THUẬT TOÁN CÂN BẰNG TẢI NHẰM GIẢM THỜI GIAN ĐÁP ỨNG DỰA VÀO NGƯỠNG THỜI GIAN TRÊN ĐIỆN TOÁN ĐÁM MÂY

Nguyễn Xuân Phi[+], **Lê Ngọc Hiếu**[+], **Trần Công Hùng**[+]
[+] Học Viện Công Nghệ Bưu Chính Viễn Thông Cơ Sở Thành Phố Hồ Chí Minh

Abstract**:** *Nhằm nâng cao hiệu suất phục vụ của các dịch vụ điện toán đám mây thì việc quản lý hiệu quả tài nguyên điện toán bao gồm: phân bổ tài nguyên, đáp ứng tài nguyên, kết nối tới tài nguyên, khám phá tài nguyên chưa sử dụng, ánh xạ các tài nguyên tương ứng, mô hình hóa tài nguyên, cung cấp tài nguyên và lập kế hoạch sử dụng các tài nguyên, là các vấn đề cấp bách. Trong đó, việc lập kế hoạch cho việc sử dụng tài nguyên dựa trên các kết nối theo thời gian, như thời gian đáp ứng của dịch vụ là rất quan trọng. Từ việc nghiên cứu thời gian đáp ứng có thể đưa ra giải pháp cho việc phân bố, cân bằng tải của các tài nguyên một cách hiệu quả. Đây là một trong những hướng nghiên cứu còn nhiều triển vọng giúp cho công nghệ đám mây ngày một hoàn thiện và tiến bộ hơn. Vì vậy, bài báo này đề xuất một thuật toán cân bằng tải nhằm giảm thời gian đáp ứng trên điện toán đám mây, ý tưởng 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.*

Keywords**: C**ân bằng tải, điện toán đám mây, thời gian đáp ứng, thời gian đáp ứng dự báo…

I. GIỚI THIỆU

Điện toán đám mây giúp chúng ta chia sẻ dữ liệu và cung cấp thêm nhiều tài nguyên cho người dùng. Người dùng chỉ cần trả tiền cho những gì họ sử dụng. Vì vậy, điện toán đám mây lưu trữ dữ liệu và phân phối tài nguyên trong một môi trường mở. Lượng dữ liệu được lưu trữ trong đám mây đang gia tăng nhanh chóng trên môi trường mở này. Do đó, cân bằng tải là thách thức lớn nhất đối với điện toán đám mây. Cân bằng tải giúp phân phối tải thông qua các nút mạng để đảm bảo rằng không có nút nào bị quá tải. Điều này tối ưu hóa các tài nguyên, cải thiện hiệu năng hệ thống. Nhiều thuật toán đã được đề xuất để cân bằng tải và tối ưu hóa tài nguyên. Có nhiều loại tải được sử dụng trên điện toán đám mây như: bộ nhớ, CPU và tải trên mạng. Cân bằng tải được coi là một quá trình tìm ra các nút mạng quá tải và do đó chuyển sang các nút khác đang tải ít hoặc không tải. Trong môi trường đám mây [1], cân bằng tải yêu cầu phân bổ lại tất cả các tải đang hoạt động giữa tất cả các nút, cân bằng tải cho phép đám mây đạt được phân bổ nguồn lực tốt nhất, linh hoạt, có thể mở rộng để tránh tắc nghẽn nhằm cải thiện năng suất và tối đa hóa việc sử dụng đám mây. Một vấn đề quan trọng khác khi xây dựng thuật toán cân bằng tải là lựa chọn các nút trong đó có các loại tải khác nhau (mức độ sử dụng CPU, dung lượng bộ nhớ, thông lượng) để tổng hợp tính toán tải tổng thể. Một trong những biểu hiện rõ nhất là dữ liệu web, nếu không có cân bằng tải, người dùng truy cập máy chủ web sẽ gặp sự cố quá tải, dữ liệu tải xuống chậm, thời gian chờ hoặc thời gian đáp ứng dài.

Phương pháp làm giảm thời gian đáp của dịch vụ đám mây khi người dùng truy cập vào dịch vụ nhằm tìm chiến lược tiết kiệm tài nguyên điện toán và tăng chất lượng dịch vụ cho người dùng, điều này ảnh hưởng trực tiếp đến hoạt động kinh doanh của nhà cung cấp dịch vụ.

Chính vì vậy bài báo này đề xuất một phương pháp nhằm giảm thời gian đáp ứng trên điện toán đám mây. Bài báo bao gồm các phần sau: Phần 1, Giới thiệu; Phần 2, Công trình liên quan; Phần 3, Đề xuất thuật toán; Phần 4, Kết quả mô phỏng; Phần 5, Kết luận

II. CÔNG TRÌNH LIÊN QUAN

Vào năm 2016, Syed Hamid Hussain Madni [2] đã nghiên cứu và đánh giá các kỹ thuật phân phối tài nguyên trên môi trường cloud. Bài báo này cũng chỉ ra tầm quan trọng của việc phân bổ nguồn tài nguyên trên đám mây, phải có chính sách phân bổ tài nguyên, chiến lược và thuật toán nhằm phân phối và chuyển các nguồn tài nguyên nhằm hỗ trợ tốt nhất cho cả nhà cung cấp lẫn

Tác giả liên hệ: Nguyễn Xuân Phi
Email: nguyenxuanphi@gmail.com
Đến tòa soạn: 10/2018, chỉnh sửa: 12/2018, chấp nhận đăng: 12/2018.

người dùng. Theo tài liệu [3], tác giả đã đưa ra một giải pháp tiếp cận cân bằng tải làm việc dựa trên các tiêu chí về QoS. Bài báo này tác giả đã sử dụng thuật toán Load Balanced Resource Scheduling (LBRS) để nâng cao chất lượng phục vụ trên Cloud. Trong đó thời gian đáp ứng (Thời gian gửi các yêu cầu và đợi trả lời trong hàng đợi cho đến khi lần đầu sử dụng CPU) được được xem xét tới khi xây dựng thuật toán phân bổ các request.

Tác giả Agraj Sharma [4], đưa ra giải thuật về cân bằng tải dựa vào thời gian đáp ứng. Ttác giả đưa ra giải thuật giảm tải dựa trên thời gian đáp ứng của server, từ đó đưa ra quyết định yêu cầu tiếp theo sẽ được xử lý bởi server nào (VMs nào). Kết quả thực nghiệm cho thấy mô hình 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.

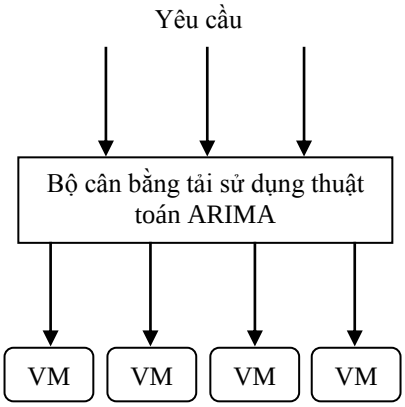
Tác giả Atyaf Dhari [5] đã đề xuất thuật toán lập lịch LDAB để đạt được cân bằng tải và QoS. Thuật toán đề xuất đã giảm thời gian đáp ứng. Thuật toán được so sánh với MaxMin, Shortest Job Firstand Round Robin. Kết quả của LBDA hiệu quả hơn các thuật toán này. Các tác giả trong thuật toán precentation [6], để giảm thời gian đáp ứng và thời gian đáp ứng. Trong bài viết, các tác giả đã đề xuất thuật toán TMA để giảm thời gian phản ứng trên điện toán đám mây dựa trên việc sửa đổi Thuật toán Throttled. Họ đã mô phỏng thuật toán được đề xuất bằng công cụ mô phỏng CloudAnalyts và thuật toán này đã cải thiện thời gian đáp ứng và thời gian xử lý của trung tâm dữ liệu đám mây.

Thời gian đáp ứng trên điện toán đám mây rất quan tâm đến [9], [10], [11], [12], [13], [14]. Với khả năng sử dụng và hiệu suất như vậy, điện toán đám mây đã trở thành một xu hướng tất yếu. Trong tương lai, sự gia tăng về số lượng người dùng đám mây yêu cầu các nhà cung cấp dịch vụ phải đáp ứng nhu cầu của người dùng với thời gian đáp ứng tối thiểu. Do đó, các phương pháp cân bằng tải trong điện toán đám mây ngày càng được phát triển, khi số lượng máy chủ hoặc cấu hình máy chủ tăng chỉ là phương pháp tạm thời. Sử dụng hiệu quả tài nguyên trên "đám mây" là điều cần thiết. Đây cũng là một thách thức lớn trong lĩnh vực điện toán đám mây. Để đáp ứng các yêu cầu trên, việc thiết lập một thuật toán cân bằng tải hiệu quả và cách sử dụng tài nguyên một cách hợp lý là mục tiêu mà điện toán đám mây muốn đạt được

III. ĐỀ XUẤT THUẬT TOÁN

Chúng tôi đề xuất một thuật toán cân bằng tải để giảm thời gian đáp ứng trên điện toán đám mây dựa vào thuật toán dự báo ARIMA [15] để dự báo thời gian đáp ứng, giúp phân bổ hiệu quả các yêu cầu.

1. Mô hình nghiên cứu



Hình 1. Mô hình cân bằng tải sử dụng thuật toán ARIMA

Trong mô hình nghiên cứu thì bộ cân bằng tải sẽ chạy thuật toán cân bằng tải như Hình 1, bộ cân bằng tải có danh sách các máy ảo và các dịch vụ mà cloud cung cấp. Bộ cân bằng tải biết trước các dịch vụ chạy trên máy ảo nào và có thể phân bổ dịch vụ mới trên một máy ảo mới theo yêu cầu. Trong mô hình này, sử dụng một tham số được gọi là ngưỡng thời gian, dựa vào ngưỡng này, mà biết trước được thời gian đáp ứng tiếp theo. Điều kiện là máy ảo có thời gian đáp ứng dự đoán nhỏ hơn ngưỡng được tính toán từ thuật toán ARIMA [15], và việc đối xử với các máy ảo là như nhau trong việc phân bổ các yêu cầu. Nếu không có máy ảo nào trong tập máy ảo đang xét thỏa mãn điều kiện ngưỡng, thì sẽ phân bổ yêu cầu tới các tập máy ảo khác kế tiếp. Máy ảo có thời gian đáp ứng trung bình và dự đoán thấp nhất sẽ được chọn để xử lý yêu cầu tiếp theo. Nếu không có máy ảo nào thỏa điều kiện ngưỡng thì ta có thể xử lý như sau:

+ Nếu có máy ảo không tải, hoặc tập máy ảo không tải, có thể sử dụng các máy này để xử lý các yêu cầu, và tất nhiên phải thỏa ngưỡng.

+ Nếu không có máy ảo không tải, hoặc tập máy ảo không tải, hoặc tất cả đều không thỏa mãn ngưỡng thì ta sẽ phân bổ dịch vụ đang chạy đó tới máy ảo có thời gian đáp ứng dự đoán gần với ngưỡng nhất.

Mô hình này là mô phỏng thuật toán một cách tự nhiên, và lên kế hoạch cho các yêu cầu tiếp theo nhằm không bị mất cân bằng tải. Theo thuật toán này sẽ giảm được các tải liên lạc giữa máy ảo và các nguồn tài nguyên hiện có, vì vậy giảm được băng thông và thông lượng không cần thiết, tăng phục vụ cho yêu cầu người dùng.

2. Thuật toán ARIMA

Theo [15], ARIMA là thuật toán dựa trên thống kê, là thuật toán tự hồi quy tích hợp trung bình trượt (Auto Regression Integrated Moving Average), được phát triển từ mô hình hồi quy

ARMA (Auto Regression Moving Avera). Đây là mô hình phát triển dựa trên số liệu chuỗi thời gian đã biết và dự báo số liệu trong tương lai gần. Dữ liệu thời gian thực hay chuỗi thời gian là một chuỗi các giá trị của một đại lượng nào đó được ghi nhận là thời gian. Bất cứ dữ liệu chuỗi thời gian nào cũng được tạo ra bằng một quá trình ngẫu nhiên. Các giá trị của chuỗi thời gian của đại lượng X được kí hiệu là X1, X2, X3,…,Xt,…, Xn với X là giá trị của X tại thời điểm t.

Việc xác định mô hình ARIMA (hay còn gọi là phương pháp Box-Jenkin) tức là xác định p, d, q trong ARIMA(p,d,q), Do mô hình Box-Jenkins chỉ mô tả chuỗi dừng hoặc những chuỗi đã sai phân hóa, nên mô hình ARIMA(p,d,q) thể hiện những chuỗi dữ liệu không dừng, đã được sai phân (ở đây, d chỉ mức độ sai phân).

Theo [16], phương pháp Box – Jenkins bao gồm các bước chung:

Quy trình dự báo bằng mô hình Arima

- Xác nhận mô hình thử nghiệm: Trước tiên, chúng ta cần nhận dạng mô hình thử nghiệm. Trong đó: d là bậc tích hợp và p,q sẽ được xác định bằng một hàm số chuyên dụng, gọi là Correlogram. Mô hình Arima có thể trình bày theo các dạng khác nhau. Phương pháp xác định mô hình thường được các nhà nghiên cứu thực hiện qua nghiên cứu chiều hướng biến đổi của hàm tương quan toàn phần hay một phần.
- Ước lượng tham số: Mô hình Arima có 2 dạng là ARMA(p,q) và Arima (p,d,q). Với dạng ARMA(p,q) sẽ có d = 0, vì thế ta có thể biến đổi về dạng Arima (p,0,q). Trong quá trình ước lượng tham số, ta cần lưu ý đến cách xác định p q trong mô hình Arima. Để xác định hai số liệu này, người ta sẽ sử dụng đồ thị Correlogram. Cụ thể, p sẽ là bậc của đồ thị AR. Xét từ độ trễ đầu tiên, thanh nào nằm ngoài đường giới hạn và sau độ giảm một cách đáng kể sau một độ trễ thì hệ số tự tương quan riêng phần đó là p. Tương tự, q sẽ là bậc của MA. Để ước lượng tham số, ta cần ước lượng khởi đầu cho các tham số a0, a1, …, ap, b1, …, bq của mô hình dự định ban đầu. Sau đó dựa trên những tham số đã ước lượng, xây dựng những ước lượng sau cùng thông quá một quá trình lặp.
- Kiểm định bằng chuẩn đoán: Sau khi các tham số của mô hình tổng quát đã được xây dựng, người ta sẽ kiểm tra mức độ chính xác và sự phù hợp của mô hình với dữ liệu đã lập. Hãy xem xét phần sai số có phải ngẫu nhiên thuần túy không? Nếu có thì mô hình đó thỏa mãn, nếu không thì ta sẽ phải thực hiện lại các bước trên.
- Dự báo: Ở bước cuối cùng này, khi mô hình phù hợp với dữ liệu đã tìm được, ta sẽ thực hiện dự báo tại thời điểm tiếp theo.

3. Đề xuất giải thuật

Dựa vào dữ liệu đã có về chuỗi thời gian của thời gian đáp ứng (Response Time), chúng tôi sử dụng thuật toán ARIMA để dự báo thời gian đáp ứng tiếp theo, từ đó biết cách phân bố tài nguyên cho các request tiếp theo. Thuật toán đề xuất gồm 3 module như sau:

- *Module 1: 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 = TNew = ARIMA(RT1, RT2,…, RT100)

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

- *Module 2: dự báo thời gian đáp ứng tiếp theo cho từng máy ảo:*

Trong module này sẽ sử dụng thuật toán ARIMA để dự báo thời gian đáp ứng tiếp theo của các máy ảo. Việc dự đoán thời gian đáp ứng tiếp theo dựa vào dữ liệu chuỗi thời gian đáp ứng trong 50 request gần nhất của máy ảo đang xét thông qua hàm getPredictedRT(). Module này đồng thời cung cấp hàm tính toán ra giá trị dự đoán gần nhất của các VM so với ngưỡng đưa vào thông qua hàm AllocateRequestToVM(VM, Request);

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

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

Module này có nhiệm vụ phân bổ các yêu cầu đến các máy ảo đạt được điều kiện ngưỡng thời gian. Nếu một yêu cầu được gửi tới VM đang xét và VM này không tải, thì yêu cầu này sẽ được chuyển tới trực tiếp VM này, và lấy được giá trị thời gian đáp ứng. Nếu thời gian đáp ứng dự đoán của VM đang xét (được tính toán từ module 2) nhỏ hơn thời gian đáp ứng tiếp theo của cloud (tính toán từ module 1) thì yêu cầu này sẽ được xử lý trên VM này. Ngược lại, không có VM nào thỏa điều kiện ngưỡng (thời gian đáp ứng dự đoán của VM không nhỏ hơn thời gian đáp ứng dự đoán của cloud) thì yêu cầu sẽ được phân bổ vào máy ảo có dự báo 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: Tinitial = RT1

Các bước của thuật toán:

- *For each Request in CloudRequests*
- *Tnew = ARIMA(RTi); // Module 1*
- *isLocated = false;*
- *For each VM in VMList*

- *If VM.getPredictedRT() < Tnew*
- *AllocateRequestToVM(VM, Request);// Module 3*
- *isLocated = true;*
- *End If*
- *End For*
- *If (!isLocated)*
- *VM = VMList.getMinDistance(Tnew);// Module 2*
- *AllocateRequestToVM(VM, Request);*
- *End If*

End For

Trong đó, ngưỡng được tính toán chính là thời gian đáp ứng lớn nhất xét trong tập các VM tuy nhiên sẽ hiệu chỉnh một số thay đổi, hoặc đưa vào các hệ số và tham số, tùy thuộc vào kết quả thực nghiệm.

IV. KẾT QUẢ MÔ PHỎNG

Giả lập môi trường cloud sử dụng bộ thư viện CloudSim và lập trình trên ngôn ngữ JAVA; Môi trường giả lập cloud là từ 3 đến 10 máy ảo, và tạo môi trường request ngẫu nhiên tới các dịch vụ trên cloud này. Bao gồm dịch vụ cung cấp máy ảo, dịch vụ cung cấp và đáp ứng người dùng của cloudSim để thử nghiệm. Cài đặt thuật toán ARIMA trên môi trường mô phỏng, và kiểm nghiệm ra kết quả. Tương tự, cài đặt thuật toán của tác giả [4], và so sánh kết quả đạt được giữa 2 thuật toán. Thực nghiệm mô phỏng thuật toán đề xuất được cài đặt trên ngôn ngữ JAVA và sử dụng NETBEAN IDE để chạy thử và hiển thị kết quả bằng STS IDE với framework SPRING BOOT. Môi trường giả lập với bộ thư viện mã nguồn mở CloudSim 4.0 (được cung cấp bởi http://www.cloudbus.org/).

- Môi trường mô phỏng gồm 1 Data center có các thông số sau (Bảng 1)
- Các Request 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 100 → 1000 (Bảng 2)

BẢNG 1. THÔNG SỐ DATACENTER

Datacenter	Host
- 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

BẢNG 2. 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

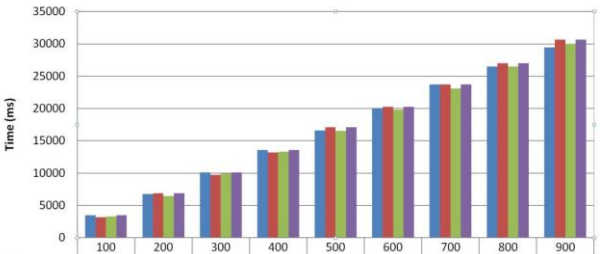
Thuật toán đề xuất được xây dựng bằng cách tạo lớp *ArimaDatacenterBroker*, kế thừa từ đối tượng DatacenterBroker, cập nhật một số phương thức và thuộc tính liên quan đến *PredictedResponseTime* và điều chỉnh các hàm dựng sẵn để khớp với thuật toán được đề xuất:

processResourceCharacteristics (SimEvent ev)
createVmsInDatacenter (int datacenterId)
processVmCreate (SimEvent ev)
processCloudletReturn (SimEvent ev)

Thử nghiệm đám mây với các tham số trên và chạy thuật toán cân bằng tải CloudSim, và chạy thuật toán được đề xuất, cùng một đầu vào, so sánh đầu ra, đặc biệt là tham số thời gian đáp ứng. Thời gian đáp ứng dự đoán của VM cũng như thời gian đáp ứng dự đoán của đám mây với lỗi thấp hơn là hiệu suất của thuật toán càng tốt.

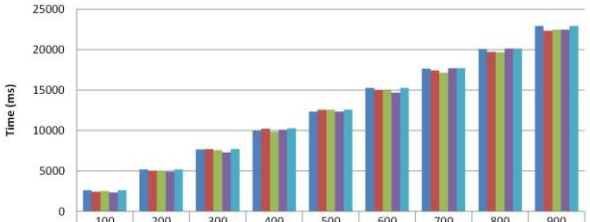
1. Kết quả mô phỏng

Mô phỏng kết quả trên CloudSim tương ứng với trường hợp 3, 4 và 5 VM được xây dựng để đáp ứng các yêu cầu, yêu cầu được khởi tạo với độ dài và kích thước ngẫu nhiên, các yêu cầu là 100, 200, ... đến 900. Thời gian đáp ứng dự đoán được hiển thị trong Hình 2, Hình 3 và Hình 4:



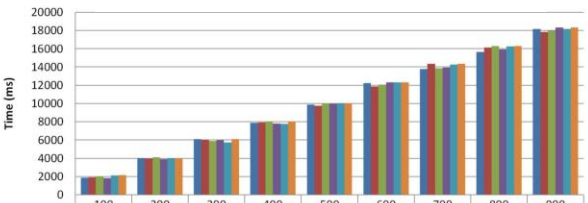
	100	200	300	400	500	600	700	800	900
VM0	3491	6736	10082	13550	16574	20049	23712	26531	29491
VM1	3162	6899	9752	13213	17094	20239	23719	26994	30649
VM2	3268	6451	9925	13327	16505	19784	23078	26484	30055
Threshold	3492	6890	10084	13547	17095	20241	23712	26992	30649

Hình 2. Thời gian đáp ứng dự báo trong trường hợp 3 VMs



	100	200	300	400	500	600	700	800	900
VM0	2635	5194	7654	10047	12324	15291	17650	20066	22941
VM1	2443	4986	7700	10251	12564	14984	17411	19700	22334
VM2	2534	5069	7584	9846	12582	14964	17151	19653	22469
VM3	2353	4928	7291	10110	12328	14685	17704	20149	22468
Threshold	2628	5192	7701	10258	12582	15291	17702	20153	22941

Hình 3. Thời gian đáp ứng dự báo trong trường hợp 4 VMs



	100	200	300	400	500	600	700	800	900
VM0	1878	4020	6087	7896	9889	12223	13771	15629	18144
VM1	1904	3922	6031	7913	9752	11875	14346	16117	17819
VM2	2042	4097	5874	8045	10019	12080	13852	16300	17967
VM3	1828	3913	5987	7820	9934	12311	13939	15929	18336
VM4	2114	3975	5724	7760	10033	12311	14279	16270	18174
Threshold	2133	4028	6089	8048	10033	12315	14347	16299	18338

Hình 4. Thời gian đáp ứng dự báo trong trường hợp 5 VMs

2. Đánh giá thuật toán

Bằng cách so sánh thời gian đáp ứng dự đoán của các máy ảo với ngưỡng tính toán (đối với trường hợp 3 máy ảo, 4 máy ảo và 5 máy ảo), chúng ta có thể thấy rằng khả năng phân phối yêu cầu của thuật toán khá ổn định, thời gian đáp ứng không quá khác với dự báo của đám mây thời gian (nghĩa là ngưỡng). Chúng ta có thể thấy lỗi dự đoán thấp của thuật toán ARIMA, làm cho nó phân bổ các yêu cầu tương ứng với các máy ảo một cách hiệu quả nhất.

Thử nghiệm này chỉ là một mô phỏng của một nhóm máy ảo, chưa kể đến việc mở rộng nhóm máy ảo để giảm tải trong trường hợp cần thiết, giả sử số lượng yêu cầu máy chủ ảo tối đa, nếu nó vượt quá hồ bơi mới được mở rộng. Tuy nhiên, các thí nghiệm mô phỏng với yêu cầu lớn hơn 1000 yêu cầu đòi hỏi máy tính mạnh hơn và bộ xử lý tốt hơn, vì vậy đây là giới hạn của thử nghiệm mô phỏng này.

V. KẾT LUẬN

Một thuật toán mới cho cân bằng tải trên đám mây bằng cách sử dụng mô hình dự đoán thời gian đáp ứng tiếp theo đã được đề xuất và thử nghiệm mô phỏng với một mô hình nhỏ. Thuật toán được đề xuất sử dụng thuật toán ARIMA để cân bằng tải dựa trên thời gian phản hồi. Đặc biệt, thời gian phản hồi được dự đoán chính xác hơn, hiệu quả thuật toán càng cao. Thuật toán này thường tiếp cận và phát triển ý tưởng về dự báo và xử lý chuỗi thời gian, thường là thuật toán ARIMA. Do đó, thuật toán đề xuất có một phương pháp khá mới trong cân bằng tải trong môi trường đám mây, trong khi đạt được một số kết quả mô phỏng khá tích cực, cho thấy hướng phát triển tốt của thuật toán.

Sự phát triển của thuật toán được đề xuất là phép đo và hiệu chuẩn chính xác hơn thời gian dự báo bằng cách kết hợp ARIMA với học máy, học không giám sát hoặc giám sát bằng cách đặt thời gian cao điểm hoặc điểm thấp của đám mây. Sự phát triển của các thuật toán tốt hơn và mô phỏng thử nghiệm sâu hơn, nhiều hơn trên máy tính là mô phỏng quy mô lớn, mạnh mẽ hơn.

Ngoài ra, việc thiết lập các thuật toán trên môi trường đám mây thực tế sẽ cho phép chúng tôi nghiên cứu sâu hơn và chi tiết hơn, vì môi trường đám mây thực tế sẽ tạo ra các vấn đề liên quan đến thời gian phản hồi, do đó điều chỉnh hợp lý và hiệu quả hơn.

TÀI LIỆU THAM KHẢO

[1] Imran Ghani & Naghmeh Niknejad & Seung Ryul Jeong, (2015), "Energy saving in green cloud computing data centers: a review", *Journal of Theoretical and Applied Information Technology*, ISSN: 1992-8645, pages 16-30.

[2] Syed Hamid Hussain Madni, (2016), "Recent advancements in resource allocation techniques for cloud computing environment: a systematic review", Springer Science+Business Media New York 2016, DOI 10.1007/s10586-016-0684-4.

[3] Ritu Kapur, (2015), "A Workload Balanced Approach for Resource Scheduling in Cloud Computing", 978-1-4673-7948-9/15/$31.00 ©2015 IEEE.

[4] Agraj Sharma & Sateesh K. Peddoju, (2014), "Response Time Based Load Balancing in Cloud Computing", 2014

International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp. 1287-1293.

[5] Atyaf Dhari, Khaldun I. Arif (2017), "An Efcient Load Balancing Scheme for Cloud Computing", Indian Journal of Science and Technology, Vol 10(11), DOI: 10.17485/ijst/2017/v10i11/110107.

[6] Bharat Khatavkar, Prabadevi Boopathy (2017), "Efficient WMaxMin Static Algorithm For Load Balancing In Cloud Computation", DOI: 10.1109/IPACT.2017.8245166, International Conference on Innovations in Power and Advanced Computing Technologies [i-PACT2017], Vellore, India.

[7] Jananta Permata Putra, Supeno Mardi Susiki Nugroho, Istas Pratomo (2017), "Live Migration Based on Cloud Computing to Increase Load Balancing", 2017 International Seminar on Intelligent Technology and Its Application, 10.1109/ISITIA.2017.8124096, Publisher: IEEE, 28-29 Aug. Surabaya, Indonesia.

[8] Nguyen Xuan Phi, Cao Trung Tin, Luu Nguyen Ky Thu and 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, DOI : 10.5121/ijcnc.2018.10307,pp 87-98, May 2018.

[9] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov ,C´ esar A. F. De Rose and Rajkumar Buyya (2010), "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software: Practice and Experience (SPE), Volume 41 Number 1, pp.23-50.

[10] Rashmi. K. S, Suma. V, Vaidehi. M (June 2012), "Enhanced Load Balancing Approach to Avoid Deadlocks in Cloud", Special Issue of International Journal of Computer Applications on Advanced Computing and Communication Technologies for HPC Applications – ACCTHPCA, pp.31-35.

Rajwinder Kaur, Pawan Luthra (2014), "Load Balancing in Cloud System using Max Min and Min Min Algorithm", International Journal of Computer Applications Proceedings on National Conference on Emerging Trends in Computer Technology (NCETCT- Number 1), pp.31-34.

[11] Navtej Singh Ghumman, Rajwinder Kaur (2015),"Dynamic Combination of Improved Max-Min and Ant Colony Algorithm for Load Balancing in Cloud System", 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT).

[12] Hafiz Jabr Younis (2015), "Efficient Load Balancing Algorithm in Cloud Computing", Islamic University Gaza Deanery of Post Graduate Studies Faculty Of Information Technology

[13] Shubham Sidana, Neha Tiwari (2016), "NBST Algorithm: A load balancing algorithm in cloud computing", International Conference on Computing, Communication and Automation (ICCCA), pp. 1178 – 1181, IEEE Conference Publications.

[14] Ross Ihaka. Time Series Analysis, Lecture Notes for 475.726, Statistics Department, University of Auckland, 2005.

[15] [10]. Roy Batchelor. Box-Jenkins Analysis. Cass Business School, City of Lodon

Nguyễn Xuân Phi. Sinh năm 1980. Tốt nghiệp Thạc sỹ chuyên ngành Truyền số liệu & Mạng máy tính tại PTIT. Hiện đang là Nghiên cứu sinh Tiến sĩ chuyên ngành Hệ thống

Lê Ngọc Hiếu. Tốt nghiệp Thạc sỹ chuyên ngành Hệ thống Thông tin tại PTIT.

.

Trần Công Hùng. Sinh năm 1961. Ông nhận bằng Tiến sĩ tại trường Đại học Bách khoa Hà Nội năm 2004. Hiện là PGS.TS của Học viện Công nghệ Bưu chính Viễn thông tại thành phố Hồ Chí Minh.