HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Nguyễn Thanh Trung

NGHIÊN CỨU CHÍNH SÁCH BỀN VỮNG NHẰM XÂY DỰNG THUẬT TOÁN NÂNG CAO HIỆU QUẢ CÂN BẰNG TẢI CỦA ĐIỆN TOÁN ĐÁM MÂY

Chuyên ngành: Hệ thống thông tin

Mã số: 8.48.01.04

TÓM TẮT LUẬN VĂN THẠC SĨ

THÀNH PHỐ HỒ CHÍ MINH – NĂM 2022

Luận văn được hoàn thành tại:

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Người hướng dẫn khoa học: PGS.TS. Trần Công Hùng

(Ghi rõ học hàm, học vị)

Phản biện 1: TS. Đàm Quang Hồng Hải

Phản biện 2: TS. Dương Thị Thùy Vân

Luận văn sẽ được bảo vệ trước Hội đồng chấm luận văn thạc sĩ tại Học viện Công nghệ Bưu chính Viễn thông

Vào lúc: 11 giờ 45 ngày 15 tháng 01 năm 2022

Có thể tìm hiểu luận văn tại:

- Thư viện của Học viện Công nghệ Bưu chính Viễn thông.

MỞ ĐẦU

Trong thời đại ngày nay, công nghệ thông tin và truyền thông ngày càng phát triển, đòi hỏi nhu cầu xử lý thông tin ngày càng cao, cần có hệ thống có khả năng lưu trữ và khai thác được một lượng dữ liệu lớn. Sự phát triển không ngừng của nền kinh tế thế giới và trong nước đòi hỏi các doanh nghiệp, các tập đoàn lớn phải có giải pháp để lưu trữ và khai thác thông tin về các dữ liệu lớn liên quan đến công việc kinh doanh của họ. Việc trang bị máy chủ vật lý đòi hỏi phải có một bộ phận kỹ thuật am hiểu về công nghệ thông tin để quản trị và vận hành hệ thống. Đồng thời cũng mất nhiều chi phí để đầu tư, nâng cấp phần mềm, phần cứng, phí bảo trì, nhân công...

Chính vì thế, điện toán đám mây (cloud computing) là một trong những giải pháp đang thu hút được một số lượng lớn các doanh nghiệp sử dụng. Điện toán đám mây thực chất là mô hình các máy chủ ảo, sử dụng các công nghệ máy tính và phát triển dựa vào mạng Internet. Ở mô hình điện toán này, mọi khả năng liên quan đến công nghệ thông tin đều được cung cấp dưới dạng các "dịch vụ". “Dịch vụ” này cho phép người sử dụng truy cập các dịch vụ công nghệ từ một nhà cung cấp nào đó "trong đám mây" mà không cần phải có các kiến thức, kinh nghiệm về công nghệ đó. Ngoài ra, người dùng cũng không cần quan tâm đến các cơ sở hạ tầng phục vụ công nghệ mà mình được cung cấp.

Điện toán đám mây giải quyết các vấn đề tối ưu hóa lưu trữ, ảo hóa máy chủ, cơ sở hạ tầng mạng với mục đích mang lại dịch vụ với chất lượng tốt nhất. Các tập đoàn, doanh nghiệp và cá nhân người dùng chỉ cần trả phí với dịch vụ tương ứng mà họ sử dụng.

Nói về chất lượng dịch vụ trên điện toán đám mây, người dùng cảm thấy chất lượng dịch vụ đáp ứng tốt trong công việc quản lý điều hành, lưu trữ và khai thác tài nguyên. Mặt khác, việc quản lý tài nguyên trở thành một công việc phức tạp đối với các nhà cung cấp dịch vụ đám mây. Có một số vấn đề được đặt ra: làm sao khắc phục vấn đề thiếu tài nguyên, giảm độ trễ trên đám mây và khả năng cải thiện hiệu suất mạng khi nhiều người dùng sử dụng cùng lúc.

Để giải quyết các vấn đề trên, hiện nay đã có hệ thống cân bằng tải để phân bổ đồng đều lưu lượng truy cập giữa hai hay nhiều các máy chủ có cùng chức năng trong cùng một hệ thống. Bằng cách đó, sẽ giúp cho hệ thống cung cấp dịch vụ của nhà cung cấp giảm thiểu một cách tối đa tình trạng một máy chủ bị quá tải và ngưng hoạt động.

Hiện nay có nhiều thuật toán cân bằng tải trên các dịch vụ đám mây. Tuy nhiên, hiệu quả của các thuật toán vẫn còn nhiều hạn chế, chưa đưa ra được giải pháp giúp bộ cân bằng tải cung cấp tài nguyên một cách hiệu quả và không tốn thời gian quay vòng lặp. Ngoài ra, các thuật toán cũng chưa có khả năng sẵn sàng và đảm bảo độ tin cậy của hệ thống.

Do đó, việc đề xuất *"Nghiên cứu chính sách bền vững nhằm xây dựng thuật toán nâng cao hiệu quả cân bằng tải của điện toán đám mây"* là vô cùng cần thiết.

Thuật toán đề xuất có khả năng chịu lỗi (Fault Tolerance), việc truy cập cũng được phân bổ đồng đều trên các nguồn tài nguyên, thậm chí là trên các Datacenter khi nhu cầu tăng lên một cách nhanh chóng. Hoặc khi một máy chủ gặp sự cố, chức năng cân bằng tải đám mây sẽ chỉ đạo phân phối công việc của máy chủ đó cho các máy chủ còn lại, đẩy thời gian (Uptime) của hệ thống lên cao nhất và cải thiện năng suất hoạt động.

Nhằm nâng cao hiệu quả cân bằng tải trên các dịch vụ điện toán đám mây đã được đề xuất, em xin đưa ra nội dung đề tài nghiên cứu như sau: *"Nghiên cứu chính sách bền vững nhằm xây dựng thuật toán nâng cao hiệu quả cân bằng tải của điện toán đám mây"*.

Luận văn được bố cục như sau:

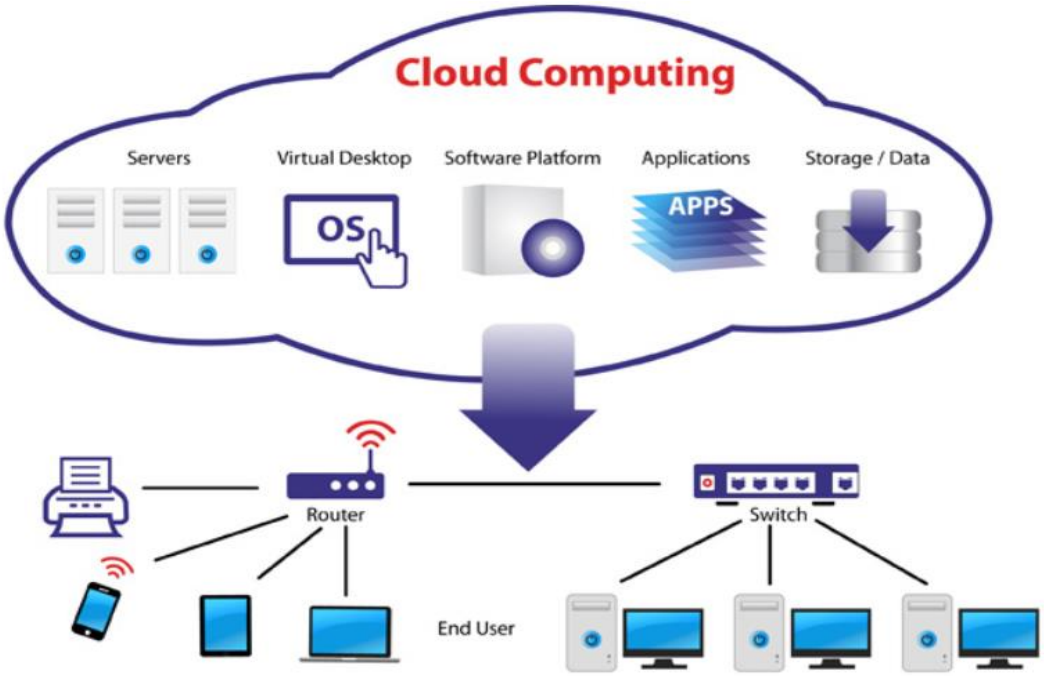
Trong chương 1 học viên sẽ giới thiệu tổng quan về hệ thống cân bằng tải của điện toán đám mây, giới thiệu sơ lược về điện toán đám mây, tổng quan về cân bằng tải, mục đích cân bằng tải để làm gì? Giới thiệu đôi nét về trí tuệ nhân tạo (AI), machine learning. Hiểu biết được những khái niệm tổng quan về điện toán đám mây. Trong chương 2, học viên sẽ giới thiệu các công trình liên quan đến đề tài ở trong và ngoài nước, thông qua việc nghiên cứu tìm hiểu được một số thuật toán và những công trình liên quan đến cân bằng tải trong điện toán đám mây, giúp luận văn

hiểu rõ hơn về cân bằng tải trên điện toán đám mây. Từ đó, hiểu được những ưu nhược điểm của các thuật toán cũng như các cách xử lý cân bằng tải, tạo tiền đề và cơ sở vững chắc cho nghiên cứu của đề tài luận văn này. Nghiên cứu các công trình này sẽ góp phần giúp củng cố hơn phần cơ sở lý thuyết và định hướng nghiên cứu, phát triển cho đề tài. Chương 3 học viên đưa ra mô hình nghiên cứu cứu sử dụng thuật toán Y nhằm mục đích loại bỏ các task tương ứng với các Request dựa trên độ ưu tiên xử lý task; trình bày một số thuật toán như Cây phân loại và hồi quy từ đó đề xuất chính sách bền vững nhằm xây dựng thuật toán nâng cao hiệu quả cân bằng tải của điện toán đám mây. Chương 4 học viên mô phỏng môi trường thực nghiệm, đưa ra kết quả thực nghiệm của mô hình, sau đó đánh giá kết quả thực nghiệm để thấy thuật toán đề xuất hiệu quả hơn các thuật toán khác.

CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN VỀ HỆ THỐNG CÂN BẰNG TẢI CỦA ĐIỆN TOÁN ĐÁM MÂY

1.1. Tổng quan về điện toán đám mây

Điện toán đám mây (cloud computing): hay còn gọi là điện toán máy chủ ảo, nơi các tính toán được "định hướng dịch vụ" và phát triển dựa vào Internet. Cụ thể hơn, trong mô hình điện toán đám mây, tất cả các tài nguyên, thông tin cũng như software đều được chia sẻ và cung cấp cho các máy tính, thiết bị, người dùng dưới dạng dịch vụ trên nền tảng một hạ tầng mạng công cộng (thường là mạng Internet). Các user thường sử dụng các dịch vụ như cơ sở dữ liệu, website, lưu trữ,… Trong mô hình cloud computing, không cần quan tâm đến vị trí địa lý cũng như các thông tin khác của hệ thống mạng đám mây - "điện toán đám mây trong suốt đối với người dùng". Người dùng cuối truy cập và sử dụng các ứng dụng đám mây thông qua các ứng dụng như trình duyệt web, các ứng dụng mobile hoặc máy tính cá nhân thông thường.



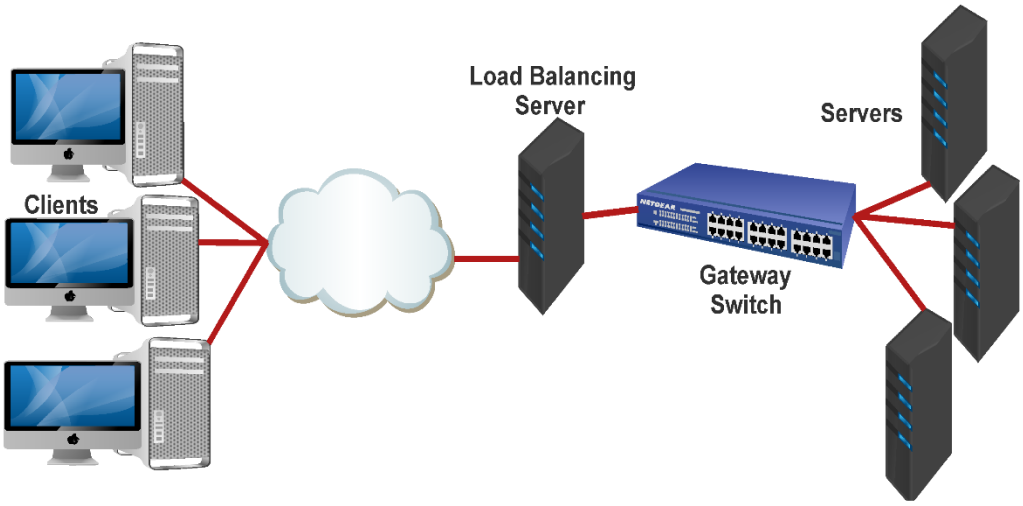
Hình 1.1. Mô hình điện toán đám mây [1]

1.2. Tổng quan về cân bằng tải trong điện toán đám mây

1.2.1. Giới thiệu về cân bằng tải

Cùng với việc phát triển rộng rãi của Internet, các website hay các ứng dụng trực tuyến hiện đang được rất nhiều người truy cập và sử dụng. Khi lượng truy cập này quá lớn thường xảy ra các vấn đề là hạ tầng mạng và khả năng xử lý của Server sẽ bị tắc nghẽn cục bộ. Vì vậy, Cân Bằng Tải luôn luôn là một trong những tính năng công nghệ rất quan trọng giúp các máy chủ ảo hoạt động đồng bộ và hiệu quả hơn thông qua việc phân phối đồng đều tài nguyên.

Giải pháp cân bằng tải là việc phân bố đồng đều lưu lượng truy cập giữa hai hay nhiều máy chủ có cùng chức năng trong cùng một hệ thống. Bằng cách đó, sẽ giúp cho hệ thống giảm thiểu tối đa tình trạng một máy chủ bị quá tải và ngưng hoạt động. Hoặc khi một máy chủ gặp sự cố, Cân Bằng Tải sẽ chỉ đạo phân phối công việc của máy chủ đó cho các máy chủ còn lại, đẩy thời gian uptime của hệ thống lên cao nhất và cải thiện năng suất hoạt động tổng thể.



Hình 1.5. Mô hình Cân bằng tải trong điện toán đám mây [8]

Cân bằng tải [11] có thể được chia thành 2 loại:

- Cân bằng tải cục bộ
- Tải toàn cầu

Cân bằng tải cục bộ được sử dụng để cân bằng dự báo tải trong một trung tâm. Nó phân phối yêu cầu từ phía máy khách sang cho máy chủ để đáp ứng nhu cầu. Thứ hai là loại cân bằng tải toàn cục. Nó quản lý và kiểm soát yêu cầu từ phía khách hàng tự động đến máy chủ qua nhiều trung tâm dữ liệu. Ngoài ra, nó cũng xử lý lưu

lượng trên cả hai mặt gói truyền tải. Xử lý cân bằng tải toàn cầu cho sự phức tạp nhưng đồng thời điều này cũng rất hữu ích cho truyền tải gói tin trên trung tâm dữ liệu mạng. Tính khả dụng đảm bảo rằng, trong trường hợp thất bại, hệ thống vẫn tiếp tục hoạt động được như mong đợi.

1.2.2. Mục đích cân bằng tải

Tăng khả năng đáp ứng, tránh tình trạng quá tải trên máy chủ, đảm bảo tính linh hoạt và mở rộng cho hệ thống.

Tăng độ tin cậy và khả năng dự phòng cho hệ thống: Sử dụng Cân bằng tải giúp tăng tính HA (High Availability) cho hệ thống, đồng thời đảm bảo cho người dùng không bị gián đoạn dịch vụ khi xảy ra lỗi sự cố tại một điểm cung cấp dịch vụ.

Tăng tính bảo mật cho hệ thống: Thông thường khi người dùng gửi yêu cầu dịch vụ đến hệ thống, yêu cầu đó sẽ được xử lý trên bộ Cân bằng tải rồi mới chuyển tiếp đến cho các máy chủ bên trong. Quá trình trả lời cho khách hàng cũng thông qua thành phần Cân bằng tải. Chính vì vậy mà người dùng không thể biết chính xác được các máy chủ bên trong cũng như phương pháp phân tải được sử dụng. Bằng cách này có thể ngăn chặn người dùng giao tiếp trực tiếp với các máy chủ, ẩn các thông tin và cấu trúc mạng nội bộ, ngăn ngừa các cuộc tấn công trên mạng hoặc các dịch vụ không liên quan đang hoạt động trên các cổng khác.

1.3. Tổng quan về trí tuệ nhân tạo (AI)

Trí tuệ nhân tạo (AI) [1] là một ngành khoa học máy tính liên quan đến việc tạo ra các chương trình nhằm mục đích tái tạo nhận thức con người và các quá trình liên quan đến việc phân tích sự phức tạp dữ liệu. Sự ra đời của khái niệm này được liên kết phổ biến với hội nghị Dartmouth năm 1956 [2]. Tuy nhiên, công nghệ tại thời điểm này đã giới hạn việc ứng dụng AI. Gần đây, những tiến bộ đáng kể đã được thực hiện trong lĩnh vực sức mạnh máy tính vì công nghệ phần cứng và phần mềm đã được cải tiến. Các cá nhân và tổ chức trong một số các ngành công nghiệp đang bắt đầu nhận ra tiềm năng của AI để cải thiện các hoạt động hiện tại. Ngoài ra, việc nghiên cứu AI cũng đã được tiến hành trên nhiều lĩnh vực như: y tế, điện toán đám mây, xử lý ảnh, …

1.4. Tổng quan về machine learning

Học máy (Machine Learning / ML) [3] là một phương pháp để tạo ra AI. ML liên quan đến các chương trình máy tính viết lập trình riêng cúng để hoàn thành một nhiệm vụ định trước. Quá trình này có thể được giám sát, bán giám sát, hoặc không giám sát. Trong học tập có giám sát, máy được cung cấp một tập dữ liệu. Với mỗi ví dụ trong tập dữ liệu, sẽ được gắn nhãn kèm theo câu trả lời. Các sau đó máy học thông qua phép thử và phép sai để dự đoán câu trả lời từ tập dữ liệu đã nhập. Học tập không giám sát liên quan đến việc phân tích dữ liệu đầu vào mà không có câu trả lời xác định. Điều này thường được sử dụng để mô hình hóa cấu trúc và phân phối dữ liệu. Cuối cùng, học tập bán giám sát là một phương pháp kết hợp liên quan đến việc kết hợp dữ liệu được gắn nhãn và không được gắn nhãn. Điều này có thể giúp giảm bớt gánh nặng của nhiệm vụ ghi nhãn. Sử dụng các thuật toán phân lớp của ML để tiến hành phân lớp người dùng dựa trên các điểm đặc trưng của họ để thực hiện việc cân bằng tải.

CHƯƠNG 2. CÁC CÔNG TRÌNH LIÊN QUAN

2.1. Các công trình nghiên cứu tại Việt Nam

Trong bài báo [1] của Trần Công Hùng và các cộng sự đăng trên tạp chí Khoa học công nghệ Thông tin và truyền thông số 04(CS.01) 2018 của Học viện Công nghệ Bưu chính viễn thông, đã đề xuất một thuật toán cân bằng tải nhằm giảm thời gian đáp ứng trên điện toán đám mây. Ý tưởng của bài báo này chính là sử dụng thuật toán dự báo ARIMA để dự báo thời gian đáp ứng. Từ đó, đưa ra cách giải quyết phân phối tài nguyên hiệu quả dựa vào giá trị ngưỡng thời gian. Bài báo đã đưa ra thuật toán cũng như thử nghiệm mô phỏng với mô hình nhỏ và đã đạt được một số kết quả mô phỏng khá tích cực, tiềm năng trong dự báo tương lai gần.

Trong bài báo [2] của tác giả Nguyễn Thanh Thủy và các cộng sự đăng trên tạp chí "International Journal of Computer Science and Network, Volume 4, Issue 2, April 2015", đã trình bày một cách tiếp cận để cải thiện thuật toán ngăn chặn bế tắc Đồng thời cũng lên lịch cho các chính sách cung cấp tài nguyên để phân bổ tài nguyên không đồng nhất. Thuật toán ngăn chặn bế tắc có độ phức tạp thời gian chạy là O (min (m, n)), trong đó m là số lượng tài nguyên và n là số lượng quy trình. Họ đề xuất thuật toán phân bổ nhiều tài nguyên cho các dịch vụ cạnh tranh đang chạy trong các máy ảo trên nền tảng phân tán không đồng nhất. Các thí nghiệm cũng so sánh hiệu suất của phương pháp đề xuất với các công việc liên quan khác.

2.2. Một số công trình nghiên cứu trên thế giới

Năm 2018, Afrianto và cộng sự [12] đã công bố nghiên cứu "Weighted Round Robin Load Balancer to Enhance Web Server Cluster in OpenFlow Networks". Nghiên cứu này nhằm mục đích thiết kế và phân tích mô hình cân bằng tải trên mạng OpenFlow và thực hiện thuật toán Round Robin trọng số (WRR). Quá trình phân tích được tiến hành bằng cách đo giá trị các thông số trình bày QoS của máy chủ web. Kết quả cho thấy thuật toán WRR có khả năng cân bằng hệ thống mạng với phân bổ tài nguyên động. Trọng lượng công việc của mỗi dịch vụ có thể lấy được từ nhu cầu và tài nguyên mạng hiện có. Hiệu suất của bộ cân bằng tải trên mạng OpenFlow tốt hơn 57% so với mạng truyền thống trong kiểm tra thời gian phản hồi.

Năm 2019, Shi và cộng sự [13] đã công bố nghiên cứu "Concury: A Fast and Light weighted Software Load Balancer". Trong nghiên cứu này, một phần mềm trạng thái cân bằng tải mới được giới thiệu, gọi là Concury - giải pháp đầu tiên để giải quyết hai vấn đề: (1) Các trạng thái được lưu trữ như các bản tiêu hóa có khả năng gây ra sự không nhất quán của gói tin do các xung đột của bản phân tích. (2) Mặt phẳng dữ liệu cần cập nhật cho mọi kết nối mới và việc cập nhật thường xuyên sẽ làm ảnh hưởng đến thông lượng cũng như tính nhất quán của gói. Cải tiến quan trọng của Concury là một cách tiếp cận theo thuật toán để lưu trữ và tra cứu các trạng thái mạng lớn. Cải tiến mới với kết nối thường xuyên và gọn gàng về chi phí bộ nhớ, nhất quán theo các thay đổi của mạng và không thường xuyên cập nhật mặt phẳng dữ liệu. Kết quả đánh giá cho thấy thuật toán Concury cung cấp thông lượng gấp 4 lần và tiêu thụ ít bộ nhớ hơn so với các thuật toán cân bằng tải khác. Đồng thời cung cấp khả năng cân bằng tải có trọng số và tự do đánh nhầm, cho cả lưu lượng trung tâm dữ liệu thực và tổng hợp.

Trong thập kỷ gần đây, sự gia tăng các yêu cầu (các ứng dụng đa dạng và phức tạp) cho các dịch vụ đám mây đang làm tăng khối lượng công việc trong môi trường đám mây. Các kỹ thuật định thời kém hiệu quả đang phải đối mặt với những thách thức về tài nguyên khi chúng được sử dụng quá mức và mất cân bằng. Điều đó dẫn đến suy giảm hiệu suất dịch vụ (do sử dụng quá mức) hoặc lãng phí tài nguyên đám mây (do chưa đáp ứng đủ). Ý tưởng cơ bản đằng sau thuật toán định thời là phân phối các nhiệm vụ (đa dạng và phức tạp) giữa các tài nguyên đám mây để tránh được vấn đề mất cân bằng. Thuật toán định thời nên tối ưu hoá các thông số trình bày chính: thời gian phản hồi, thời gian chờ, độ tin cậy, tính khả dụng, mức tiêu thụ năng lượng, chi phí, mức sử dụng tài nguyên,… Thế nên, vào năm 2019, Kumar và cộng sự [14] đã công bố nghiên cứu "A comprehensive survey for scheduling techniques in cloud computing" nhằm cung cấp đánh giá hệ thống cũng như phân loại các kỹ thuật lập lịch đề xuất cùng với những ưu điểm và hạn chế của chúng. Đây là một cuộc khảo sát có hệ thống và toàn diện, là bước đệm tốt cho các nhà nghiên cứu điện toán đám mây cũng như kỹ thuật định thời trong tương lai.

Cân bằng tải có thể cải thiện các chỉ số Chất lượng dịch vụ (QoS), bao gồm thời gian phản hồi, chi phí, thông lượng, hiệu suất và sử dụng tài nguyên. Vậy nên,

Ghomi và cộng sự [15] vào năm 2017 đã công bố bài nghiên cứu "Load-balancing algorithms in cloud computing: A survey" với tài liệu về các thuật toán định thời tác vụ và cân bằng tải. Đồng thời, họ còn trình bày một phân loại mới của các thuật toán. Ví dụ: danh mục cân bằng tải Hadoop MapReduce, danh mục cân bằng tải dựa trên Hiện tượng tự nhiên, danh mục cân bằng tải dựa trên tác nhân, danh mục Cân bằng tải chung, danh mục hướng ứng dụng, danh mục nhận biết mạng và danh mục quy trình làm việc cụ thể. Họ đã cung cấp các đánh giá về từng loại danh mục trên với thông tin chi tiết về việc xác định các vấn đề mở và hướng dẫn cho các nghiên cứu trong tương lai.

Điện toán đám mây đã trở nên phổ biến hơn do các dịch vụ mà nó cung cấp. Trong môi trường điện toán đám mây, cân bằng tải là một vấn đề rất quan trọng. Bởi người dùng và yêu cầu của họ đối với các dịch vụ khác nhau trên nền tảng điện toán đám mây ngày càng tăng, nên cần thiết phải sử dụng hiệu quả tài nguyên trong môi trường đám mây. Thuật toán cân bằng tải hiệu quả là thuật toán phải đảm bảo được việc sử dụng tài nguyên hiệu quả bằng cách cung cấp đầy đủ tài nguyên cho người dùng theo yêu cầu. Để ưu tiên người dùng Cân bằng tải sử dụng định thời, thời gian phản hồi và thời gian chờ là các chỉ số hoạt động của các thuật toán cân bằng tải. Trong bài báo "A Comparative Study of Static and Dynamic Load Balancing Algorithms in Cloud Computing" năm 2017, Deepa và cộng sự đã cung cấp một nghiên cứu so sánh có hệ thống về các thuật toán cân bằng tải hiện có trong điện toán đám mây. [16]

CHƯƠNG 3. NGHIÊN CỨU CHÍNH SÁCH BỀN VỮNG NHẰM XÂY DỰNG THUẬT TOÁN NÂNG CAO HIỆU QUẢ CÂN BẰNG TẢI CỦA ĐIỆN TOÁN ĐÁM MÂY

3.1. Mô hình nghiên cứu

Mô hình nghiên cứu sử dụng thuật toán Cây phân loại hồi quy (CART) nhằm mục đích loại bỏ các task tương ứng với các Request dựa trên chính sách bền vững. Chính sách bền vững được tính toán dựa trên mức độ tiêu thụ năng lượng của task (Power Consumed), mức độ sử dụng CPU (CPU Usages), mức độ sử dụng RAM (RAM Usages) và chi phí (Costing) để thực hiện các task trong cloud thông qua thuật toán K-Means. Sau khi phân loại các job/task theo chính sách bền vững, bộ cân bằng tải sẽ phân bổ các request tương ứng với các task có độ bền vững cao hơn vào những máy ảo/host có năng lực xử lý tốt hơn, tức là mức độ rảnh task cao hơn. Từ đó, phân bổ các request có nhu cầu xử lý cao vào các máy ảo/host có mức độ hoạt động thấp nhất. Với cách tiếp cận này, thuật toán đề xuất sẽ cải thiện thời gian xử lý cân bằng tải trên cloud đồng thời ứng dụng trên môi trường cloud theo thời gian thực. Trong luận văn này tạm đặt tên thuật toán là RCVKA.

Thuật toán được đề xuất như sau:

* Bước 1: Tiếp nhận request

* Bước 2: Áp dụng các chính sách của thuật toán để cân bằng tải

* Bước 3: Dựa vào kết quả thu được tiến hành cân bằng tải

Về mục tiêu:

- Giảm thiểu các rủi ro cho hệ thống máy chủ.
- Giảm thiểu tối đa thời gian sống cho các yêu cầu trong điện toán đám mây.
- Hạn chế tối đa và ngăn chặn sự mất cân bằng tải giữa các máy ảo.

Giả định:

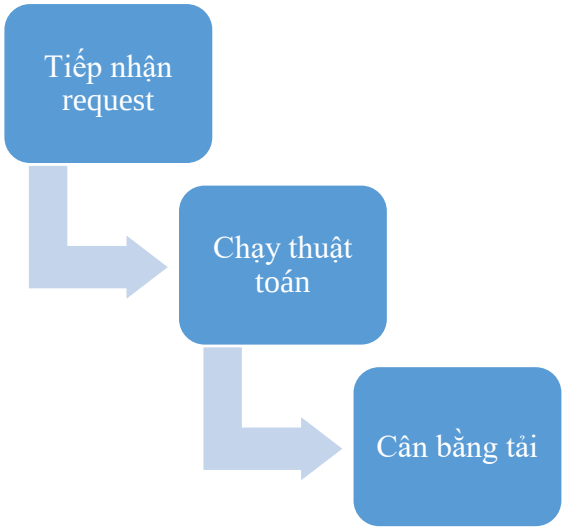
- Bộ cân bằng tải sẽ biết trước được những dịch vụ nào đang chạy trên các máy ảo vào bất kỳ thời điểm nào.

- Luận văn này chỉ tập trung vào dịch vụ Web (Web Service), các máy chủ web sẽ biết trước thời gian xử lý của từng dịch vụ chạy trên web và trên từng máy ảo.
- Nếu hai máy ảo có cấu hình tương đương nhau về RAM, vi xử lý và I/O thì thời gian thực thi của các dịch vụ sẽ không có nhiều khác biệt.

Mô hình nghiên cứu:

Thuật toán được đề xuất như sau:

* Bước 1: Tiếp nhận các request
* Bước 2: Áp dụng các chính sách của thuật toán để thực hiện cân bằng tải
* Bước 3: Dựa vào kết quả thu được tiến hành thực hiện cân bằng tải



Hình 3.1. Mô hình cân bằng tải

3.2. Thuật toán Cây phân loại và hồi quy (Classification and Regression Tree - CART)

Thuật toán CART [21] là một loại thuật toán phân loại cần thiết để xây dựng cây quyết định trên cơ sở "chỉ số tạp chất Gini". Nó là một thuật toán học máy cơ bản và cung cấp nhiều trường hợp sử dụng. Nhà thống kê Leo Breiman đã đặt ra cụm từ này để mô tả các thuật toán Cây quyết định có thể được sử dụng cho các vấn đề phân loại hoặc mô hình dự báo hồi quy.

CART là một từ bao hàm dùng để chỉ các loại cây quyết định sau:

- Cây phân loại: Khi biến mục tiêu liên tục, cây được sử dụng để tìm "lớp" mà biến mục tiêu có nhiều khả năng rơi vào nhất.
- Cây hồi quy: Được sử dụng để dự báo giá trị của một biến liên tục.

Trong cây quyết định, các nút được chia thành các nút con trên cơ sở giá trị ngưỡng của một thuộc tính. Thuật toán CART thực hiện điều đó bằng cách tìm kiếm sự đồng nhất tốt nhất cho các nút con, với sự trợ giúp của tiêu chí Gini Index.

Nút gốc được lấy làm tập huấn luyện và được chia thành hai bằng cách xem xét giá trị ngưỡng và thuộc tính tốt nhất. Hơn nữa, các tập hợp con cũng được chia theo cùng một logic. Điều này tiếp tục cho đến khi tìm thấy bộ con thuần túy cuối cùng trên cây hoặc số lá tối đa có thể có trên cây đang phát triển đó. Điều này còn được gọi là Cắt tỉa cây (Tree Pruning).

Công thức tính toán chỉ số Gini.

$$GI = \Sigma_{i=0}^{c} P_i(1 - P_i) \text{ hoặc } GI = 1 - \Sigma_{i=0}^{c} P_i^2$$

Ở đây c là tổng số lớp và P là xác suất của lớp i.

3.4. Thuật toán K-Means

Thuật toán K-Means là thuật toán rất quan trọng và được sử dụng phổ biến trong kỹ thuật phân cụm, được đề xuất bởi J.B.MacQueen. Thuật toán không giám sát này thường được sử dụng trong khai thác dữ liệu và nhận dạng mẫu. Hướng đến việc giảm thiểu chỉ số hiệu suất cụm, tiêu chí lỗi bình phương và lỗi là nền tảng của thuật toán này. Để tìm kiếm kết quả tối ưu hóa, thuật toán này cố gắng tìm K bộ phận để thỏa mãn một tiêu chí nhất định. Đầu tiên, chọn một số dấu chấm để đại diện cho các tiêu điểm cụm ban đầu (thông thường, chọn K điểm mẫu đầu tiên của thu nhập để đại diện cho tiêu điểm cụm ban đầu); thứ hai, gom các chấm mẫu còn lại về tiêu điểm theo tiêu chí khoảng cách tối thiểu thì ta sẽ được phân loại ban đầu, phân loại nếu không hợp lý thì sửa đổi (tính lại từng tiêu điểm từng cụm), lặp đi lặp lại cho đến khi có được một sự phân loại.

3.5. Thuật toán đề xuất RCVKA

Dựa vào tham khảo từ tài liệu [20], luận văn này xin đề xuất thuật toán gồm 3 nhóm module chính:

(1) Module tính toán ra các thông số của Request bằng thuật toán K-Means:

Thuật toán K-Means trong Module 1 đóng vai trò tính toán các thông số sử dụng tài nguyên của các task/job dựa trên các thuộc tính của Request sau khi chọn lọc bởi chính sách bền vững. Các thuộc tính của Request bao gồm: Size, Response Length, Max Length,…

Po_{New} = K-Means(Request, Power)

CPU_{New} = K-Means(Request, CPU)

RAM_{New} = K-Means(Request, RAM)

Request = $\{X_1, X_2, \ldots, X_n\}$, với X_i là các thuộc tính của Request khi gửi lên cloud.

Trong đó:

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

CPU_{New} (CPU dự 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, ta có thể sử dụng nhóm 3 yếu tố {Po, CPU, RAM} để tổng hợp tính toán hoặc để tính toán riêng biệt từng đại lượng.

(2) Module phân lớp tác vụ theo hành vi người dùng:

Module này sẽ sử dụng thuật toán CART để phân lớp Request đang xét, dựa vào chính sách bền vững trên cloud của các tác vụ.

VM_{select} = RandomForest(Po, CPU, RAM);

Trong đó:

VM_{select}: máy ảo được chọn

Cart: hàm phân lớp từ mô hình Cây quyết định CART đã được xây dựng dựa trên bộ dữ liệu trước đây của các Request

Po: Power dự đoán tính toán từ Module 1

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

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

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

Nhiệm vụ của Module này là phân bổ các loại Request đến các máy ảo phù hợp. Nếu một Request được gửi tới thì Request đó sẽ được phân loại bởi Module 1 và các VM đang xét bao gồm VM không tải cũng được phân cụm theo Module 2. Ngoài ra, Module 3 có nhiệm vụ phân bổ Request đang xét vào máy ảo đã được tìm thấy từ Module 2, từ đó xử lý và trả về kết quả của Request. Sau đó, lưu kết quả vào

lịch sử bộ nhớ các Request gần nhất đã xử lý để làm dữ liệu đầu vào cho quá trình xây dựng mô hình Cây quyết định CART ở Module 2.

Thuật toán RCVKA (**R**equest **C**ART Classification & **V**M **K**-Means Clustering **A**lgorithm)

```
1.  For each Request in CloudRequests
2.      isLocated = false;
3.      SustainablePolicy = {Po, CPU, RAM}new = K-Means (T1,T2,…,Tn); //
        Module 1
4.      Request.SustainablePolicyClass = Cart(SustainablePolicy); //Cart là mô
        hình phân lớp tác vụ
5.      For each VM in VMList
6.          If isFitSituation(Request.SustainablePolicyClass, VM)
7.              AllocateRequestToVM(VM, Request); // Module 3
8.              isLocated = true;
9.          End If
10.     End For
11.     If(!isLocated)
12.         VM = VMList.getSelectedVM(); // Module 2
13.         AllocateRequestToVM(VM, Request);
14.     End If
15. End For
```

Trong đoạn mã giả trên, thuật toán RCVKA đã sử dụng một vòng lặp để lắng nghe tất cả các Request có trong danh sách hàng đợi được gửi lên bộ cân bằng tải (CloudRequests). Thuật toán sẽ tiếp tục phân bổ các Request cho đến khi hết danh sách hàng đợi. Cụ thể, thuật toán sử dụng biến isLocated (kiểu luận lý – boolean) để làm cờ (flag) đánh dấu Request đang xét để xem Request đó đã được phân bổ hay chưa. Khi bắt đầu vòng lặp, biến isLocated được khởi tạo giá trị mặc định là false. Bước tiếp theo, thuật toán sẽ tính toán ra vector SustainablePolicy 3 chiều tương ứng với PowerConsume, CPU Usage và RAM Usage (Priority = {Po, CPU, RAM}) cần dùng để thực hiện Request đang xét. Việc tính toán này dựa trên số liệu của các Request T1, T2,… Tn trước đó, trong đó n là số Request đã được lưu trong lịch sử. Ti chính là các đại lượng của Request thứ i đã được lưu lại. Ti 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, RAM Usage. N Request trong lịch sử này là dữ liệu xây dựng nên hàm K-Means để dự báo và tính toán ra các đại lượng SustainablePolicy cho Request đang xét. Đại lượng SustainablePolicy cũng chính là dữ liệu đầu vào để chạy CART phân lớp cho Request đang xét, lớp sau khi phân sẽ được gán vào thuộc tính

SustainablePolicyClass của Request. Sau khi chạy ra thông số SustainablePolicy cho Request, thuật toán sẽ thực hiện duyệt vòng lặp để duyệt qua các máy ảo hiện đang có trên cloud. Với từng máy ảo tương ứng, thuật toán sẽ xem xét mức độ phù hợp đối với độ ưu tiên của Request đang xét, thông qua hàm isFitSituation(Request.SustainablePolicyClass, VM). Nếu thỏa, RCVKA sẽ phân bổ Request đang xét vào máy ảo: AllocateRequestToVM(VM, Request) đồng thời gán giá trị biến isLoacated = true. Trong trường hợp không tìm được máy ảo nào thích hợp thì vòng lặp sẽ kết thúc. Lúc này, thuật toán chạy hết vòng lặp, biến isLocated vẫn mang giá trị false và Request vẫn chưa được phân bổ. Vì vậy, nhiệm vụ chính của thuật toán RCVKA hiện giờ là phân bổ Request trên vào máy ảo đầu tiên trong danh sách thông qua đoạn lệnh VM = VMList.getSelectedVM(). Việc phân bổ này đảm bảo các Request được dự báo nếu không nằm trong dữ liệu của thuật toán vẫn được phân bổ và xử lý để phục vụ người dùng.

Phương pháp đánh giá thuật toán RCVKA

Kết quả đạt được từ thuật toán mà luận văn đề xuất **RCVKA** đã đáp ứng các mục tiêu được đề ra trước đó. Chẳng hạn như, giới hạn số lượng yêu cầu xếp hàng để phân phối, giảm thiểu thời gian xử lý và phản hồi của đám mây trung tâm, đều tốt hơn so với các thuật toán cũ. Điều này đồng nghĩa với việc hiệu năng của điện toán đám mây được thực hiện bởi thuật toán **RCVKA** đã cải thiện hơn so với bốn thuật toán được đề cập đến trong luận văn là **FCFS**, **MaxMin**, **MinMin** và **Round Robin**.

CHƯƠNG 4. MÔ PHỎNG CHƯƠNG TRÌNH VÀ ĐÁNH GIÁ KẾT QUẢ

4.1. Giới thiệu chung

Trong chương này, luận văn sẽ trình bày về cài đặt mô phỏng thuật toán RCVKA. Cụ thể, chương này sẽ sử dụng thuật toán RCVKA phân bổ các task tương ứng với các Request dựa trên độ bền vững các task đó đã được tính toán. Mức độ tiêu thụ năng lượng của task (Power Consumed), mức độ sử dụng CPU (CPU Usages), mức độ sử dụng RAM và chi phí (Costing) để thực hiện các task trong cloud đã được tính toán kỹ càng tạo nên độ bền vững của thuật toán. Sau khi phân loại các task theo độ ưu tiê độ bền vững, bộ cân bằng tải sẽ phân bổ các request ứng với các task có độ bền vững cao hơn vào những máy ảo/host có năng lực xử lý tốt hơn hay có mức độ rảnh task cao. Từ đó, phân bổ vào các máy ảo/host có mức độ hoạt động thấp nhất các request có nhu cầu xử lý cao. Với cách tiếp cận này, thuật toán đề xuất RCVKA sẽ cải thiện thời gian xử lý cân bằng tải trên cloud sau đó ứng dụng trên môi trường cloud theo thời gian thực. Các kết quả thu được sau khi tiến hành các bước như trên dùng để phân tích tính hiệu quả của thuật toán đã đề xuất.

4.2. Môi trường mô phỏng thực nghiệm

Dựa vào bộ dữ liệu chứa các Request, luận văn đã sử dụng thuật toán K-Means để phân loại Request dựa trên các đặc trưng của chúng (hành vi người dùng cloud).

Bước 1: Tiếp nhận giá trị input

Bước 2: Phân tích các input để rút trích các đặc trưng của các giá trị input

Bước 3: Dựa vào các đặc trưng trên, chúng ta sử dụng machine learning để phân lớp các giá trị đầu vào.

Bước 4: Dựa vào kết quả phân lớp, ta tiến hành cân bằng tải cho hệ thống.

Dựa vào dữ liệu có thể biết của các Request, ta sử dụng thuật toán Regression để phân loại chúng bằng cách tính toán ra bộ Priority = {Power, CPU, RAM}. Từ đó, ta biết được cách phân bổ tài nguyên cho các Request vào các máy ảo đã được phân cụm sẵn trước đó. Kết hợp đánh giá số lần sai và sai số, ta có thể cải thiện thuật toán

bằng cách áp dụng máy học (machine learning). Tuy nhiên, máy học sẽ ít được áp dụng để cải thiện thuật toán vì vẫn xảy ra sai số cho phép.

Khi giả lập môi trường cloud, luận văn đã sử dụng ngôn ngữ lập trình JAVA và bộ thư viện CloudSim. Bao gồm 5 đến 15 máy ảo, môi trường cloud là nơi để các Request ngẫu nhiên truy cập đến các dịch vụ trên cloud. Các dịch vụ gồm có: dịch vụ cung cấp máy ảo, dịch vụ cung cấp và đáp ứng người dùng của CloudSim để tiến hành thử nghiệm.

Sau đó, luận văn tiếp tục tiến hành cài đặt thuật toán K-Means và J48 trên môi trường mô phỏng được phát triển bởi bộ thư viện WEKA. Cuối cùng là kiểm nghiệm kết quả của thuật toán đề xuất.

Các tham số của mô hình mạng mô phỏng:

Quá trình thực nghiệm mô phỏng thuật toán được cài đặt trên Eclipse IDE hoặc NETBEAN IDE và sử dụng ngôn ngữ JAVA để chạy thử với kết quả được hiển thị dưới dạng console. Môi trường giả lập dùng bộ thư viện mã nguồn mở CloudSim (được cung cấp bởi http://www.cloudbus.org/) kết hợp với bộ thư viện về datamining là WEKA.

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

- 1 Datacenter với thông số sau:

Bảng 4.1. Thông số cấu hình Datacenter

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 - Bandwidth: 10000

- Các máy ảo có cấu hình giống nhau khi được khởi tạo:

Bảng 4.2. Cấu hình máy ảo

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

- Các Request (chạy trên web, WebRequest) đại diện bởi Cloudlet trong CloudSim và kích thước của các Cloudlet được khởi tạo một các ngẫu nhiên bằng hàm Random của JAVA. Số lượng Cloudlet lần lượt từ 500 → 1500.

Bảng 4.3. Cấu hình thông số các Request

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

- Thuật toán đề xuất được xây dựng bằng cách tạo ra lớp **RJVKASchedulingAlgorithm** kế thừa từ đối tượng **BaseSchedulingAlgorithm**. Song, thuật toán đề xuất đã có cập nhật thêm một số phương thức và thuộc tính liên quan tới **predictRequestKmeans** đồng thời điều chỉnh các hàm dựng sẵn để trở nên phù hợp hơn:

@Override

*public void **run()** // Module 3*

*public CondorVM **getFittingVm** (double label)*

// Module 2

*public String **predictRequestPowerConsume**(Cloudlet req)*

*public String **predictRequestCpuUsage**(Cloudlet req)*

*public String **predictRequestRamUsage** (Cloudlet req)*

// Module 1

Tiêu chí đánh giá

Thực nghiệm mô phỏng cloud với các tham số như trên để chạy thuật toán cân bằng tải của CloudSim có sẵn. Sau đó, kết hợp chạy thuật toán đề xuất RCVKA mới cài đặt với dữ liệu đầu vào. Cuối cùng, so sánh kết quả đầu ra, đặc biệt là thông số thời gian thực hiện (Makespan). Sai số của thời gian đáp ứng dự đoán của các máy ảo và thời gian đáp ứng dự đoán của cloud càng thấp thì hiệu quả của thuật toán càng được đánh giá cao.

4.3. Kết quả thực nghiệm của mô hình.

Ta tiến hành chạy thực nghiệm mô phỏng trên CloudSim với 5 máy ảo đã được dựng sẵn để đáp ứng các Request. Các request được khởi tạo có chiều dài và kích thước ngẫu nhiên với số lượng Request lần lượt là 30, 60, 100 và 1000. Dưới đây là kết quả ghi nhận mô phỏng 30 và 1000 request:

Bảng 4.4. Kết quả thực nghiệm mô phỏng với 30 request

Thời gian thực hiện (ms)	FCFS	RCVKA	MaxMin	MinMin	Round Robin
AVG	554.70	206.43	253.53	261.94	271.48
MAX	8,618.72	2,734.52	3,068.34	3,191.11	4,687.36
MIN	0.26	0.11	0.14	0.14	0.12

Với kết quả thực nghiệm của 30 Request, có thể thấy rằng thuật toán đề xuất RCVKA đang có lợi thế hơn so với các thuật toán còn lại. Thuật toán MaxMin và MinMin cũng theo sát thuật toán RCVKA với thời gian xử lý khá ổn định. Trong khi đó, thuật toán FCFS lại có thời gian xử lý khá cao, chiếm gấp gần 4 lần so với thuật toán đề xuất. Tuy nhiên, để chứng tỏ thuật toán đề xuất thật sự có hiệu quả, ta sẽ tiến hành xử lý nhiều request hơn và quan sát kết quả.

Bảng 4.7. Kết quả thực nghiệm mô phỏng với 1000 request

Thời gian thực hiện (ms)	FCFS	RCVKA	MaxMin	MinMin	Round Robin
AVG	380.90	326.59	545.62	639.75	514.52
MAX	15,971.02	6,360.17	54,741.18	30,587.87	60,263.71
MIN	0.12	0.18	0.15	0.16	0.13

Với trường hợp 1000 request này, thuật toán RCVKA đã chứng minh mình vượt trội hơn so với 4 thuận toán còn lại. Nếu lượng request bùng nổ, việc xử lý của

thuật toán vẫn đáp ứng được, phụ thuộc vào cấu hình thiết bị chạy bộ cân bằng tải vì vậy không quá khó khăn trong việc xử lý cân bằng tải nếu lượng request nhiều.

4.4. Đánh giá kết quả

Mô hình thực nghiệm mô phỏng với các thông số cũng như kịch bản đưa ra đều dựa vào quá trình request của các browser trên môi trường cloud. Qua đó, luận văn ghi nhận được các thông số về thời gian xử lý của các máy ảo cũng như của cloud. Việc chạy thực nghiệm với thông số của 5 máy ảo, chịu tải từ 30 đến 1000 Request đã cho ra kết quả tương đối khả quan. Ngoài ra, việc phân bổ các Request đến các máy ảo cũng xử lý khá đồng đều và có tính khả thi cao.

KẾT LUẬN

Luận văn "**NGHIÊN CỨU CHÍNH SÁCH BỀN VỮNG NHẰM XÂY DỰNG THUẬT TOÁN NÂNG CAO HIỆU QUẢ CÂN BẰNG TẢI CỦA ĐIỆN TOÁN ĐÁM MÂY**" nghiên cứu các thuật toán tiến hành phân lớp từ các đặc trưng của request chọn lọc bởi chính sách bền vững. Sau đó, phân bổ các tác vụ vào các máy ảo sao cho hợp lý và nâng cao cân bằng tải trong môi trường điện toán đám mây, sử dụng hợp lý và có hiệu quả nguồn tài nguyên đám mây. Dựa vào các thuật toán đã có để phân tích làm rõ chúng, sau đó có thể đánh giá đưa ra nhược điểm và lợi thế của từng thuật toán. Từ các nhược điểm đã phân tích, đề xuất một thuật toán kết hợp với chính sách bền vững nhằm cải tiến và nâng cao khả năng cân bằng tải so với các thuật toán cũ. Quá trình nghiên cứu đã đạt được nhiều mục tiêu đề ra như sau:

- Nghiên cứu tổng quan đám mây và các đám mây với ba mô hình chính đang được sử dụng. Các kỹ thuật cân bằng tải được dùng trong môi trường điện toán đám mây.
- Nghiên cứu các thuật toán hỗ trợ để phân lớp, phân cụm các request. Nghiên cứu chính sách bền vững, áp dụng vào để phân cụm các request dựa trên các đặc trưng của nó.
- Nghiên cứu cách tiếp cận đám mây điện toán thông qua mô phỏng sử dụng công cụ giao diện thân thiện và dễ sử dụng của CloudSim. Cài đặt và mô phỏng các kỹ thuật cân bằng tải, các thuật toán Round Robin, MaxMin**,** MinMin và thuật toán tự nhiên FCFS. Các giá trị thu được khi mô phỏng đưa ra dùng để phân tích so sánh với nhau, nhằm tóm lại được các nhược điểm và ưu điểm của các thuật toán. Từ đó, luận văn sẽ đề xuất một thuật toán sửa đổi để khắc phục mặt hạn chế đó.
- Kết quả đạt được từ thuật toán đề xuất đáp ứng được các mục tiêu như việc đáp ứng thời gian được cải thiện, hạn chế các tài nguyên bị đói cũng như máy ảo có năng lực xử lý mạnh sẽ được xử lý nhiều yêu cầu hơn. Qua đó, giúp cân bằng tải hiệu quả hơn các thuật toán được so sánh là Round Robin, MaxMin, MinMin và thuật toán tự nhiên FCFS.

- Thuật toán đề xuất RCVKA có thể dùng để đưa vào áp dụng trên thực tế.

Hạn chế luận văn

- Chưa được ứng dụng vào môi trường thực tế.
- Thời gian đáp ứng và xử lý chưa cải thiện được nhiều.

Vấn đề kiến nghị và hướng đi tiếp theo của nghiên cứu:

- Đưa thuật toán đề xuất vào ứng dụng thực tế.

Áp dụng mô hình năng lượng tiêu thụ của Datacenter hoặc cloud tương ứng để xây dựng biểu đồ phân bổ tải cho cloud.