HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Vương Duy Thanh

NGHIÊN CỨU ỨNG DỤNG AI XÂY DỰNG THUẬT TOÁN DỰ BÁO CÁC TÁC VỤ TRÊN ĐÁM MÂY NHẰM NÂNG CAO HIỆU QUẢ CÂN BẰNG TẢI

LUẬN VĂN THẠC SỸ KỸ THUẬT

(Theo định hướng ứng dụng)

TP. HCM - 2022

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Vương Duy Thanh

NGHIÊN CỨU ỨNG DỤNG AI XÂY DỰNG THUẬT TOÁN DỰ BÁO CÁC TÁC VỤ TRÊN ĐÁM MÂY NHẰM NÂNG CAO HIỆU QUẢ CÂN BẰNG TẢI

CHUYÊN NGÀNH: HỆ THỐNG THÔNG TIN

MÃ SỐ: 8.48.01.04

TÓM TẮT LUẬN VĂN THẠC SỸ KỸ THUẬT

(Theo định hướng ứng dụng)

TP. HCM – 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

Phản biện 1:

……

Phản biện 2:

……

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: ..... giờ ..... ngày ..... tháng .... năm .....

Có thể tìm hiểu luận văn tại:

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

I. MỞ ĐẦU

1. Tính cấp thiết của đề tài

Trong thời đại ngày nay, Công nghệ thông tin và truyền thông ngày càng phát triển, đòi hỏi nhu cầu về xử lý thông tin ngày càng cao. Khi đó, nhu cầu về khả năng lưu trữ được một lượng dữ liệu to lớn là vô cùng cấp thiết. Sự phát triển không ngừng của nền kinh tế thế giới đã đẩy các doanh nghiệp, các tập đoàn lớn vào tình thế buộc phải thay đổi. Lúc này, cần có một giải pháp giúp họ lưu trữ được một khối lượng khổng lồ các dữ liệu liên quan đến công việc kinh doanh của họ.

Vì vậy, giải pháp để đáp ứng tất cả các nhu cầu nói trên trong nhiều năm qua đã xuất hiện. Đó chính là Điện toán đám mây [1] (Cloud computing).

Nhiều phương pháp dựa trên trí tuệ nhân tạo (AI) đã được đề xuất để dự báo các tác vụ nhằm nâng cao hiệu quả cân bằng tải trên đám mây. Tuy nhiên, để cải thiện hơn nữa độ chính xác dự đoán các tác vụ của các phương pháp dựa trên máy học và học sâu trước đây, ứng dụng AI xây dựng thuật toán dự báo các tác vụ trên đám mây nhằm nâng cao hiệu quả cân bằng tải được đề xuất trong luận văn này, đề tài như sau: *"Nghiên cứu ứng dụng AI xây dựng thuật toán dự báo các tác vụ trên đám mây nhằm nâng cao hiệu quả cân bằng tải"*.

Với việc dự báo được các tác vụ và phân bổ phù hợp cho các tài nguyên, bộ cân bằng tải sẽ làm việc hiệu quả hơn. Ngoài ra, hiệu quả kinh doanh của nhà cung cấp dịch vụ đám mây cũng được cải thiện bằng cách giảm sự từ chối về số lượng công việc được gửi.

Luận văn bao gồm: Phần mở đầu, nội dung gồm bốn chương và phần kết luận.

2. Tổng quan về vấn đề nghiên cứu

Theo tài liệu [2], điện toán đám mây (Cloud Computing), còn gọi là điện toán máy chủ ảo, là mô hình điện toán sử dụng các công nghệ máy tính và phát triển dựa vào mạng Internet. Cụ thể hơn là trong mô hình điện toán đám mây, tất cả các tài nguyên, thông tin và software đều được chia sẻ và cung cấp cho các máy tính, thiết bị cùng với người dùng dưới dạng dịch vụ trên nền tảng một hạ tầng mạng công cộng (thường là mạng Internet). Người dùng cuối truy cập và sử dụng các ứng dụng đám mây thông qua các ứng dụng như trình duyệt web, ứng dụng mobile hoặc máy tính cá nhân.

2.1. Lợi ích của điện toán đám mây

Giúp tiết kiệm chi phí: Vì không cần trung tâm dữ liệu tại chỗ nên không cần phải lắp đặt máy chủ, phần cứng, phần mềm...

2.2. Các mô hình dịch vụ [3]

Mô hình dịch vụ của điện toán đám mây được các nhà cung cấp dịch vụ chia thành 3 loại lớn:

2.2.1. Cơ sở hạ tầng như một dịch vụ (Infrastructure as a Service - IaaS)

2.2.2. Nền tảng như một dịch vụ (Platform as a Service - PaaS)

2.2.3. Phần mềm như một dịch vụ (Software as a Service - SaaS)

2.3. Tổng quan về cân bằng tải

2.3.1. Cân bằng tải [1]

Cân bằng tải là một phương pháp quan trọng trong điện toán đám mây giúp các máy chủ hoạt động hiệu quả thông qua việc phân phối tài nguyên một cách đồng đều, giảm hoặc tránh trình trạng tắc nghẽn.

2.3.2. Cân bằng tải trên điện toán đám mây [2]

Cân bằng tải trong điện toán đám mây cung cấp giải pháp cho các vấn đề khác nhau về thiết lập và sử dụng tài nguyên trong môi trường điện toán đám mây.

2.4. Một số công trình nghiên cứu liên quan đến cân bằng tải

3. Mục đích nghiên cứu

- Mục tiêu chính: Nghiên cứu ứng dụng Machine Learning xây dựng thuật toán dự báo các tác vụ trên đám mây nhằm nâng cao hiệu quả cân bằng tải..

4. Đối tượng và phạm vi nghiên cứu

- Đối tượng nghiên cứu
  - Đối tượng nghiên cứu chính là tác vụ trên điện toán đám mây.
  - Nghiên cứu các thuật toán dự báo áp dụng vào dự báo tác vụ.
- Phạm vi nghiên cứu.

- Xây dựng mô hình mô phỏng đám mây ở mức độ nhỏ: khoảng 10 – 20 máy ảo (Có thể sử dụng máy thật tuy nhiên để tiết kiệm chi phí nên trong đề cương này, thuật toán sẽ mô phỏng và áp dụng trên máy ảo).
- Độ phức tạp trên mỗi máy ảo chỉ ở mức độ thấp: khoảng 1 – 4 ứng dụng trên các máy ảo đó.
- Yêu cầu (Request) gửi về máy chủ cũng đơn giản, dung lượng dữ liệu gửi về nhỏ: Khoảng dưới 1 Mb.

5. Phương pháp nghiên cứu

- Phương pháp luận:

  Dựa trên cơ sở là các lý thuyết về điện toán đám mây, Task Prediction, cân bằng tải trên cloud.

- Phương pháp đánh giá dựa trên cơ sở toán học:

  Trên cơ sở các lý thuyết về điện toán đám mây. Đề xuất ra thuật toán để dự báo tác vụ trên đám mây dựa trên các thuật toán dự báo (thống kê, AI, …). Chứng minh thuật toán và đánh giá hiệu quả của thuật toán.

- Phương pháp đánh giá bằng mô phỏng thực nghiệm

  Xây dựng mô hình mô phỏng và thực nghiệm thuật toán đã đề xuất.

II. NỘI DUNG

CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG DỰ BÁO CÁC TÁC VỤ TRÊN ĐÁM MÂY NHẰM NÂNG CAO HIỆU QUẢ CÂN BẰNG TẢI

1.1. Tổng quan về điện toán đám mây

Điện toán đám mây (cloud computing): [5], [6] hay còn gọi là điện toán máy chủ ảo nơi các tính toán được "định hướng dịch vụ" và phát triển dựa vào Internet.

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

1.2.1 Giới thiệu về cân bằng tải

Cân bằng tải là một trong những chủ đề quan trọng nhất trong môi trường phân tán. Vì Cloud Computing được coi là một trong những nền tảng tốt nhất giúp lưu trữ dữ liệu với chi phí tối thiểu và có thể truy cập mọi lúc qua internet, cân bằng tải cho điện toán đám mây đã trở thành một lĩnh vực nghiên cứu rất thú vị và quan trọng. Cân bằng tải nhằm mục đích thỏa mãn người dùng và sử dụng tỷ lệ tài nguyên cao bằng cách đảm bảo phân bổ hợp lý. Có rất nhiều khó khăn trong các kỹ thuật cân bằng tải như bảo mật, khả năng chịu lỗi, v.v ... vốn phổ biến trong môi trường điện toán đám mây hiện đại. Nhiều nhà nghiên cứu đã

đề xuất một số kỹ thuật thuật toán để tăng cường nhằm tìm ra những phương án tốt nhất cho Cân bằng tải.

1.2.2 Mục đích cân bằng tải

Tăng khả năng đáp ứng và tránh tình trạng quá tải trên máy chủ đồng thời 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 sự cố lỗi tại một điểm cung cấp dịch vụ.

Tăng tính bảo mật cho hệ thống:

1.3. Tổng quan về tác vụ

Tác vụ [17], [18] là quá trình sắp xếp các yêu cầu (request) đến theo một cách thức nhất định để lập lịch tài nguyên sẵn có sẽ được sử dụng hợp lý. Vì điện toán đám mây là công nghệ cung cấp dịch vụ thông qua phương tiện Internet, người dùng dịch vụ phải gửi yêu cầu của họ thông qua Internet.

1.4. Vai trò của dự báo tác vụ đối với cân bằng [19] tải trên cloud

Việc dự báo các tác vụ [20], [21] có vai trò quan trọng trong việc cân bằng tải trên cloud. Với việc có thể dự báo trước các tác vụ cần

sử dụng những tài nguyên nào sẽ giúp cho việc cân bằng tải hiệu quả và tránh việc phân phối các tác vụ không hợp lý.

1.5. Tổng quan về AI

Trí tuệ nhân tạo (AI) [22], [23] là một ngành khoa học máy tính liên quan đến việc tạo ra các chương trình nhằm mục đích tái tạo con người nhận thức 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 [24].

1.6. Tổng quan về Machine Learning

Học máy (Machine Learning / ML) [22] là một tập hợp con của AI, cho phép máy móc có thể học từ bộ dữ liệu bất kì hoặc học từ những kinh nghiệm trước đó mà không cần phải lập trình một cách cụ thể và chi tiết. ML liên quan đến các chương trình máy tính viết lập trình của riêng chúng để hoàn thành một nhiệm vụ định trước.

1.7. Kết luận chương

Hiểu biết được những khái niệm tổng quan về điện toán đám mây, trí tuệ nhân tạo và học máy. Hiểu biết thuật toán điện toán đám mây để dự báo các tác vụ trên đám mây thông qua môi trường điện toán. Mục đích cân bằng tải để làm tăng hiệu năng của hệ thống.

CHƯƠNG 2: CÁC CÔNG TRÌNH LIÊN QUAN

2.1. Ở Việt Nam.

Năm 2018, Nguyễn Xuân Phi, Lê Ngọc Hiếu và Trần Công Hùng [11] đã công bố nghiên cứu "Thuật toán cân bằng tải nhằm giảm thời gian đáp ứng dựa vào ngưỡng thời gian trên điện toán đám mây". Nhóm tác giả đã nghiên cứu một thuật toán mới cho cân bằng tải trên đám mây bằng cách sử dụng mô hình dự đoán thời gian đáp ứng được đề xuất và thử nghiệm mô phỏng với một mô hình nhỏ. Thuật toán được sử dụng trong nghiên cứu này là thuật toán ARIMA.

2.2. Trên thế giới.

Năm 2021, Ibrahim Mahmood Ibrahim và các cộng sự [12] đã công bố nghiên cứu *"Task Scheduling Algorithms in Cloud Computing: A Review"*. Bài báo này đưa ra ý tưởng về các thuật toán lập lịch tác vụ khác nhau trong điện toán đám mây môi trường được sử dụng bởi các nhà nghiên cứu.

2.3. Tổng kết chương

Trong chương này 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.

CHƯƠNG 3 : ĐỀ XUẤT THUẬT TOÁN DỰ BÁO TÁC VỤ TRÊN ĐIỆN TOÁN ĐÁM MÂY NHẰM NÂNG CAO HIỆU QUẢ CÂN BẰNG TẢI

3.1. Giới thiệu chung

Cân bằng tải trên điện toán đám mây là một trong những công nghệ thu hút được sự chú ý của nhiều nhà khoa học cũng như các tổ chức, doanh nghiệp lớn, nhỏ và nhà cung cấp dịch vụ trong những năm gần đây. Các ứng dụng của AI và thuật toán ML được phát triển, cải tiến và ứng dụng trong các hệ thống cân bằng tải trên điện toán đám mây. Trong chương này, thuật toán đề xuất sử dụng trong quá trình cân bằng tải trên cloud được trình bày với mục tiêu dự báo các tác vụ dựa trên lịch sử thực hiện tác vụ trước đó. Mục đích là để phân bổ các request vào các máy ảo phù hợp, có khả năng đáp ứng cao.

3.2. Mô hình nghiên cứu

Mô hình nghiên cứu sử dụng thuật toán phân lớp AdaBoost nhằm mục đích dự báo các request tương ứng với các task dựa trên lịch sử về thời gian thực hiện cũng như xử lí các request. Lịch sử xử lí ở đây 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 task đó trong cloud. Sau khi phân loại các job/task theo lịch sử thực hiện, bộ cân bằng

tải sẽ phân bổ các request có các job/task đó vào những máy ảo/host có năng lực xử lý tốt hơn. Từ đó, phân bổ request có nhu cầu xử lý nhiều vào 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 và ứng dụng trên môi trường cloud theo thời gian thực. Trong luận văn này tạm đặt tên thuật toán là ACTPA (AdaBoost Classification of Task-Prediction Algorithm).

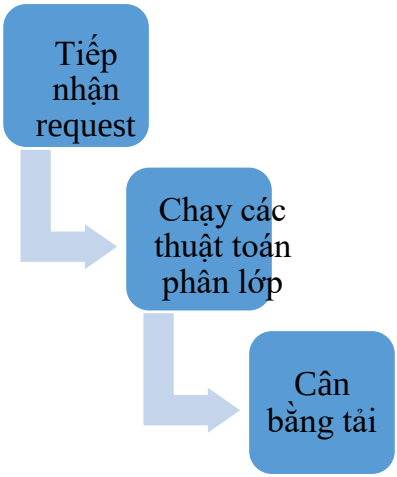
Về mục tiêu:

- Giảm thiểu rủi ro cho hệ thống máy chủ.
- Giảm thiểu thời gian sống cho các yêu cầu trong điện toán đám mây.
- Hạn chế tối đa cũng như ngăn chặn sự mất cân bằng tải giữa các máy ảo.
- 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). Đồng thời sử dụng hiệu quả hơn nguồn tài nguyên trên cloud để đáp ứng tốt nhất các yêu cầu của người dùng.
- Phân lớp được các yêu cầu (Coming Request) tiếp theo tương ứng với độ ưu tiên đã được phân 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.

Mô hình nghiên cứu:

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



Hình 3.1: Mô hình dự đoán tác vụ

3.3. Thuật toán AdaBoost

Adaboost là một thuật toán boosting dùng để xây dựng bộ phân lớp (classifier). Trong đó, boosting là thuật toán học quần thể bằng cách xây dựng nhiều thuật toán học cùng lúc (ví dụ như cây quyết định) và kết hợp chúng lại. Mục đích là để có một cụm hoặc một nhóm các *weak learner*, sau đó kết hợp chúng lại để tạo ra một *strong learner* duy nhất.

Thuật toán Adaboost có thể kết hợp với nhiều thuật toán khác để cải thiện hiệu suất. Đầu ra của một thuật toán (thường được gọi là "weak learners") được kết hợp lại thành một tổng có trọng số đại diện cho đầu ra cuối cùng của bộ phân loại tăng cường (boosted classifier).

Trong máy tính, một task 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,… Tất cả đều được tính toán ra chi phí thực hiện công việc đó theo thời gian hoặc MIPS. Chính vì thế, bài luận văn này dựa vào các đặc điểm đó để tính toán ra độ ưu tiên của task mà máy tính phục vụ. Task có mức tiêu hao năng lượng (Power Consumed), mức độ sử dụng CPU (CPU Usage) cũng như mức độ sử dụng RAM (RAM Usage) hay chi phí (Cost) cao hơn thì sẽ có độ ưu tiên cao hơn và ngược lại.

3.4. Thuật toán đề xuất ACTPA

Thuật toán đề xuất ACTPA (*AdaBoost Classification Task Prediction 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 AdaBoost 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 đó, 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 máy học vào. Tuy nhiên, việc áp dụng này sẽ ít diễn ra vì có sai số cho phép.

Luận văn này xin đề xuất thuật toán gồm 3 nhóm Module chính:

(1) Module phân lớp các request bằng thuật toán Random Forest (RF):

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

Nhóm Thời Gian xử lý = MK_{New} = DT (X_1, X_2,…, X_n)

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 nhiều hơn dựa vào độ biến thiên của Request.

(2) Module phân lớp tác vụ dựa trên thời gian dự báo:

Trong Module này sẽ sử dụng thuật toán phân lớp AdaBoost 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 AdaBoost của các request đã được xử lý trong quá khứ và đánh nhãn từ 1 đến 5, tương ứng với 5 mức độ ưu tiên. Mức 1 là độ ưu tiên thấp nhất, mức 5 là độ ưu tiên cao nhất. Dựa vào mô hình này, ta phân chia được 44 lớp từ các request đang cần xử lý và xác định được label (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 hoạt động thấp hay ít tải của máy. Tức, mức 1 là máy tải nhiều nhất và mức 5 là máy tải ít nhất nhưng tính sẵn sàng cao nhất.

VMselect = AdaBoost(Po, CPU, RAM);

Trong đó:

VMselect là máy ảo được chọn ra

AdaBoost là hàm phân lớp từ bộ thư viện WEKA

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 dự báo tác vụ và máy ảo phù hợp. Nếu một yêu cầu được gửi đến thì sẽ 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. Từ đó xử lý và trả về kết quả cho request đồng thời 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 AdaBoost ở Module 2.

Thuật toán ACTPA

```
1.     For each Request in CloudRequests
2.           isLocated = false;
3.           RT = RandomForest(T1,T2….. Tn); //Module 1
4.           Request.Class = AdaBoost(RT); // AdaBoost là mô
hình phân lớp tác vụ
5.           For each VM in VMList
6.                 If isFitSituation(Request.Class, 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
```

3.5. Kết luận chương

Chương này giới thiệu vì sao tác giả lại chọn việc dự báo thời gian tải tối đa và tải tối thiểu cũng như giá trị của tải đó thông qua thời gian xử lý để phục vụ công việc cân bằng tải. Với mục tiêu duy trì trạng thái an toàn và hoạt động liên tục của cloud, nghiên cứu hướng đến mục đích tối ưu hóa nguồn tài nguyên của cloud và giúp cho cân bằng tải

hoạt động tốt nhất. Thuật toán đề xuất sẽ giải quyết được cân bằng tải dựa trên cải thiện thời gian thực thi.

CHƯƠNG 4. MÔ PHỎNG THUẬT TOÁN VÀ ĐÁNH GIÁ KẾT QUẢ

4.1. Giới thiệu chung

Để có thể hiểu rõ hơn về thuật toán đề xuất, chương này sẽ trình bày về cài đặt mô phỏng thuật toán ACTPA (AdaBoost Classification of Task Prediction Algorithm). Quá trình mô phỏng cân bằng tải sẽ sử dụng thuật toán đề xuất ACTPA để phân loại và dự báo các tác vụ dựa trên thời gian xử lí. Sau đó, điều phối các tác vụ đến các máy ảo phù hợp. Các tác vụ có thời gian xử lí càng cao sẽ được phân bổ vào các máy ảo có mức độ sử dụng thấp, tức máy ảo có tính sẵn sàng cao và ngược lại. Với cách tiếp cận này, thuật toán đề xuất ACTPA sẽ tối ưu hóa thời gian xử lý cân bằng tải trên cloud và ứng dụng trên môi trường cloud theo thời gian thực. Sau khi tiến hành các bước thực nghiệm và thu được các kết quả, ta sẽ phân tích cũng như so sánh tính hiệu quả của thuật toán đề ra với các thuật toán cân bằng tải nổi tiếng khác. Các thuật toán được so sánh lần lượt là Round Robin, MaxMin, MinMin và FCFS.

4.2. Môi trường mô phỏng thực nghiệm

Dựa vào dữ liệu của các request, ta sử dụng thuật toán SVM để phân loại chúng bằng cách tính toán ra thời gian xử lý. Cũng từ đó, ta biết cách phân bố tài nguyên cho cái request vào các máy ảo đã phân cụm. Kết hợp với đánh giá số lần sai và sai số, ta sẽ cải thiện thuật toán bằng cách áp dụng máy học vào. Dù vậy, việc áp dụng này sẽ ít diễn ra vì có sai số cho phép.

Giả lập môi trường cloud sử dụng bộ thư viện CloudSim và lập trình trên ngôn ngữ JAVA. Môi trường giả lập cloud từ 5 đến 15 máy ảo chính là môi trường request ngẫu nhiên đến các dịch vụ trên cloud. Các dịch vụ trên cloud bao gồm: cung cấp máy ảo, cung cấp và đáp ứng người dùng muốn thử nghiệm của CloudSim.

Cài đặt thuật toán SVM, AdaBoost trên môi trường mô phỏng và kiểm nghiệm kết quả.

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

- 01 Datacenter với thông số như 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	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

- TimeZone: +7 GMT - Cost: 3.0 - Cost per Memory: 0.05 - Cost per Storage: 0.1 - Cost per Bandwidth: 0.1	- 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 (các request chạy trên web, WebRequest) được 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 là 30 □ 1000.

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 ACTPA được xây dựng bằng cách tạo ra lớp **ACTPASchedulingAlgorithm**, lớp này đã được kế thừa từ đối tượng có sẵn là **DataAwareSchedulingAlgorithmExample**. Đồng thời, thuật toán cũng cập nhật thêm một số phương thức và thuộc tính liên quan tới **predictRequestSVM**, sau đó điều chỉnh các hàm dựng sẵn để phù hợp với thuật toán đề xuất:

@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 và chạy thuật toán cân bằng tải của CloudSim có sẵn. Sau đó, chạy thuật toán đề xuất mới cài đặt với cùng dữ liệu đầu vào và so sánh kết quả đầu ra, đặc biệt là thông số thời gian xử lý.

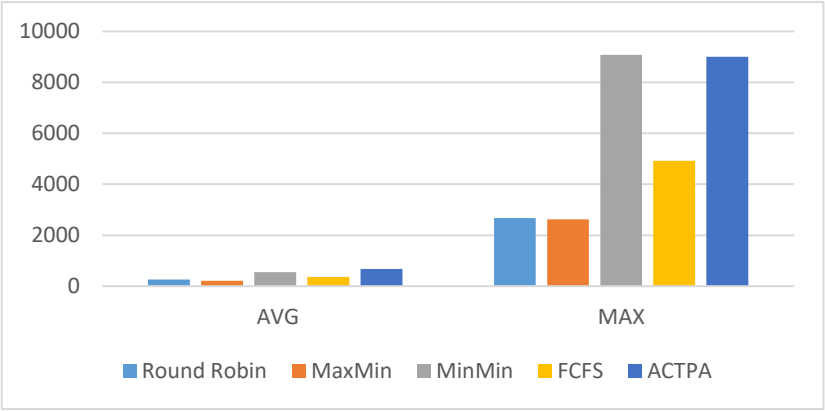
Thời gian xử lý của các máy ảo cũng như thời gian xử lý của cloud với sai số càng thấp thì hiệu quả của thuật toán càng tốt.

4.3. Thực nghiệm và kết quả mô phỏng

Kết quả chạy thực nghiệm mô phỏng trên CloudSim với 5 máy ảo được dựng sẵn để đáp ứng các yêu cầu. Các yêu cầu này được khởi tạo với chiều dài và kích thước ngẫu nhiên cùng số lượng Request từ 30, 60, 100 đến 1000. Kết quả thu được của thuật toán đề xuất được đem đi so sánh với các thuật toán Round-Robin, MaxMin, MinMin và FCFS có thời gian thực hiện là:

Bảng 4.4: Kết quả thực nghiệm mô phỏng với 30 Request

Thuật toán	Thời gian thực hiện (ms)		
	AVG	MAX	MIN
Round Robin	259.37	2677.21	0.26
MaxMin	217.71	2621.24	0.12
MinMin	548.94	9080.72	1.03
FCFS	364.19	4925.67	1.3
ACTPA	679.49	9001.37	0.38



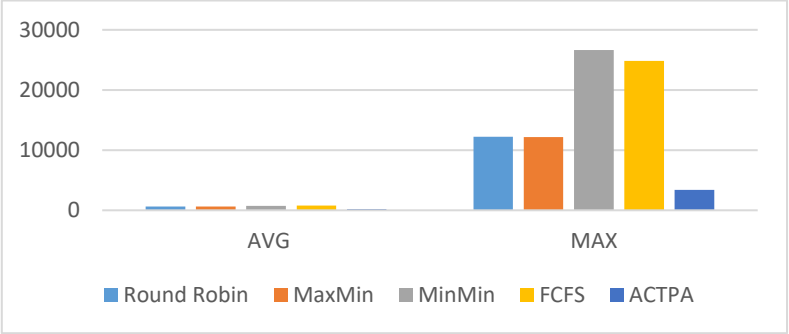
Hình 4.1: Biểu đồ so sánh thời gian thực hiện của 5 thuật toán với 30 Request

Quá trình thực nghiệm với 30 request trở lại, thuật toán MaxMin đã chiếm ưu thế với thời gian xử lí thấp nhất ở cả hai mốc thời gian trung bình và thời gian cao nhất. Ngoài ra, thuật toán Round Robin cũng có thời gian xử lí tác vụ rất tốt, chỉ kém thuật toán MaxMin 41.66 ms ở mức trung bình và 55.97 ms ở mức cao nhất. Thuật toán đề xuất trong phạm vi dưới 30 request vẫn chưa phát huy được sự tối ưu trong việc xử lí các tác vụ so với các thuật toán còn lại.

Kết quả chạy thực nghiệm mô phỏng trên CloudSim với 5 máy ảo được dựng sẵn để đáp ứng các yêu cầu. Chúng được khởi tạo cùng chiều dài và kích thước ngẫu nhiên, với số lượng Request là 60:

Bảng 4.5: Kết quả thực nghiệm mô phỏng với 60 request

Thuật toán	**Thời gian thực hiện (ms)**		
	AVG	**MAX**	**MIN**
Round Robin	650.77	12226.29	0.13
MaxMin	648.04	12194.58	0.18
MinMin	731.74	26626.42	0.11
FCFS	770.81	24847.42	0.11
ACTPA	222.83	3411.38	0.11



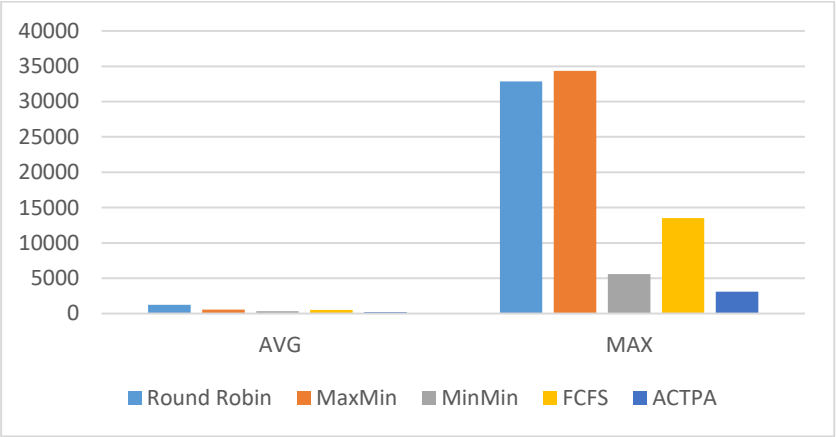
Hình 4.2: Biểu đồ so sánh thời gian thực hiện của 5 thuật toán với 60 Request

Kết quả thu được trong phạm vi dưới 60 request dần chứng minh được khả năng tối ưu hóa thời gian xử lí tác vụ của thuật toán đề xuất ở cả hai mốc thời gian. Từ biểu đồ ta có thể thấy được tốc độ của thuật toán đề xuất ACTPA đã vượt xa các thuật toán cân bằng tải còn lại và vẫn đang có chiều hướng phát triển tốt.

Bảng 4.6: Kết quả thực nghiệm mô phỏng với 100 request

Thuật toán	**Thời gian thực hiện (ms)**		
	AVG	**MAX**	**MIN**
Round Robin	1221.92	32851.68	0.15

MaxMin	578.37	34341.74	0.12
MinMin	339.68	5578.17	0.18
FCFS	503.30	13517.15	0.17
ACTPA	196.54	3104.3	0.11



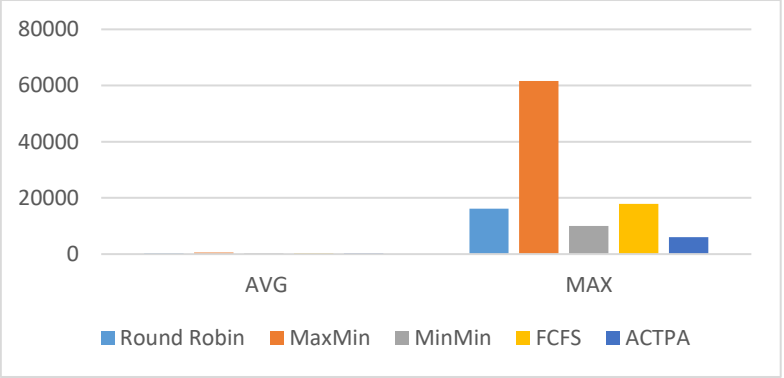
Hình 4.3: Biểu đồ so sánh thời gian thực hiện của 5 thuật toán với 100 Request

Từ request thứ 100 trở đi, thuật toán ACTPA vượt trội hơn hẳn so với MaxMin và Round Robin. Tuy nhiên, vẫn chưa đủ lợi thế so với RoundRobin. Nhưng với số lượng request càng lớn thì ACTPA càng lợi thế hơn. Dần dà, thuật toán đề xuất càng chiếm ưu thế tuyệt đối so với

các thuật toán còn lại. Rõ ràng FCFS thể hiện sự thiếu tự nhiên và tính thông minh của giải thuật.

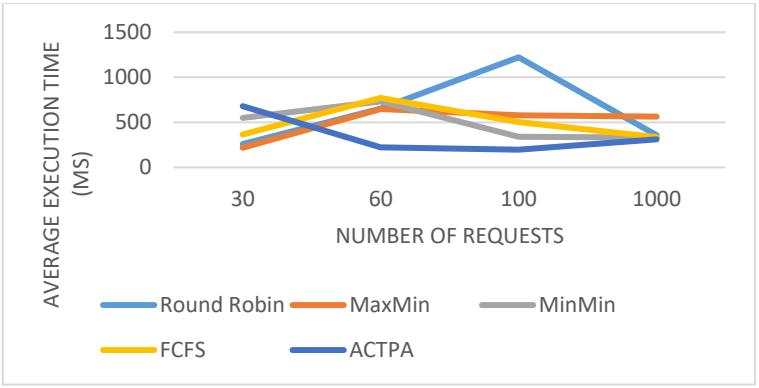
Bảng 4.7: Kết quả thực nghiệm mô phỏng với 1000 request

Thuật toán	Thời gian thực hiện (ms)		
	AVG	**MAX**	**MIN**
Round Robin	356.87	16134.21	0.11
MaxMin	563.26	61632.4	0.11
MinMin	331.16	10018.56	0.1
FCFS	335.86	17833.29	0.11
ACTPA	310.09	6038.66	0.17

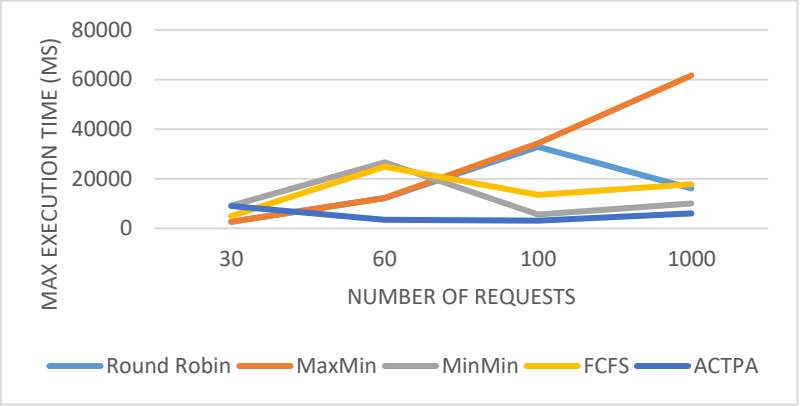


Hình 4.4: Biểu đồ so sánh thời gian thực hiện của 5 thuật toán với 1000 Request

Kết quả thực nghiệm 1000 request cho thấy thuật toán đề xuất vượt trội hơn hẳn các thuật toán cân bằng tải khác ở cả hai mốc thời gian là AVG và MAX.



Hình 4.5: Thời gian thực hiện trung bình của 5 thuật toán với 1000 Request



Hình 4.6: Thời gian thực hiện trung bình của 5 thuật toán với 1000 Request

Thông qua 02 biểu đồ tổng hợp 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 ACTPA. Cụ thể, thời gian xử lý của các máy ảo khả quan hơn so với thời gian xử lý của các thuật toán khác trên cloud (cả trường hợp ít và nhiều request).

Thực nghiệm mô phỏng này chỉ là mô phỏng nhóm các máy ảo và chưa tính đến việc mở rộng tập các máy ảo (VM pool) để giảm tải trong trường hợp cần thiết. Vì ta chỉ giả định nhóm các máy ảo này xử lý tối đa bao nhiêu request, nếu vượt quá ta mới mở rộng pool. Tuy nhiên, việc thí nghiệm mô phỏng với lượng request lớn trên 1000 request đòi

hỏi máy tính mạnh hơn và có bộ xử lý tốt hơn. Chính vì vậy, đây cũng là hạn chế của thí nghiệm mô phỏng này.

4.4. Tổng kết chương

Chương 4 của luận văn trình bày mô hình thực nghiệm mô phỏng, các thông số cũng như kịch bản đưa ra là dựa vào quá trình request của các browser trên môi trường cloud. Từ đó, ghi nhận các thông số về thời gian đáp ứng dự báo của các máy ảo và của cloud. Việc chạy thực nghiệm mô phỏng với thông số 5 máy ảo, chịu tải từ 30 đến 1000 request đã cho thấy kết quả tương đối tốt. Đồng thời, việc phân bổ các request đến các máy ảo xử lý cũng khá đồng đều và có tính khả thi cao.

KẾT LUẬN VÀ KIẾN NGHỊ

Luận văn "**Nghiên cứu ứng dụng AI xây dựng thuật toán dự báo các tác vụ trên đám mây nhằm nâng cao hiệu quả cân bằng tải**" đã cơ bản hoàn thành được các mục tiêu đề ra. Thuật toán đề xuất dựa trên ý tưởng từ nhiều công trình nghiên cứu trong và ngoài nước. Từ đó, tiến hành tổng hợp, phân tích và đánh giá. Nhờ sự hỗ trợ mạnh mẽ từ bộ thư viện WEKA và công cụ mô phỏng điện toán đám mây Cloudsim, các máy ảo và thuật toán được cài đặt một cách dễ dàng, nhanh chóng thông qua môi trường APACHE NETBEAN với ngôn ngữ lập trình Java. Quá trình mô phỏng sử dụng thuật toán phân lớp Adaboost nhằm phân loại các máy ảo và thuật toán Random Forest cũng như phân loại các các tác vụ dựa trên thời gian xử lí. Từ đó, dự báo các tác vụ và phân bổ vào các máy ảo tương ứng. Các kết quả thu được từ quá trình thực nghiệm thuật toán đề xuất được phân tích và so sánh với các thuật toán cân bằng tải trên đám mây khác như Round Robin, MaxMin, MinMin, FCFS. Dựa vào kết quả so sánh dựa trên ba mốc thời gian là AVG, MAX và MIN, nhìn chung kết quả cho thấy rằng thuật toán đề xuất có tính hiệu quả cao và khả năng vượt trội hơn các thuật toán khác về việc dự báo các tác vụ trên điện toán đám mây.

Một thuật toán mới về cân bằng tải trên môi trường cloud bằng phương pháp phân loại các request theo thời gian xử lý đã được đề xuất và thực nghiệm mô phỏng với mô hình nhỏ. Dựa trên ý tưởng và các

công trình nghiên cứu trước, đưa ra một giải thuật mới ứng dụng khai phá dữ liệu là thuật toán AdaBoost để cân bằng tải dựa vào thời gian xử lý. Trong đó, việc tính toán ra thời gian xử lí càng chính xác thì hiệu quả thuật toán càng cao. Tuy nhiên, việc tính toán càng chính xác thì càng đòi hỏi tốn nhiều bộ nhớ và phải xử lí nhiều. Bên cạnh đó, người dùng trên môi trường cloud có các request vô cùng đa dạng và phong phú nên thời gian xử lý cũng biến đổi khôn lường trên cloud. Thuật toán được đề xuất trong luận văn này tiếp cận một cách khái quát cũng như phát huy ý tưởng của phân lớp theo Regression, toán học và thời gian xử lý, điển hình là thuật toán AdaBoost. Do đó, thuật toán đề ra đã có một hướng tiếp cận khá mới trong cân bằng tải ở môi trường đám mây đồng thời đạt được một số kết quả thực nghiệm mô phỏng khá tích cực, cho thấy hướng phát triển tốt của thuật toán.

Hướng phát triển của thuật toán đề xuất là việc đo lường và hiệu chỉnh chính xác hơn bằng cách kết hợp AdaBoost với học máy, học không giám sát hoặc có giám sát bằng cách đưa ra các khoảng thời gian cao điểm hoặc thấp điểm của cloud. Để phát triển thuật toán tốt hơn và sâu hơn, cần thực nghiệm mô phỏng trên máy tính có cấu hình mạnh hơn và quy mô lớn hơn.

Bên cạnh đó, việc cài đặt thuật toán trên cloud thực tế sẽ giúp ta nghiên cứu chuyên sâu và cụ thể hơn. Bởi, môi trường cloud thực tế sẽ phát sinh nhiều vấn đề liên quan đến thời gian xử lý. Từ đó, ta sẽ biết hiệu chỉnh thuật toán một cách hợp lý hơn và hiệu quả hơn.