin cũng cho thấy khả năng cân bằng tải tốt hơn, trong khi Round Robin và FCFS có hiệu năng thấp hơn và khả năng cân bằng tải kém hơn. Biểu đồ cột trên thể hiện sự khác biệt về hiệu năng và khả năng cân bằng tải của các thuật toán trong việc xử lý các request trên môi trường đám mây.

Trường hợp 4000 Request *(Epigenomics_4000)* với kết quả như sau:

Bảng 4. 9 So sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 4000 request

Trường hợp 4000 Request	*Thời gian đáp ứng (Overall response time)*	
	Avg (ms)	**Max (ms)**
FCFS	754.196,95	1.020.849,06
MaxMin	439.485,10	**756.334,06**
Round Robin	570.041,72	779.839,31
MinMin	574.321,37	881.864,81
PDOA	**527.492,89**	1.471.354,87



Hình 4. 13 Biểu đồ so sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 4000 request

Bảng 4.9 và hình 4.13 về thời gian đáp ứng trung bình (ms) và thời gian đáp ứng tối đa (max) của các thuật toán trong trường hợp 4000 Request (Epigenomics_4000), chúng ta thấy rằng PDOA có thời gian đáp ứng trung bình thấp nhất là 527,492.89 ms, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. MaxMin cũng có thời gian đáp ứng trung bình tương đối thấp, là 439,485.10 ms, cho thấy hiệu năng khá tốt. Round Robin, MinMin và FCFS có thời gian đáp ứng trung bình cao hơn, với FCFS có hiệu năng thấp nhất trong số các thuật toán được đánh giá.

Về khả năng cân bằng tải thì PDOA có thời gian đáp ứng tối đa cao nhất là 1,471,354.87 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. MaxMin, Round Robin và MinMin cũng có thời gian đáp ứng tối đa không quá cao, cho thấy khả năng cân bằng tải tốt hơn so với FCFS. FCFS có thời gian đáp ứng tối đa cao nhất, là 1,020,849.06 ms, cho thấy khả năng cân bằng tải kém hơn so với các thuật toán khác.

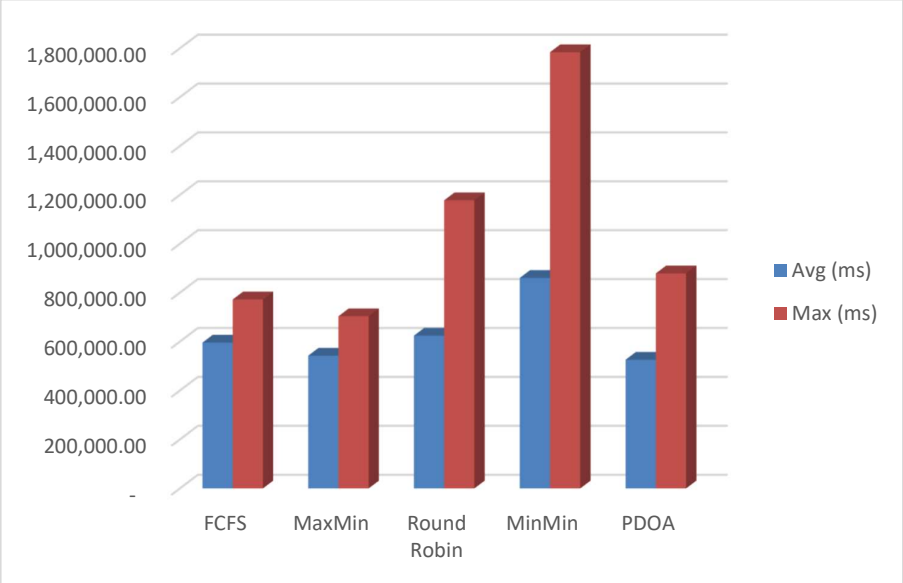
Như vậy, trong trường hợp 4000 Request (Epigenomics_4000), PDOA có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. MaxMin cũng cho thấy hiệu năng tốt và khả năng cân bằng tải tốt. Round Robin, MinMin và FCFS có hiệu năng thấp hơn và khả năng cân bằng tải kém hơn. Biểu đồ cột trên thể hiện sự khác biệt về hiệu năng và khả năng cân bằng tải của các thuật toán trong việc xử lý các request trên môi trường đám mây.

Trường hợp 5000 Request *(Epigenomics_5000)* với kết quả như sau:

Bảng 4. 10 So sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 5000 request

Trường hợp 5000 Request	*Thời gian đáp ứng (Overall response time)*	
	Avg (ms)	**Max (ms)**
FCFS	596.140,40	772.364,58
MaxMin	542.803,93	**703.800,06**
Round Robin	624.502,55	1.178.137,87
MinMin	860.489,68	1.784.249,58

PDOA	**526.067,73**	879.050,88



Hình 4. 14 Biểu đồ so sánh thời gian đáp ứng các thuật toán với thuật toán PDOA ở trường hợp 5000 request

Theo hình 4.14 và bảng 4.10 về thời gian đáp ứng trung bình (ms) và thời gian đáp ứng tối đa (max) của các thuật toán trong trường hợp 5000 Request (Epigenomics_5000), chúng ta có thể thấy PDOA có thời gian đáp ứng trung bình thấp nhất là 526,067.73 ms, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. MaxMin và FCFS cũng có thời gian đáp ứng trung bình tương đối thấp, tuy nhiên vẫn cao hơn so với PDOA. Round Robin và MinMin có thời gian đáp ứng trung bình cao hơn, với MinMin có hiệu năng thấp nhất trong số các thuật toán được đánh giá.

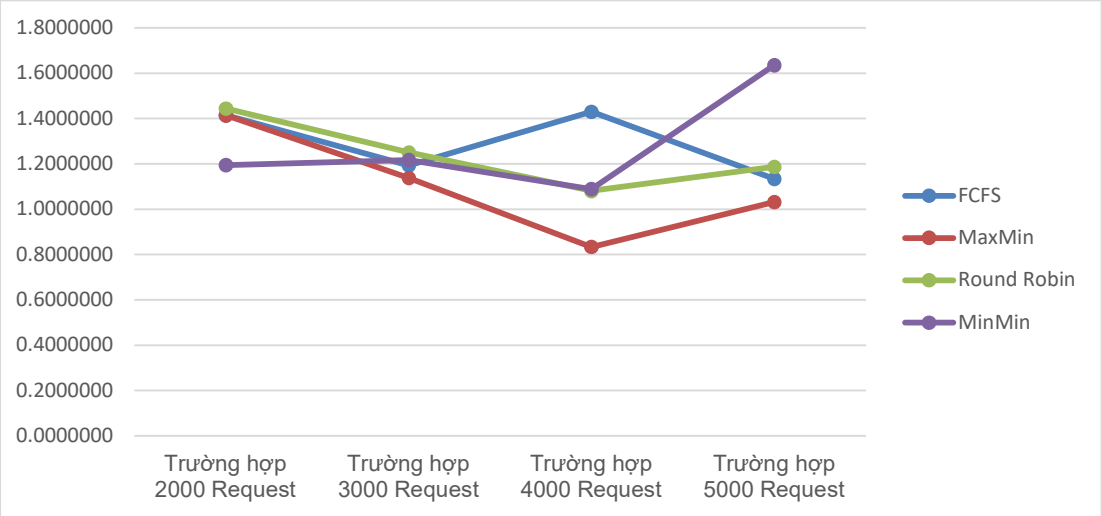
Về khả năng cân bằng tải thì PDOA có thời gian đáp ứng tối đa thấp nhất là 879,050.88 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. MaxMin, FCFS và Round Robin cũng có thời gian đáp ứng tối đa không quá cao, tuy nhiên vẫn cao hơn so với PDOA, cho thấy khả năng cân bằng tải tốt hơn so với MinMin. MinMin có thời gian đáp ứng tối đa cao nhất, là 1,784,249.58 ms, cho thấy khả năng cân bằng tải kém hơn so với các thuật toán khác.

Trong trường hợp 5000 Request (Epigenomics_5000), PDOA có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. MaxMin và FCFS cũng cho thấy hiệu năng tốt và khả năng cân bằng tải tốt. Round Robin có hiệu năng

tương đối và khả năng cân bằng tải tốt. Tuy nhiên, MinMin có hiệu năng thấp và khả năng cân bằng tải kém hơn. Biểu đồ cột trên thể hiện sự khác biệt về hiệu năng và khả năng cân bằng tải của các thuật toán trong việc xử lý các request trên môi trường đám mây.

Bảng 4. 11 So sánh Speedups các thuật toán với thuật toán PDOA ở 4 trường hợp từ 2000 đến 5000 request

Speedup	Trường hợp 2000 Request	Trường hợp 3000 Request	Trường hợp 4000 Request	Trường hợp 5000 Request
FCFS	1,4146552	1,1937972	1,4297765	1,1332009
MaxMin	1,4141032	1,1376229	*0,8331583*	1,0318138
Round Robin	1,4436748	1,2507451	1,0806624	1,1871143
MinMin	1,1946557	1,2176093	1,0887756	1,6357013



Hình 4. 15 Biểu đồ so sánh Speedups các thuật toán với thuật toán PDOA ở 4 trường hợp từ 2000 đến 5000 request

Theo bảng 4.11 và hình 4.15 về số liệu so sánh khả năng tăng tốc (speedup, tính bằng thời gian đáp ứng trung bình) của thuật toán PDOA so với các thuật toán phổ biến khác (FCFS, MaxMin, MinMin, Round Robin) trong 4 trường hợp, ta có thể nhận xét như sau:

- *Trường hợp 2000 Request:* PDOA có khả năng tăng tốc lớn nhất so với Round Robin, với tỷ lệ speedup là 1,4436748. PDOA có khả năng tăng tốc cao tiếp theo so với FCFS và MaxMin, với tỷ lệ speedup lần lượt là 1,4146552 và

1,4141032. PDOA có khả năng tăng tốc thấp hơn nhất khi so với MinMin, với tỷ lệ speedup là 1,1946557.

- *Trường hợp 3000 Request:* PDOA có khả năng tăng tốc lớn nhất với MinMin, với tỷ lệ speedup là 1,2176093. PDOA có khả năng tăng tốc cao tiếp theo là so với FCFS và MaxMin, với tỷ lệ speedup lần lượt là 1,1937972 và 1,1376229. PDOA có khả năng tăng tốc thấp nhất khi so với Round Robin, với tỷ lệ speedup là 1,2507451.
- *Trường hợp 4000 Request:* PDOA có khả năng tăng tốc lớn nhất là so với Round Robin, với tỷ lệ speedup là 1,0806624. PDOA có khả năng tăng tốc cao tiếp theo là so với FCFS, với tỷ lệ speedup lần lượt là 1,4297765. Tuy nhiên trong trường hợp này, PDOA so vớp MaxMin không tăng tốc mà là giảm đi tương ứng với tỷ lệ Speedup là 0,8331583. PDOA có khả năng tăng tốc thấp nhất là so với MinMin, với tỷ lệ speedup là 1,0887756.
- *Trường hợp 5000 Request:* PDOA có khả năng tăng tốc lớn nhất là so với Round Robin, với tỷ lệ speedup là 1,1871143. PDOA có khả năng tăng tốc cao tiếp theo là so với FCFS và MaxMin, với tỷ lệ speedup lần lượt là 1,1332009 và 1,0318138. PDOA có khả năng tăng tốc thấp nhất so với MinMin, nhưng vẫn có tỷ lệ speedup cao hơn là 1,6357013.

Tổng quát, dựa trên khả năng tăng tốc của PDOA so với các thuật toán khác, PDOA thể hiện khả năng cân bằng tải tốt và có khả năng tăng tốc tương đối so với FCFS, MaxMin và Round Robin trong các trường hợp 2000, 3000 và 5000 Request. Tuy nhiên, PDOA có khả năng tăng tốc thấp hơn so với MinMin trong một số trường hợp.

4.3 HÀNH VI NGƯỜI DÙNG CLOUD VÀ THUẬT TOÁN K-CTPA

4.3.1 Hành vi người dùng cloud với độ ưu tiên tác vụ

Hành vi người dùng cloud đã được chú ý và nghiên cứu, điển hình như Oracle [132] cũng đã nghiên cứu về người dùng cloud và hành vi của người dùng cloud. Trong bài báo của Maryam Alruwaythi [133] và nghiên cứu của Chen [134] thì hành vi người dùng cloud sẽ dẫn đến những lòng tin trên đám mây, từ đó tạo ra độ ưu tiên của người dùng trên đám mây.

Tác vụ (task) [135] là một tiến trình hoặc nhiều tiến trình sẽ được thực thi trên một nút tính toán hiện hữu bởi một máy ảo trong điện toán đám mây.

Độ ưu tiên của các tác vụ là việc xác định mức độ ưu tiên của tác vụ cần thực hiện trong lập lịch tác vụ, vì chúng có ảnh hưởng lớn đến chất lượng dịch vụ mà nhà cung cấp dịch vụ cam kết. Độ ưu tiên của các tác vụ (priority task) phụ thuộc vào nhiều yếu tố như: khả năng sử dụng CPU, RAM, băng thông, chiều dài và kích thước của mỗi tác vụ, thời gian hoàn thành của mỗi tác vụ…việc xác định mức độ ưu tiên của tác vụ có ảnh hưởng rất lớn đến vấn đề lập lịch/cân bằng tải trong môi trường điện toán đám mây một cách tối ưu.

Trong nghiên cứu hành vi người dùng cloud của luận án này, mục đích là áp dụng thuật toán kNN (k Nearest Neighbours) để phân loại các công việc dựa trên mức độ ưu tiên của chúng, tương ứng với các yêu cầu nhận được. Các yếu tố xác định mức độ ưu tiên bao gồm lượng năng lượng tiêu thụ của công việc (Power consumed), mức độ sử dụng CPU (CPU Usages), mức độ sử dụng RAM (RAM Usages) và chi phí (Costing) để thực hiện công việc đó trong môi trường cloud. Khi đã xác định được độ ưu tiên của các jobs/tasks, cơ chế cân bằng tải sẽ phân phối các yêu cầu chứa các task ưu tiên cao đến những máy ảo/hosts có khả năng xử lý mạnh mẽ hơn, tức là có khả năng nhàn rỗi cao hơn. Đồng thời, những request yêu cầu xử lý nặng sẽ được chuyển tới máy ảo/hosts có tải hoạt động thấp nhất. Bằng cách tiếp cận này, thuật toán được đề xuất mong muốn tối ưu hóa quá trình xử lý và cân bằng tải trên cloud, giúp ứng dụng có thể hoạt động hiệu quả trong môi trường cloud thực tế. Trong luận án này, tác giả tạm đặt tên thuật toán là k-CTPA (k-NN Classification of Task-Priority Algorithm).

4.3.2 Thuật toán k-CTPA

Về mục tiêu:

- Giảm thiểu rủi ro cho hệ thống máy chủ.
- Giảm thiểu thời gian đáp ứng cho các yêu cầu trong môi trường điện toán đám mây.
- Hạn chế tối đa sự mất cân bằng tải giữa các máy ảo, ngăn chặn mất cân bằng tải.
- Giúp giải quyết các yêu cầu nhanh hơn, phân loại được các tác vụ (Task) với các độ ưu tiên khác nhau tương ứng với các yêu cầu (Request), sử dụng hiệu quả hơn nguồn tài nguyên trên cloud, đáp ứng tốt nhất cho người dùng.

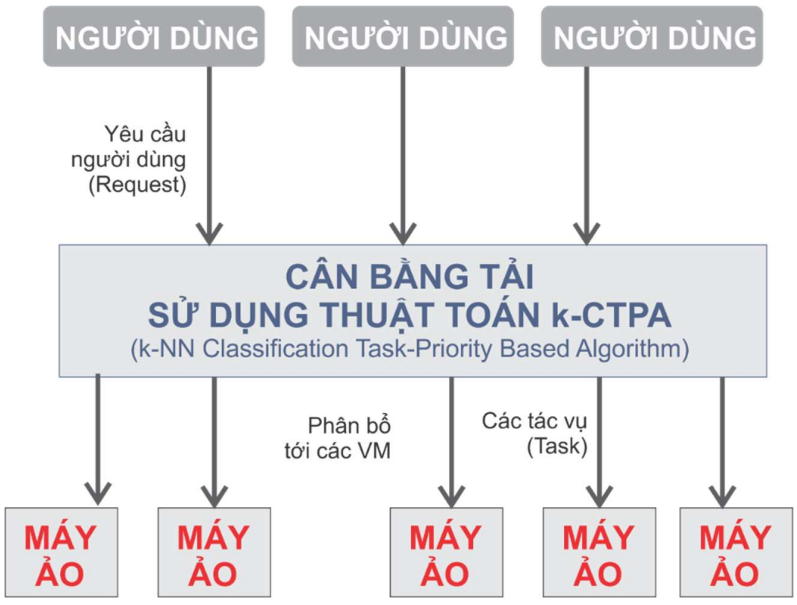
- Phân lớp được các yêu cầu (Coming Request) tiếp theo tương ứng với độ ưu tiên đã được phan lớp ở trên, từ đó có kế hoạch đưa các yêu cầu này sang những máy ảo / host có khả năng xử lý tải tương ứng.
- Sắp xếp các máy ảo / host / tài nguyên theo mức độ sử dụng từ cao đến thấp để phân bổ task cho hợp lý.

Giả định:

- Cơ chế cân bằng tải là sẽ luôn nắm được thông tin về các dịch vụ đang hoạt động trên từng máy ảo tại mọi thời điểm.
- Kịch bản mô phỏng này tập trung vào mô phỏng dịch vụ Web, vì các server web có khả năng xác định trước được thời gian xử lý cho mỗi dịch vụ web đang chạy trên mỗi máy ảo.
- Khi hai máy ảo có cùng cấu hình về RAM, bộ vi xử lý, và I/O, thì thời gian để thực thi các dịch vụ có đặc điểm tương đồng nhau sẽ gần như không có sự chênh lệch.

Mô hình nghiên cứu:

- Trong mô hình nghiên cứu trước giờ thì bộ cân bằng tải sẽ chạy theo sơ đồ như sau:



Hình 4. 16 Sơ đồ mô hình nghiên cứu trên Cloud thuật toán k-CTPA

- Thuật toán đề xuất là nơi xử lý các yêu cầu và đưa vào các máy ảo phù hợp để cân bằng tải.
- Trong mô hình này sử dụng Regression (dựa vào các tính chất của request) để phân loại các request đầu vào, dự báo các thông số cloud cần để xử lý task mà request này đem tới (Power, CPU Usage, RAM Usage). Để phân lớp với kỹ thuật Regression này, thuật toán sử dụng bộ data trong lịch sử cloud được lưu lại (sử dụng dữ liệu gần nhất).
- Sau đó với số liệu (Power, CPU Usage, RAM Usage) mà cloud cấn có để xử lý job / task tương ứng, đã được tính toán ở trên, thuật toán sử dụng tiếp theo là k-NN để phân lớp tác vụ (task /job) trong đó bộ dữ liệu là dữ liệu thực đã lưu lại kết hợp với dữ liệu dự đoán mới tính toán ra ở trên, phân lớp các các vụ dựa vào độ ưu tiên, từ đó phân bổ vào các máy ảo tương ứng.
- Mô hình này mô phỏng thuật toán một cách hiệu quả, đồng thời lập kế hoạch cho các yêu cầu sắp tới để đảm bảo tải trên hệ thống được cân đối. Thông qua việc áp dụng thuật toán này, khả năng giảm thiểu tải truyền thông giữa các máy ảo và nguồn tài nguyên sẵn có được nâng cao, từ đó giảm băng thông không cần thiết và tăng cường khả năng phục vụ các yêu cầu từ người dùng.

Thuật toán k-NN và ứng dụng vào phân lớp task theo độ ưu tiên

Độ ưu tiên của tác vụ (Task)

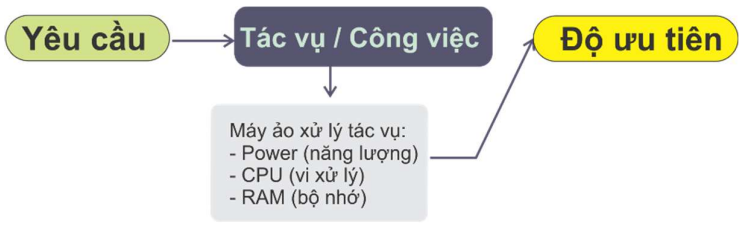
Trong máy tính, một task / job được thực hiện sẽ tiêu hao nguồn năng lượng nhất định, kèm theo đó là mức độ sử dụng CPU của máy tính, mức độ sử dụng bộ nhớ tạm thời RAM, v/v. và tất cả đều được tính toán ra chi phí thực hiện công việc đó theo thời gian hoặc MIPS. Chính vì thế, luận án này dựa vào các đặc điểm đó để tính toán ra độ ưu tiên của task / job mà máy tính phục vụ. Task có mức tiêu hao năng lượng cao (Power consumed), mức độ sử dụng CPU (CPU Usage) lớn, cũng như mức độ sử dụng RAM nhiều (RAM Usage), hoặc có chi phí cao (Cost) thì sẽ có độ ưu tiên cao và ngược lại.

Độ ưu tiên trong bài luận án này là đại lượng được nghiên cứu 3 chiều, tổng hợp từ Power (Power Consume), CPU và RAM

$$Priority = \{ Po, CPU, RAM \} \quad (9.1)$$

Phân lớp tác vụ / task dựa theo độ ưu tiên

Dựa vào bộ dữ liệu trong quá khứ khi xử lý những task / job đầu tiên, ta sử dụng k-NN để phân lớp độ ưu tiên cho các task tiếp theo. Tương ứng với mỗi yêu cầu (Request) sẽ có một task / job mà máy tính cần phải thực hiện để phục vụ người dùng. Chính vì thế, bất kỳ một request được gửi đến cloud đề có thể được phân lớp dựa trên task tương ứng của nó.



Hình 4. 17 Tính toán độ ưu tiên của các Request

Thuật toán đề xuất ***k-CTPA*** (*kNN Classification Task Priority Algorithm)*, dựa vào yếu tố độ ưu tiên của tác vụ (Task Priority được mô tả ở trên) tương ứng với các request, kèm theo đó là một số thuộc tính khác, ta sử dụng thuật toán k-NN để phân lớp các request này, từ đó ta biết cách phân bố tài nguyên cho các request này một cách tối ưu nhất. Song song, các tài nguyên (máy ảo/ host) được sắp xếp theo mức độ sử dụng tăng dần. Kết hợp với đánh giá số lần sai, và sai số, ta cải thiện thuật toán bằng cách áp dụng học máy, tuy nhiên, việc áp dụng này sẽ ít diễn ra vì có sai số cho phép.

Dựa vào tham khảo các công trình liên quan, luận án này xin đề xuất thuật toán gồm 3 nhóm module chính:

(1) Module tính toán ra các thông số của request bằng thuật toán Regression:

Trong module này, thuật toán Regression sẽ dựa vào các thuộc tính của request và các yếu tố mà tính toán ra các thông số sử dụng tài nguyên của Task/job tương ứng với request đó. Các thuộc tính bao gồm: Size, Response Length, Max Length, …

$$Po_{New} = Regression(Request, Power) \quad (9.2)$$

$$CPU_{New} = Regression(Request, CPU) \quad (9.3)$$

$$RAM_{New} = Regression(Request, RAM) \quad (9.4)$$

Trong đó X_i = là các thuộc tính của Request khi gửi lên cloud.

Request = { $X_1, X_2, \ldots, X_n$ }, với X_i là các thuộc tính của Request

Po_{New} : Power dự đoán Power new: Power ghi nhận được trong quá khứ

CPU_{New} : CPUdự đoán CPU new: CPU ghi nhận được trong quá khứ

RAM_{New} : RAM dự đoán RAM new: RAM ghi nhận được trong quá khứ

Ở đây có thể sử dụng nhóm 3 yếu tố {Po, CPU, RAM} để tổng hợp tính toán, hoặc tính toán riêng biệt từng đại lượng.

(2) Module phân lớp tác vụ theo độ ưu tiên:

Trong module này sẽ sử dụng thuật toán phân lớp k-NN (với k=3~5) để phân lớp request đang xét, dựa vào tính chất của độ ưu tiên các tác vụ. Việc phân lớp này sẽ thông qua việc xây dựng mô hình phân lớp k-NN của các Request đã được xử lý trong quá khứ, và đánh nhãn tương ứng từ 1 tới 5, là tương ứng với mức độ ưu tiên từ 1 tới 5. Mức 1 là độ ưu tiên thấp nhất, mức 5 là độ ưu tiên cao nhất. Dựa vào mô hình này, ta phân lớp được Request đang cần xử lý, xác định được label tương ứng (từ 1 đến 5). Sau đó ta chọn ra máy ảo có thứ tự tương ứng 1 đến 5. Thứ tự này được sắp xếp dựa trên mức độ rảnh hay ít tải của máy, tức là mức 1 là máy tải nhiều nhất, và mức 5 là máy tải ít nhất, rãnh nhất.

$$VM_{select} = k\text{-}NN(Po, CPU, RAM) \quad (9.5)$$

Trong đó:

VM_{select} là máy ảo được chọn ra

k-NN là hàm phân lớp từ mô hình KNN đã được xây dựng dựa trên bộ dữ liệu quá khứ của các request

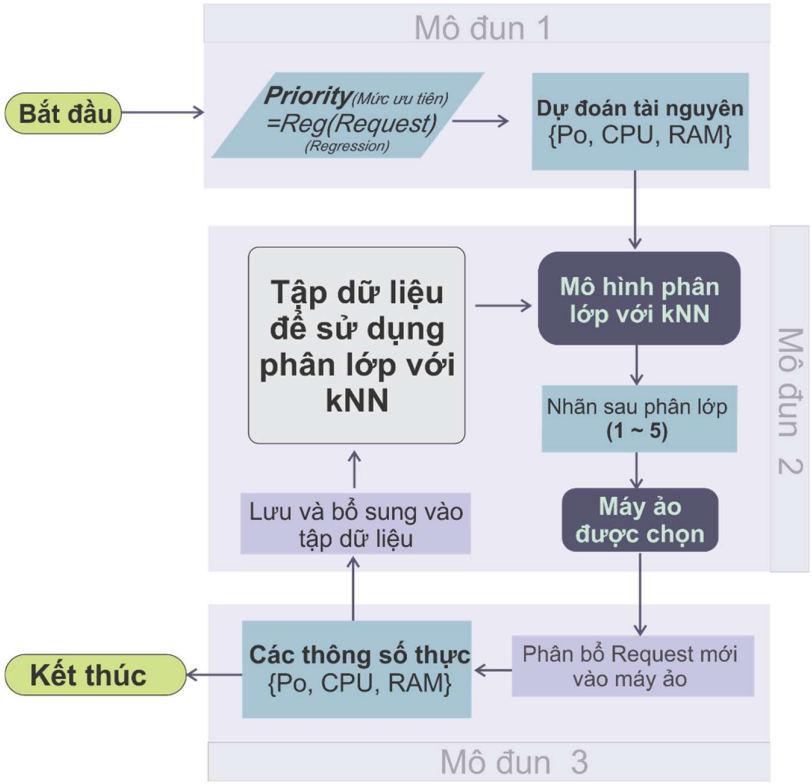
Po : là Power dự đoán tính toán từ Module 1

CPU: là mức sử dụng CPU dự đoán tính toán từ Module 1

RAM: là mức sử dụng RAM dự đoán tính toán từ Module 1

(3) Module phân bổ các dịch vụ (chọn máy ảo)

Module này có nhiệm vụ phân bổ các yêu cầu đến các máy ảo thông qua loại request và máy ảo phù hợp. Nếu một yêu cầu được gửi tới thì yêu cầu này được phân loại bởi module 1, và các VM đang xét kể cả VM không tải cũng được phân cụm theo module 2. Ở đây, Module 3 có nhiệm vụ phân bổ Request đang xét vào máy ảo đã tìm thấy từ Module 2, và từ đó xử lý cho request đó, và trả về kết quả của request, lưu vào lịch sử bộ nhớ các request gần nhất đã xử lý, làm dữ liệu đầu vào cho quá trình xây dựng mô hình KNN ở Module 2.



Hình 4. 18 Sơ đồ thuật toán đề xuất k-CTPA

Sơ đồ mã giả thuật toán ***k-CTPA***

Thuật toán k-CTPA
Input: Tập Requests
Output: Phân bổ máy ảo cho các request trong tập Requests

```
14. For each Request in CloudRequests
15.     isLocated = false;
16.     Priority = {Po, CPU, RAM}new = Regression(T1,T2…. Tn); // Module 1
17.     Request.PriorClass = k-NN(Priority); //k-NN là mô hình phân lớp tác vụ
18.     For each VM in VMList
19.         If isFitSituation(Request.PriorClass, VM)
20.             AllocateRequestToVM(VM, Request); // Module 3
21.             isLocated = true;
22.         End If
23.     End For
24.     If(!isLocated)
25.         VM = VMList.getSelectedVM(); // Module 2
26.         AllocateRequestToVM(VM, Request);
27.     End If
28. End For
```

Trong đoạn mã giả trên của thuật toán k-CTPA, thuật toán sẽ sử dụng một vòng lặp để lắng nghe tất cả các Request có trong danh sách hang đợi các Request được gửi lên bộ cân bằng tải (ở đây là ***CloudRequests***). Khi nào hết danh sách này thì sẽ không phân bổ nữa. Trong đó, thuật toán sử dụng biến ***isLocated*** (kiểu luận lý) để làm cờ đánh dấu rằng Request đang xét đã được phân bổ hay chưa. Mới vào vòng

lặp, biến ***isLocated*** được tạo giá trị mặc định là *false*. Sau đó, thuật toán tính toán ra vector Priority với 3 chiều là PowerConsume, CUP Usage và RAM Usage (*Priority ={Po, CPU, RAM}*) cần dùng để thực hiện Request đang xét. Việc tính toán này dựa trên lịch sử số liệu của các Request trước đó $T_1, T_2, \ldots T_n$, trong đó n là số request đã được lưu. T_i là các đại lượng của Request thứ *i* được lưu lại, T_i bao gồm các đại lượng đầu vào: *MaxLength, FileSize, OutputSize*…; và các đại lượng xử lý do Cloud đã thực hiện để xử lý Request thứ *i* bao gồm *PowerConsume*, *CPU Usage* và *RAM Usage*. Dữ liệu n Request trong lịch sử này sẽ xây dựng nên hàm Regression (hồi quy tuyến tính) để dự báo và tính toán ra các đại lượng *Priority* cho Request đang xét. Sau đó, dung dữ liệu Priority này để chạy k-NN phân lớp cho Request đang xét. Và lớp này được gán vào thuộc tính *PriorClass* của Request. Sau khi chạy ra thông số Priority cho Request, thì thuật toán duyệt vòng lặp để duyệt qua các máy ảo đang có trên cloud. Tương ứng với từng máy, thuật toán xem xét máy ảo đó có phù hợp với độ ưu tiên của Request đang xét hay không, thông qua hàm ***isFitSituation(Request.PriorClass, VM)***. Nếu thỏa thì sẽ phân bổ request đang xét vào máy ảo đó ***AllocateRequestToVM(VM, Request),*** và đồng thời gán giá trị biến ***isLoacated = true***. Nếu trường hợp không tìm ra máy ảo nào phù hợp, thì sẽ kết thúc vòng lặp. Lúc này, chạy hết vòng lặp và biến ***isLocated*** vẫn mang giá trị *false*, và lúc này Request chưa được phân bổ. Vì vậy, thuật toán phân bổ Request này vào máy ảo đầu tiên của danh sách máy ảo thông qua đoạn lệnh **VM = VMList.getSelectedVM()**. Việc phân bổ này đảm bảo nếu có request nào được dự báo không nằm trong dữ liệu của thuật toán, vẫn được phân bổ và xử lý phục vụ người dùng.

Theo thuật toán đề xuất, đầu ra của phân lớp request được tính toán chính là thời gian xử lý xét, và không biết được giá trị max hay giá trị min, nên có thể lưu lại 1 số lượng nhất định thời gian xử lý của các request trước nhằm thực hiện tính toán và phân bổ, số lượng này chính là *n* được nhắc đến trên sơ đồ mã giả trên. Chính vì thế, luận án này xin được sử dụng lại phương pháp loại suy hoặc newton để tính toán ra các máy ảo đang thuộc lớp nào. Cụ thể, là lấy max min của thời gian xử lý của n Request trong quá khứ tưng ứng với các máy ảo xử lý, từ đó chia thành k phần bằng nhau (tương ứng với k lớp của k-NN), và phân lớp cho máy ảo tương ứng từ 1 tới k. Từ đó, phân bổ Request được phân lớp bởi k-NN vào đúng máy ảo có phân lớp k

(phân lớp này là loại suy newton) phân bổ phù hợp. Tuy nhiên sẽ hiệu chỉnh một số thay đổi, hoặc đưa vào các hệ số và tham số, tùy thuộc vào kết quả mô phỏng giả lập.

Đánh giá thuật toán k-CTPA

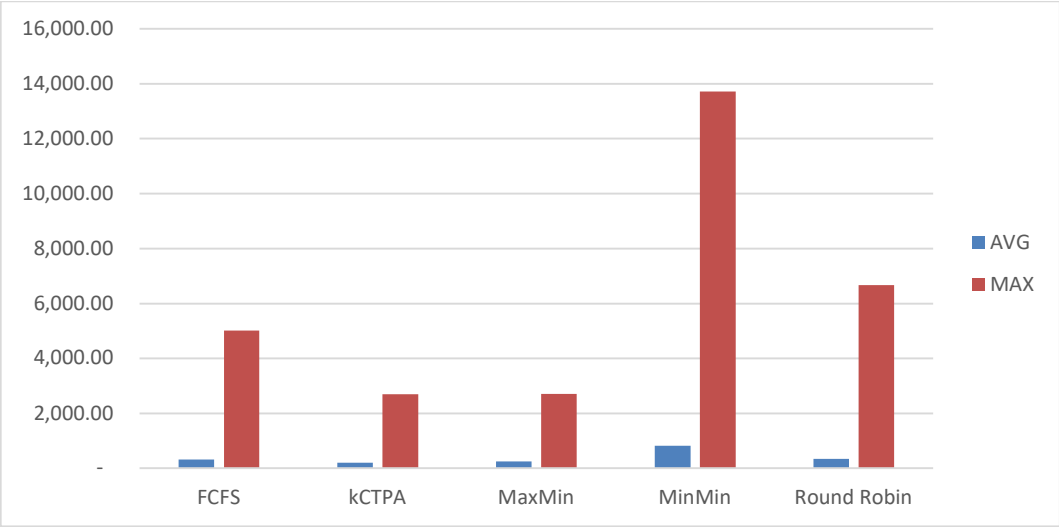
Kết quả thu được từ thuật toán k-CTPA đã thành công trong việc đạt được các định hướng đã đề ra, bao gồm việc hạn chế số lượng yêu cầu đợi xử lý, nâng cao hiệu suất thời gian xử lý và giảm thời gian phản hồi khi so sánh với bốn thuật toán truyền thống khác. Điều này cũng có nghĩa là với thuật toán được đề xuất, hiệu suất của điện toán đám mây được cải thiện so với bốn thuật toán ***MaxMin, Round Robin, MinMin*** và ***FCFS***. Thuật toán đề xuất đã cho thấy hiệu quả khi máy ảo có số lượng cao lên thì k-CTPA đảm bảo thời gian phản hồi và thời gian xử lý tốt, giảm chi phí của các trung tâm dữ liệu đám mây.

CÀI ĐẶT THUẬT TOÁN k-CTPA

Kết quả chạy mô phỏng giả lập trên CloudSim với 5 máy ảo được dựng sẵn để đáp ứng các yêu cầu, các yêu cầu được khởi tạo với chiều dài và kích thước ngẫu nhiên, số lượng Request lần lượt là 30, 60, 100 và 1000. Sau đó kết quả này được so sánh với các thuật toán Round-Robin, MaxMin, MinMin và FCFS. Đầu tiên ta xét trường hợp với 30 Requets, ta có thời gian thực hiện như bảng 4.4.

Bảng 4. 12 So sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 30 request

Thời gian thực hiện (ms)	FCFS	kCTPA	MaxMin	MinMin	Round Robin
AVG	306,69	203,39	246,42	810,12	340,91
MAX	5.009,65	2.694,31	2.713,20	13.719,94	6.659,64
MIN	0,24	0,11	0,12	0,15	0,11



Hình 4. 19 Biểu đồ so sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 30 request

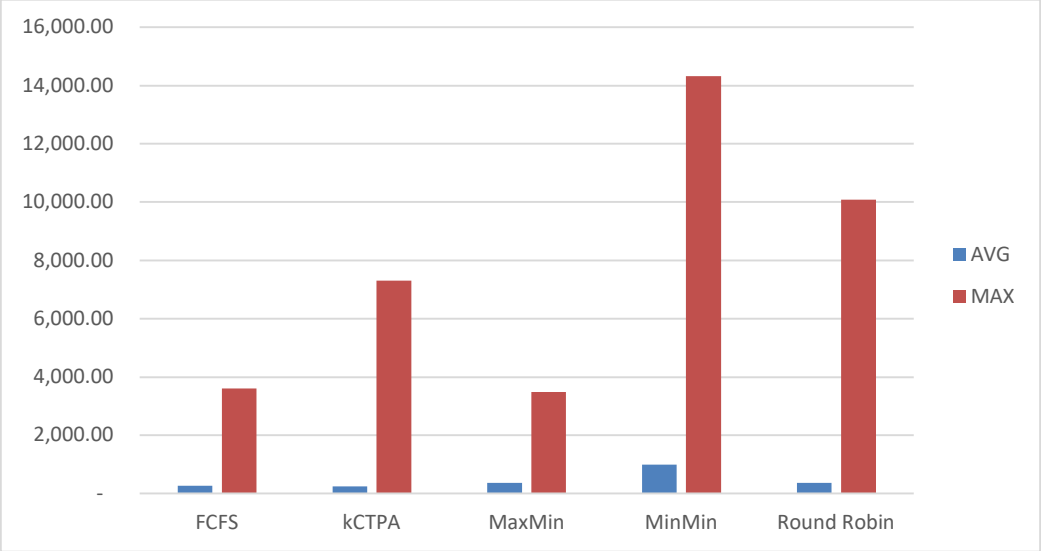
Dựa trên bảng 4.12 và hình 4.19 về thời gian thực hiện (ms) của các thuật toán (FCFS, MaxMin, MinMin, Round Robin) so với thuật toán kCTPA trong trường hợp 30 request, chúng ta có thể thấy kCTPA có thời gian thực hiện trung bình thấp nhất, chỉ là 203,39 ms, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. FCFS và MaxMin cũng có thời gian thực hiện trung bình tương đối thấp, lần lượt là 306,69 ms và 246,42 ms. Round Robin có thời gian thực hiện trung bình cao hơn, là 340,91 ms. MinMin có thời gian thực hiện trung bình cao nhất, là 810,12 ms, cho thấy hiệu năng thấp nhất trong số các thuật toán được đánh giá.

Về khả năng cân bằng tải thì kCTPA có thời gian thực hiện tối đa thấp nhất, chỉ là 2.694,31 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. FCFS, MaxMin và Round Robin cũng có thời gian thực hiện tối đa không quá cao, tuy nhiên vẫn cao hơn so với kCTPA, cho thấy khả năng cân bằng tải tốt hơn so với MinMin. MinMin có thời gian thực hiện tối đa cao nhất, là 13.719,94 ms, cho thấy khả năng cân bằng tải kém hơn so với các thuật toán khác. Tóm lại, trong trường hợp 30 request, kCTPA có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. FCFS và MaxMin cũng cho thấy hiệu năng tốt và khả năng cân bằng tải tốt. Round Robin có hiệu năng tương đối và khả năng cân bằng tải tốt. Tuy nhiên, MinMin có hiệu năng thấp và khả năng cân bằng tải kém hơn. Biểu đồ cột trên thể hiện sự khác biệt về hiệu năng và khả năng cân bằng tải của các thuật toán trong việc xử lý các request trên môi trường đám mây.

Với kết quả mô phỏng giả lập với 30 Request trở lại, ta thấy thuật toán Round-Robin chiếm ưu thế và xử lý nhanh, thuật toán MaxMin cũng khá ổn định. Thuật toán FCFS thì chưa có thế mạnh. Tuy nhiên thuật toán đề xuất k-CTPA cũng khá ổn định, và chứng tỏ dần ổn định và tốt hơn khi xử lý nhiều request hơn.

Bảng 4. 13 So sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 60 request

Thời gian thực hiện (ms)	FCFS	kCTPA	MaxMin	MinMin	Round Robin
AVG	268,58	249,45	364,92	986,94	369,41
MAX	3.599,42	7.311,10	3.491,21	14.318,26	10.084,81
MIN	0,19	0,35	0,13	0,15	0,16



Hình 4. 20 Biểu đồ so sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 60 request

Theo bảng 4.12 và hình 4.20 về thời gian thực hiện (ms) của các thuật toán (FCFS, MaxMin, MinMin, Round Robin) so với thuật toán kCTPA trong trường hợp 60 request, ta thấy kCTPA có thời gian thực hiện trung bình gần nhất với FCFS, chỉ là 249,45 ms, cho thấy hiệu năng tương đương so với các thuật toán khác trong việc xử lý các request. MaxMin và Round Robin có thời gian thực hiện trung bình cao hơn, lần lượt là 364,92 ms và 369,41 ms. MinMin có thời gian thực hiện trung bình cao nhất, là 986,94 ms, cho thấy hiệu năng thấp nhất trong số các thuật toán được đánh giá. Về khả năng cân bằng tải thì kCTPA có thời gian thực hiện tối đa thấp nhất,

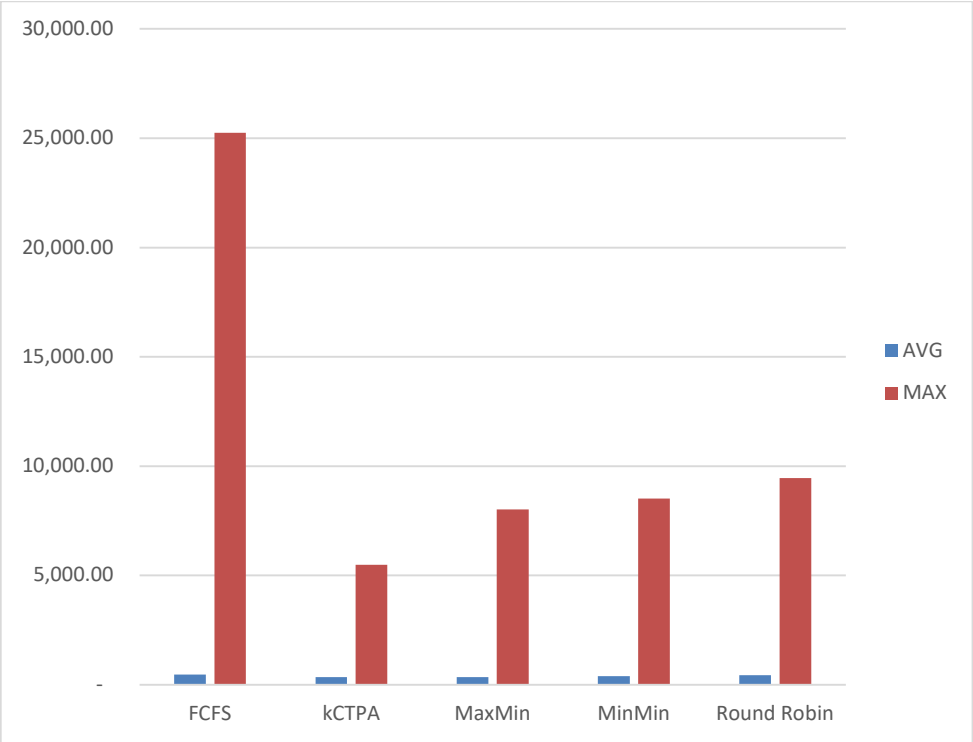
chỉ là 7.311,10 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. FCFS, MaxMin và Round Robin cũng có thời gian thực hiện tối đa không quá cao, tuy nhiên vẫn cao hơn so với kCTPA, cho thấy khả năng cân bằng tải tốt hơn so với MinMin. MinMin có thời gian thực hiện tối đa cao nhất, là 14.318,26 ms, cho thấy khả năng cân bằng tải kém hơn so với các thuật toán khác.

Ở trường hợp 60 request, kCTPA có hiệu năng tương đương với FCFS và khả năng cân bằng tải tốt. MaxMin và Round Robin có hiệu năng tương đối và khả năng cân bằng tải tốt. MinMin có hiệu năng thấp và khả năng cân bằng tải kém hơn. Biểu đồ cột trên thể hiện sự khác biệt về hiệu năng và khả năng cân bằng tải của các thuật toán trong việc xử lý các request trên môi trường đám mây.

Từ request thứ 100 trở đi, thuật toán k-CTPA vượt trội hơn hẳn so với MaxMin, MinMin. Tuy nhiên vẫn chưa thấy ưu thế so với RoundRobin. Nhưng với số lượng request càng lớn thì k-CTPA càng lợi thế hơn hẳn. Và dần dần chiếm ưu thế tuyệt đối so với các thuật toán còn lại. Rõ ràng FCFS thể hiện sự thiếu thông minh và tính tự nhiên của giải thuật.

Bảng 4. 14 So sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 100 request

Thời gian thực hiện (ms)	FCFS	kCTPA	MaxMin	MinMin	Round Robin
AVG	474,86	346,66	351,66	386.87	424,85
MAX	25.252,80	5.475,33	8.010,48	8.522.92	9.462,06
MIN	0,11	0,20	0,13	0.15	0,17



Hình 4. 21 Biểu đồ so sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 100 request

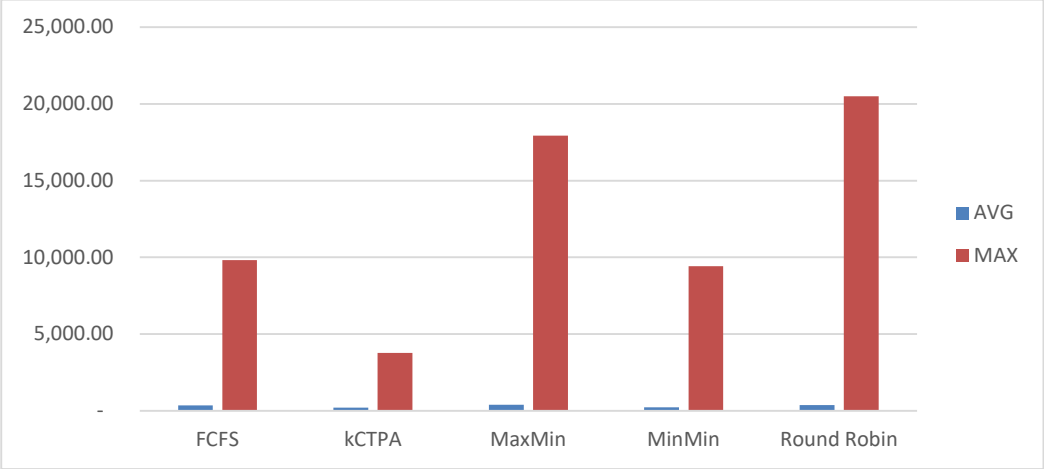
Theo bảng 4.13 và hình 4.21 về thời gian thực hiện (ms) của các thuật toán (FCFS, MaxMin, MinMin, Round Robin) so với thuật toán kCTPA trong trường hợp 100 request, thì kCTPA có thời gian thực hiện trung bình thấp nhất, chỉ là 346,66 ms, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. MaxMin, MinMin và Round Robin cũng có thời gian thực hiện trung bình tương đối thấp, lần lượt là 351,66 ms, 386,87 ms và 424,85 ms. FCFS có thời gian thực hiện trung bình cao nhất, là 474,86 ms, cho thấy hiệu năng thấp nhất trong số các thuật toán được đánh giá.

Về khả năng cân bằng tải, kCTPA có thời gian thực hiện tối đa thấp nhất, chỉ là 5.475,33 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. FCFS, MaxMin và Round Robin cũng có thời gian thực hiện tối đa không quá cao, tuy nhiên vẫn cao hơn so với kCTPA, cho thấy khả năng cân bằng tải tốt hơn so với MinMin. MinMin có thời gian thực hiện tối đa cao nhất, là 8.522,92 ms, cho thấy khả năng cân bằng tải kém hơn so với các thuật toán khác. Như vậy ở trường hợp 100 request, kCTPA có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. MaxMin, MinMin và Round Robin cũng cho thấy hiệu năng tốt và

khả năng cân bằng tải tốt. FCFS có hiệu năng thấp nhất và khả năng cân bằng tải kém hơn. Biểu đồ cột trên thể hiện sự khác biệt về hiệu năng và khả năng cân bằng tải của các thuật toán trong việc xử lý các request trên môi trường đám mây.

Bảng 4. 15 So sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 1000 request

Thời gian thực hiện (ms)	FCFS	kCTPA	MaxMin	MinMin	Round Robin
AVG	360,43	192,82	391,75	230,98	378,06
MAX	9.829,39	3.755,07	17.947,42	9.433,95	20.515,09
MIN	0,12	0,11	0,15	0,10	0,12

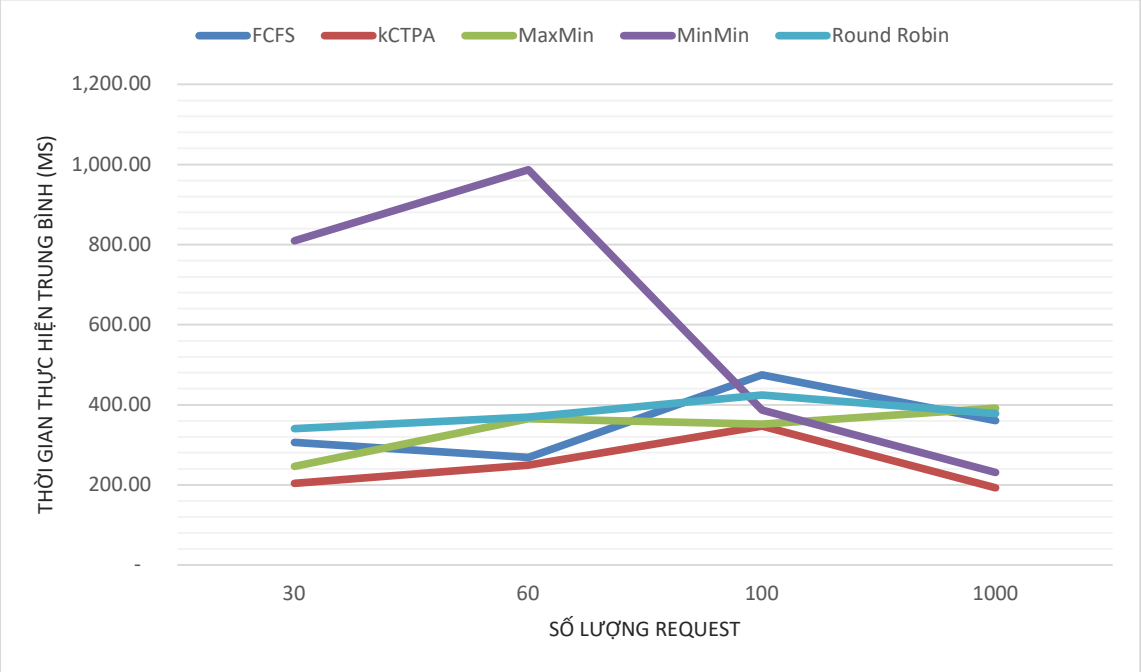


Hình 4. 22 Biểu đồ so sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 1000 request

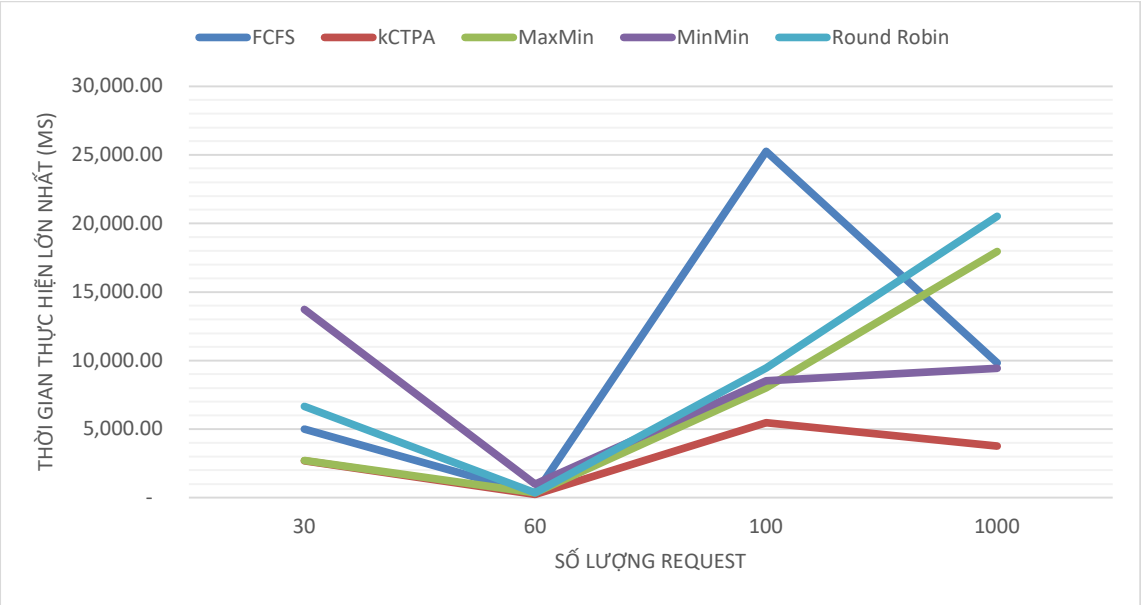
Theo bảng 4.13 và hình 4.21 về thời gian thực hiện (ms) của các thuật toán (FCFS, MaxMin, MinMin, Round Robin) so với thuật toán kCTPA trong trường hợp 1000 request, thì kCTPA có thời gian thực hiện trung bình thấp nhất, chỉ là 192,82 ms, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. FCFS, MaxMin và Round Robin có thời gian thực hiện trung bình cao hơn, lần lượt là 360,43 ms, 391,75 ms và 378,06 ms. MinMin có thời gian thực hiện trung bình thấp hơn so với các thuật toán khác, là 230,98 ms, nhưng vẫn thấp hơn kCTPA. Về khả năng cân bằng tải, kCTPA có thời gian thực hiện tối đa thấp nhất, chỉ là 3.755,07 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. FCFS, MaxMin và Round Robin cũng có thời gian thực hiện tối đa không quá cao, tuy nhiên vẫn cao

hơn so với kCTPA, cho thấy khả năng cân bằng tải tốt hơn so với MinMin. MinMin có thời gian thực hiện tối đa cao nhất, là 9.433,95 ms, cho thấy khả năng cân bằng tải kém hơn so với các thuật toán khác.

Như vậy trong trường hợp 1000 request, kCTPA có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. FCFS, MaxMin và Round Robin cũng cho thấy hiệu năng tốt và khả năng cân bằng tải tốt. MinMin có hiệu năng tốt nhưng khả năng cân bằng tải kém hơn. Biểu đồ cột trên thể hiện sự khác biệt về hiệu năng và khả năng cân bằng tải của các thuật toán trong việc xử lý các request trên môi trường đám mây. Ở trường hợp 1000 Request ta thấy k-CTPA vượt trội hơn hẳn so với các thuật toán khác, bỏ xa các thuật toán khác.



Hình 4. 23 Biểu đồ so sánh thời gian thực hiện trung bình của các thuật toán với kCTPA trong các trường hợp từ 30-1000 Request



Hình 4. 24 Biểu đồ so sánh thời gian thực hiện lớn nhất của các thuật toán với kCTPA trong các trường hợp từ 30-1000 Request

Thông qua 04 trường hợp là 30, 60, 100 và 1000 so sánh thời gian xử lý của các thuật toán với điều kiện như nhau ta có thể thấy sự phân bổ khá ổn định và hợp lý của thuật toán đề xuất k-CTPA, thời gian xử lý của các máy ảo không quá khác biệt so với thời gian xử lý của các thuật toán khác trên cloud (ở trường hợp ít và nhiều request). Hình 4.23 và hình 4.24 cho thấy k-CTPA luôn thấp nhất, kể cả giá trị trung bình lẫn giá trị max.

Thuật toán đề xuất đã cho thấy hiệu quả khi máy ảo có số lượng cao lên thì k-CTPA đảm bảo thời gian phản hồi và thời gian xử lý tốt, giảm chi phí của các trung tâm dữ liệu đám mây. Tuy nhiên thuật toán vẫn còn 1 số nhược điểm như:

- Nếu số lượng máy ảo nhiều thì việc tìm ra máy có Usage nhỏ nhất là khó khăn hơn, nên có thể tìm một máy có Usage phù hợp là đạt.
- Chưa sắp xếp các máy ảo theo danh sách tăng dần.
- Thuật toán dự báo càng chính xác thì phân bổ càng hiệu quả.
- Tính toán và xử lý các ngưỡng động, tức là các ngưỡng này thay đổi theo VM, theo request hoặc theo đặc điểm cloud.

Đánh giá mô hình Regression trong thuật toán k-CTPA

Để đánh giá độ chính xác của Mô hình hồi quy tuyến tính sử dụng trong k-CTPA, luận án sử dụng số sai số RAE để đánh giá mô hình chạy như thế nào và đưa ra giá

trị dự đoán chính xác cho bộ cân bằng tải. Bảng 4.16 cho thấy RAE tồi tệ nhất xảy ra trong dự đoán sử dụng RAM trong trường hợp 1 và tốt nhất là trong trường hợp 3 nhưng đó là dự đoán Mức tiêu thụ nguồn. Chúng ta có thể thấy rằng, RAE có thể chấp nhận được cho thử nghiệm này nhưng nó không tốt ở mọi trường hợp do biến thiên của các request.

Bảng 4. 16 Sai số RAE trong 4 trường hợp mô phỏng giả lập thuật toán kCTPA

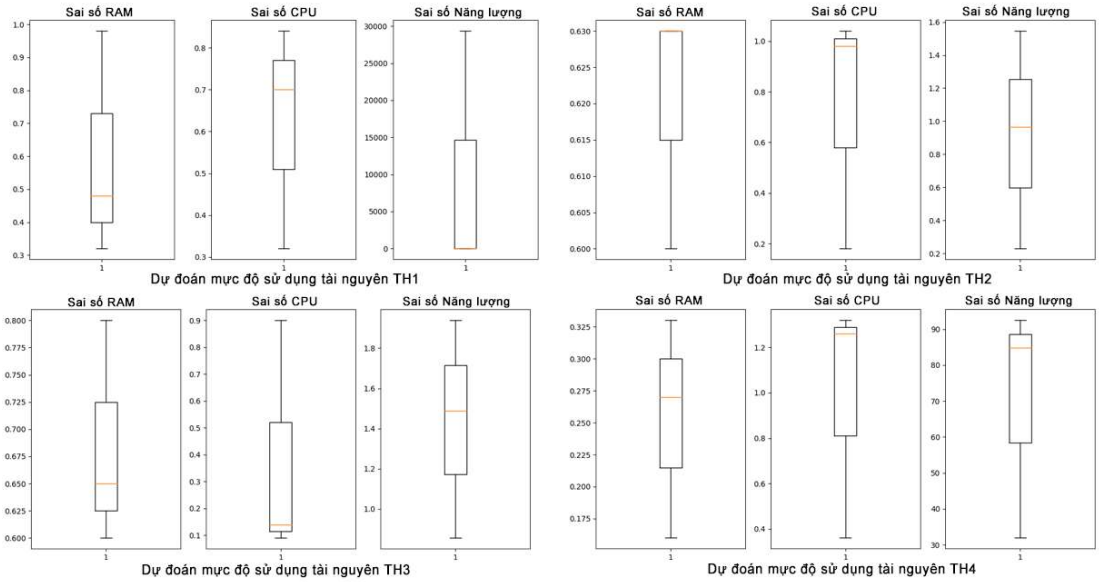
	RAE			
	Trường hợp 1 (30 requests)	Trường hợp 2 (60 requests)	Trường hợp 3 (100 requests)	Trường hợp 4 (1000 requests)
Power Consume	0,085039	0,095735	0,075476	0,084136
CPU Usage	0,295400	0,284387	0,236654	0,248776
RAM Usage	0,326888	0,292676	0,295666	0,300793

Dựa trên bảng 4.16 về sai số RAE của Regression sử dụng trong thuật toán kCTPA trong 4 trường hợp mô phỏng giả lập, chúng ta có thể nhận xét như sau:

- Mức sử dụng năng lượng (Power Consume): Sai số RAE trong các trường hợp từ 0,075476 đến 0,095735, cho thấy sai số khá nhỏ và chấp nhận được. Điều này cho thấy Regression có khả năng dự đoán khá chính xác mức sử dụng năng lượng trong thuật toán kCTPA.
- Mức độ sử dụng CPU: Sai số RAE trong các trường hợp từ 0,236654 đến 0,295400, cũng cho thấy sai số khá nhỏ và khả năng dự đoán chính xác của Regression trong việc dự đoán mức độ sử dụng CPU trong kCTPA.
- Mức độ sử dụng BỘ NHỚ (RAM): Sai số RAE trong các trường hợp từ 0,292676 đến 0,326888, cho thấy sai số cũng khá nhỏ và Regression có khả năng dự đoán chính xác mức độ sử dụng BỘ NHỚ (RAM) trong thuật toán kCTPA.

Dựa trên sai số RAE, Regression trong kCTPA cho thấy khả năng dự đoán chính xác trong việc ước lượng mức sử dụng năng lượng, mức độ sử dụng CPU và mức độ sử dụng BỘ NHỚ (RAM). Tuy nhiên, cần lưu ý rằng các sai số vẫn tồn tại, cho thấy Regression không hoàn toàn chính xác. Do đó, cần tiếp tục nghiên cứu và cải thiện mô hình Regression để đạt được độ chính xác cao hơn trong dự đoán các thông số liên quan đến kCTPA. Như vậy, Regression trong kCTPA cho thấy khả năng

dự đoán chính xác và sai số RAE ở mức chấp nhận được trong việc ước lượng các thông số liên quan đến năng lượng, CPU và BỘ NHỚ (RAM). Tuy nhiên, cần tiếp tục nghiên cứu để nâng cao độ chính xác của mô hình.



Hình 4. 25 Biểu đồ boxplot sai số dự đoán các tài nguyên sử dụng trong 4 trường hợp mô phỏng giả lập thuật toán k-CTPA

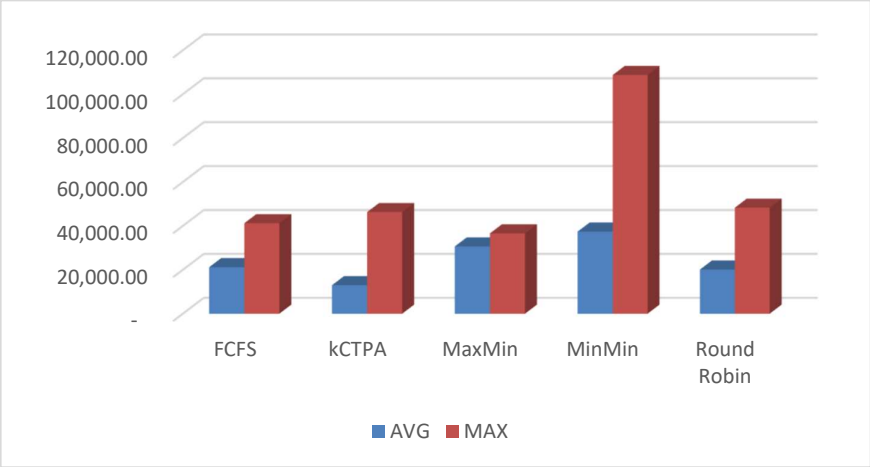
Bên cạnh sai số REA, chúng ta có thể sử dụng lỗi dự đoán của Hồi quy tuyến tính để hiểu sâu hơn về độ chính xác của các tài nguyên được dự đoán. Hình 4.25 cho thấy các hộp lỗi của 4 trường hợp. Trong trường hợp 1 và trường hợp 2, sai số vẫn còn cao một chút, nhưng trong trường hợp 3 và trường hợp 4, sai số có vẻ ổn định và giảm xuống. Còn lại, cho thấy các sai số có thể chấp nhận được trong việc dự đoán các tài nguyên này của đám mây.

Với cấu hình tương tự, ta mô phỏng giả lập ***với 10 máy ảo***, và tăng số request lên lần lượt là ***1800, 2700, 3600 và 4500***, ta thu được các kết quả khá khả quan. Bên cạnh đó, với số lượng request lớn, luận án bổ sung tham số speedups để tính toán và đánh giá.

Trường hợp 1800 request như sau:

Bảng 4. 17 So sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 1800 request

Thời gian thực hiện (ms)	FCFS	kCTPA	MaxMin	MinMin	Round Robin
AVG	21.199,96	12.984,27	30.661,48	37.416,05	20.111,42
MAX	41.215,69	46.300,03	36.598,23	108.743,24	48.302,83



Hình 4. 26 Biểu đồ so sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 1800 request

Theo bảng 4.17 và hình 4.26 về thời gian thực hiện (ms) của các thuật toán (FCFS, MaxMin, MinMin, Round Robin) so với thuật toán kCTPA trong trường hợp 1800 request, ta thấy kCTPA có thời gian thực hiện trung bình thấp nhất, chỉ là 12.984,27 ms, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. FCFS và Round Robin cũng có thời gian thực hiện trung bình khá thấp, lần lượt là 21.199,96 ms và 20.111,42 ms. MaxMin và MinMin có thời gian thực hiện trung bình cao hơn, lần lượt là 30.661,48 ms và 37.416,05 ms.

Về khả năng cân bằng tải, kCTPA có thời gian thực hiện tối đa thấp nhất, chỉ là 46.300,03 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. FCFS và Round Robin cũng có thời gian thực hiện tối đa không quá cao, tuy nhiên vẫn cao hơn so với kCTPA, cho thấy khả năng cân bằng tải tốt hơn so với MaxMin và MinMin. MaxMin có thời gian thực hiện tối đa cao nhất, là 108.743,24 ms, cho thấy khả năng cân bằng tải kém hơn so với các thuật toán khác.

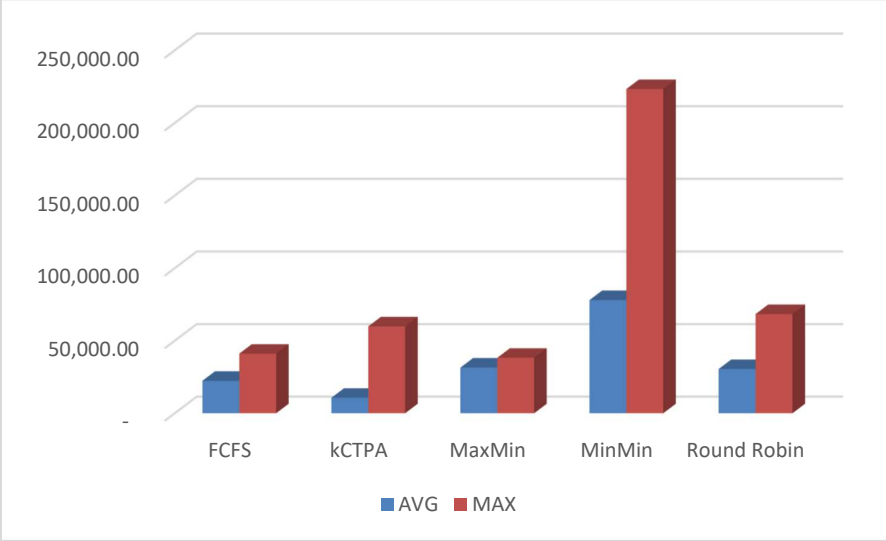
Ở trường hợp 1800 request, kCTPA có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. FCFS và Round Robin cũng cho thấy hiệu năng tốt và khả năng cân bằng tải tốt. MaxMin và MinMin có hiệu năng thấp hơn và khả năng cân bằng tải kém hơn. Biểu đồ cột trên thể hiện sự khác biệt về hiệu

năng và khả năng cân bằng tải của các thuật toán trong việc xử lý các request trên môi trường đám mây.

Trường hợp 2700 request như sau:

Bảng 4. 18 So sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 2700 request

Thời gian thực hiện (ms)	FCFS	kCTPA	MaxMin	MinMin	Round Robin
AVG	22,371.68	10,688.25	31,499.81	78,016.46	30,493.80
MAX	41,100.68	59,889.57	38,261.72	223,336.56	68,378.88



Hình 4. 27 Biểu đồ so sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 2700 request

Theo bảng 4.18 và hình 4.27 về thời gian thực hiện (ms) của các thuật toán (FCFS, MaxMin, MinMin, Round Robin) so với thuật toán kCTPA trong trường hợp 2700 request, thì kCTPA có thời gian thực hiện trung bình thấp nhất, chỉ là 10,688.25 ms, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. FCFS, MaxMin và Round Robin có thời gian thực hiện trung bình khá tương đồng, lần lượt là 22,371.68 ms, 31,499.81 ms và 30,493.80 ms. MinMin có thời gian thực hiện trung bình cao nhất, lên đến 78,016.46 ms.

Về khả năng cân bằng tải, kCTPA có thời gian thực hiện tối đa thấp nhất, chỉ là 59,889.57 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc.

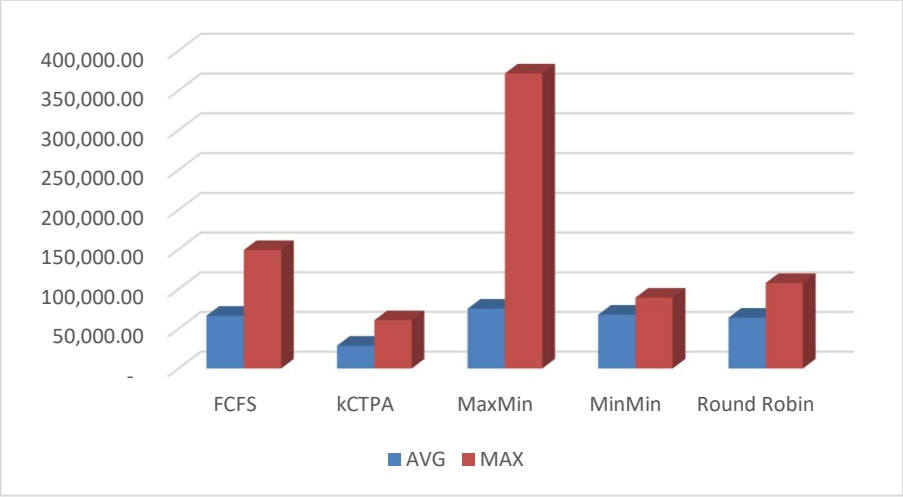
FCFS, MaxMin và Round Robin cũng có thời gian thực hiện tối đa không quá cao, tuy nhiên vẫn cao hơn so với kCTPA, cho thấy khả năng cân bằng tải tốt hơn so với MinMin. MinMin có thời gian thực hiện tối đa cao nhất, lên đến 223,336.56 ms, cho thấy khả năng cân bằng tải kém hơn so với các thuật toán khác.

Như vậy, trong trường hợp 2700 request, kCTPA có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. FCFS, MaxMin và Round Robin cũng cho thấy hiệu năng tốt và khả năng cân bằng tải tốt. MinMin có hiệu năng thấp hơn và khả năng cân bằng tải kém hơn. Biểu đồ cột trên thể hiện sự khác biệt về hiệu năng và khả năng cân bằng tải của các thuật toán trong việc xử lý các request trên môi trường đám mây.

Trường hợp 3600 request như sau:

Bảng 4. 19 So sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 3600 request

Thời gian thực hiện (ms)	FCFS	kCTPA	MaxMin	MinMin	Round Robin
AVG	66,533.26	28,782.21	75,358.77	67,741.85	63,940.76
MAX	149,229.80	60,790.54	371,356.46	89,354.45	107,700.03



Hình 4. 28 Biểu đồ so sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 3600 request

Theo bảng 4.19 và hình 4.28 về thời gian thực hiện (ms) của các thuật toán (FCFS, MaxMin, MinMin, Round Robin) so với thuật toán kCTPA trong trường hợp

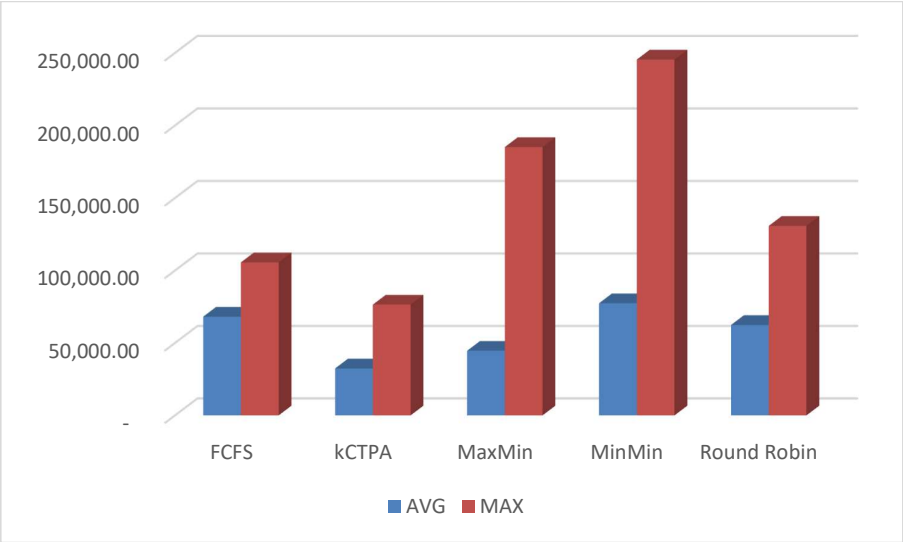
3600 request, kCTPA có thời gian thực hiện trung bình thấp nhất, chỉ là 28,782.21 ms, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. FCFS, MaxMin và Round Robin có thời gian thực hiện trung bình tương đối gần nhau, lần lượt là 66,533.26 ms, 75,358.77 ms và 63,940.76 ms. MinMin có thời gian thực hiện trung bình cao nhất, lên đến 67,741.85 ms. Về khả năng cân bằng tải, kCTPA có thời gian thực hiện tối đa thấp nhất, chỉ là 60,790.54 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. FCFS, MaxMin và Round Robin cũng có thời gian thực hiện tối đa không quá cao, tuy nhiên vẫn cao hơn so với kCTPA, cho thấy khả năng cân bằng tải tốt hơn so với MinMin. MinMin có thời gian thực hiện tối đa cao nhất, lên đến 89,354.45 ms, cho thấy khả năng cân bằng tải kém hơn so với các thuật toán khác.

Với trường hợp 3600 request, kCTPA có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. FCFS, MaxMin và Round Robin cũng cho thấy hiệu năng tốt và khả năng cân bằng tải tốt. MinMin có hiệu năng thấp hơn và khả năng cân bằng tải kém hơn. Biểu đồ cột trên thể hiện sự khác biệt về hiệu năng và khả năng cân bằng tải của các thuật toán trong việc xử lý các request trên môi trường đám mây.

Trường hợp 4500 request như sau:

Bảng 4. 20 So sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 4500 request

Thời gian thực hiện (ms)	FCFS	kCTPA	MaxMin	MinMin	Round Robin
AVG	68,166.80	32,471.17	44,629.60	77,437.24	62,437.37
MAX	105,650.00	76,499.60	185,148.60	245,442.89	130,797.97



Hình 4. 29 Biểu đồ so sánh thời gian thực hiện của các thuật toán với kCTPA ở trường hợp 4500 request

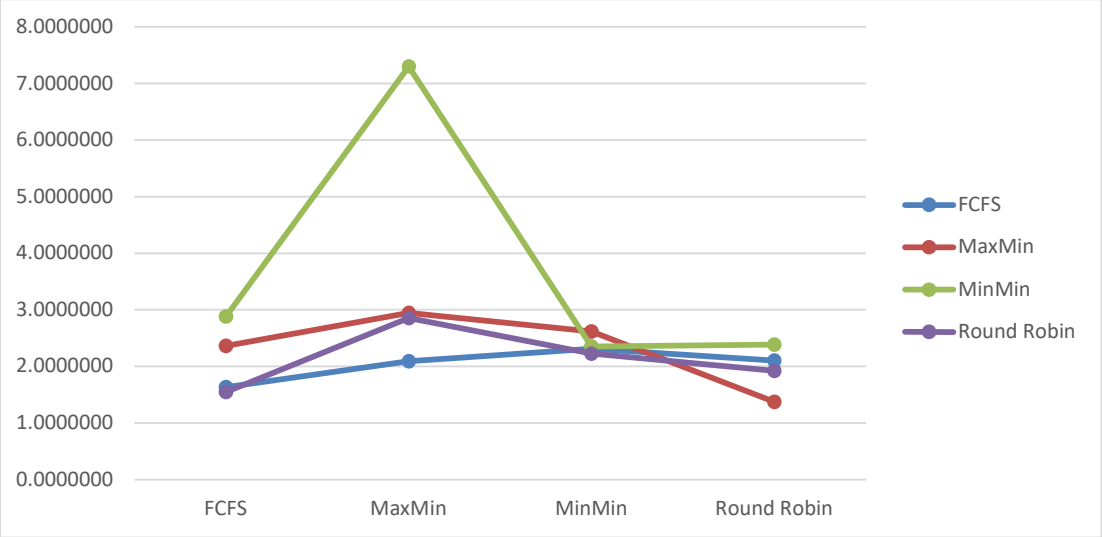
Theo bảng 4.20 và hình 4.29 về thời gian thực hiện (ms) của các thuật toán (FCFS, MaxMin, MinMin, Round Robin) so với thuật toán kCTPA trong trường hợp 4500 request, kCTPA có thời gian thực hiện trung bình thấp nhất, chỉ là 32,471.17 ms, cho thấy hiệu năng tốt hơn so với các thuật toán khác trong việc xử lý các request. FCFS, MaxMin và Round Robin có thời gian thực hiện trung bình tương đối gần nhau, lần lượt là 68,166.80 ms, 44,629.60 ms và 62,437.37 ms. MinMin có thời gian thực hiện trung bình cao nhất, lên đến 77,437.24 ms.

Về khả năng cân bằng tải, kCTPA có thời gian thực hiện tối đa thấp nhất, chỉ là 76,499.60 ms, cho thấy khả năng cân bằng tải tốt trong việc phân phối công việc. FCFS, MaxMin và Round Robin cũng có thời gian thực hiện tối đa không quá cao, tuy nhiên vẫn cao hơn so với kCTPA, cho thấy khả năng cân bằng tải tốt hơn so với MinMin. MinMin có thời gian thực hiện tối đa cao nhất, lên đến 245,442.89 ms, cho thấy khả năng cân bằng tải kém hơn so với các thuật toán khác.

Tóm lại, trong trường hợp 4500 request, kCTPA có hiệu năng tốt nhất và khả năng cân bằng tải cao hơn so với các thuật toán khác. FCFS, MaxMin và Round Robin cũng cho thấy hiệu năng tốt và khả năng cân bằng tải tốt. MinMin có hiệu năng thấp hơn và khả năng cân bằng tải kém hơn. Biểu đồ cột trên thể hiện sự khác biệt về hiệu năng và khả năng cân bằng tải của các thuật toán trong việc xử lý các request trên môi trường đám mây.

Bảng 4. 21 So sánh Speedups các thuật toán với thuật toán kCTPA ở 4 trường hợp từ 1800 đến 4500 request

Speedups	FCFS	MaxMin	MinMin	Round Robin
Trường hợp 1800 Request	1,6327418	2,3614328	2,8816448	1,5489068
Trường hợp 2700 Request	2,0931096	2,9471444	7,2992743	2,8530213
Trường hợp 3600 Request	2,3116106	2,6182413	2,3536015	2,2215375
Trường hợp 4500 Request	2,0993024	1,3744379	2,3848002	1,9228558



Hình 4. 30 Biểu đồ so sánh Speedups các thuật toán với thuật toán kCTPA ở 4 trường hợp từ 1800 đến 4500 request

Theo bảng 4.21 và hình 4.30 về số liệu so sánh khả năng tăng tốc (speedup) của thuật toán kCTPA so với các thuật toán phổ biến (FCFS, MaxMin, MinMin, Round Robin) ở 4 trường hợp, ta có thể nhận xét và đánh giá hiệu năng cân bằng tải của kCTPA như sau:

- *Trường hợp 1800 Request:* kCTPA có khả năng tăng tốc (speedup) so với FCFS là 1,6327418; MaxMin là 2,3614328; MinMin là 2,8816448 và Round Robin là 1,5489068. kCTPA đạt hiệu năng cân bằng tải tốt hơn so với FCFS, MaxMin và Round Robin, và tương đương với MinMin.
- *Trường hợp 2700 Request:* kCTPA có khả năng tăng tốc (speedup) so với FCFS là 2,0931096; MaxMin là 2,9471444; MinMin là 7,2992743 và Round Robin là 2,8530213. kCTPA đạt hiệu năng cân bằng tải tốt hơn rõ rệt so với

FCFS và Round Robin, và có hiệu năng cân bằng tải cao nhất trong số các thuật toán.

- *Trường hợp 3600 Request:* kCTPA có khả năng tăng tốc (speedup) so với FCFS là 2,3116106; MaxMin là 2,6182413; MinMin là 2,3536015 và Round Robin là 2,2215375. kCTPA đạt hiệu năng cân bằng tải tương đương với MaxMin và MinMin, và cao hơn so với FCFS và Round Robin.
- *Trường hợp 4500 Request:* kCTPA có khả năng tăng tốc (speedup) so với FCFS là 2,0993024; MaxMin là 1,3744379; MinMin là 2,3848002 và Round Robin là 1,9228558. kCTPA đạt hiệu năng cân bằng tải tương đương với MinMin và cao hơn so với FCFS và Round Robin, nhưng thấp hơn so với MaxMin.

Nhìn chung, kCTPA thể hiện khả năng tăng tốc và hiệu năng cân bằng tải tốt hơn so với FCFS và Round Robin trong các trường hợp mô phỏng giả lập. Nó cũng có hiệu năng cân bằng tải cao nhất trong trường hợp 2700 Request và cân bằng tải tương đương với các thuật toán khác trong các trường hợp khác. Điều này cho thấy kCTPA có tiềm năng để cải thiện hiệu suất và cân bằng tải trong môi trường đám mây.

4.4 KẾT LUẬN CHƯƠNG

Với ý tưởng tiếp cận từ bên ngoài bộ cân bằng tải từ phân tích SWOT, chương này trình bày cách tiếp cận và quan sát các yếu tố bên ngoài bộ cân bằng tải trên môi trường đám mây, từ đó trình bày lý do tại sao sao tác giả lại chọn việc dự báo Deadlock để phục vụ thuật toán cân bằng tải cũng như lý do tại sao chọn phân lớp độ ưu tiên các tác vụ để cân bằng tải. Với mục tiêu nâng cao hiệu năng cân bằng tải, luận án kết hợp các thuật toán học máy và phân tích dữ liệu để xây dựng hai thuật toán cân bằng tải là PDOA và k-CTPA. Các thuật toán được cài đặt và mô phỏng giả lập, và kết quả cho thấy sự nổi trội của trí tuệ nhân tạo, đặc biệt là học máy và phân tích dữ liệu, trong việc cân bằng tải trên môi trường đám mây.

Với việc cài đặt và mô phỏng giả lập các thuật toán đề xuất, các kết quả cho thấy sự nổi trội của AI mà đại diện là ML và phân tích dữ liệu, tiềm năng của AI trong bộ cân bằng tải trên điện toán đám mây Tuy nhiên, khi áp dụng các thuật toán này cho dữ liệu lớn, sự mạnh mẽ của thuật toán cân bằng tải và khả năng xử lý thời gian thực là những thách thức đáng kể. Đồng thời, việc tăng độ chính xác của các thuật

toán phân lớp, gom cụm và dự báo cũng là những vấn đề quan trọng cần được giải quyết.

Nhìn chung, việc kết hợp trí tuệ nhân tạo, học máy và phân tích dữ liệu mang lại tiềm năng đáng kể trong việc cân bằng tải trên môi trường đám mây. Tuy nhiên, cần tiếp tục nghiên cứu và phát triển để vượt qua các thách thức hiện tại và đạt được hiệu năng cân bằng tải tốt nhất trong các tình huống thực tế.

PHẦN KẾT LUẬN

1. Các kết quả đã đạt được

Việc nghiên cứu nâng cao hiệu năng với ý tưởng "TIẾP CẬN SWOT CHO CÂN BẰNG TẢI TRÊN ĐIỆN TOÁN ĐÁM MÂY" đã mở ra một cách nhìn mới và toàn diện về cân bằng tải trên điện toán đám mây, mở ra các hướng nghiên cứu sâu rộng dưới các gọc độ khác nhau đang được nhiều nhà nghiên cứu đặc biệt quan tâm trong thời gian gần đây. Mục tiêu của các công trình nghiên cứu nhằm tối ưu và nâng cao hiệu năng cân bằng tải trên cloud với nền tảng công nghệ hiện đại, đặc biệt là cá kỹ thuật học máy. Trong bối cảnh trí tuệ nhân tạo (AI) phát triển không ngừng, tác giả của luận án này đã chú trọng vào việc khám phá và phát triển các thuật toán cân bằng tải hiệu quả, tận dụng những lợi thế của AI, đặc biệt là học máy (ML) và phân tích dữ liệu (big data analytics) trong môi trường điện toán đám mây. Qua quá trình học tập và nghiên cứu để hoàn thành luận án, tác giả đã thu được một số kết quả đáng chú ý dưới đây:

1.1 Từ ý tưởng phân tích SWOT đưa ra hướng tiếp cận phân tích cân bằng tải, nhằm nâng cao hiệu năng cân bằng tải trên môi trường điện toán đám mây

- Nghiên cứu tổng quan CBT trên đám mây. Các kỹ thuật cân bằng tải được dùng trong môi trường điện toán đám mây, các đặc tính của cân bằng tải trên môi trường đám mây, từ đó đưa ra phân tích SWOT cho CBT trên môi trường đám mây.
- Thông qua phân tích SWOT, đưa ra 02 hướng tiếp cận chính là tiếp cận từ bên trong và tiếp cận từ bên ngoài để nâng cao hiệu năng cân bằng tải. Phân tích từ bên trong bao gồm các yếu tố nội tại của CBT, cải tiến các vấn đề hiện có của CBT v/v. Phân tích các yếu tố bên ngoài như mối nguy cơ, đe dọa hoặc thách thức mà nhà phát triển CBT không thể kiểm soát được, cụ thể là môi trường mạng và người dùng cloud.

1.2 Đề xuất xây dựng nhóm thuật toán cân bằng tải theo hướng tiếp cận từ bên trong, bao gồm đề xuất xây dựng và phát triển bộ cân bằng tải kết hợp các tham số của nó với các thuật toán ML và cải tiến thuật toán CBT hiện có.

- Cải tiến thuật toán Throttle thông qua thuật toán ITA, cho thấy tính ưu việt của thuật toán cải tiến.
- Kết hợp dự báo chuỗi thời gian ARIMA thông qua thuật toán APRTA
- Kết hợp các thuật toán phân lớp như SVM, phân cụm k-Means để xây dựng thuật toán MCCVA trong cân bằng tải.
- Kết hợp các thuật toán phân lớp Naïve Bayes, phân cụm k-Means để xây dựng thuật toán RCBA trong cân bằng tải.

1.3 Đề xuất xây dựng nhóm thuật toán cân bằng tải theo hướng tiếp cận từ bên ngoài, bao gồm các thuật toán liên quan đến Deadlock và hành vi người dùng cloud, đại diện là độ ưu tiên tác vụ (Task priority).

- Đề xuất hướng tiếp cận mới thông qua việc kiểm soát deadlock và không cho tài nguyên cloud ở trạng thái treo. Thuật toán PDOA thông qua regression để dự báo các thông số tài nguyên (RAM, CPU, STORAGE) từ đó dự báo khả năng xảy ra treo và deadlock. Thuật toán PDOA có thể dùng để đưa vào áp dụng trên thực tế.
- Với hướng tiếp cận lấy người dùng làm trung tâm, nghiên cứu hành vi người dùng trong cân bằng tải là một hướng tiếp cận mới và đáng chú ý. Với sự phát triển của AI & ML nên đây là hướng đi tiềm năng mà luận án đề xuất. Xây dựng được thuật toán k-CTPA thông qua việc phân lớp độ ưu tiên của tác vụ để cân bằng tải.

2. Hướng phát triển của đề tài luận án

Vấn đề kiến nghị và hướng đi tiếp theo của nghiên cứu:

- Tiếp tục cải tiến và phát triển các thuật toán theo 2 hướng tiếp cận, tăng độ chính xác và hiệu năng lên cao hơn nữa trong môi trường cloud biến thiên lớn hơn.
- Kết hợp phát triển các thuật toán hoặc tổ hợp thuật toán theo cả 02 hướng tiếp cận.

- Tiếp tục nghiên cứu và phát triển theo hướng cơ hội và thách thức, nghiên cứu các yếu tố bên ngoài nhưng tác động mạnh mẽ đến cân bằng tải và hiệu năng hoạt động của nó.

CÁC CÔNG TRÌNH NGHIÊN CỨU CỦA TÁC GIẢ

TẠP CHÍ KHOA HỌC

[CT1] H. Le Ngoc, T. N. Thi Huyen, X. Phi Nguyen, and C. Hung Tran, "MCCVA: A New Approach Using SVM and Kmeans for Load Balancing on Cloud", *International Journal on Cloud Computing: Services and Architecture (IJCCSA)* Vol. 10, No.3, June 2020 DOI: 10.5121/ijccsa.2020.10301, 1-14.

➔Tạp chí khoa học quốc tế có phản biện độc lập, được chỉ mục bởi ProQuest

➔Đã cập nhật trong Google Scholar và Research Gate

➔ NCS là tác giả đầu và là tác giả liên hệ

[CT2] N. Xuan Phi, **L. Ngoc Hieu**, and T. Cong Hung, "Load Balancing Algorithm on Cloud Computing for Optimize Respone Time", *International Journal on Cloud Computing: Services and Architecture (IJCCSA)* Vol. 10, No.3, June 2020 DOI: 10.5121/ijccsa.2020.10302 15.

➔Tạp chí khoa học quốc tế có phản biện độc lập, được chỉ mục bởi ProQuest

➔Đã cập nhật trong Google Scholar và Research Gate

➔ NCS là tác giả đầu

[CT3] Hieu Le Ngoc and Hung Tran Cong, "ITA: The Improved Throttled Algorithm of load balancing on cloud computing," *International journal of computer network and communication.,* vol. 14, no. 1, pp. 25–39, 2022.

➔ Tạp chí khoa học quốc tế có phản biện độc lập, được đánh chỉ mục bởi SCOPUS (Elsevier)

➔ H-Index 2022: 8.0, Q4 – Scopus

➔ NCS là tác giả đầu và là tác giả liên hệ

[CT4] Hieu Le Ngoc và Hung Tran Cong, "PDOA: dự báo Deadlock để nâng cao cân bằng tải trên điện toán đám mây", *tạp chí Khoa học Công nghệ thông tin và Truyền thông*, PTIT

➔ Tạp chí khoa học của PTIT, được hội đồng GSNN công nhận 0.5 điểm

➔ NCS là tác giả đầu và là tác giả liên hệ

[CT5] Hieu Le Ngoc and Hung Tran Cong, "Enhancing Load Balancing in Cloud Computing through Adaptive Task Prioritization", *Journal of Computer Science and Technology Studies (JCSTS),* vol. 5, no. 2, 2023, DOI: https://doi.org/10.32996/jcsts.2023.5.2.1.

➔ Tạp chí khoa học quốc tế có phản biện độc lập, được đánh chỉ mục bởi Crossref, WorldCat (OCLC), Google Scholar

➔ Impact Factor 2022: 5.711

➔ H-Index 2022: 29, Q2 - Scopus

➔ NCS là tác giả đầu và là tác giả liên hệ

HỘI NGHỊ KHOA HỌC

[CT6] Hieu Le Ngoc and Hung Tran Cong, "Enhancing Load Balancing in Cloud Computing through Deadlock Prediction", *EAI INISCOM 2023 - 9th EAI International Conference on Industrial Networks and Intelligent Systems*, https://doi.org/10.1007/978-3-031-47359-3_19

➔ Hội thảo khoa học quốc tế có phản biện độc lập, được đánh chỉ mục bởi Web of Science, Compendex, Scopus, BLP, EU Digital Library, Google Scholar, IO-Port, MathSciNet, Inspec, and Zentralblatt MATH

➔ Impact Factor 2021: 3.007

➔ Được xuất bản trong ấn phẩm của nhà xuất bản Springer với tiêu đề *EAI/Springer Innovations in Communication and Computing*

➔ NCS là tác giả đầu và là tác giả liên hệ

[CT7] Hung Tran Cong, Duy Tien Tran and **Hieu Le Ngoc**, "A proposed load balancer using naïve Bayes to enhance response time on cloud computing" in *2022 24th International Conference on Advanced Communication Technology (ICACT)*, 2022

➔ Bài báo hội nghị khoa học quốc tế có phản biện độc lập, được chỉ mục bởi IEEE Xplore, SCOPUS, INSPEC, Engineering Index (EI), Conference Proceedings Citation Index (CPCI)

➔ H-Index 2022: 34

➔ NCS là tác giả liên hệ

TÀI LIỆU THAM KHẢO

[1] "12 benefits of cloud computing," [Online]. Available: https://www.salesforce.com/ap/products/platform/best-practices/benefits-of-cloud-computing/. [Accessed 01 07 2022].

[2] "Advantages of cloud computing," Google Cloud, [Online]. Available: https://cloud.google.com/learn/advantages-of-cloud-computing. [Accessed 1 7 2022].

[3] "Top 10 benefits of cloud computing," Oracle.com, [Online]. Available: https://www.oracle.com/cloud/what-is-cloud-computing/top-10-benefits-cloud-computing/. [Accessed 1 7 2022].

[4] "Benefits of cloud computing," Ibm.com, [Online]. Available: https://www.ibm.com/cloud/learn/benefits-of-cloud-computing. [Accessed 1 7 2022].

[5] "What Is Cloud Load Balancing?," Nginx.com, [Online]. Available: https://www.nginx.com/resources/glossary/cloud-load-balancing/. [Accessed 1 7 2022].

[6] "Cloud load balancing," Google Cloud, [Online]. Available: https://cloud.google.com/load-balancing. [Accessed 1 7 2022].

[7] "Cloud Load Balancing overview," Google Cloud, [Online]. Available: https://cloud.google.com/load-balancing/docs/load-balancing-overview. [Accessed 1 7 2022].

[8] S. M. S. Suntharam, "Load Balancing By Max-Min Algorithm in Private Cloud Environment," *International Journal of Science and Research (IJSR),* 2013.

[9] T. Kokilavani and D. I. George Amalarethinam, "Load balanced MinMin algorithm for static MetaTask scheduling in grid computing," *International journal of computer applications,* vol. 20, no. 2, pp. 42-48, 2011.

[10] "Round Robin Load Balancing," Avi Networks, 2 7 2019. [Online]. Available: https://avinetworks.com/glossary/round-robin-load-balancing/. [Accessed 1 7 2022].

[11] "What Is Round-Robin Load Balancing?," Nginx.com, [Online]. Available: https://www.nginx.com/resources/glossary/round-robin-load-balancing/. [Accessed 1 7 2022].

[12] M. Z. Branko Radojevic, "Analysis of issues with load balancing algorithms in hosted (cloud) environments.," in *MIPRO, 2011 Proceedings of the 34th International Convention*, Opatija, Croatia, 2011.

[13] Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi and Jameela Al-Jaroodi, "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms," *2012 IEEE Second Symposium on Network Cloud Computing and Applications,* 2012.

[14] Y. Liu and Y.-K. Fang, "Optimizing WLC scheduling algorithm of LVS," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, 2010.

[15] Brototi Mondala, Kousik Dasguptaa, Paramartha Duttab, "Load Balancing in Cloud Computing using Stochastic Hill Climbing-A Soft Computing Approach," *Procedia Technology,* 2012.

[16] 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,* 2012.

[17] Huankai Chen ,Professor Frank Wang, Dr Na Helian , Gbola Akanmu, "User-Priority Guided Min-Min Scheduling Algorithm For Load Balancing in Cloud Computing," *2013 National Conference on Parallel Computing Technologies (PARCOMPTECH),* 2013.

[18] Dhinesh Babu L.D., P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing, Applied Soft Computing 13,* 2013.

[19] Seokho Son, Gihun Jung, Sung Chan Jun, "An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider," *The Journal of Supercomputing,* vol. 64, p. 606–637, 2013.

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

[21] Rajwinder Kaur, Pawan Luthra, "Load Balancing in Cloud Computing," in *Proc. of Int. Conf. on Recent Trends in Information, Telecommunication and Computing, ITC*, 2014.

[22] Geethu Gopinath P P, Shriram K Vasudevan, "An in-depth analysis and study of Load balancing techniques in the cloud computing environment," *2nd International Symposium on Big Data and Cloud Computing (ISBCC'15), Procedia Computer Science,* vol. 50, pp. 427-432, 2015.

[23] Ritu Kapur, "A Workload Balanced Approach for Resource Scheduling in Cloud Computing," in *2015 Eighth International Conference on Contemporary Computing (IC3)*, Noida, India, 2015.

[24] Keng-Mao Cho, Pang-Wei Tsai, Chun-Wei Tsai, Chu-Sing Yang, "A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing," *Neural Computing and Applications,* vol. 26, pp. 1297-1309, 2015.

[25] Mohit Kumara, S.C.Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," in *7th International Conference on Advances in Computing & Communications, ICACC-2017*, Cochin, India, 2017.

[26] Deepali Simaiya, Raj Kumar Paul, "Review of Various Performcae Evaluation Issues and Efficient Load Balancing for Cloud Computing," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2018 IJSRCSEIT,* 2018.

[27] Deepak Puthal, Mohammad S. Obaidat, Priyadarsi Nanda, Mukesh Prasad, Saraju P. Mohanty, and Albert Y. Zomaya, "Secure and Sustainable Load Balancing of, Edge Data Centers in Fog Computing," *ACHIEVING ENERGY EFFICIENCY AND SUSTAINABILITY IN EDGE/FOG DEPLOYMENT,* vol. 56, no. 5, pp. 60-65, 2018.

[28] Mohammad Alkhalaileh, Rodrigo N. Calheiros, Quang Vinh Nguyen and Bahman Javadi, "Dynamic Resource Allocation in Hybrid Mobile Cloud Computing for Data-Intensive Applications," in *14th International Conference, GPC 2019*, Uberlândia, Brazil, 2019.

[29] Kothapuli Venkata Subba Reddy, Jagirdar Srinivas and Ahmed Abdul Moiz Qyser, "A Dynamic Hierarchical Load Balancing Service Architecture for Cloud Data Centre Virtual Machine Migration," in *Proceedings of the Second International Conference on SCI 2018*, 2019.

[30] Shahbaz Afzal, G. Kavitha, "Load balancing in cloud computing – A hierarchical taxonomical classification," *Journal of Cloud Computing: Advances, Systems and Applications,* 2019.

[31] R. Kanakala and V. K. Reddy, "Performance analysis of load balancing techniques in cloud computing environment," *TELKOMNIKA Indonesian Journal of Electrical Engineering,* vol. 13, no. 3, 2015.

[32] D. A. Shafiq, N. Z. Jhanjhi and A. Abdullah, "Load balancing techniques in cloud computing environment: A review," *Journal of King Saud University - Computer and Information Sciences,* vol. 34, no. 7, pp. 3910-3933, 2022.

[33] "What is cloud computing? A beginner's guide," Microsoft.com, [Online]. Available: https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/. [Accessed 19 4 2022].

[34] "What is cloud computing?," Amazon.com, [Online]. Available: https://aws.amazon.com/what-is-cloud-computing/. [Accessed 19 4 2022].

[35] E. Knorr, "What is cloud computing? Everything you need to know now," InfoWorld, 2018. [Online]. Available: https://www.infoworld.com/article/2683784/what-is-cloud-computing.html. [Accessed 19 4 2022].

[36] "What is Cloud Computing? Types and Examples," Salesforce.com, [Online]. Available: https://www.salesforce.com/products/platform/best-practices/cloud-computing/. [Accessed 19 4 2022].

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

[38] "Cloud computing services," Google Cloud, [Online]. Available: https://cloud.google.com/. [Accessed 1 7 2022].

[39] "Cloud computing with AWS," Amazon.com, [Online]. Available: https://aws.amazon.com/what-is-aws/. [Accessed 1 7 2022].

[40] "Achieve more with the Microsoft Cloud," Microsoft.com, [Online]. Available: https://www.microsoft.com/en-us/microsoft-cloud. [Accessed 1 7 2022].

[41] P. M. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Gaithersburg, MD, 2011.

[42] Bui Thanh Khiet, Nguyen Thi Nguyet Que, Ho Dac Hung, Pham Tran Vu, Tran Cong Hung, "A Fair VM Allocation for Cloud Computing based on Game Theory," in *Proceedings of the 10th National Conference on Fundamental and Applied Information Technology Research (FAIR'10)*, Da Nang, Vietnam, 2017.

[43] J. Zhang, Q. Liu and J. Chen, "An Advanced Load Balancing Strategy for Cloud Environment," in *17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, Guangzhou, China, 2019.

[44] J. Zhao, K. Yang, X. Wei, Y. Ding, L. Hu and G. Xu, "A Heuristic Clustering-Based Task Deployment Approach for Load Balancing Using Bayes Theorem in Cloud Environment," *IEEE Transactions on Parallel and Distributed Systems,* vol. 27, no. 2, pp. 305-316, 2016.

[45] GIBET TANI, H. and C. EL AMRANI, "Smarter Round Robin Scheduling Algorithm for Cloud Computing and Big Data," *Journal of Data Mining and Digital Humanities,* 2018.

[46] Matthias Sommer, Michael Klink, Sven Tomforde, Jörg Hähner, "Predictive Load Balancing in Cloud Computing Environments Based on Ensemble Forecasting," in *IEEE International Conference on Autonomic Computing (ICAC2016)*, Wurzburg, Germany, 2016.

[47] G. Shao and J. Chen, "A Load Balancing Strategy Based on Data Correlation in Cloud Computing," in *IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, Shanghai, China, 2016.

[48] Ashok Kumar Upadhaya, C.K. Jha, Shikha Pandey, "Suboptimal Mechanism For Load Balancing In CloudEnvironment," in *International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, Chennai, India, 2017.

[49] K. S. Umadevi and P. Chaturvedi, "Predictive load balancing algorithm for cloud computing," in *International conference on Microelectronic Devices, Circuits and Systems (ICMDCS)*, Vellore, 2017.

[50] J. Bao and W. Huang, "The development status and prospects of cloud computing," in *The 2nd International Conference on Information Science and Engineering*, 2010.

[51] Yang, Wenyan;, "A brief analysis of development situations and trend of cloud computing," in *IOP conference series. Earth and environmental science*, 2017.

[52] "What is load balancing?," NGINX, [Online]. Available: https://www.nginx.com/resources/glossary/load-balancing/. [Accessed 19 4 2020].

[53] A. A. M. S. A. Ahmad AA Alkhatib, "Load Balancing Techniques in Cloud Computing: Extensive Review," *Advances in Science Technology and Engineering Systems Journal,* vol. 6, no. 2, pp. 860-870, 2021.

[54] P. Wang, H. Xu, Z. Niu, D. Han and Y. Xiong, "Expeditus: Congestion-Aware Load Balancing in Clos Data Center Networks," *IEEE/ACM Transactions on Networking,* vol. 25, no. 5, pp. 3175-3188, 2017.

[55] C. Jayashri, P. Abitha, S. Subburaj, S. Y. Devi, Suthir S and Janakiraman S, "Big data transfers through dynamic and load balanced flow on cloud networks," in *Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, Chenmai, 2017.

[56] Imran Ghani & Naghmeh Niknejad & Seung Ryul Jeong, "Energy saving in green cloud computing data centers: a review," *Journal of Theoretical and Applied Information Technology,* pp. 16-30, 2015.

[57] Mishra, Sambit Kumar; Sahoo, Bibhudatta; Parida, Priti Paramita;, "Load balancing in cloud computing: A big picture," *Journal of King Saud University - Computer and Information Sciences,* vol. 32, no. 2, pp. 149-158, 2020.

[58] Thakur, Avnish; Goraya, Major Singh;, "A taxonomic survey on load balancing in cloud," *Journal of network and computer applications,* vol. 98, pp. 43-57, 2017.

[59] Jiawei Han, Micheline Kamber and Jian Pei, Data Mining: Concepts and Techniques, Elsevier Inc, 2012.

[60] Bishop, Christopher M, Pattern recognition and Machine Learning, Springer, 2016.

[61] Shehroz S.Khan, Amir Ahmad, "Cluster center initialization algorithm for K-means clustering," *Pattern Recognition Letters,* vol. 25, no. 11, pp. 1293-1302, 2004.

[62] Ross Ihaka, Time Series Analysis, Statistics Department, University of Auckland,, 2005.

[63] Roy Batchelor, Box-Jenkins Analysis, London: Cass Business School, 2006.

[64] "Machine Laerning Cơ bản," Vũ Hữu Tiệp, [Online]. Available: https://machinelearningcoban.com/2016/12/28/linearregression/. [Accessed 09 04 2021].

[65] E. Gürel, "Swot analysis: A theoretical review," *Journal of International Social Research,* vol. 10, no. 51, pp. 994-1006, 2017.

[66] M. N. H. S. O. J. Ify Evangel OBIM, "A SWOT ANALYSIS OF CLOUD COMPUTING AS AN INNOVATIVE TECHNOLOGY FOR LIBRARY SERVICE DELIVERY," *Journal of applied Information Science and Technology,* vol. 13, no. 1, pp. 212-221, 2020.

[67] K. V. M. R. a. K. A. Sonal Dubey, "SWOT Analysis of Cloud Computing Environment," in *CSI-2015, Golden Jubilee of the Computer Society of India (CSI) 50th Annual Convention CSI@50*, New Delhi, 2015.

[68] "Load Balancing metrics," Oracle.com, 2022. [Online]. Available: https://docs.oracle.com/en-us/iaas/Content/Balance/Reference/loadbalancermetrics.htm. [Accessed 19 4 2022].

[69] "Performance study for load balancer," Ibm.com, [Online]. Available: https://www.ibm.com/support/pages/performance-study-load-balancer. [Accessed 19 4 2022].

[70] "Load balancing metrics," Google Cloud, [Online]. Available: https://cloud.google.com/load-balancing/docs/metrics. [Accessed 19 4 2022].

[71] Moses Ashawa, Oyakhire Douglas, Jude Osamor, Riley Jackie, "Improving cloud efficiency through optimized resource allocation technique for load balancing using LSTM machine learning algorithm," *Improving cloud efficiency through optimized resource allocation technique for load balancing using LSTM machine learning algorithm,* vol. 11, 2022.

[72] Ashish Mishra, Saurabh Sharma, Divya Tiwari, "A Survey on Load Balancing in Cloud Computing," in *Intelligent Computing and Innovation*

on Data Science. Lecture Notes in Networks and Systems, Singapore, Springer, 2020.

[73] Sreelekshmi S, K R Remesh Babu, "Synchronized Multi-Load Balancer with Fault Tolerance in Cloud," *International Journal of Computer Information Systems and Industrial Management Applications.,* vol. 10, pp. 107-114, 2018.

[74] G. Punetha Sarmila, G. Punetha Sarmila, P. Dinadayalan, "Survey on fault tolerant — Load balancing algorithmsin cloud computing," in *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, Coimbatore, India, 2015.

[75] S. Roy, D. M. A. Hossain, S. Kumar Sen, N. Hossain and M. R. Al Asif, "Measuring the performance on load balancing algorithms," *Global Journal of Computer Science and Technology,* pp. 41-49, 2019.

[76] Kudriavtceva, Arina;, "SWOT-analysis of digital technologies for an industrial enterprise," *IOP conference series. Materials science and engineering,* vol. 497, p. 012012, 2019.

[77] "Consumer technology SWOT analysis and tech monitor tool," Infotech.com, [Online]. Available: https://www.infotech.com/research/it-consumer-technology-swot-analysis-and-tech-monitor-tool. [Accessed 1 7 2022].

[78] G. K. K. T. Sugandhi Midha, "Cloud Deep Down – SWOT Analysis," in *2017 2nd International Conference on Telecommunication and Networks (TEL-NET 2017)*, India, 2017.

[79] Atharva Agashe, Shivani Pande, Rupesh C. Jaiswal, "A Survey Paper on Cloud Computing and Migration to the Cloud," *Journal of Emaerging Technologies and Innovative Research (JETIR),* vol. 9, no. 10, pp. 258-265, 2022.

[80] Yunusa Simpa Abdulsalam, Mustapha Hedabou, "Security and Privacy in Cloud Computing: Technical Review," *Future Internet,* vol. 14, no. 11, 2022.

[81] Belen Bermejo, Carlos Juiz, "Improving cloud/edge sustainability through artificial intelligence: A systematic review," *Journal of Parallel and Distributed Computing,* vol. 176, pp. 41-54, 2023.

[82] H. N. Alshareef, "Current Development, Challenges and Future Trends in Cloud Computing: A Survey," *(IJACSA) International Journal of*

Advanced Computer Science and Applications, vol. 14, no. 3, pp. 329-339, 2023.

[83] Shajunyi Zhao, Jianchun Miao, Jingfeng Zhao, Nader Naghshbandi, "A comprehensive and systematic review of the banking systems based on pay-as-you-go payment fashion and cloud computing in the pandemic era," *Information Systems and e-Business Management,* 2023.

[84] Laura-Diana Radu, "Green Cloud Computing: A Literature Survey," *Symmetry,* vol. 9, no. 295, 2019.

[85] Archana Patil, Rekha Patil, "An Analysis Report on Green Cloud Computing Current Trends and Future Research Challenges," in *International Conference on Sustainable Computing in Science, Technology & Management (SUSCOM-2019)*, Karnataka, India, 2019.

[86] Nesma Abd El-Mawla, Hegazi Ibrahim, "Green Cloud Computing (GCC), Applications, Challenges and Future Research Directions," *Nile Journal of Communication &Computer Science,* vol. 1, no. 1, pp. 1-12, 2022.

[87] J Sylvia Grace, G Meeragandhi, "Green Cloud Computing and Environmental Impact Management for an IT Infrastructure," *International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies,* vol. 13, no. 6, 2022.

[88] Avita Katal, Susheela Dahiya, Tanupriya Choudhury, "Energy efficiency in cloud computing data centers: a survey on software technologies," *Cluster Computing,* vol. 26, pp. 1845-1875, 2023.

[89] Jason M. Pittman, Shaho Alaee, "A green scheduling algorithm for cloud-based honeynets," *Frontiers in Sustainability,* 2023.

[90] Jorge Pérez, Jessica Díaz, Javier Berrocal, Ramón López-Viana, Ángel González-Prieto, "Edge computing: A grounded theory study," *Computing,* vol. 104, pp. 2711-2747, 2022.

[91] Gustavo Caiza, Morelva Saeteros, William Oñate, Marcelo V. Garcia, "Fog computing at industrial level, architecture, latency, energy, and security: A review," *Heliyon,* vol. 6, 2020.

[92] Mohammed Al Masarweh,, Tariq Alwada'n , Waleed Afandi, "Fog Computing, Cloud Computing and IoT Environment: Advanced Broker Management System," *Journal of Sensor and Actuator Networks,* vol. 11, no. 84, 2022.

[93] D. Choi, "Fog computing application of cyber-physical models of IoT devices with symbolic approximation algorithms," *Journal of Cloud Computing,* pp. 11-63, 2022.

[94] S. S. Gill, "A Manifesto for Modern Fog and Edge Computing: Vision, New Paradigms, Opportunities, and Future Directions," in *Operationalizing Multi-Cloud Environments. EAI/Springer Innovations in Communication and Computing*, Springer, 2022.

[95] Sundas Iftikhar,Sukhpal Singh Gill , Chenghao Song, Minxian Xu, "AI-based fog and edge computing: A systematic review, taxonomy and future directions," *Internet of Things,* 2023.

[96] Pankaj Sharma, P K Gupta, "Optimization of IoT-Fog Network Path and fault Tolerance in Fog Computing based Environment," in *International Conference on Machine Learning and Data Engineering*, India, 2023.

[97] Gwanggil Jeon, Marcelo Albertini, Valerio Bellandi, Abdellah Chehri, "Intelligent mobile edge computing for IoT big data," *Complex & Intelligent Systems,* vol. 8, 2022.

[98] Kai Peng, Peichen Liu, Peng Tao, Qingjia Huang, "Security-Aware computation offloading for Mobile edge computing-Enabled smart city," *Journal of Cloud Computing: Advances, Systems and Applications,* 2021.

[99] Tarik Taleb, Sunny Dutta,Adlen Ksentini,Muddesar Iqbal, Hannu Flinck, "Mobile Edge Computing Potential in Making Cities Smarter," *IEEE Communications Magazine,* 2017.

[100] M. Cinque, "Real-Time FaaS: serverless computing for Industry 4.0," *Service Oriented Computing and Applications,* 2023.

[101] Yasmina Bouizem, Djawida Dib, Nikos Parlavantzas, Christine Morin, "Integrating request replication into FaaS platforms: an experimental evaluation," *Journal of Cloud Computing,* 2023.

[102] Urmil Bharti, Anita Goel, S. C. Gupta, "ReactiveFnJ: A choreographed model for Fork-Join Workflow in Serverless Computing," *Journal of Cloud Computing,* 2023.

[103] Tran Cong Hung & Nguyen Xuan Phi, "Study the effect of parameters to load balancing in cloud computing," *International Journal of Computer Networks & Communications (IJCNC),* vol. 8, no. 3, pp. 33-45, 2016.

[104] Khiet Thanh Bui, Tran Vu Pham, & Hung Cong Tran, "A Load Balancing Game Approach for VM Provision Cloud Computing Based on Ant

Colony Optimization," in *Context-Aware Systems and Applications International Conference, ICCASA*, Thu Dau Mot, Vietnam, 2016.

[105] N. Sasikaladevi, "Minimum makespan task scheduling algorithm in cloud computing," *International Journal of Advances in Intelligent Informatics ,* vol. 2, no. 3, pp. 123-130, 2016.

[106] Sambit Kumar Mishra, Md Akram Khan, Bibhudatta Sahoo, Deepak Puthal and Mohammad S. Obaidat, "Time Efficient Dynamic Threshold-based load balancing technique for cloud computing," *Fellow of IEEE, and KF Hsiao ,* 2017.

[107] Atyaf Dhari, Khaldun I.Arif , "An Efficient Load Balancing Scheme for Cloud Computing," *Arif Indian Journal of Science and Technology,* vol. 10, no. 11, 2017.

[108] Subasish Mohapatra , Ishan Aryendu, Anshuman Panda, Aswini Kumar Padhi, "A Modern Approach For Load Balancing Using Forest Optimization Algorithm," in *Proceedings of the Second International Conference on Computing Methodologies and Communication (ICCMC 2018)* , 2018.

[109] Nguyen Xuan Phi, Cao Trung Tin, Luu Nguyen Ky Thu, Tran Cong Hung , "Proposed Load Balancing Algorithm To Reduce Response Time And Processing Time On Cloud Computing," *International Journal of Computer Networks & Communications (IJCNC),* vol. 10, no. 3, 2010.

[110] İ. Çağlar and D. T. Altılar, "Look-ahead energy efficient VM allocation approach for data centers," *Journal of Cloud Computing Advances Systems and Applications,* vol. 11, no. 1, 2022.

[111] Rashmi. K. S & Suma. V & Vaidehi. M, "Enhanced Load Balancing Approach to Avoid Deadlocks in Cloud," *Special Issue of International Journal of Computer Applications,* 2012.

[112] Mahitha.O & Suma. V, "Deadlock Avoidance through Efficient Load Balancing to Control Disaster in Cloud Environment," in *4th ICCCNT* , 2013.

[113] Ha Huy Cuong Nguyen, Hung Vi Dang, Nguyen Minh Nhut Pham, Van Son Le, & Thanh Thuy Nguyen, "Deadlock Detection for Resource Allocation in Heterogeneous Distributed Platforms," *Advances in Intelligent Systems and Computing ,* vol. 361, 2015.

[114] Ha Huy Cuong Nguyen, Van Son Le, "Detection and Avoidance Deadlock for Resource Allocation in Heterogeneous Distributed Platforms," *International Journal of Computer Science and Telecommunications,* vol. 6, no. 2, 2015.

[115] Ha Huy Cuong Nguyen, Van Thang Doan, "Avoid Deadlock Resource Allocation (ADRA) Model V VM-out-of-N PM," *International Journal of Innovative Technology and Interdisciplinary Science,* vol. 2, no. 1, pp. 98-107, 2019.

[116] C. St-Onge, S. Benmakrelouf, N. Kara, H. Tout, C. Edstrom and R. Rabipour, "Generic SDE and GA-based workload modeling for cloud systems," *Journal of Cloud Computing Advances Systems and Applications,* vol. 10, no. 1, 2021.

[117] T.-P. Pham, J. J. Durillo and T. Fahringer, "Predicting workflow task execution time in the cloud using A two-stage machine learning approach," *IEEE transactions on cloud computing,* pp. 256-268, 2020.

[118] Rodrigo N. Calheiros, Rajiv Ranjan, César A. F. De Rose, Rajkumar Buyya, "CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing," Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, Melbourne, 2009.

[119] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, Rajkumar Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *SOFTWARE – PRACTICE AND EXPERIENCE,* vol. 41, p. 23–50, 2011.

[120] Pravesh Humane, J.N. Varshapriya, "Simulation of Cloud Infrastructure using CloudSim Simulator: A Practical Approach for Researchers," in *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, Chennai, T.N., India, 2015.

[121] Saydul Akbar Murad,, Abu Jafar Md Muzahid, Zafril Rizal M Azmi, Md Imdadul Hoque, Md Kowsher, "A review on job scheduling technique in cloud computing and priority Computer and Information Sciences,"

Journal of King Saud University - Computer and Information Sciences, vol. 34, 2023.

[122] M. Menaka, K.S. Sendhil Kumar, "Workflow scheduling in cloud environment – Challenges, tools, limitations & methodologies: A review," *Measurement: Sensors,* 2022.

[123] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: practice & experience,* vol. 41, no. 1, pp. 23-50, 2011.

[124] "Oracle java technologies," Oracle.com, [Online]. Available: https://www.oracle.com/java/technologies/. [Accessed 19 4 2022].

[125] A. NetBeans, "Welcome to Apache NetBeans," Apache.org, [Online]. Available: https://netbeans.apache.org/. [Accessed 19 4 2022].

[126] "The data platform for AI," WEKA, [Online]. Available: https://www.weka.io/. [Accessed 19 4 2022].

[127] "java: Java bindings for TensorFlow," TensorFlow, [Online]. Available: https://github.com/tensorflow/java. [Accessed 19 4 2022].

[128] B. Wickremasinghe, R. N. Calheiros and R. Buyya, "CloudAnalyst: A CloudSim-based visual modeller for analysing cloud computing environments and applications," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*.

[129] C. Guindon, "Eclipse desktop & web IDEs," Eclipse.org, [Online]. Available: https://www.eclipse.org/ide/. [Accessed 19 4 2022].

[130] "Wikipedia," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Deadlock. [Accessed 9 4 2021].

[131] Deep Shikha, Lalit Kumar, "Deadlock Prevention by Mutual Exclusion Process in Cloud Storage," *Journal of Emerging Technologies and Innovative Research (JETIR),* vol. 8, no. 9, pp. 436-442, 2021.

[132] "The importance of user behavior analytics for cloud service security," Oracle.com, [Online]. Available: https://www.oracle.com/assets/user-behavior-analytics-3497541.pdf. [Accessed 19 4 2022].

[133] K. K. a. K. E. N. Maryam Alruwaythi, "User Behavior and Trust Evaluation in Cloud Computing," *EPiC Series in Computing,* vol. 58, pp. 378-386, 2019.

[134] Z. Chen, L. Tian and C. Lin, "Trust evaluation model of cloud user based on behavior data," *International journal of distributed sensor networks,* vol. 14, no. 5, p. 155014771877692, 2018.

[135] N. Er-raji and F. Benabbou, "Priority task scheduling strategy for heterogeneous multi-datacenters in cloud computing," *International journal of advanced computer science and applications : IJACSA,* vol. 8, no. 2, 2017.