Kỷ yếu Hội nghị Quốc gia lần thứ X về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Đà Nẵng, ngày 17-18/08/2017
DOI: 10.15625/vap.2017.00064

MỘT MÔ HÌNH HỌC TĂNG CƯỜNG CHO VẤN ĐỀ ĐIỀU CHỈNH TỰ ĐỘNG TÀI NGUYÊN TRONG ĐIỆN TOÁN ĐÁM MÂY DỰA TRÊN FUZZY Q-LEARNING

Nguyễn Khắc Chiến[1], Bùi Thanh Khiết[2,3], Hồ Đắc Hưng[3], Nguyễn Hồng Sơn[4], Hồ Đắc Lộc[5]

[1] Bộ môn Toán - Tin học, trường Đại học Cảnh sát nhân dân, TP Hồ Chí Minh
[2] Khoa Khoa học & Kỹ thuật máy tính, Đại học Bách khoa TP HCM
[3] Khoa Kỹ thuật Công nghệ, trường Đại học Thủ Dầu Một
[4] Khoa CNTT2, Học viện Công nghệ Bưu chính Viễn thông
[5] Trường Đại học Công nghệ TP Hồ Chí Minh

nkchienster@gmail.com, khietbt@tdmu.edu.vn, hunghd@ptithcm.edu.vn, ngson@ptithcm.edu.vn, hdloc@hcmhutech.edu.vn

TÓM TẮT: *Điện toán đám mây đã trở thành một trong những xu hướng nổi bật trong lĩnh vực Công nghệ thông tin. Trong đó, vấn đề điều chỉnh tự động tài nguyên là một thách thức lớn, ảnh hưởng lớn đến việc khai thác hiệu quả tài nguyên được nhiều nhà nghiên cứu quan tâm trong thời gian gần đây. Hầu hết các giải thuật điều chỉnh tự động tài nguyên trong điện toán đám mây hiện nay dựa trên cơ chế ngưỡng. Tuy nhiên, các kỹ thuật dựa trên ngưỡng không linh hoạt và khó thích nghi khi có biến động lớn về yêu cầu tài nguyên, đặc biệt là trong môi trường điện toán đám mây. Trong nghiên cứu này, chúng tôi xây dựng mô hình điều chỉnh tự động tài nguyên dựa trên logic mờ và học tăng cường Fuzzy Q-Learning để tăng khả năng thích nghi cũng như khả năng đáp ứng tức thời yêu cầu tài nguyên trong cấp phát/thu hồi tài nguyên tự động theo hướng cân bằng tải cho dịch vụ cơ sở hạ tầng trong điện toán đám mây.*

Từ khóa: *Logic mờ, Q-Learning, Điện toán đám mây, điều chỉnh tự động (auto-scaling).*

I. GIỚI THIỆU

Công nghệ Điện toán đám mây (ĐTĐM) cho phép người sử dụng dịch vụ công nghệ từ một nhà cung cấp mà không cần nhiều kiến thức, kinh nghiệm cũng như không cần quan tâm đến các cơ sở hạ tầng phục vụ công nghệ đó. Trong dịch vụ cơ sở hạ tầng, khả năng điều chỉnh tự động tài nguyên là một trong những đặc trưng quan trọng của ĐTĐM, có thể đáp ứng với sự thay đổi ngữ cảnh bên ngoài hoặc bên trong một cách tự động. Tài nguyên được cung cấp theo nhu cầu, như các ứng dụng di động [1], các hệ Cơ sở dữ liệu [2, 3], các ứng dụng luồng dữ liệu [4], các ứng dụng y khoa thời gian thực [5],... Các ứng dụng trên ĐTĐM đa số có tải biến động do vậy vấn đề quản lý tài nguyên, sử dụng tài nguyên cần độ linh động tùy biến theo nhu cầu. Do đó, tránh được việc cung cấp tài nguyên quá ít hay quá nhiều, trong khi đó vẫn duy trì chất lượng dịch vụ (QoS) ở mức cao là thách thức lớn. Khách hàng chỉ trả tiền cho những tài nguyên đã sử dụng, nhà cung cấp dịch vụ có thể phục vụ cho nhiều khách hàng với cùng một cơ sở hạ tầng. Khái niệm về điều chỉnh tự động tài nguyên đã được định nghĩa từ nhiều quan điểm của các học giả và các nhà cung cấp dịch vụ ĐTĐM trong các bối cảnh đa dạng. Theo đó, hệ thống có thể điều chỉnh tự động mở rộng hay thu hồi tài nguyên để đảm bảo tính sẵn sàng cho các ứng dụng.

Điều chỉnh tự động tài nguyên trên ĐTĐM có thể được xây dựng dựa trên quản lý tự trị theo vòng lặp MAPE-K [6] gồm giám sát, phân tích, lên kế hoạch và thực thi. Trạng thái của hệ thống được giám sát và phân tích từ đó hoạch định kế hoạch cho giai đoạn tiếp theo để vận hành hệ thống. Việc lập kế hoạch tuân theo các luật có thể đơn giản như một chính sách sự kiện - điều kiện - hành động (ECA - Event Condition Action), dễ thực hiện và nhanh chóng tính toán, hoặc có dạng hàm tối ưu một số tính năng nhất định của các hệ thống được quản lý. Kế hoạch dựa vào tri thức có sẵn có để quản lý tự trị. Tri thức có thể được hình thành khi vận hành hệ thống hoặc có thể được phát triển bằng cách rút trích luật từ dữ liệu của quá trình giám sát. Có nhiều hướng tiếp cận, giải quyết bài toán điều chỉnh tự động tài nguyên như phương pháp luật dựa vào ngưỡng tĩnh [7], lý thuyết điều khiển [8], lý thuyết hàng đợi [9] và phân tích chuỗi thời gian,... Theo đó, đa số chiến lược điều chỉnh tự động tài nguyên bị động với các yêu cầu tài nguyên của ứng dụng trên ĐTĐM, do đó có thể không đủ khả năng đáp ứng với những thay đổi đột ngột của tải công việc. Để giải quyết hạn chế này, chiến lược điều chỉnh tự động chủ động (proactive auto-scaling) đưa ra dựa vào khả năng dự đoán nhu cầu sử dụng tài nguyên trong tương lai để nâng cao khả năng đáp ứng yêu cầu của người dùng. Bên cạnh đó, các phương pháp theo hướng dựa trên bộ điều khiển tự tổ chức phù hợp cho bộ điều khiển đám mây [10]. Tuy nhiên, bộ điều khiển đám mây trong thực tế vẫn phụ thuộc vào người sử dụng/vận hành khi xác định các thông số điều khiển cũng như tập luật dùng để điều khiển. Việc xác định tập luật phần lớn dựa vào kinh nghiệm của người sử dụng/vận hành. Do vậy, phương pháp sử dụng cơ chế học tăng cường giúp giải quyết vấn đề trên. Cụ thể, sự kết hợp của điều khiển mờ và học tăng cường là một cơ chế tự thích nghi mạnh mẽ, trong đó điều khiển mờ giúp cho việc lập luận ở một mức độ trừu tượng cao - lập luận giống con người và học tăng cường cho phép điều chỉnh thích nghi với bộ điều khiển [11, 12]. Tuy nhiên, trong nghiên cứu [11, 12] chỉ mới dừng lại ở việc xử lý điều chỉnh tài nguyên cho một công việc tại một thời điểm, trong khi đó hệ thống ĐTĐM lại có rất nhiều ứng dụng khác nhau thực thi cùng lúc tại một thời điểm cũng như chỉ quan tâm đến thời gian đáp ứng và tải của hệ thống.

Trong nghiên cứu này, chúng tôi sẽ xây dựng mô hình điều chỉnh tự động tài nguyên dựa trên logic mờ và học tăng cường, cụ thể là Fuzzy Q-Learning nhằm giúp giải quyết việc điều chỉnh tài nguyên cho nhiều tải công việc khác nhau cùng lúc theo hướng cân bằng tải. Theo đó hệ thống có khả năng: (i) đáp ứng tức thời yêu cầu tài nguyên, (ii) điều chỉnh tự động tài nguyên theo hướng cân bằng tải, (iii) tiết kiệm chi phí sử dụng dịch vụ. Phần còn lại trình bày các nghiên cứu liên quan trong phần II. Phần III trình bày mô hình điều chỉnh tự động tài nguyên. Thuật toán Fuzzy Q-Learning được trình bày trong phần IV. Phần V trình bày kết quả thực nghiệm của mô hình. Phần VI trình bày kết luận.

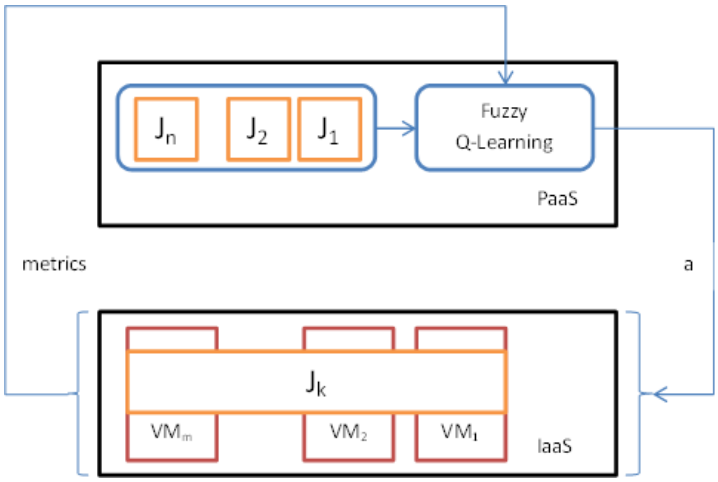
II. CÁC NGHIÊN CỨU LIÊN QUAN

Điều chỉnh tự động tài nguyên có thể chia thành hai hướng sau: (i) Khả năng điều chỉnh theo chiều ngang: là khả năng thêm nhiều thực thể phần cứng hoặc phần mềm, chẳng hạn như máy chủ, để chúng làm việc như một đơn vị logic thống nhất. Nghĩa là việc thêm các đơn vị tài nguyên riêng để cùng thực hiện một công việc. Thay vì một máy chủ, có thể có hai hoặc nhiều máy chủ cùng thực hiện một công việc; (ii) khả năng điều chỉnh theo chiều dọc: tăng cường khả năng của phần cứng hiện có bằng cách thêm tài nguyên như thêm sức mạnh xử lý đến một máy chủ để tăng khả năng xử lý. Thực hiện thông qua việc bổ sung thêm phần cứng như: CPU, RAM, ổ cứng,… Khả năng điều chỉnh theo chiều dọc cung cấp nhiều tài nguyên chia sẻ hơn cho hệ điều hành và các ứng dụng.

Naskos, A. và đồng nghiệp [13] đã đưa ra tổng quan về tính năng đàn hồi trong ĐTĐM. Việc quản lý hiệu quả tính năng đàn hồi trong ĐTĐM là một thách thức. Các hướng nghiên cứu cũng như phân loại các hướng nghiên cứu dựa trên hướng tiếp cận tính toán tự trị, được chia làm 4 giai đoạn của vòng lặp MAPE. Để giảm mức độ sử dụng tài nguyên cho dịch vụ ĐTĐM, Ghobaei-Arani [14] và đồng nghiệp đã đề xuất mô hình tự trị cho việc cấp phát tài nguyên dựa trên mô hình MAPE. Theo đó, mô hình này có khả năng đáp ứng cho các trường hợp thay đổi đột ngột tải công việc,… mà vẫn đảm bảo mục tiêu thỏa thuận ở mức dịch vụ (SLA). Qu, C. và đồng nghiệp [15] đưa ra kiến trúc của ứng dụng điều chỉnh tự động tài nguyên web, gồm 3 loại chính: single, multier, SOA. Từ đó có những giải pháp thích hợp để xây dựng bộ điều khiển Auto-scaler. Sun, Y. và đồng nghiệp [16] đã đưa ra mô hình ROAR để đơn giản hóa, tối ưu và tự động hóa việc cấp phát tài nguyên đảm bảo mục tiêu QoS cho ứng dụng web. Đề xuất trên gọi là GROWL (Generic Rescource Opimization for Web aplication Language) để định nghĩa kế hoạch kiểm tra tải và yêu cầu QoS. Tiêu chuẩn ứng dụng điều chỉnh được sử dụng để đánh giá hệ thống. Tải đầu vào của hệ thống có thể được tạo ra thông qua phần mềm giả lập hoặc được thu thập từ dữ liệu thực tế. Sahni và đồng nghiệp [17] đưa ra thuật toán điều chỉnh tự động nhận biết không đồng nhất thích nghi với sự thay đổi tải mà vẫn đảm bảo được QoS. Đề xuất sử dụng phương pháp online profiling của tài nguyên ĐTĐM và lịch sử tải để ước lượng yêu cầu tài nguyên trong tương lai. Sau đó tìm giải pháp dựa trên heuristic. Chiến lược greedy heuristic được áp dụng vào thuật toán điều chỉnh tự động thích nghi cho phép điều chỉnh đa dạng với chi phí tối ưu mà vẫn đảm bảo QoS trong môi trường ĐTĐM. Thông qua thực nghiệm trên các mẫu web mô tả và các mẫu tải công việc khoa học cho thấy hệ thống có khả năng mở rộng và thu hồi tài nguyên để đáp ứng yêu cầu tải cũng như yêu cầu về hiệu quả khai thác hệ thống. Đối với hệ thống dịch vụ ĐTĐM cung cấp tài nguyên thực thi các công việc chứa các tác vụ phụ thuộc lẫn nhau với độ ưu tiên và cấu trúc không biết trước xuất hiện trong thời gian thực thi hệ thống. De Coninck, E. và đồng nghiệp [18] đề xuất áp dụng hướng mô hình gồm tri thức trên tải công việc, nhu cầu tài nguyên và hành vi mở rộng để giải quyết vấn đề. Thông tin được thu thập thông qua hệ thống giám sát và giả lập kịch bản dạng 'what - if'. Mô hình tri thức tạo hiệu quả trong việc lên kế hoạch và điều phối tài nguyên cũng như giúp nhà cung cấp dịch vụ ĐTĐM đánh giá nhu cầu cơ sở hạ tầng. Arabnejad, H. và đồng nghiệp đã triển khai auto-scaling trên hệ thống OpenStack [12]. Theo đó, điều chỉnh tự động phân phối tài nguyên động mà không cần đến tri thức ban đầu dựa trên thuật toán logic mờ để học và phân phối tài nguyên tại thời điểm thực thi. Đề xuất dựa trên nền tảng Heat Orchestration template có thể quản lý nhiều tình huống biến động tải khác nhau mà vẫn đảm bảo tài nguyên theo nhu cầu và giảm chi phí sử dụng cơ sở hạ tầng, chi phí quản lý. Tuy nhiên, đề xuất chỉ áp dụng phân phối tài nguyên cho một công việc tại một thời điểm. Grimaldi, D. và đồng nghiệp đưa ra mô hình logic mờ dựa trên thông tin không đồng nhất cho bài toán điều phối tài nguyên điều chỉnh theo chiều ngang trên đám mây công cộng [19]. Đề xuất dựa trên tối ưu hóa thông tin phản hồi nhờ vào logic mờ để tự điều chỉnh tham số trên môi trường có biến động theo thời gian lớn và khó dự đoán trong ĐTĐM.

III. MÔ HÌNH ĐIỀU CHỈNH TỰ ĐỘNG TÀI NGUYÊN

ĐTĐM cung cấp tài nguyên cho ứng dụng của khách hàng. Thành phần giám sát các ứng dụng theo vòng lặp tự trị MAPE-K liên tục theo dõi nhằm giám sát mức độ sử dụng tài nguyên. Việc sử dụng tài nguyên phải đáp ứng các mục tiêu về SLA. Bộ giám sát thu thập dữ liệu về tình trạng sử dụng tài nguyên và chuyển đến bộ phận điều chỉnh tự động (auto-scaler) được tạo thành từ các thành phần điều khiển mờ và thành phần học tăng cường. Bộ auto-scaler đưa ra các lệnh nhằm điều chỉnh tự động tài nguyên thích hợp cho từng yêu cầu của công việc. Cụ thể, bộ auto-scaler sẽ đưa ra quyết định tăng/giảm số lượng máy ảo phục công việc cụ thể. Thành phần học tăng cường được sử dụng trong quá trình ra quyết định. Ban đầu, thành phần học tăng cường này không cần tri thức (ở đây có nghĩa là luật), theo thời gian tri thức sẽ được tích lũy qua những lần ra quyết định (Thành phần học tăng cường được cập nhật tri thức qua mỗi vòng lặp MAPE). Thành phần điều khiển mờ đảm nhận biểu diễn tri thức theo dạng tập luật. Từ mức độ sử dụng tài nguyên của các công việc tại một thời điểm cụ thể, thành phần điều khiển mờ suy luận ra trạng thái của cụm máy ảo phục vụ cho công việc đó bằng biến ngôn ngữ S. Tri thức sẽ được biểu diễn đưới dạng tập luật [IF … THEN], ví dụ: ***[IF** Trạng thái cụm máy ảo phục vụ công việc J có trạng thái là S **THEN** Số lượng máy ảo cần được điều chỉnh là a]*.



Hình 1. Kiến trúc hệ thống

Kiến trúc hệ thống được đề xuất trong bài báo này như Hình 1.

Giả sử ta có n ứng dụng $= \{j_1, j_2, \dots, j_n\}$. Mỗi ứng dụng được thực thi trên một cụm k máy ảo đồng nhất. Mỗi máy ảo được biểu diễn đưới dạng vector $v_j^i = \{c_j^i, r_j^i, d_j^i\} \forall\, i \in [1, k]$, trong đó c_j^i là thông số CPU đã được sử dụng, r_j^i là lượng bộ nhớ RAM đã được sử dụng, d_j^i là lượng ổ đĩa đã được sử dụng của máy ảo thứ i trong cụm thứ j. Giả sử dữ liệu máy ảo được thu thập dưới dạng phần trăm. Tải cho mỗi máy ảo i trong cụm thứ j là w_j^i được tính theo công thức sau:

$$w_j^i = \lambda_1 c_j^i + \lambda_2 r_j^i + \lambda_3 d_j^i, \quad (1)$$

trong đó λ_1, λ_2 và $\lambda_3 > 0$ là các trọng số thỏa mãn điều kiện $(\lambda_1 + \lambda_2 + \lambda_3 = 1)$.

Mức độ cân bằng tải của k máy ảo dành cho ứng dụng j được tính toán dựa trên phương của tải trong từng cụm máy ảo. Nếu phương sai càng gần 0 có nghĩa hệ thống đang đạt mức cân bằng cao và ngược lại nếu phương sai gần bằng 1 thì hệ thống đang mất cân bằng tải.

$$l_j = \frac{\sum_{i=1}^{k} (w_j^i - \overline{w^j})^2}{k} \in [0,1], \quad (2)$$

Mức độ cân bằng tải của k máy ảo dành cho ứng dụng j được mờ hóa thành biến ngôn ngữ theo logic mờ $S = \{S_1, S_2, .., S_P\}$ để diễn tả trạng thái của tải. Tương ứng với từng loại trạng thái của tải, hệ thống sẽ quyết định xem cần thêm vào hay giảm bớt số lượng máy ảo phục vụ ứng dụng đó, số lượng máy ảo được thêm hoặc thu hồi trong A= $\{a_1, a_2, .., a_e\} \forall\, a \in N$

Chi phí sử dụng máy ảo của ứng dụng j sử dụng trong thời gian t được tính như sau:

$$U_i(t) = 1 - \frac{vm_i(t)}{vm_{max}}, \quad (3)$$

trong đó, $vm_i(t)$ là số lượng máy ảo thuê trong thời gian t của ứng dụng j, vm_{max} là số lượng máy ảo tối đa cho ứng dụng j.

A. Thành phần điều khiển mờ

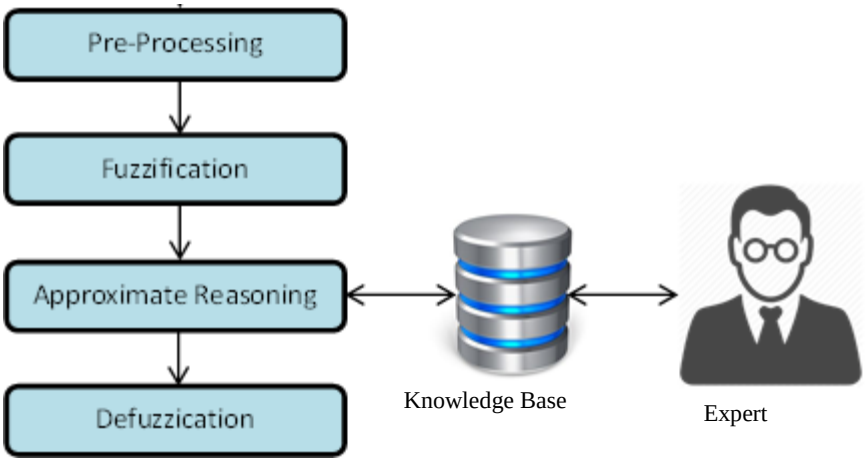
Suy luận mờ là một quá trình ánh xạ một tập giá trị điều khiển đầu vào thành giá trị điều khiển đầu ra thông qua các tập luật mờ (Hình 2). Thành phần điều khiển mờ thường được áp dụng vào vấn đề khó biểu diễn bằng mô hình toán học rõ. Thành phần điều khiển mờ gồm:

- **Cơ sở tri thức (Knowledge Base)**: gồm các luật được cấu hình động theo tham số ngưỡng nhằm phù hợp với nhiều điều kiện của hệ thống.

- **Quá trình mờ hóa (Fuzzification)**: nhằm làm mờ tập dữ liệu đầu vào phục vụ cho quá trình lập luận xấp xỉ. Để quá trình mờ hóa đạt hiệu quả cao cần phải tiền xử lý dữ liệu.

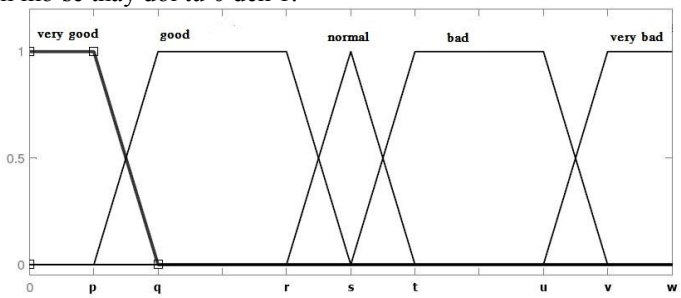
- **Quá trình lập luận xấp xỉ (Approximate Reasoning)**: dựa trên các luật cung cấp bởi cơ sở tri thức, quá trình này đưa ra các kết luận về trạng thái hệ thống.

- **Quá trình khử mờ (Defuzzication)**: đưa ra hành động hợp lý với từng trạng thái cụ thể của ứng dụng.



Hình 2. Thành phần điều khiển mờ

Giả sử để mô tả các giá trị của mức độ cân bằng tải trong việc sử dụng tài nguyên của một ứng dụng cụ thể tại một thời điểm được biểu diễn bằng năm trạng thái: *rất tốt (very good), tốt (good), bình thường (normal), xấu (bad), rất xấu (very bad)* (Hình 3). Những giá trị của tham số đầu vào sẽ quyết định giá trị của biến mờ và phụ thuộc vào trạng thái hệ thống mà giá trị của biến mờ sẽ thay đổi từ 0 đến 1.



Hình 3. Đồ thị đánh giá mức cân bằng tải cụm máy ảo đang thực thi ứng dụng

Công thức mờ hóa với giá trị very good:

$$\mu_{verygood}(l) = \begin{cases} 1 \text{ nếu } l < p, \\ \frac{q-l}{q-p} \text{ nếu } p \le l \le q, \\ 0 \text{ nếu } l > q. \end{cases} \quad (4)$$

Công thức mờ hóa với giá trị good:

$$\mu_{good}(l) = \begin{cases} 0 \text{ nếu } l < p \text{ hoặc } l > s, \\ \frac{l-p}{q-p} \text{ nếu } p \le l \le q, \\ 1 \text{ nếu } l > q \text{ và } l < r, \\ \frac{s-l}{s-r} \text{ nếu } r \le l \le s. \end{cases} \quad (5)$$

Công thức mờ hóa với giá trị normal:

$$\mu_{normal}(l) = \begin{cases} 0 \text{ nếu } l < r \text{ hoặc } l > t, \\ \frac{l-r}{s-r} \text{ nếu } r \le l \le s, \\ \frac{t-l}{t-s} \text{ nếu } s \le l \le t. \end{cases} \quad (6)$$

Công thức mờ hóa với giá trị bad:

$$\mu_{bad}(l) = \begin{cases} 0 \text{ nếu } l < s \text{ hoặc } l > v, \\ \frac{l-s}{t-s} \text{ nếu } s \le l \le t, \\ 1 \text{ nếu } t < l < u, \\ \frac{v-l}{v-u} \text{ nếu } u \le l \le v. \end{cases} \quad (7)$$

Công thức mờ hóa với giá trị very bad:

$$\mu_{verybad}(l) = \begin{cases} 1 \, nếu \, l > v, \\ \frac{v-l}{v-u} \, nếu \, u \le l \le v, \\ 0 \, nếu \, l < u. \end{cases} \quad (8)$$

Quá trình khử mờ sẽ đưa ra hành động thêm hoặc bớt tài nguyên phục vụ ứng dụng. Giá trị đầu ra được tính toán theo công thức sau:

$$y(x) = \sum_i^N \mu_i(x) * a_i \quad (9)$$

trong đó, N là số luật, $\mu_i(x)$ là mức độ/xác suất luật i của giá trị đầu vào x và a_i là hành động tương ứng với luật đó.

B. Thành phần học tăng cường

Thành phần học được dựa trên kỹ thuật Q-Learning. Tại mỗi vòng lặp, bộ điều khiển sẽ đưa ra một hành động a được dựa vào trạng thái s được ký hiệu $Q(s,a)$. Thành phần điều khiển sẽ chọn hành động mà có giá trị phần thưởng (reward) tốt trong quá khứ.

$$R_t = r_{t+1} + \gamma r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (10)$$

trong đó, γ là hệ số thể hiện mức độ quan trọng của giá trị reward trong tương lai. Mỗi phần thưởng được tính toán dựa vào chi phí sử dụng máy ảo như sau:

$$r_t = U(t) - U(t-1) \quad (11)$$

Chính sách $\pi(s,a)$ là xác suất chọn hành động a từ trạng thái s, được tính theo công thức sau:

$$Q^{\pi} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\} \quad (12)$$

trong đó, $E_{\pi}\{.\}$ là hàm kỳ vọng của chính sách π. Khi nào chính sách thích hợp được tìm thấy, khi đó vấn đề học tăng cường được giải quyết. Q-learning là một kỹ thuật không cần bất kỳ chính sách cụ thể cho trước.

Giá trị $Q(s,a)$ được tính như sau:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \eta[r_{t+1} + \gamma \, max_a \, Q(s_{t+1},a_t) - Q(s_t,a_t)] \quad (13)$$

trong đó, η là tỷ lệ học. Chính sách thích hợp chọn hành động ngẫu nhiên với một xác suất ε hoặc chọn hành động có giá trị hàm Q lớn nhất cho trạng thái hiện tại với xác suất $1-\varepsilon$, hành động được chọn như sau:

$$a(s) = arg \, max_k \, Q(s,k). \quad (14)$$

C. Thuật toán điều chỉnh tự động tài nguyên

Thuật toán: Autosacling Fuzzy Q-learning

Input: η, γ

Output: a

1. Khởi tạo gía trị q-values: $q[i,k] = 0$ cho từng ứng dụng $J = \{j_1, j_2, \dots, j_n\}$
2. Sinh luật điều khiển mờ
3. Tính giá trị đầu ra của bộ điều khiển cho các ứng dụng J tại thời điểm t
4. Xấp xỉ giá trị Q cho từng ứng dụng ở trạng thái hiện tại $s(t)$ và hành động a:

$$Q(s(t),a) = \sum_i^N \mu_i(s(t)) * q[i,a_i]$$

5. Triển khai hành động a xuống cơ sở hạ tầng phục vụ ứng dụng j_i
6. Tính giá các trị tại thời điểm $t+1$:

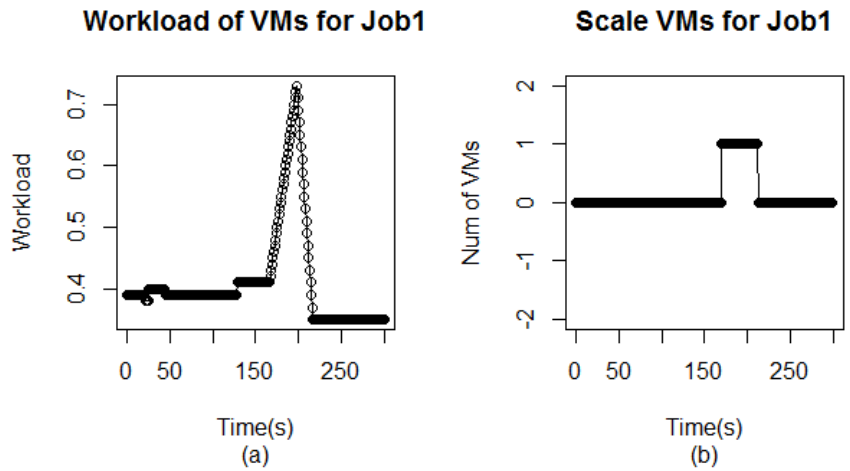
$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \eta\left[r_{t+1} + \gamma \max_a Q(s_{t+1},a_t) - Q(s_t,a_t)\right]$$

7. Lặp lại tiến trình cho trạng thái mới của ứng dụng tới khi nào hội tụ giá trị q-value

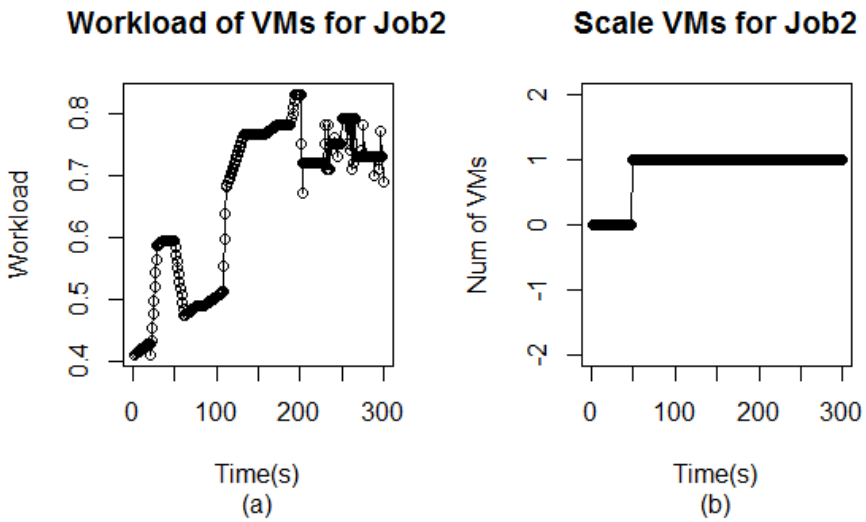
IV. KẾT QUẢ THỰC NGHIỆM

Trong phần này, các thực nghiệm mô phỏng thực hiện để đánh giá khả năng đáp ứng tức thời yêu cầu điều chỉnh tự động tài nguyên theo hướng cân bằng tải cũng như khả năng tiết kiệm chi phí sử dụng dịch vụ. Các thực nghiệm được thực hiện trên một mô phỏng tải theo sự kiện rời rạc có thể được sử dụng để tái tạo môi trường giống nhau. Để đánh giá khả năng đáp ứng của hệ thống, chúng tôi sử dụng 5 mẫu tải công việc điển hình cho dịch vụ web [14]. Theo đó, tải thể hiện lưu lượng truy cập internet thực với yêu cầu đáp ứng khác nhau gồm: (i) yêu cầu tăng đột ngột từ thấp lên cao - Job 1, (ii) yêu cầu từ thấp tăng lên cao rồi xuống thấp - Job 2, (iii) yêu cầu tăng/giảm có biên độ lớn với tốc độ thay đổi chậm - Job 3, (iv) yêu cầu tăng/giảm có biên độ lớn với tốc độ thay đổi nhanh - Job 4, (v) yêu cầu tăng/giảm

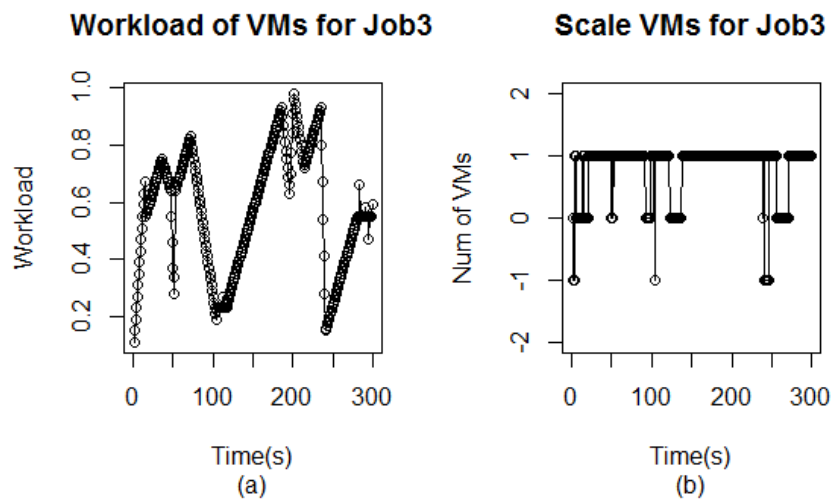
biến động chậm - Job 5. Mô hình trên được thực nghiệm trên máy tính Core i3, 8 GB RAM, 500 GB HDD, hệ điều hành Windows 10.



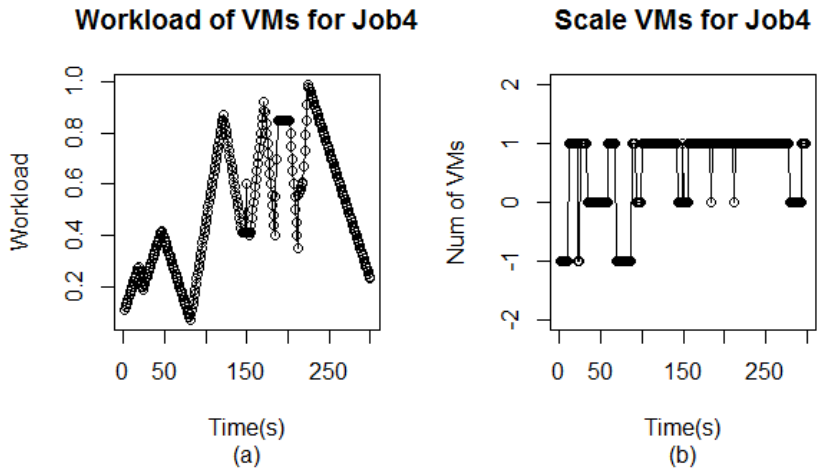
Hình 4. Yêu cầu tăng đột ngột từ thấp lên cao - Job 1



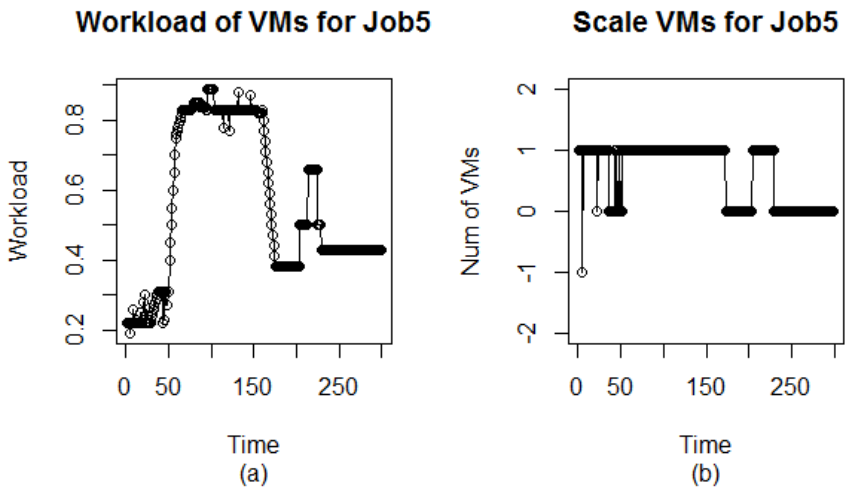
Hình 5. Yêu cầu từ thấp tăng lên cao rồi xuống thấp - Job 2



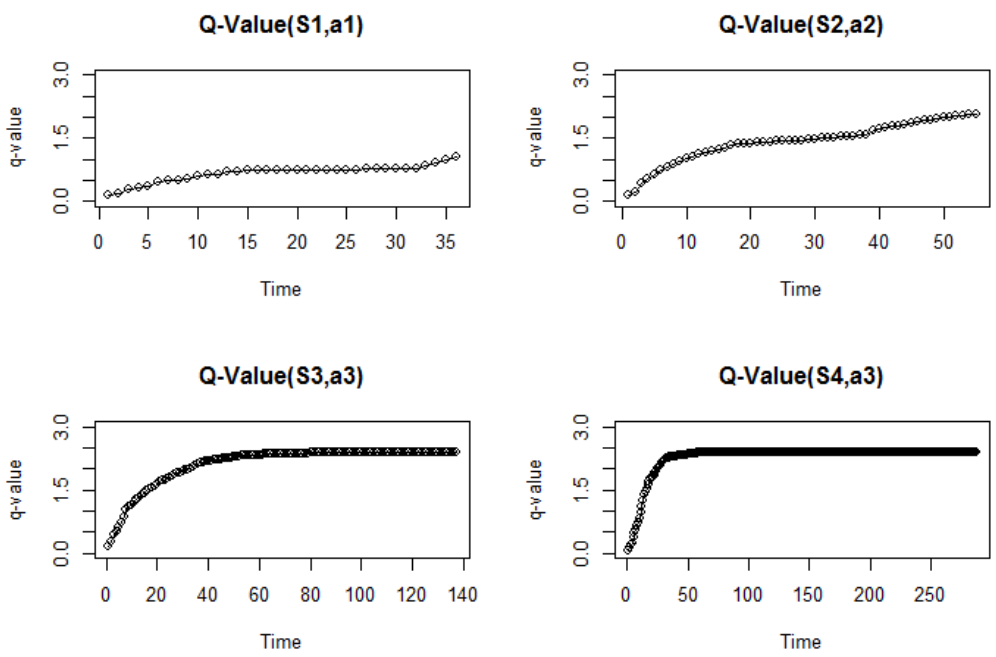
Hình 6. Yêu cầu tăng/giảm có biên độ lớn với tốc độ thay đổi chậm - Job 3



Hình 7. Yêu cầu tăng/giảm có biên độ lớn với tốc độ thay đổi nhanh - Job 4



Hình 8. Yêu cầu tăng/giảm biến động chậm - Job 5

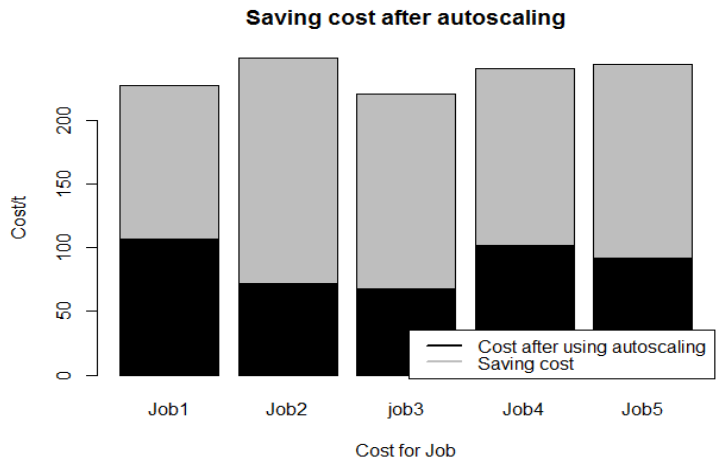


Hình 9. Biểu đồ giá trị Q-Value

Tương ứng với từng thời điểm t, bộ điều khiển autoscaler sẽ điều khiển việc tăng/giảm số lượng máy ảo cho các công việc. Số lượng máy ảo sẽ được tăng/giảm trong [-2,-1,0,+1,+2], kết quả được biểu diễn trong *Hình 4-8*. Theo đó ta có thể thấy, khả năng đáp ứng sự biến động của tải của từng công việc gần như là tức thời ví dụ như job1 có tải tăng giảm đột ngột trong quãng thời gian t từ 170 đến 200 tương ứng bộ autoscaler điều khiển tăng số lượng máy ảo lên.

Giá trị tải được tính theo công thức (1) với các trọng số được lấy trong thực nghiệm này là $\lambda_1 = \lambda_2 = \lambda_3 = \frac{1}{3}$, mức độ cân bằng tải của hệ thống được tính theo công thức (2) và được mờ hóa để cho ra trạng thài S, Q-Value là giá trị biểu diễn mức độ chọn hành động a khi hệ thống ở trạng thái S. Ở đây chúng tôi có 5 trạng thái cho hệ thống và 5 hình động tương ứng với 5 trạng thái đó. Tại thời điểm t cụ thể, giá trị Q-Value được tính toán và cập nhật lại. Đây thể hiện khả năng đáp ứng của phương pháp học tăng cường.

Trong Hình 9 cho thấy giá trị Q-Value là thông số quan trọng của thuật toán Q-Learning. Thuật toán cho thấy một số giá trị Q-Value hội tụ nhanh khi thực nghiệm.



Hình 10. Biểu đồ chi phí sử dụng dịch vụ cho các job trong khoảng thời gian t = 300, chi phí cho 1 VM là 0.1 đơn vị chi phí

Trong *Hình 10* biểu diễn chi phí tiết kiệm khi sử dụng bộ điều khiển autoscaler. Theo đó, chi phí được tiết kiệm khá nhiều so với không sử dụng bộ điều khiển autoscaler. Theo đó, đối với Job1 chi phí khi sử khiển autoscaler chỉ có hơn 100 trong khi không sử dụng bộ autoscaler là hơn 200.

V. KẾT LUẬN

Trong nghiên cứu này, chúng tôi xây dựng mô hình tự động điều chỉnh tài nguyên dựa trên logic mờ và học tăng cường Fuzzy Q-Learning để tăng khả năng đáp ứng tức thời yêu cầu tài nguyên trong cấp phát/thu hồi tài nguyên tự động theo hướng cân bằng tải. Bên cạnh đó, bộ điều khiển autoscaler còn cho thấy hiệu quả về mặt chi phí. Tại một thời điểm bộ autoscaler có khả năng điều khiển việc tăng/giảm máy ảo cho nhiều công việc với những tình trạng khác nhau. Thuật toán không cần thiết lập các giá trị liên quan đến tri thức ban đầu của nhà quản trị ĐTĐM như giá trị Q-Value. Thông qua thực nghiệm, chúng tôi nhận thấy điều này cũng ảnh hưởng đến tốc độ hội tụ của giá trị Q-Value. Do vậy, hướng nghiên cứu sắp tới của chúng tôi sẽ cải thiện tốc độ hội tụ của giá trị Q-Value cũng như vấn đề xác định số lượng máy ảo cần tăng/giảm.

TÀI LIỆU THAM KHẢO

[1]. Bernstein, D., Vidovic, N., and Modi, S.: "A cloud PAAS for high scale, function, and velocity mobile applications-with reference application as the fully connected car", in Editor (Ed.)^(Eds.): 'Book A cloud PAAS for high scale, function, and velocity mobile applications-with reference application as the fully connected car' (IEEE, 2010, edn.), pp. 117-123.

[2]. Agrawal, D., El Abbadi, A., Das, S., and Elmore, A.J.: 'Database scalability, elasticity, and autonomy in the cloud', in Editor (Ed.)^(Eds.): 'Book Database scalability, elasticity, and autonomy in the cloud' (Springer, 2011, edn.), pp. 2-15.

[3]. Huang, C.-W., Shih, C.-C., Hu, W.-H., Lin, B.-T., and Cheng, C.-W.: 'The improvement of auto-scaling mechanism for distributed database-A case study for MongoDB', in Editor (Ed.)^(Eds.): 'Book The improvement of auto-scaling mechanism for distributed database-A case study for MongoDB' (IEEE, 2013, edn.), pp. 1-3.

[4]. Vijayakumar, S., Zhu, Q., and Agrawal, G.: 'Dynamic resource provisioning for data streaming applications in a cloud environment', in Editor (Ed.)^(Eds.): 'Book Dynamic resource provisioning for data streaming applications in a cloud environment' (IEEE, 2010, edn.), pp. 441-448.

[5]. Raveendran, A., Bicer, T., and Agrawal, G.: 'A framework for elastic execution of existing mpi programs', in Editor (Ed.)^(Eds.): 'Book A framework for elastic execution of existing mpi programs' (IEEE, 2011, edn.), pp. 940-947.

[6]. Horn, P.: 'Autonomic computing: IBM\'s Perspective on the State of Information Technology', 2001.

[7]. Jamshidi, P., Ghafari, M., Ahmad, A., and Pahl, C.: 'A framework for classifying and comparing architecture-centric software evolution research', in Editor (Ed.)^(Eds.): 'Book A framework for classifying and comparing architecture-centric software evolution research' (IEEE, 2013, edn.), pp. 305-314.

[8]. Filieri, A., Maggio, M., Angelopoulos, K., D'Ippolito, N., Gerostathopoulos, I., Hempel, A.B., Hoffmann, H., Jamshidi, P., Kalyvianaki, E., and Klein, C.: 'Software engineering meets control theory', in Editor (Ed.)^(Eds.): 'Book Software engineering meets control theory' (IEEE Press, 2015, edn.), pp. 71-82.

[9]. Ardagna, D., Casale, G., Ciavotta, M., Pérez, J.F., and Wang, W.: 'Quality-of-service in cloud computing: modeling techniques and their applications', Journal of Internet Services and Applications, 2014, 5, (1), pp. 11.

[10]. Gambi, A., Pezzè, M., and Toffetti, G.: 'Kriging-based self-adaptive cloud controllers', IEEE Transactions on Services Computing, 2016, 9, (3), pp. 368-381.

[11]. Jamshidi, P., Sharifloo, A. M., Pahl, C., Metzger, A., and Estrada, G.: 'Self-learning cloud controllers: Fuzzy Q-learning for knowledge evolution', in Editor (Ed.)^(Eds.): 'Book Self-learning cloud controllers: Fuzzy Q-learning for knowledge evolution' (IEEE, 2015, edn.), pp. 208-211.

[12]. Arabnejad, H., Jamshidi, P., Estrada, G., El Ioini, N., and Pahl, C.: 'An Auto-Scaling Cloud Controller Using Fuzzy Q-Learning-Implementation in OpenStack', in Editor (Ed.)^(Eds.): 'Book An Auto-Scaling Cloud Controller Using Fuzzy Q-Learning-Implementation in OpenStack' (Springer, 2016, edn.), pp. 152-167.

[13]. Naskos, A., Gounaris, A., and Sioutas, S.: 'Cloud elasticity: a survey': 'Algorithmic Aspects of Cloud Computing' (Springer, 2016), pp. 151-167.

[14]. Ghobaei-Arani, M., Jabbehdari, S., and Pourmina, M.A.: 'An autonomic approach for resource provisioning of cloud services', Cluster Computing, 2016, 19, (3), pp. 1017-1036.

[15]. Qu, C., Calheiros, R. N., and Buyya, R.: 'Auto-scaling Web Applications in Clouds: A Taxonomy and Survey', arXiv preprint arXiv:1609.09224, 2016.

[16]. Sun, Y., White, J., Eade, S., and Schmidt, D. C.: 'ROAR: A QoS-oriented modeling framework for automated cloud resource allocation and optimization', Journal of Systems and Software, 2016, 116, pp. 146-161.

[17]. Sahni, J., and Vidyarthi, D. P.: 'Heterogeneity-aware adaptive auto-scaling heuristic for improved QoS and resource usage in cloud environments', Computing, 2016, pp. 1-31.

[18]. De Coninck, E., Verbelen, T., Vankeirsbilck, B., Bohez, S., Simoens, P., and Dhoedt, B.: 'Dynamic auto-scaling and scheduling of deadline constrained service workloads on IaaS clouds', Journal of Systems and Software, 2016, 118, pp. 101-114.

[19]. Grimaldi, D., Pescape, A., Salvi, A., and Persico, V.: 'A Fuzzy Approach based on Heterogeneous Metrics for Scaling Out Public Clouds', IEEE Transactions on Parallel and Distributed Systems, 2017.

AN REINFORCEMENT LEARNING APPROACH FOR AUTOSCALING IN CLOUD COMPUTING BEASED ON FUZZY Q-LEARNING

Nguyen Khac Chien, Bui Thanh Khiet, Ho Dac Hung, Nguyen Hong Son, Ho Dac Loc

ABSTRACT: *Cloud computing has become one of the most prominent trends in the IT industry. In particular, the problem of auto-scaling (allocation /deallocation) of resources is a great challenge, greatly affecting the efficient exploitation of resources should always be subject to research in recent times. Most of the algorithms for auto-scaling resources in the cloud computing are based on threshold mechanisms. However, threshold-based techniques are not flexible and adaptable when there are major fluctuations in resource requirements, especially in the cloud computing environment. In this paper, we built a model for auto-scaling resources based on fuzzy logic and reinforcement learning - Fuzzy Q-Learning to increase adaptability as well as the ability to immediately respond to resource requirements in dynamic allocation/deallocation of resources in the direction of load balancing.*

Keywords: *Auto-scaling, Fuzzy Logic, Q-Learning, Cloud Computing.*