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

Nguyễn Khắc Chiến

NGHIÊN CỨU GIẢI PHÁP TỰ ĐỘNG ĐIỀU CHỈNH TÀI NGUYÊN HIỆU QUẢ TRONG ĐIỆN TOÁN ĐÁM MÂY

LUẬN ÁN TIẾN SĨ KỸ THUẬT

Hà Nội - 2019

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

Nguyễn Khắc Chiến

NGHIÊN CỨU GIẢI PHÁP TỰ ĐỘNG ĐIỀU CHỈNH TÀI NGUYÊN HIỆU QUẢ TRONG ĐIỆN TOÁN ĐÁM MÂY

CHUYÊN NGÀNH: KỸ THUẬT MÁY TÍNH
MÃ SỐ: 9.48.01.06

LUẬN ÁN TIẾN SĨ KỸ THUẬT

NGƯỜI HƯỚNG DẪN KHOA HỌC:
GS.TSKH. HỒ ĐẮC LỘC
TS. NGUYỄN HỒNG SƠN

Hà Nội - 2019

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các kết quả được viết chung với các tác giả khác đều được sự đồng ý của đồng tác giả trước khi đưa vào luận án. Các kết quả nêu trong luận án là trung thực và chưa từng được công bố trong các công trình nào khác.

Người cam đoan

Nguyễn Khắc Chiến

LỜI CẢM ƠN

Thực hiện luận án tiến sĩ là một thử thách lớn, đòi hỏi sự kiên trì và tập trung cao độ. Tôi thực sự hạnh phúc với kết quả đạt được trong đề tài nghiên cứu của mình. Những kết quả đạt được không chỉ là nỗ lực cá nhân, mà còn có sự hỗ trợ và giúp đỡ của các thầy hướng dẫn, nhà trường, bộ môn, đồng nghiệp và gia đình. Tôi muốn bày tỏ tình cảm của mình đến với họ.

Nghiên cứu sinh xin được bày tỏ lòng biết ơn sâu sắc đến Thầy giáo GS. TSKH. Hồ Đắc Lộc và TS. Nguyễn Hồng Sơn đã tận tình hướng dẫn, giúp đỡ, trang bị phương pháp nghiên cứu, kiến thức khoa học để tôi hoàn thành các nội dung nghiên cứu của luận án.

Nghiên cứu sinh xin bày tỏ lòng biết ơn chân thành tới các thầy, cô của Học viện Công nghệ Bưu chính Viễn thông, các đồng chí thuộc Trường Đại học cảnh sát nhân dân đã đóng góp nhiều ý kiến quý báu giúp tôi hoàn thành các nội dung nghiên cứu của luận án.

Nghiên cứu sinh xin trân trọng cảm ơn Học viện Công nghệ Bưu chính Viễn thông, Khoa Sau đại học là cơ sở đào tạo và đơn vị quản lý, các đồng chí lãnh đạo trường Đại học Cảnh sát nhân dân, cùng các đồng chí trong Bộ môn Toán - Tin học, trường Đại học Cảnh sát nhân dân, nơi tôi đang công tác đã tạo điều kiện thuận lợi, hỗ trợ và giúp đỡ tôi trong suốt quá trình học tập, nghiên cứu thực hiện luận án.

Tôi xin trân trọng cảm ơn các bạn bè người thân và gia đình đã cổ vũ, động viên giúp đỡ, tạo điều kiện cho tôi hoàn thành luận án.

Nghiên cứu sinh

MỤC LỤC

LỜI CAM ĐOAN i
LỜI CẢM ƠN ii
MỤC LỤC iii
DANH MỤC CÁC BẢNG v
DANH MỤC CÁC HÌNH VẼ vi
DANH MỤC CÁC TỪ VIẾT TẮT ix
PHẦN MỞ ĐẦU 1
1. GIỚI THIỆU 1
2. TÍNH CẤP THIẾT CỦA LUẬN ÁN 2
3. MỤC TIÊU CỦA LUẬN ÁN 7
4. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU 7
5. PHƯƠNG PHÁP NGHIÊN CỨU 7
6. CÁC ĐÓNG GÓP CỦA LUẬN ÁN 8
7. BỐ CỤC CỦA LUẬN ÁN 8
CHƯƠNG 1 : TỔNG QUAN VỀ TỰ ĐỘNG ĐIỀU CHỈNH TÀI NGUYÊN TRONG ĐIỆN TOÁN ĐÁM MÂY 10
1.1. CƠ SỞ LÝ THUYẾT 10
1.2. TỔNG QUAN VỀ CÁC CÔNG TRÌNH LIÊN QUAN 27
KẾT LUẬN CHƯƠNG 1: 38
CHƯƠNG 2 : CÁC ĐỀ XUẤT CƠ CHẾ CƠ BẢN CÓ ẢNH HƯỞNG ĐẾN TỰ ĐỘNG ĐIỀU CHỈNH TÀI NGUYÊN TRONG ĐIỆN TOÁN ĐÁM MÂY 39
2.1. ĐẶT VẤN ĐỀ 39
2.2. ĐỀ XUẤT MỘT KỸ THUẬT CÂN BẰNG TẢI ĐỘNG HIỆU QUẢ TRONG ĐIỆN TOÁN ĐÁM MÂY 40
2.3. ĐỀ XUẤT MỘT KỸ THUẬT DI TRÚ HIỆU QUẢ TRONG ĐIỆN TOÁN ĐÁM MÂY 54
2.4. ẢNH HƯỞNG CỦA CÂN BẰNG TẢI ĐẾN CƠ CHẾ TỰ ĐỘNG ĐIỀU CHỈNH TÀI NGUYÊN 62

KẾT LUẬN CHƯƠNG 2:.....69
CHƯƠNG 3 : ĐỀ XUẤT MỘT MÔ HÌNH MẠNG HÀNG ĐỢI CHO HỆ THỐNG ĐIỆN TOÁN ĐÁM MÂY 71
3.1. ĐẶT VẤN ĐỀ.....71
3.2. MÔ HÌNH HÓA ỨNG DỤNG ĐA TẦNG TRÊN ĐIỆN TOÁN ĐÁM MÂY.....74
3.3. THỰC NGHIỆM VÀ ĐÁNH GIÁ 85
KẾT LUẬN CHƯƠNG 3.....88
CHƯƠNG 4 : GIẢI PHÁP TỰ ĐỘNG ĐIỀU CHỈNH TÀI NGUYÊN CHO ỨNG DỤNG ĐA TẦNG TRÊN ĐIỆN TOÁN ĐÁM MÂY 89
4.1. ĐẶT VẤN ĐỀ.....89
4.2. KIẾN TRÚC ỨNG DỤNG ĐA TẦNG 93
4.3. ĐỀ XUẤT MÔ HÌNH TỰ ĐỘNG ĐIỀU CHỈNH.....94
4.4. THỰC NGHIỆM VÀ ĐÁNH GIÁ 110
KẾT LUẬN CHƯƠNG 4.....134
KẾT LUẬN 135
DANH MỤC CÁC CÔNG TRÌNH CÔNG BỐ.....137
TÀI LIỆU THAM KHẢO 139

DANH MỤC CÁC BẢNG

Bảng 2.1: Thuật toán cân bằng tải đề xuất AMLB-New 48
Bảng 2.2: Thiết lập các tham số cho đám mây 49
Bảng 2.3: Các tham số cho VM 50
Bảng 2.4: Thuật toán đề xuất MM-New 58
Bảng 2.5: Thông tin về trung tâm dữ liệu 59
Bảng 2.6: Bảng số liệu máy chủ trong mô phỏng 59
Bảng 2.7: Tiêu thụ năng lượng của hai loại máy chủ 59
Bảng 2.8: Thông tin VM trong sử dụng trong mô phỏng 60
Bảng 2.9: Kết quả một số thông số khi chạy thực nghiệm 60
Bảng 3.1: Thuật toán cân bằng tải Round Robin 77
Bảng 3.2. Bảng mô tả trạng thái 81
Bảng 3.3. Cấu hình cụm VM 86
Bảng 3.4. Khoảng tin cậy 95% của thời gian đợi trung bình của mỗi VM trong từng cụm 87
Bảng 4.1: Thuật toán tự động điều chỉnh (AS AFQL) 108
Bảng 4.2: Cấu hình các VM 110
Bảng 4.3: Các phân hoạch của giá trị tài nguyên khả dụng trung bình 111
Bảng 4.4: Các phân hoạch của giá trị phương sai 111
Bảng 4.5: Các phân hoạch của thời gian đáp ứng 111
Bảng 4.6: Tập luật được xây dựng dựa theo tri thức chuyên gia 112
Bảng 4.7: Tập luật thu được sau khi học xong với bộ tham số ε=0.2, γ=0.8 và η=0.2 124

DANH MỤC CÁC HÌNH VẼ

Hình 0.1: Ví dụ một trường hợp tự động điều chỉnh 3
Hình 1.1: Một dạng hệ thống AS đơn giản 11
Hình 1.2: Phương pháp điều chỉnh trong CC.....16
Hình 1.3: Ví dụ về điều chỉnh theo chiều dọc.....16
Hình 1.4: Ví dụ về điều chỉnh theo chiều ngang.....17
Hình 1.5: Phân loại các kỹ thuật AS 18
Hình 1.6: Ví dụ một ứng dụng web có kiến trúc ba tầng.....23
Hình 2.1: Mô hình cân bằng tải.....41
Hình 2.2: Ảnh hưởng của các chính sách khác nhau đối với việc thực thi các tác vụ [15] 43
Hình 2.3: Ví dụ có 2 VM đang hoạt động như trên 45
Hình 2.4: Thời gian đáp ứng trung bình theo trường hợp 1.....50
Hình 2.5: Thời gian xử lý trung bình theo trường hợp1 50
Hình 2.6: Thời gian đáp ứng trung bình theo trường hợp 2.....51
Hình 2.7: Thời gian xử lý trung bình theo trường hợp 2 51
Hình 2.8: Thời gian đáp ứng trung bình theo trường hợp 3.....52
Hình 2.9: Thời gian xử lý trung bình theo trường hợp 3 52
Hình 2.10: Thời gian đáp ứng trung bình theo trường hợp 4.....53
Hình 2.11: Thời gian xử lý trung bình theo trường hợp 4 53
Hình 2.12:Thời gian đáp ứng trung bình cho các trường hợp lập lịch.....53
Hình 2.13: Thời gian xử lý trung bình cho các trường hợp lập lịch 53
Hình 2.14. Minh hoạ kỹ thuật di trú giúp cân bằng hoạt động của hệ thống [78]. ...54
Hình 2.15: Ví dụ cho trường hợp phải di trú nhiều VM 56
Hình 2.16: Kết quả so sánh hài thuật toán MM-Old và MM-New 61
Hình 2.17: Phấn bố tải công việc đến trong ClarkNet trace 67
Hình 2.18: Số lượng VM được bổ sung khi sử dụng Round Robin 68
Hình 2.19: Số lượng VM được bổ sung khi sử dụng AMLB-New 69

Hình 2.20: Độ lệch Dev(t) theo bộ cân bằng tải sử dụng Round Robin và sử dụng AMLB-New 70
Hình 3.1: Mô hình mạng hàng đợi mở cho ứng dụng đa tầng trên CC 75
Hình 3.2: Mạng hàng đợi mở tổng quát 80
Hình 3.3: Ví dụ mô hình mạng hàng đợi mở cho ứng dụng đa tầng trên CC 84
Hình 3.4: Thời gian đợi trung bình của cụm VM cho thực nghiệm 1 86
Hình 3.5: Thời gian đợi trung bình của các VM cho thực nghiệm 2 87
Hình 3.6: Thời gian đợi trung bình của các VM cho thực nghiệm 3 88
Hình 4.1: Mô hình tương tác giữa tác nhân và hệ thống.....95
Hình 4.2: Cấu trúc cơ bản của hệ suy diễn mờ 101
Hình 4.3: Hàm thành viên mờ cho tài nguyên khả dụng trung bình.....102
Hình 4.4: Hàm thành viên mờ cho phương sai của tài nguyên khả dụng 102
Hình 4.5: Hàm thành viên mờ cho thời gian đáp ứng trung bình 104
Hình 4.6: Bộ tự động điều chỉnh đề xuất 108
Hình 4.7: Kết quả thực nghiệm áp dụng bộ AS với tham số ε=0.2, $\gamma = 0.5$, $\eta = 0.5$. 114
Hình 4.8: Xác định tham số thăm dò/khai thác (ε) 115
Hình 4.9: Kết quả thực nghiệm áp dụng bộ AS với bộ tham số $\varepsilon = 0.2, \gamma = 0.2, \eta = 0.5$.....116
Hình 4.10: Xác định hệ số suy giảm (γ).....117
Hình 4.11: Kết quả thực nghiệm áp dụng bộ AS với bộ tham số $\varepsilon = 0.2, \gamma = 0.8, \eta = 0.2$.....118
Hình 4.12: Xác định tham số tốc độ học (η) 119
Hình 4.13: Sự tiến hóa của các giá trị q với các tham số ε=0.2, γ=0.8 và η=0.2. ..120
Hình 4.14: So sánh dựa vào tài nguyên khả dụng trung bình và phương sai 121
Hình 4.15: Biểu đồ hộp so sánh giữa tài nguyên khả dụng trung bình và phương sai 121
Hình 4.16: So sánh dựa vào thời gian đáp ứng trung bình và chi phí.....122
Hình 4.17: Biểu đồ hộp so sánh giữa thời gian đáp ứng trung bình và chi phí 122

Hình 4.18: So sánh thời gian đáp ứng trung bình đối với bốn bộ tham số khác nhau 123
Hình 4.19: Dữ liệu đầu vào của bộ dữ liệu ClarkNet.....125
Hình 4.20: Kết quả thực hiện điều chỉnh trên bộ dữ liệu ClarkNet 126
Hình 4.21: So sánh tổng chi phí đối với 3 trường hợp trên dữ liệu ClarkNet 127
Hình 4.22: So sánh thời gian đáp ứng trung bình đối với 3 trường hợp trên bộ dữ liệu ClarkNet 127
Hình 4.23: Dữ liệu đầu vào của bộ dữ liệu Wiki 128
Hình 4.24: Kết quả thực hiện điều chỉnh với bộ dữ liệu Wiki.....129
Hình 4.25: So sánh tổng chi phí đối với 3 trường hợp trên dữ liệu Wiki 130
Hình 4.26: So sánh thời gian đáp ứng trung bình đối với 3 trường hợp trên bộ dữ liệu Wiki.....130
Hình 4.27: Dữ liệu đầu vào của bộ dữ liệu sinh ngẫu nhiên.....131
Hình 4.28: Kết quả thực hiện điều chỉnh với bộ dữ liệu sinh ngẫu nhiên 132
Hình 4.29: So sánh tổng chi phí đối với 3 trường hợp trên dữ liệu ngẫu nhiên 133
Hình 4.30: So sánh thời gian đáp ứng trung bình đối với 3 trường hợp trên bộ dữ liệu ngẫu nhiên 133

DANH MỤC CÁC TỪ VIẾT TẮT

KÝ HIỆU	DIỄN GIẢI	
	TIẾNG ANH	TIẾNG VIỆT
AI	Artificial Intelligence	Trí tuệ nhân tạo
AMLB	Active Monitoring Load Balancing	Cân bằng tải giám sát tích cực
AR	Autoregression	Tự hồi quy
ARMA	Autoregressive moving average	Trung bình động tự hồi quy
AS	Auto Scaling	Tự động điều chỉnh
CC	Cloud Computing	Điện toán đám mây
CPU	Control Process Unit	Bộ xử lý trung tâm
ECA	Event Condition Action	Hành động-Điều kiện-Sự kiện
FCFS	First Come First Served	Đến trước phục vụ trước
HTM	Hierarchical Temporal Memory	Bộ nhớ tạm phân cấp
IO	Input/Output	Vào/ra
IoT	Internet of Things	Internet vạn vật
IT	Information Technology	Công nghệ thông tin
LCFS	Last Come First Served	Đến sau phục vụ trước
LT	Lower Threshold	Ngưỡng dưới
MA	Moving Average	Trung bình động
MAPE	Monitor-Analyze-Plan-Execute	Vòng lặp điều khiển MAPE
MDP	Markov Decisson Process	Quá trình quyết định Markov
MI	Millions Instructions	Triệu chỉ thị
MIPS	Milions Instructions Per Second	Triệu chỉ thị trên giây
MISO	Multi input Single Output	Nhiều đầu vào, một đầu ra
MM	Minimization Migration	Di trú ít nhất
MPC	Model Predictive Controllers	Bộ điều khiển dự báo mô hình
MVA	Mean Value Analysis	Phân tích giá trị trung bình

KÝ HIỆU	DIỄN GIẢI	
	TIẾNG ANH	TIẾNG VIỆT
NIST	National Institute of Standards and Technology	Viện nghiên cứu tiêu chuẩn và công nghệ quốc gia Hoa Kỳ
PE	Processing Element	Phần tử xử lý
PM	Physical Machine	Máy chủ vật lý
PS	Processor Sharing	Chia sẻ bộ xử lý
QoS	Quality Of Service	Chất lượng dịch vụ
RAM	Random Access Menmory	Bộ nhớ truy xuất ngẫu nhiên
RC	Random choice	Chọn ngẫu nhiên
SLA	Service Level Agreement	Thỏa thuận mức dịch vụ
SLO	Service Level Objective	Mục tiêu mức dịch vụ
SS	Space-shared	Chính sách chia sẻ không gian
TS	Time-shared	Chính sách chia sẻ thời gian
UT	Upper Threshold	Ngưỡng trên
VM	Virtual Machine	Máy ảo

PHẦN MỞ ĐẦU

1. GIỚI THIỆU

Theo số liệu thống kê, Việt Nam là nước có nhịp độ tăng chi tiêu cho điện toán đám mây (CC) trong giai đoạn 2010-2016 cao nhất (64,4%/năm), cao hơn hẳn mức bình quân của cả khối ASEAN (49,5%). Đến năm 2018, Việt Nam đạt 41/100 điểm và trở thành nước đứng thứ 14 trong bảng xếp hạng về độ phủ dịch vụ CC. Điều này cho thấy mô hình CC đang trở nên phổ biến và bắt đầu chiếm ưu thế hơn so với mô hình công nghệ thông tin (IT) truyền thống. Trong tương lai, việc ứng dụng mô hình này tại Việt Nam được dự đoán sẽ còn tăng mạnh và đa dạng hơn. Việc các tổ chức, doanh nghiệp, và cơ quan nhà nước dần dần đưa các hệ thống IT của mình lên CC là một yếu tố quan trọng để tham gia vào cuộc cách mạng công nghệ 4.0. Khi nói về cách mạng 4.0, thường nói về bốn công nghệ: CC, Internet vạn vật (IoT), dữ liệu lớn (Big Data) và trí tuệ nhân tạo (AI). Trong đó, CC là nền tảng ở dưới cùng, bất kỳ một ứng dụng nào về AI, IoT hay dữ liệu lớn đều cần có hạ tầng ở bên dưới là CC thì mới chạy được. Dịch vụ CC được xem là "móng nhà" trong cuộc cách mạng công nghệ 4.0, là hạ tầng cơ bản cho sự phát triển của cách mạng 4.0 ở Việt Nam trong thời gian tới.

CC đóng vai trò là nền tảng, chuyển thế giới thực thành thế giới số hóa, giúp các tổ chức, doanh nghiệp có thể khai thác, sử dụng một kho dữ liệu khổng lồ, phân tích và đánh giá các chiến lược một cách chính xác và khoa học. CC tham gia vào các quá trình chuẩn hóa sản phẩm để cắt giảm chi phí sản xuất, khai thác lợi thế kinh tế theo quy mô, tiết kiệm thời gian phát triển sản phẩm, cung cấp các phần mềm kinh doanh, quản lý hiện đại,...

CC được Gartner xếp đầu bảng trong các công nghệ chiến lược từ năm 2010. Tuy nhiên, CC vẫn là một mô hình đang tiến tới hoàn chỉnh, các hãng công nghệ cũng như các tổ chức tiêu chuẩn trên thế giới đã đưa ra các định nghĩa và cách nhìn riêng. Theo Viện nghiên cứu tiêu chuẩn và công nghệ quốc gia Hoa Kỳ (NIST) [75]: *"CC là mô hình điện toán cho phép truy cập qua mạng để lựa chọn và sử dụng tài nguyên tính toán (ví dụ: mạng, PM, bộ lưu trữ, ứng dụng và dịch vụ) theo nhu cầu một cách thuận tiện và nhanh chóng; đồng thời cho phép kết thúc sử dụng dịch vụ, giải phóng*

tài nguyên dễ dàng, giảm thiểu các giao tiếp với nhà cung cấp". Như vậy, có thể hiểu CC là việc ảo hóa các tài nguyên tính toán và các ứng dụng. Thay vì việc sử dụng một hoặc nhiều máy chủ vật lý (PM) thật, thì nay sử dụng các tài nguyên được ảo hóa thông qua môi trường Internet.

Theo NIST, CC có thể triển khai trên bốn mô hình: đám mây riêng, đám mây công cộng, đám mây hỗn hợp và đám mây cộng đồng; có ba mô hình dịch vụ là cơ sở hạ tầng như là dịch vụ (IaaS), nền tảng như là dịch vụ (PaaS) và phần mềm như là dịch vụ (SaaS); có năm đặc trưng quan trọng là tính tự phục vụ theo nhu cầu, truy cập diện rộng, dùng chung tài nguyên, khả năng co giãn nhanh chóng và trả tiền theo tài nguyên thực sự dùng.

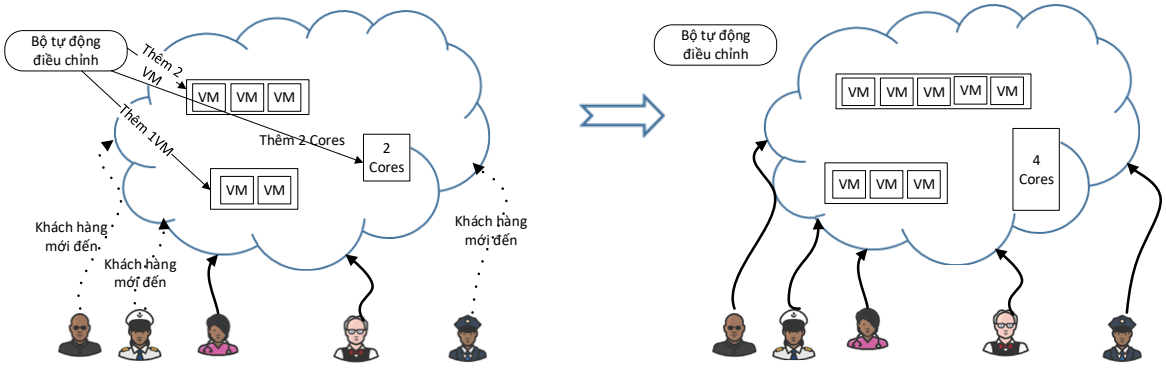
Trong đó, đặc trưng về khả năng co giãn nhanh chóng được nhiều nhà nghiên cứu quan tâm trong thời gian gần đây. Để sử dụng tính co giãn đám mây hiệu quả, điều quan trọng là phải cung cấp và giải phóng tài nguyên đám mây tự động, kịp thời mà không cần có sự can thiệp của con người, vì việc cung cấp tài nguyên quá nhiều dẫn đến lãng phí tài nguyên làm chi phí tăng lên, ngược lại, việc cung cấp tài nguyên quá ít là nguyên nhân làm suy giảm hiệu năng và gây ra vi phạm thỏa thuận mức dịch vụ (SLA). Cơ chế tự động cung cấp hoặc giải phóng tài nguyên để đáp ứng các yêu cầu chất lượng dịch vụ (QoS) được gọi là tự động điều chỉnh (AS).

Việc thiết kế và triển khai thực hiện một bộ AS tổng quát cho các ứng dụng web là một nhiệm vụ khó khăn do các yếu tố khác nhau, chẳng hạn như: đặc tính về tải công việc động, các yêu cầu tài nguyên của ứng dụng đa dạng, các tài nguyên đám mây và các mô hình giá phức tạp. Luận án này sẽ tập trung vào nghiên cứu và đề xuất giải pháp AS hiệu quả trong môi trường CC. Ngoài ra, luận án còn nghiên cứu một số vấn đề có ảnh hưởng đến chất lượng dịch vụ AS đó là các kỹ thuật cân bằng tải và di trú trong CC. Để đánh giá các số đo hiệu năng của hệ thống CC, luận án nghiên cứu và đề xuất một mô hình hệ thống CC sử dụng mạng hàng đợi.

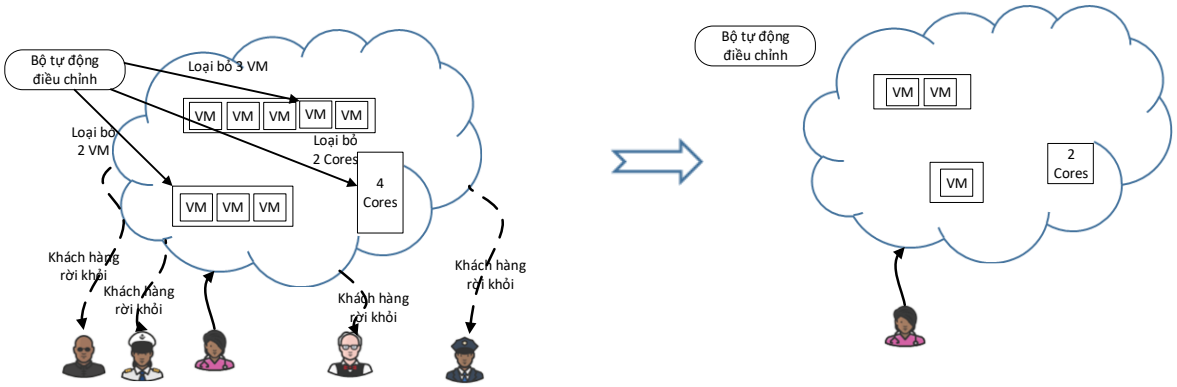
2. TÍNH CẤP THIẾT CỦA LUẬN ÁN

Trong CC, vấn đề AS cho các ứng dụng đa tầng có thể được xác định là cách tự chủ, tự động cung cấp và giải phóng các tài nguyên để phục vụ cho các tải ứng dụng

biến động mà không có sự can thiệp của con người sao cho chi phí tài nguyên ít nhất và thỏa mãn SLA hay SLO của ứng dụng. Hình 0.1 minh họa các trường hợp AS. Hình 0.1(a), vì tăng các yêu cầu đến, tài nguyên hiện tại rơi vào tình trạng quá tải, bộ AS quyết định cung cấp một số tài nguyên nhất định bằng cách thêm các VM mới hoặc là thêm khả năng tính toán (ví dụ: thêm core) cho các VM hiện có. Ngược lại, trong Hình 0.1(b), bộ AS giải phóng một số tài nguyên khi số lượng yêu cầu giảm xuống bằng cách tắt một số VM hiện có hoặc là giảm công suất của các VM hiện có (ví dụ: giảm số core).



(a)



(b)

Hình 0.1: Ví dụ một trường hợp tự động điều chỉnh

Đây là một bài toán điều khiển tự động cổ điển, đòi hỏi một bộ điều khiển AS loại tài nguyên và số lượng tài nguyên được cấp phát để đạt được các mục tiêu hiệu năng nhất định, được phản ánh trong SLA. Cụ thể, nó thường được thể hiện như một vòng lặp điều khiển MAPE: Giai đoạn giám sát (M), phân tích (A), lập kế hoạch (P) và thực thi (E) [88]. Chu kỳ điều khiển liên tục lặp đi lặp lại theo thời gian.

Một số khó khăn cho các kỹ thuật AS hiện tại là: (1) môi trường liên tục, việc lựa chọn hành động hiện tại sẽ ảnh hưởng đến các hành động trong tương lai; (2) cơ sở hạ tầng CC phức tạp, không đồng nhất về cơ sở hạ tầng và dịch vụ gây khó khăn cho việc mô phỏng các mô hình chính xác môi trường thực thi; (3) tải công việc có tính chất động và không theo quy luật làm cho việc dự đoán tải công việc và việc ra các quyết định điều chỉnh chưa được chính xác. Trong môi trường CC, bộ điều khiển AS hoạt động như một tác nhân yêu cầu giải quyết các bài toán tuần tự, trong đó hành động ở trạng thái cụ thể sẽ ảnh hưởng đến hành động trong tương lai. Như vậy, cần xây dựng giải pháp cho các hành động theo tình huống cụ thể. Từ đó, giải pháp sẽ chỉ ra các quyết định mà bộ AS cần thực hiện dựa trên các tình huống khác nhau (tức là: các trạng thái của ứng dụng trong CC).

Một số khó khăn và thách thức cần được giải quyết trong việc xây dựng các giải pháp AS đã được chỉ ra trong các khảo sát [3, 68, 88], như:

- Các nghiên cứu về AS ở mức dịch vụ còn hạn chế. AS bao gồm các mô hình dịch vụ đám mây đa dạng, nhưng hầu hết các nghiên cứu chỉ tập trung vào mức cơ sở hạ tầng. AS ở mức dịch vụ là quan trọng như các dịch vụ đang chạy trên một tập hợp các VM được kết nối và QoS phụ thuộc vào cách AS xử lý các tài nguyên cho các VM như thế nào. Các số đo mức độ dịch vụ như số giao dịch trên mỗi đơn vị thời gian cần phải được ánh xạ tới các số đo mức hệ thống như là: tỷ lệ mức độ sử dụng CPU, mạng và vấn đề vào ra (IO) của ổ đĩa.

- Các công cụ để giám sát không đủ, việc kết hợp các phép đo ở mức nền tảng và mức dịch vụ để hỗ trợ việc ra quyết định AS còn hạn chế.

- AS trong môi trường CC hỗn hợp cũng không được hỗ trợ tốt. Đám mây hỗn hợp, trong đó có một phần ứng dụng được triển khai trên đám mây riêng và phần khác được triển khai trên đám mây công cộng. Trong trường hợp này, các đám mây công cộng và đám mây riêng có thể cung cấp các giải pháp AS khác nhau mà không tương thích với nhau, do đó sẽ có vấn đề không tương thích giữa các tài nguyên AS trên hai đám mây.

- Hiệu quả của AS theo độ tin cậy của quá trình AS không được quản lý tốt. Thất bại của quá trình AS có thể dẫn đến vi phạm các yêu cầu về QoS của hệ thống như là: vấn đề hiệu năng, khả năng co giãn và thậm chí phải chịu chi phí không cần thiết.

- Hạn chế về các nghiên cứu chỉ ra mối quan hệ giữa AS và các thuộc tính khác như: độ sẵn sàng, độ tin cậy và độ an ninh. Ví dụ, các cuộc tấn công từ chối dịch vụ DDoS có thể gây ra một dịch vụ AS để mở rộng hệ thống không cần thiết và do đó làm tăng chi phí vận hành.

- Xác định ảnh hưởng của các cơ chế hệ thống cơ bản đến dịch vụ AS trong CC, như kỹ thuật cân bằng tải, lập lịch hay di trú, từ đó đưa ra các giải pháp mới thích nghi trong điều kiện AS nhằm nâng cao hiệu năng đám mây.

- Các giải pháp AS đã được đề xuất trong các công trình được đánh giá và thực nghiệm trên các môi trường và công cụ khác nhau. Mỗi giải pháp đều có những ưu và hạn chế nhất định. Do đó, hiện vẫn chưa có giải pháp AS hiệu quả nhất trên một môi trường hay công cụ nhất định. Nghiên cứu về các giải pháp AS trong CC hiện vẫn là vấn để mở cho các nhà nghiên cứu.

Cộng đồng nghiên cứu đã đề xuất ra một số giải pháp AS, như giải pháp dựa trên lý thuyết điều khiển [2], dự trên lý thuyết hàng đợi [34, 101], điều khiển mờ [50], phân tích chuỗi thời gian và dự đoán [35] nhưng chưa giải quyết đầy đủ tính động của các ứng dụng CC. Các xu hướng gần đây, giải pháp dựa vào bộ điều khiển tự tổ chức có thể phù hợp với độ phức tạp của bộ điều khiển đám mây [11, 32]. Tuy nhiên, nó vẫn phụ thuộc vào người sử dụng đám mây phải xác định và cấu hình bộ điều khiển đám mây. Do các nhà cung cấp đám mây không biết chi tiết về các ứng dụng được triển khai làm cho khó khăn để đưa ra một tập luật tối ưu. Như vậy, việc xác định tập luật này lại được giao cho những người sử dụng CC, những người này lại không hiểu biết về tải công việc, cơ sở hạ tầng hay mô hình hiệu năng. Như trong công trình [50] đã sử dụng logic mờ để khám phá tri thức trực quan của con người, chuyển tri thức này thành một tập luật NẾU-THÌ để thực hiện AS. Tuy nhiên, do tập luật này phải được xác định ngay từ khi bắt đầu xây dựng bộ AS nên có thể gặp những vấn đề sau: (1) tri thức có thể không có sẵn, người dùng không thể tạo ra bất kỳ luật nào; (2) tri thức có thể có sẵn

nhưng có từng phần, người dùng chỉ có thể xác định các luật một phần cho một số trường hợp; (3) tri thức không phải luôn tối ưu, người dùng có thể xác định các luật nhưng chúng không có hiệu quả, ví dụ: các luật dư thừa; (4) tri thức có thể được chính xác đối với một số luật nhưng có thể ít chính xác đối với một số luật khác, ví dụ: các luật chứa sự không chắc chắn và tùy thuộc vào mức độ tri thức ưu tiên; (5) tri thức có thể cần phải thay đổi trong thời gian thực hiện, các luật có thể được chính xác tại thời gian thiết kế nhưng có thể thay đổi trong thời gian chạy. Kết quả là, người sử dụng các luật đã được định nghĩa trước có thể dẫn đến quyết định điều chỉnh tối ưu cục bộ và tốn tiền trả cho các nhà cung cấp ứng dụng CC.

Để khắc phục hạn chế trên, luận án nghiên cứu và đề xuất giải pháp AS tài nguyên hiệu quả trong CC bằng cách sử dụng bộ điều khiển mờ kết hợp với phương pháp học tăng cường – học Q mờ [54] để điều chỉnh và cải thiện các chính sách AS khi thực hiện. Phương pháp học tăng cường được sử dụng là học Q, cho phép hệ thống học từ sự tương tác với môi trường, trong đó việc học được thực hiện thông qua cơ chế phần thưởng [23]. Sự kết hợp của điều khiển mờ và học Q mờ tạo ra một cơ chế tự thích nghi mạnh mẽ, trong đó điều khiển mờ thực hiện việc lập luận ở mức độ trừu tượng cao hơn, lập luận giống con người và học Q cho phép thích nghi với bộ điều khiển. Bộ AS này có thể bắt đầu làm việc mà không cần có tri thức ban đầu (với tri thức ban đầu rỗng), và tri thức sẽ có được trong thời gian thực hiện, thông qua cơ chế tiến hóa tri thức để học được tập luật tối ưu dùng để điều chỉnh tài nguyên theo các tham số đầu vào.

Ngoài ra, luận án còn nghiên cứu một số vấn đề có ảnh hưởng đến hiệu năng của hoạt động dịch vụ AS như các kỹ thuật cơ bản của CC: kỹ thuật cân bằng tải, di trú. Tiếp theo, luận án cũng nghiên cứu các mô hình hàng đợi để mô hình hệ thống CC và đề xuất một mô hình CC sử dụng mạng hàng đợi – mạng Jackson mở nhằm đánh giá một số số đo hiệu năng trong hệ thống CC. Trong tương lai, có thể tiến hành thử nghiệm một cơ chế AS trên CC, trong đó các công thức phân tích của mô hình đề xuất được sử dụng để phân tích tài nguyên tự động nhằm đáp ứng các mục tiêu QoS theo tải công việc thay đổi.

3. MỤC TIÊU CỦA LUẬN ÁN

Mục tiêu của luận án là nghiên cứu đề xuất giải pháp AS tài nguyên trong CC nhằm duy trì hiệu năng hoạt động của các trung tâm dữ liệu CC, tận dụng tối đa tài nguyên cấp phát đáp ứng kịp thời nhu cầu của người dùng, đồng thời giảm chi phí cho người dùng đám mây.

Luận án tập trung vào các vấn đề sau:

- Nghiên cứu một số cơ chế cơ bản có ảnh hưởng đến hoạt động AS trong CC, đó là kỹ thuật cân bằng tải và di trú trong CC. Sau đó đề xuất một kỹ thuật cân bằng tải và một kỹ thuật di trú hiệu quả trong CC. Đánh giá ảnh hưởng của các kỹ thuật cân bằng tải đến hiệu quả dịch vụ AS trong môi trường CC.

- Nghiên cứu và đề xuất một mô hình mạng hàng đợi mở cho mô hình CC không đồng nhất về cơ sở hạ tầng nhằm đánh giá một số số đo hiệu năng quan trọng trong CC, như thời gian đáp ứng trung bình, số lượng máy chủ đang hoạt động, thời gian đợi trung bình, thông lượng,… Đây là các tham số quan trong cho hệ thống AS tài nguyên trong môi trường CC.

- Nghiên cứu và đề xuất giải pháp AS tài nguyên theo nhu cầu hiệu quả trong môi trường CC, đảm bảo được hiệu năng của CC, duy trì QoS, đồng thời tiết kiệm chi phí cho khách hàng.

4. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

Đối tượng nghiên cứu của luận án là các giải pháp kỹ thuật cơ bản như kỹ thuật cân bằng tải và di trú trong CC; mô hình hóa môi trường CC bằng một mô hình mạng hàng đợi mở; và giải pháp AS tài nguyên hiệu quả trong CC.

Phạm vi nghiên cứu của luận án tập trung áp dụng các giải pháp, kỹ thuật trên nền tảng cơ sở hạ tầng CC.

5. PHƯƠNG PHÁP NGHIÊN CỨU

Phương pháp nghiêu cứu của luận án là:

- Với nội dung 1: Luận án nghiên cứu các kỹ thuật cân bằng tải và di trú trong CC hiện có, chỉ ra vấn đề hạn chế, sau đó đề xuất giải pháp để khắc phục những hạn chế đó. Sử dụng thực nghiệm để đánh giá tính hiệu quả của kỹ thuật đề xuất.

- Với nội dung 2: Luận án nghiên cứu các mô hình mạng hàng đợi – mạng Jackson mở, để đề xuất ra một mô hình cho hệ thống CC. Sử dụng mô phỏng để đánh giá tính đúng đắn của mô hình được đề xuất.

- Với nội dung 3: Luận án nghiên cứu logic mờ và học tăng cường để đề xuất ra bộ AS hiệu quả cho CC. Sử dụng thực nghiệm để so sánh, đánh giá với các giải pháp hiện có.

Về thực nghiệm và mô phỏng: Luận án thực hiện thử nghiệm các kỹ thuật, giải pháp đề xuất dựa vào công cụ mô phỏng CC hoặc tự lập trình mô phỏng trên ngôn ngữ lập trình Java và công cụ mô phỏng CloudSim [15], sử dụng cơ sở dữ liệu sinh ngẫu nhiên, hoặc lấy bộ dữ liệu trên Internet, hoặc được thiết lập cố định theo cấu hình nhất định,…

6. CÁC ĐÓNG GÓP CỦA LUẬN ÁN

Đóng góp của luận án bao gồm:

- Đề xuất một kỹ thuật cân bằng tải hiệu quả và một kỹ thuật di trú hiệu quả trong CC tạo điều kiện thuận lợi cho việc điều chỉnh tài nguyên giúp cải thiện hiệu năng của các trung tâm CC. Đã minh chứng cơ chế cân bằng tải có ảnh hưởng đến tính hiệu quả của cơ chế AS tài nguyên và kỹ thuật cân bằng tải được đề xuất có tác động tích cực đến AS tài nguyên. Kết quả nghiên cứu được công bố trong các công trình TCNN1, TCTN1, HNQT1, HNQT2 và được trình bày trong Chương 2 của luận án.

- Đề xuất một mô hình CC sử dụng mạng hàng đợi – mạng Jackson mở, làm cơ sở để đánh giá các thông số quan trọng về hiệu năng của CC. Kết quả được công bố trong công trình TCNN3 và được trình bày trong Chương 3 của luận án.

- Đề xuất một giải pháp AS tài nguyên hiệu quả trong CC: xây dựng một bộ AS sử dụng kết hợp kỹ thuật học tăng cường và điều khiển logic mờ. Kết quả được công bố trong các công trình TCNN2, HNQT3, HNTN1 và được trình bày trong Chương 4 của luận án.

7. BỐ CỤC CỦA LUẬN ÁN

Luận án được xây dựng thành bốn chương như sau:

Chương 1: Tổng quan về tự động điều chỉnh tài nguyên trong điện toán đám mây

Giới thiệu tổng quan về AS trong CC. Trình bày khái niệm về AS trong CC; các phương pháp điều chỉnh trong CC; phân loại các kỹ thuật AS tài nguyên cùng với các công trình đã công bố, đánh giá những ưu điểm và hạn chế của từng kỹ thuật. Ngoài ra, còn trình bày về kiến trúc ứng dụng đa tầng trong CC, nền tảng thực nghiệm và tải công việc trong CC.

Chương 2: Các đề xuất cơ chế cơ bản có ảnh hưởng đến tự động điều chỉnh tài nguyên trong điện toán đám mây

Chương này, luận án sẽ trình bày kết quả nghiên cứu các cơ chế hệ thống cơ bản có ảnh hưởng đến chất lượng dịch vụ AS trong CC, cụ thể là đề xuất được một kỹ thuật cân bằng tải và một kỹ thuật di trú hiệu quả. Sau đó, luận án đã đánh giá về ảnh hưởng của các kỹ thuật cân bằng tải đến tính hiệu quả của dịch vụ AS tài nguyên trong CC.

Chương 3: Đề xuất một mô hình mạng hàng đợi cho hệ thống điện toán đám mây

Trình bày về mô hình mạng hàng đợi – mạng Jackson mở được đề xuất cho môi trường CC không đồng nhất, đánh giá các số đo hiệu năng của hệ thống: thời gian đáp ứng trung bình, thời gian chờ, thông lượng,… Đây là những thông số quan trọng, có thể làm đầu vào cho các bộ AS tài nguyên trong CC.

Chương 4: Giải pháp tự động điều chỉnh tài nguyên cho ứng dụng đa tầng trên điện toán đám mây

Trình bày giải pháp AS hiệu quả được đề xuất trong CC sử dụng bộ điều khiển mờ kết hợp với phương pháp học tăng cường. Việc đánh giá và kiểm chứng giải pháp đề xuất thông qua thực nghiệm, so sánh với các giải pháp AS hiện có.

Cuối cùng là một số kết luận và hướng phát triển của luận án.

CHƯƠNG 1 : TỔNG QUAN VỀ TỰ ĐỘNG ĐIỀU CHỈNH TÀI NGUYÊN TRONG ĐIỆN TOÁN ĐÁM MÂY

Chương này trình bày tổng quan về AS tài nguyên trong môi trường CC và những vấn đề liên quan đến nội dung luận án, như: khái niệm về AS tài nguyên, phân loại các kỹ thuật AS tài nguyên, tổng quan về các công trình liên quan đến các kỹ thuật AS. Bên cạnh đó cũng trình bày về vòng lặp điều khiển MAPE trong AS tài nguyên trên CC, kiến trúc ứng dụng đa tầng, nền tảng thực nghiệm và tải công việc trong CC.

1.1. CƠ SỞ LÝ THUYẾT

1.1.1. Khái niệm về tự động điều chỉnh

Hiện có một số hãng công nghệ, các tổ chức và nhà nghiên cứu đã đưa ra các định nghĩa về AS theo cách riêng của họ. Dưới đây là một số định nghĩa về AS.

Theo Gartner[1]: *"AS là tự động mở rộng hay thu nhỏ khả năng của hệ thống tạo nên tính sẵn sàng cho các ứng dụng và là một đặc trưng mong muốn nhất trong đám mây IaaS và PaaS cung cấp. Khi có thể, người thuê công nghệ nên sử dụng để phù hợp với khả năng được cung cấp cho nhu cầu của ứng dụng và tiết kiệm chi phí."*

Theo TechTarget[2]: *"AS là một đặc trưng dịch vụ CC để tự động thêm hoặc loại bỏ các tài nguyên tính toán tùy thuộc vào việc sử dụng thực tế. AS đôi khi được gọi là tính đàn hồi tự động."*

AS là cách để điều chỉnh tăng lên hoặc điều chỉnh giảm xuống một cách tự động số lượng tài nguyên tính toán đang được cấp phát cho các ứng dụng của khách hàng dựa vào nhu cầu của khách hàng ở bất kỳ thời điểm nào.

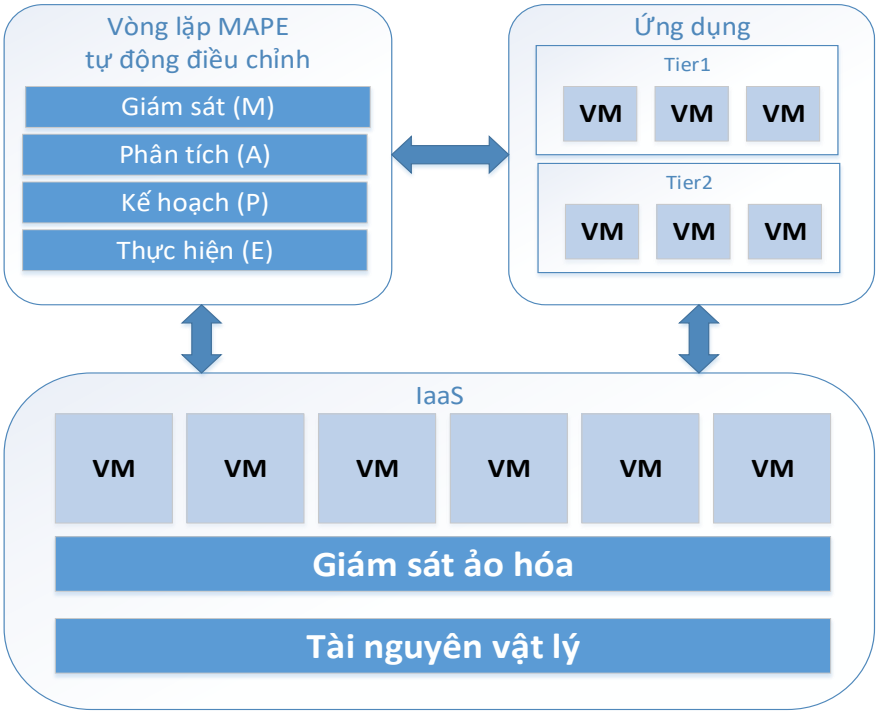
Theo Techopedia[3], AS là một đặc trưng của CC cho phép người dùng AS các dịch vụ đám mây, như các VM và khả năng của VM, tăng lên hoặc giảm xuống, tùy thuộc vào tình huống cụ thể. Các nhà cung cấp CC, như Amazon Web Services (AWS), cung cấp đặc trưng này. AS cũng đảm bảo các VM mới được tăng lên kịp

[1] https://www.gartner.com/doc/2651015/technology-overview-autoscaling. *Truy cập ngày 10/7/2018.*
[2] https://searchcloudapplications.techtarget.com/definition/autoscaling. *Truy cập ngày 10/7/2018.*
[3] https://www.techopedia.com/definition/29432/auto-scaling. *Truy cập ngày 10/7/2018.*

thời khi nhu cầu tăng đột biến và giảm xuống khi nhu cầu giảm, cho phép thực hiện nhất quán đảm bảo chi phí thấp hơn.

Tuy nhiên, trong một số trường hợp, AS thực sự làm tăng chi phí vì nó không phải luôn luôn có thể xác định việc tăng đột biến tải công việc là đúng (hợp lệ) hay nó là kết quả của một cuộc tấn công từ chối dịch vụ (DDoS), khi các máy chủ điều chỉnh để đáp ứng lưu lượng tăng đột biến này. Điều này tạo ra một chi phí lãng phí không có lợi. Giải pháp là cần có một công cụ giám sát thông minh hơn để có thể phân biệt giữa lưu lượng thực tế và một cuộc tấn công.

Theo quan điểm học thuật, AS là khả năng của cơ sở hạ tầng CC cho phép cung cấp tự động các tài nguyên ảo hóa [14, 44]. Tài nguyên được sử dụng bởi các ứng dụng dựa trên đám mây có thể được tự động tăng lên hoặc giảm xuống, do đó thích hợp cho việc sử dụng tài nguyên theo nhu cầu của ứng dụng.



Hình 1.1: Một dạng hệ thống AS đơn giản

Một hệ thống AS đơn giản được chỉ ra như trong Hình 1.1. Lõi là bộ AS, có thể bao gồm nhiều thành phần xử lý logic. Quá trình AS phù hợp với vòng lặp điều khiển MAPE của hệ thống tự trị [88]. Đầu tiên, hệ thống giám sát thu thập thông tin về trạng thái hệ thống và trạng thái ứng dụng. Bộ AS bao gồm các giai đoạn phân tích và lập

kế hoạch: nó sử dụng thông tin thu thập được để ước tính mức sử dụng và nhu cầu tài nguyên trong tương lai (phân tích) và sau đó lập kế hoạch hành động sửa đổi tài nguyên phù hợp, ví dụ: loại bỏ VM hoặc thêm bộ nhớ bổ sung. Sau đó, nhà cung cấp sẽ thực hiện các hành động theo yêu cầu của bộ AS.

Tuy nhiên, dạng hệ thống AS đơn giản như vậy có những hạn chế, vì nó xử lý không hiệu quả khi tăng độ phức tạp trong thời gian chạy của môi trường đám mây, như, tải công việc động/ thay đổi và tải công việc không chắc chắn, hiệu năng QoS và tính không đồng nhất của các dịch vụ trên đám mây. Với cơ sở hạ tầng được chia sẻ của đám mây, quá trình AS cần phải biết và có thể xử lý các yêu cầu QoS. Điều này do việc cải thiện hiệu năng QoS của một dịch vụ có thể làm suy giảm các dịch vụ và các VM lân cận, làm ảnh hưởng tiêu cực đến chất lượng tổng thể của AS và tính đàn hồi.

Để cải thiện chất lượng thích nghi, các hệ thống AS hiện đại thường bổ sung thêm các khía cạnh tinh vi khác, gồm mô hình hóa, xác định độ chi tiết của việc điều khiển và ra quyết định. Mô hình hóa liên quan đến mô hình QoS, điều kiện môi trường (ví dụ: tải công việc) và nhu cầu của các nút điều khiển (ví dụ: cấu hình phần mềm và tài nguyên phần cứng). Các mô hình kết quả là một công cụ mạnh mẽ để hỗ trợ quá trình ra quyết định AS.

Việc xác định mức độ chi tiết điều khiển trong bộ AS là điều cần thiết để đảm bảo lợi ích (ví dụ: QoS và các mục tiêu chi phí) cho tất cả dịch vụ trên đám mây. Một số mục tiêu cụ thể có thể được xem xét độc lập so với một số mục tiêu khác. Hiện có mục tiêu phụ thuộc cho cả các dịch vụ bên trong và bên ngoài. Tức là, sự phụ thuộc mục tiêu không chỉ do tính chất (dịch vụ bên ngoài), ví dụ: mục tiêu thông lượng và chi phí của một dịch vụ; mà còn phải xử lý yêu cầu QoS (các dịch vụ bên trong) do các dịch vụ cùng nằm trên VM và các VM cùng trên một PM [13, 71, 79, 113].

Tiếp theo là quá trình ra quyết định tự động của bộ AS để tạo ra quyết định tối ưu (hoặc gần tối ưu), bao gồm các giá trị mới được cấu hình của các yếu tố điều khiển và cho tất cả các mục tiêu liên quan. Theo khách quan, việc ra quyết định AS đòi hỏi phải giải quyết sự cân bằng phức tạp giữa yêu cầu SLA và ngân sách. Quyết định cân bằng

có thể được thực thi bằng cách sử dụng các hành động điều chỉnh theo chiều dọc và/hoặc theo chiều ngang, điều chỉnh tương ứng với các dịch vụ trên đám mây và/hoặc các VM.

Như đã đề cập ở trên, bài toán AS liên quan trực tiếp đến vòng lặp điều khiển MAPE. Trong phần tiếp theo sẽ tìm hiểu về vòng lặp điều khiển MAPE.

1.1.2. Vòng lặp điều khiển MAPE

Mục đích của bộ AS là điều chỉnh các tài nguyên tự động để gán cho các ứng dụng đàn hồi, tùy thuộc vào tải công việc đầu vào. Bộ AS có thể là triển khai/ thực hiện adhoc cho một ứng dụng cụ thể hoặc dịch vụ được cung cấp bởi nhà cung cấp IaaS. Trong mọi trường hợp, hệ thống cần phải tìm sự cân bằng giữa việc đáp ứng SLA của ứng dụng và việc tối thiểu chi phí thuê tài nguyên đám mây. Với bất kỳ bộ AS nào cũng phải đối mặt với một số vấn đề sau:

- *Cung cấp tài nguyên quá ít*: Ứng dụng không có đủ tài nguyên để xử lý tất cả các yêu cầu đến trong giới hạn thời gian được thiết lập trong SLA.

- *Cung cấp tài nguyên quá nhiều*: Trong trường hợp này, ứng dụng có nhiều tài nguyên hơn so với nhu cầu để thỏa mãn SLA. Điều này đáp ứng được SLA, nhưng khách hàng đang phải trả một chi phí không cần thiết nếu các VM không có tải hoặc ít tải.

- *Dao động:* Một sự kết hợp của cả hai quá trình cung cấp ở trên sẽ tạo ra những tác động không mong muốn. Dao động xảy ra khi các hành động điều chỉnh được thực hiện quá nhanh, trước khi có thể thấy tác động đến từng hành động điều chỉnh trên ứng dụng. Giữ bộ đệm dung lượng (nhằm giữ cho VM ở mức 80% thay vì 100%) hoặc sử dụng chu kỳ làm mát là các phương pháp phổ biến để tránh dao động.

Đây là bài toán điều khiển tự động cổ điển, đòi hỏi một hệ thống AS các loại tài nguyên và số lượng tài nguyên được cấp phát để đạt được mục tiêu hiệu năng nhất định, được phản ánh trong SLA. Nó thường được thực hiện như một vòng lặp điều khiển MAPE [88]. Chu kỳ kiểm soát liên tục lặp đi lặp lại theo thời gian. Mỗi giai đoạn trong vòng lặp MAPE như sau:

1.1.2.1. Giai đoạn giám sát (M)

Một hệ thống AS đòi hỏi sự hỗ trợ của một hệ thống giám sát cung cấp các phép đo về nhu cầu của người dùng, trạng thái hệ thống (ứng dụng) và tuân theo SLA dự kiến. Thông qua API, cơ sở hạ tầng đám mây cung cấp, truy cập thông tin hữu ích cho bản thân nhà cung cấp (ví dụ: để kiểm tra chức năng chính xác của cơ sở hạ tầng và mức độ sử dụng) và khách hàng (ví dụ: để kiểm tra việc tuân theo SLA).

Hệ thống cần phải giám sát một số số đo hiệu năng để xác định xem các hoạt động AS có cần thiết hay không và chúng làm thế nào để thực hiện.

1.1.2.2. Giai đoạn phân tích (A)

Giai đoạn phân tích bao gồm việc xử lý các số liệu được thu thập trực tiếp từ hệ thống giám sát, thu thập từ dữ liệu về mức sử dụng hệ thống hiện tại và các dự đoán tùy chọn về các nhu cầu trong tương lai. Một số bộ AS không thực hiện bất kỳ loại dự đoán nào, chỉ đáp ứng với trạng thái hệ thống hiện tại: đó là *bộ AS bị động*. Tuy nhiên, sử dụng các kỹ thuật khác để dự đoán nhu cầu trong tương lai nhằm sắp xếp việc cung cấp tài nguyên với dự đoán đủ số lượng: đó là *bộ AS chủ động*. Dự đoán là quan trọng vì luôn có độ trễ thời gian khi một hành động AS được thực hiện (ví dụ: thêm một VM) cho đến khi nó có hiệu lực (ví dụ: phải mất vài phút để gán PM và triển khai VM, di chuyển hình ảnh VM vào nó, khởi động hệ điều hành và ứng dụng, và máy chủ hoạt động hoàn toàn [73]). Bộ AS bị động có thể không thể điều chỉnh trong trường hợp tăng lưu lượng bất ngờ (ví dụ: sự kiện ưu đãi đặc biệt hoặc hiệu ứng Slashdot). Do đó, bộ AS chủ động có thể được sử dụng để đối phó với nhu cầu biến động và có thể điều chỉnh trước.

Trong giai đoạn phân tích, hệ thống sẽ xác định xem có cần để thực hiện hoạt động điều chỉnh dựa trên thông tin được giám sát.

1.1.2.3. Giai đoạn lập kế hoạch (P)

Khi trạng thái hiện tại (hoặc tương lai) được biết (hoặc được dự đoán), bộ AS lập kế hoạch điều chỉnh các tài nguyên được gán cho ứng dụng để tìm sự cân bằng thích hợp giữa chi phí và việc đáp ứng theo SLA. Ví dụ về các quyết định điều chỉnh loại bỏ một VM hoặc thêm nhiều bộ nhớ hơn cho một VM cụ thể. Các quyết định sẽ

được đưa ra bằng cách xem xét dữ liệu thu được từ giai đoạn phân tích (hoặc trực tiếp từ hệ thống giám sát) và mục tiêu SLA, cũng như các yếu tố khác liên quan đến cơ sở hạ tầng đám mây, bao gồm các mô hình giá và thời gian khởi động VM.

Giai đoạn lập kế hoạch ước lượng bao nhiêu tài nguyên trong toàn bộ tài nguyên cần được cung cấp/giải phóng trong hành động điều chỉnh tiếp theo. Nó cũng cần được tối ưu hóa các thành phần của tài nguyên để giảm thiểu chi phí tài chính.

1.1.2.4. Giai đoạn thực hiện (E)

Giai đoạn này gồm việc thực hiện các hành động điều chỉnh được quyết định trong bước trước. Về mặt khái niệm, đây là giai đoạn đơn giản, được triển khai thông qua API của nhà cung cấp CC. Sự phức tạp thực sự được ẩn chứa bên phía khách hàng. Cần phải mất một thời gian từ thời điểm tài nguyên được yêu cầu cho đến khi nó thực sự sẵn sàng và giới hạn về những độ trễ này có thể là một phần trong SLA.

Giai đoạn thực thi có trách nhiệm xử lý các kế hoạch điều chỉnh để cung cấp/giải phóng các tài nguyên. Điều này đơn giản và có thể được thực hiện bằng cách gọi các hàm API của nhà cung cấp đám mây. Tuy nhiên, từ quan điểm kỹ thuật, có thể hỗ trợ các hàm API của các nhà cung cấp khác nhau là một nhiệm vụ rất khó khăn.

1.1.3. Phương pháp điều chỉnh

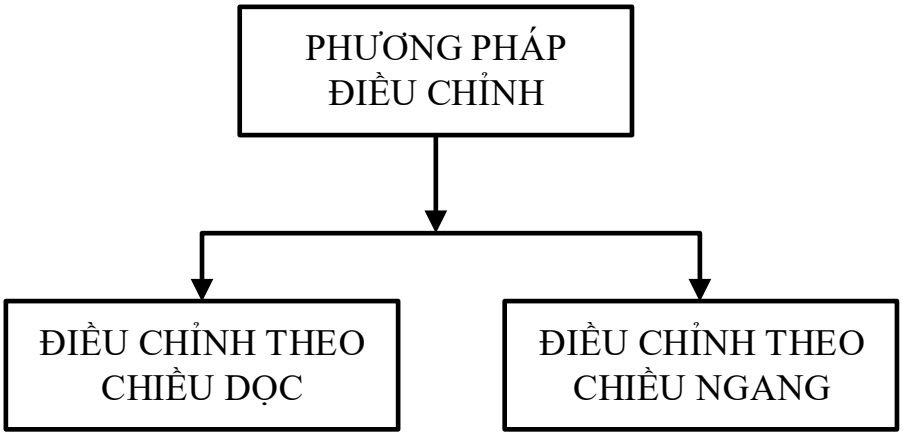
AS tài nguyên trong môi trường CC có thể được thực hiện điều chỉnh theo chiều dọc, theo chiều ngang hoặc có thể là kết hợp của hai phương pháp này (Hình 1.2). Mỗi phương pháp đều có ưu điểm và hạn chế của nó.

1.1.3.1. Điều chỉnh theo chiều dọc

Điều chỉnh theo chiều dọc có nghĩa là thêm hoặc loại bỏ tài nguyên, bao gồm CPU, bộ nhớ, IO và mạng, vào hoặc từ VM hiện có (ví dụ Hình 1.3). Để tự động thực hiện các hành động này, bộ giám sát sử dụng các cơ chế như chia sẻ CPU và tăng bộ nhớ, để hỗ trợ hot-plug CPU và bộ nhớ. Tuy nhiên, các nhà cung cấp CC lớn, như Amazon, Google và Microsoft, không hỗ trợ phương pháp điều chỉnh các tài nguyên trong thời gian chạy. Trong những nền tảng này, trước tiên là cần phải tắt VM để thêm tài nguyên. Một số nhà cung cấp như CenturyLink[4] cho phép người dùng điều

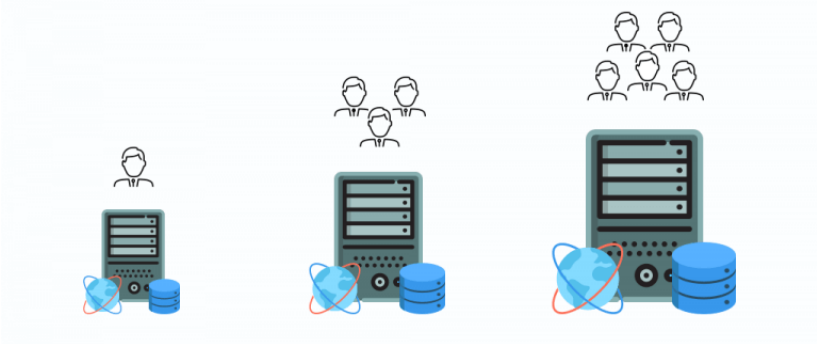
[4] *https://www.ctl.io/autoscale/*

chỉnh các lõi CPU mà không có thời gian chết theo chiều dọc. Profitbricks[5] cho phép thêm cả CPU và bộ nhớ cho VM một cách tự động.



Hình 1.2: Phương pháp điều chỉnh trong CC

Điều chỉnh theo chiều dọc được xem là không phù hợp cho các ứng dụng có khả năng điều chỉnh cao do những hạn chế của nó. Về mặt lý tưởng, khả năng tối đa một VM có thể điều chỉnh được bằng kích thước của PM. Tuy nhiên, nhiều VM thường trú trên cùng một PM có cạnh tranh tài nguyên, do đó có giới hạn hoặc ranh giới về khả năng điều chỉnh. Mặc dù có hạn chế, AS theo chiều dọc nhanh hơn so với AS theo chiều ngang về mặt thời gian cung cấp vì nó có thể có hiệu lực ngay lập tức. Bên cạnh đó, một số dịch vụ hoặc các thành phần mà khó có thể nhân bản trong thời gian chạy, chẳng hạn như máy chủ cơ sở dữ liệu và máy chủ ứng dụng ở trạng thái hết dung lượng, có thể sử dụng điều chỉnh theo chiều dọc.

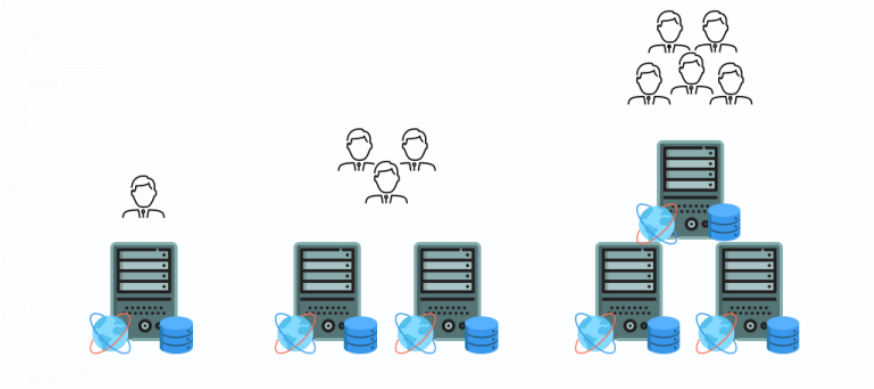


Hình 1.3: Ví dụ về điều chỉnh theo chiều dọc[6]

[5] *https://www.profitbricks.com/help/Live Vertical Scaling*
[6] *https://dzone.com/articles/vertical-scaling-and-horizontal-scaling-in-aws. Truy cập ngày 08/10/2018.*

1.1.3.2. Điều chỉnh theo chiều ngang

Điều chỉnh theo chiều ngang liên quan đến việc thêm/loại bỏ một số đơn vị xử lý hoặc PM vào cụm/hệ thống. Phương pháp này là cốt lõi của tính năng đàn hồi trong CC, thực hiện thêm/loại bỏ một số VM trong cụm/hệ thống (ví dụ Hình 1.4). Hầu hết các nhà cung cấp CC cung cấp các VM được chuẩn hóa về kích cỡ khác nhau để khách hàng lựa chọn. Một số nhà cung cấp khác cho phép người dùng tùy chỉnh cấu hình các VM với số lượng core CPU, bộ nhớ và băng thông mạng. Bên cạnh đó, nhiều mô hình giá cùng tồn tại trên thị trường đám mây hiện nay, làm tăng thêm tính phức tạp của bài toán cung cấp tài nguyên.



Hình 1.4: Ví dụ về điều chỉnh theo chiều ngang[7]

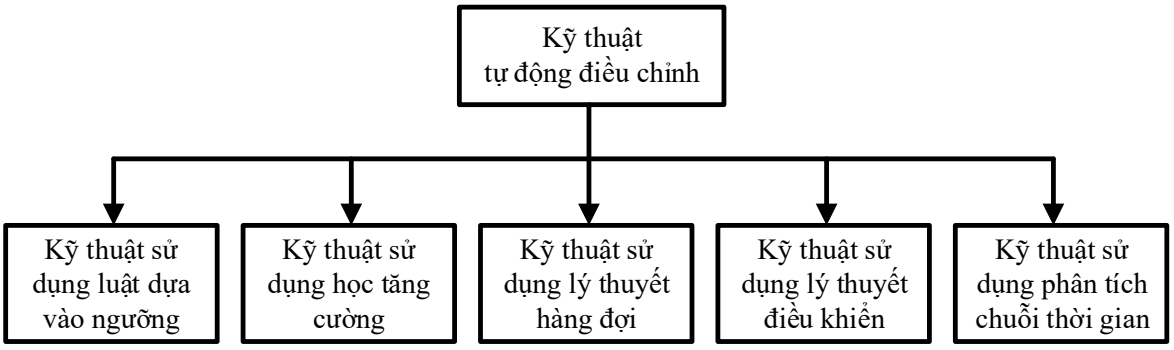
Tính không đồng nhất: Với một tầng/dịch vụ đơn trong ứng dụng web, nếu việc thanh toán là hằng số, thì việc sử dụng các VM đồng nhất có thể chấp nhận được vì nó dễ dàng quản lý. Khi đó, nhà cung cấp đám mây cung cấp các dịch vụ AS chỉ cho phép sử dụng các VM đồng nhất. Lựa chọn loại VM nào đó là trách nhiệm của người sử dụng trong bộ AS thương mại. Giải pháp tối ưu phụ thuộc vào cấu trúc tài nguyên của tầng/dịch vụ, cho dù đó là ứng dụng chuyên sâu về CPU hoặc bộ nhớ và đặc trưng tải công việc. Nếu tải công việc luôn đủ lớn, kích cỡ các VM khác nhau làm cho có sự khác biệt trong xử lý. Trong khi đó đối với một tải công việc nhỏ và biến động, các VM nhỏ hơn được sử dụng vì thực hiện điều chỉnh có thể được tiến hành ở mức độ chi tiết mịn hơn và do đó tiết kiệm chi phí hơn.

[7] https://dzone.com/articles/vertical-scaling-and-horizontal-scaling-in-aws. *Truy cập ngày 08/10/2018.*

Hiệu quả chi phí của VM rất cao liên quan đến loại ứng dụng và tải công việc. Nếu có những biến động xảy ra, việc lựa chọn loại VM cũng nên được cấu hình lại. Grozev và Buyya [39] đề xuất phương pháp phát hiện những thay đổi trực tuyến bằng cách sử dụng mô hình bộ nhớ tạm thời phân cấp và một mạng nơron nhân tạo được huấn luyện tự động và sau đó chọn lại loại VM hiệu quả nhất về chi phí.

1.1.4. Kỹ thuật tự động điều chỉnh

Theo khảo sát của Lorido-Bortran và đồng nghiệp [68] đã phân chia kỹ thuật điều chỉnh thành năm loại: kỹ thuật sử dụng luật dựa vào ngưỡng, kỹ thuật sử dụng học tăng cường, kỹ thuật sử dụng lý thuyết hàng đợi, kỹ thuật sử dụng lý thuyết điều khiển và kỹ thuật sử dụng phân tích chuỗi thời gian (Hình 1.5).



Hình 1.5: Phân loại các kỹ thuật AS

1.1.4.1. Kỹ thuật sử dụng luật dựa vào ngưỡng

Các luật hoặc chính sách AS dựa trên ngưỡng là rất phổ biến giữa các nhà cung cấp CC, như Amazon EC2 và các công cụ của bên thứ ba như RightScale. Tính đơn giản và trực quan của kỹ thuật này khiến nó trở nên rất hấp dẫn với các khách hàng đám mây. Tuy nhiên, việc thiết lập các ngưỡng tương ứng cho mỗi ứng dụng là một nhiệm vụ đòi hỏi có sự hiểu biết chuyên sâu về các xu hướng tải công việc.

Từ vòng lặp điều khiển MAPE, các luật hoàn toàn là một kỹ thuật ra quyết định (giai đoạn lập kế hoạch). Số lượng VM hoặc số lượng tài nguyên được gán cho ứng dụng đích sẽ thay đổi theo một tập luật, thường có hai loại ra quyết định: một là điều chỉnh mở rộng/thu nhỏ và hai là điều chỉnh giảm xuống/tăng lên.

1.1.4.2. Kỹ thuật sử dụng học tăng cường

Kỹ thuật sử dụng học tăng cường [97] là một loại phương pháp ra quyết định tự động đã được sử dụng trong bộ AS. Không có bất kỳ tri thức ưu tiên nào, các kỹ thuật học tăng cường có thể xác định hành động điều chỉnh tốt nhất để thực hiện cho mọi trạng thái của ứng dụng, với tải công việc đầu vào.

Kỹ thuật sử dụng học tăng cường [97] tập trung vào việc học thông qua tương tác trực tiếp giữa một tác nhân (ví dụ: bộ AS) và môi trường. Bộ AS sẽ học từ kinh nghiệm (phương pháp thử sai) hành động điều chỉnh tốt nhất để thực hiện, phụ thuộc vào trạng thái hiện tại, được đưa ra bởi tải công việc đầu vào, hiệu năng hoặc tập hợp các biến khác. Sau khi thực hiện một hành động, bộ AS nhận được phản hồi hoặc phần thưởng (ví dụ: cải thiện hiệu năng) từ hệ thống, về hành động đó tốt như thế nào. Vì vậy, bộ AS sẽ có xu hướng thực thi các hành động mang lại phần thưởng cao (hành động tốt nhất được tăng cường). Hạn chế của phương pháp này là hiệu năng khởi tạo không tốt, thời gian huấn luyện dài và vấn đề xử lý đột biến tải công việc đầu vào.

Ảnh hưởng của quyết định điều chỉnh sẽ mất một thời gian để có tác động đến ứng dụng và phần thưởng sẽ đến sau một thời gian trễ. Phương pháp học tăng cường đưa ra quyết định với thông tin hiện tại (trạng thái ứng dụng) về phần thưởng tương lai (ví dụ: thời gian đáp ứng). Phương pháp này bao gồm hai giai đoạn trong vòng lặp điều khiển MAPE: giai đoạn phân tích (A) và giai đoạn lập kế hoạch (P). Trước tiên, thông tin về ứng dụng và phần thưởng được thu thập trong bảng tra cứu (hoặc bất kỳ cấu trúc nào khác) để sử dụng sau này (giai đoạn phân tích). Sau đó, trong giai đoạn lập kế hoạch, thông tin này được sử dụng để ra quyết định hành động điều chỉnh tốt nhất.

1.1.4.3. Kỹ thuật sử dụng lý thuyết hàng đợi

Lý thuyết hàng đợi cổ điển đã được sử dụng rộng rãi để mô hình hóa các ứng dụng Internet và các máy chủ truyền thống. Lý thuyết hàng đợi có thể được sử dụng cho giai đoạn phân tích nhiệm vụ AS trong các ứng dụng đàn hồi, tức là ước lượng các số đo hiệu năng như: độ dài hàng đợi hoặc thời gian chờ trung bình cho các yêu cầu. Lý thuyết hàng đợi là một nhánh của xác suất thống kê, được ứng dụng trong

nhiều lĩnh vực khác nhau như: mạng truyền thông, hệ thống bán vé, thanh toán trong siêu thị, làm thủ tục tại sân bay,... Lý thuyết hàng đợi tập trung trả lời các câu hỏi như: thời gian chờ trung bình trong hàng đợi, thời gian đáp ứng trung bình của hệ thống (thời gian chờ trong hàng đợi cộng với thời gian phục vụ), tức là việc sử dụng các thiết bị phục vụ, phân phối số lượng khách hàng trong hàng đợi, phân phối khách hàng trong hệ thống.

Lý thuyết hàng đợi được sử dụng để phân tích các hệ thống có tính chất dừng, được đặc trưng bởi tốc độ đến và tốc độ phục vụ không đổi. Mục tiêu để lấy các số đo hiệu năng dựa trên mô hình hàng đợi và một số tham số đã biết (ví dụ: tốc độ đến λ). Ví dụ về các số đo hiệu năng là thời gian chờ trung bình trong hàng đợi và thời gian đáp ứng trung bình. Trong trường hợp có các điều kiện thay đổi (tức là tốc độ đến và tốc độ phục vụ không là hằng số), chẳng hạn như các ứng dụng có khả năng mở rộng, các tham số của mô hình hàng đợi phải được tính toán lại định kỳ và các số đo được tính toán lại.

Có hai phương pháp để giải quyết các bài toán mô hình hàng đợi: bằng phân tích và bằng mô phỏng. Trước đây chỉ có thể sử dụng các mô hình hàng đợi đơn giản có các phân bố được xác định rõ ràng cho các quá trình đến và quá trình phục vụ, như hàng đợi M/M/1 và G/G/1. Một phương pháp phân tích nổi tiếng để giải quyết các mạng hàng đợi là phân tích giá trị trung bình (MVA) [76]. Khi phương pháp phân tích không khả thi đối với các mô hình tương đối phức tạp, phương pháp mô phỏng vẫn có thể được sử dụng để thu được các số đo mong muốn.

Trong mô hình hàng đợi, việc áp dụng Luật của Little rất hữu ích, cho biết số lượng khách hàng trung bình (hoặc yêu cầu) $\mathrm{E}[\mathrm{C}]$ trong hệ thống bằng với tốc độ khách hàng đến trung bình λ nhân với thời gian phục vụ trung bình của mỗi khách hàng trong hệ thống $\mathrm{E}[\mathrm{T}]$: $\mathrm{E}[\mathrm{C}] = \lambda \times \mathrm{E}[\mathrm{T}]$.

1.1.4.4. Kỹ thuật sử dụng lý thuyết điều khiển

Lý thuyết điều khiển đã được áp dụng để tự động hoá việc quản lý các hệ thống xử lý thông tin khác nhau, chẳng hạn như hệ thống máy chủ web, hệ thống lưu trữ và các cụm trung tâm dữ liệu /cụm máy chủ. Đối với các ứng dụng đàn hồi được lưu trữ

trên đám mây, một hệ thống điều khiển có thể kết hợp cả hai giai đoạn của nhiệm vụ AS (giai đoạn phân tích và giai đoạn lập kế hoạch).

Mục tiêu chính của bộ điều khiển là tự động hóa việc quản lý (ví dụ: tác vụ điều chỉnh) của hệ thống đích. Bộ điều khiển phải duy trì giá trị của biến điều khiển y (ví dụ: tải CPU), gần với mức mong muốn hoặc điểm được thiết lập y_{ref}, bằng cách điều chỉnh biến thao tác u (ví dụ: số VM). Biến thao tác là đầu vào cho hệ thống đích, trong khi đó biến điều khiển được đo bằng bộ cảm biến và xem là đầu ra của hệ thống.

Có ba loại hệ thống điều khiển chính: vòng lặp mở, phản hồi và chuyển tiếp (feed-forward). Bộ điều khiển vòng lặp mở, cũng được gọi là không phản hồi (non-feedback), tính toán đầu vào cho hệ thống đích chỉ sử dụng trạng thái hiện tại và mô hình của hệ thống. Chúng không sử dụng thông tin phản hồi để xác định liệu hệ thống đầu ra y đã đạt được mục tiêu mong muốn y_{ref} hay chưa. Ngược lại, bộ điều khiển phản hồi quan sát đầu ra của hệ thống và có thể sửa bất kỳ độ lệch nào so với giá trị mong muốn. Bộ điều khiển chuyển tiếp cố gắng dự đoán lỗi ở đầu ra. Chúng dự đoán, sử dụng một mô hình, hành vi của hệ thống, và đáp ứng trước khi lỗi thực sự xảy ra. Dự đoán có thể không thành công và vì lý do này, bộ điều khiển phản hồi và bộ điều khiển chuyển tiếp thường được kết hợp với nhau.

Bộ điều khiển có thể được phân thành nhiều loại [85]: Bộ điều khiển khuếch đại cố định; Bộ điều khiển thích nghi; Bộ điều khiển dự báo mô hình (MPC);…

1.1.4.5. Kỹ thuật sử dụng phân tích chuỗi thời gian

Phân tích chuỗi thời gian sử dụng dữ liệu lịch sử để dự đoán việc sử dụng trong tương lai. Chuỗi thời gian được sử dụng trong nhiều lĩnh vực bao gồm tài chính, kỹ thuật, kinh tế và tin sinh học, thường đại diện cho sự thay đổi của một phép đo theo thời gian. Một chuỗi thời gian là một chuỗi các điểm dữ liệu, được đo thường xuyên tại các khoảng thời gian liên tiếp được thiết lập tại các khoảng thời gian như nhau. Ví dụ là số lượng yêu cầu đến ứng dụng, được thực hiện trong khoảng thời gian một phút. Phân tích chuỗi thời gian có thể được sử dụng trong giai đoạn phân tích của quá trình AS, để tìm các mẫu lặp lại trong tải công việc đầu vào hoặc để cố gắng dự đoán các giá trị trong tương lai.

Một công cụ cơ bản để biểu diễn chuỗi thời gian là biểu đồ. Nó liên quan đến việc phân bố các giá trị của chuỗi thời gian vào các khoảng có chiều rộng bằng nhau và biểu diễn tần số cho mỗi khoảng đó. Nó đã được sử dụng trong các nghiên cứu để biểu diễn cho mẫu sử dụng tài nguyên hoặc phân bố và sau đó dự đoán các giá trị trong tương lai.

1.1.5. Kiến trúc ứng dụng đa tầng

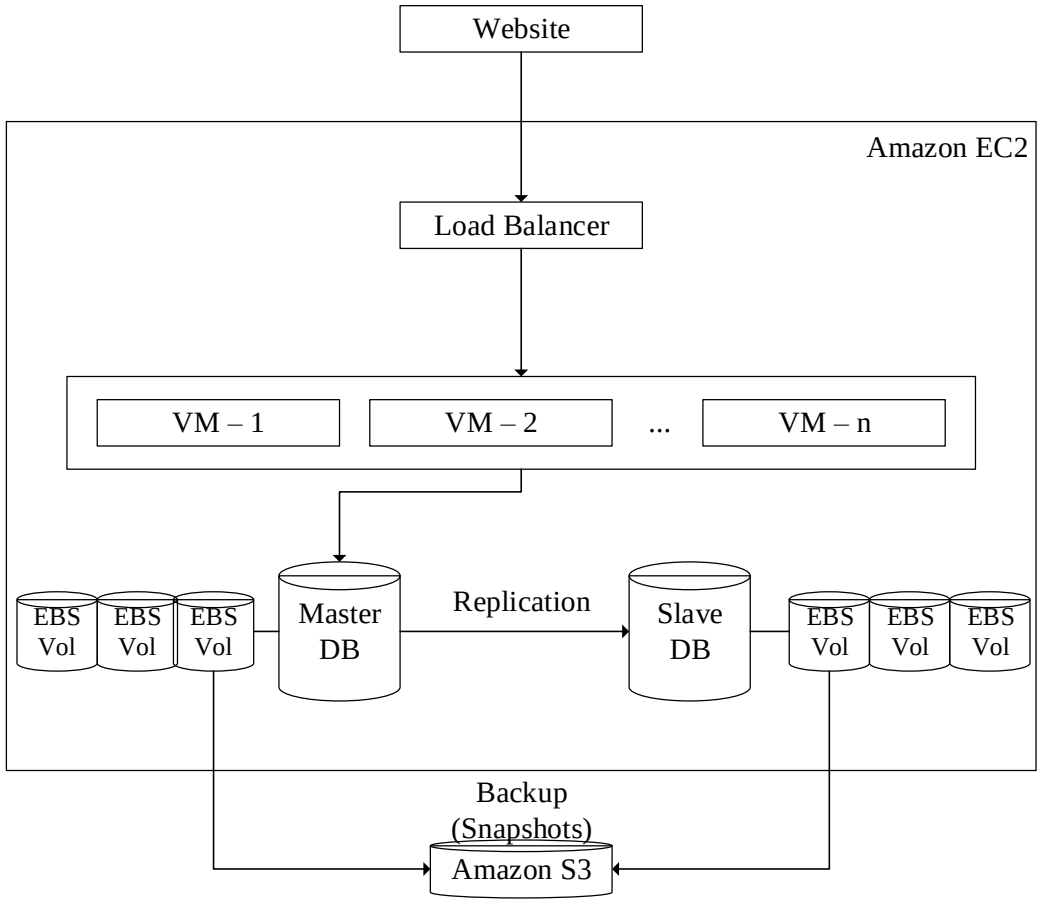
Các ứng dụng đa tầng bao gồm các tầng kết nối tuần tự. Tại mỗi tầng, các yêu cầu dựa vào tầng trước để hoàn thành xử lý hoặc nó được chuyển đến tầng tiếp theo để xử lý và cuối cùng, kết quả xử lý chuyển đến người dùng.

Một kiến trúc thích nghi mở rộng thuộc loại này thường bao gồm ba tầng: một tầng trước, một tầng logic ứng dụng và một tầng cơ sở dữ liệu (ví dụ Hình 1.6). Tầng cơ sở dữ liệu thường được xem xét là ít điều chỉnh và thường bỏ qua tính năng AS.

Nhiều công trình đã hướng đến các ứng dụng đa tầng. Một số công trình sử dụng các phương pháp chia để trị mà phá vỡ SLA tổng thể thành SLA của mỗi tầng, chẳng hạn như các công trình được thực hiện bởi Urgaonkar và đồng nghiệp [101], Iqbal và đồng nghiệp [46], Upendra và đồng nghiệp [94]. Những công trình khác, xem xét SLA của toàn bộ ứng dụng và cung cấp các tài nguyên đến mỗi tầng một cách toàn diện. Chiến lược này đòi hỏi sự cố gắng trong việc xây dựng mô hình và dự đoán lượng tiêu thụ tài nguyên sử dụng cho các mạng hàng đợi phức tạp và kỹ thuật học máy, do đó bộ AS chỉ áp dụng cho các ứng dụng đa tầng, có các công trình nghiên cứu và được đề xuất của Zhang và đồng nghiệp [112], Padala và đồng nghiệp [83], Han và đồng nghiệp [40], và Lama và Zhou [64].

Hãy xem xét một ứng dụng đa tầng đàn hồi được triển khai trên một nhóm n VM (Hình 1.6). Các VM có thể có một lượng tài nguyên được gán giống nhau hoặc khác nhau (ví dụ: 1GB bộ nhớ, 2 CPU,...), nhưng mỗi VM có số định danh duy nhất (có thể là địa chỉ IP của nó). Các yêu cầu nhận được bởi bộ cân bằng tải có thể đến từ người dùng cuối thực tế hoặc từ ứng dụng khác. Giả sử thời gian thực hiện của một yêu cầu có thể thay đổi giữa mili giây và phút.

Bộ cân bằng tải sẽ nhận tất cả các yêu cầu gửi đến và chuyển tiếp chúng đến một trong các máy chủ trong cụm. Nó có thể được thực hiện theo nhiều cách khác nhau. Cho dù công nghệ được chọn là gì, giả định rằng bộ cân bằng tải đã cập nhật thông tin về các VM đang hoạt động: nó sẽ dừng ngay lập tức gửi các yêu cầu tới các VM đã được loại bỏ, và nó sẽ bắt đầu gửi tải công việc tới các VM mới được bổ sung. Nó cũng giả định rằng mỗi yêu cầu sẽ được gán cho một VM duy nhất, sẽ chạy nó cho đến khi hoàn thành nhiệm vụ liên quan đến nó. Một số chính sách cân bằng tải có thể được sử dụng, ví dụ: chính sách ngẫu nhiên, Round Robin hoặc kết nối ít nhất. Trong trường hợp các cụm VM không đồng nhất, việc điều phối tải công việc phải tỷ lệ thuận với sức mạnh xử lý của các VM.



Hình 1.6: Ví dụ một ứng dụng web có kiến trúc ba tầng

Trong luận án này nghiên cứu các kỹ thuật AS theo chiều ngang, tức là AS số lượng VM tăng lên hoặc giảm xuống theo nhu cầu của tải công việc làm sao tối thiểu việc sử dụng các tài nguyên tính toán. Đối với ứng dụng đa tầng, mỗi tầng có thể sử

dụng nhiều VM, mức sử dụng tài nguyên của VM thứ i là $v_i = \{c_i, r_i, d_i\}$, trong đó, c_i là mức sử dụng CPU, r_i là mức sử dụng bộ nhớ và d_i là mức sử dụng ổ đĩa của VM thứ i. Các mức sử dụng này được thu thập theo dạng tỷ lệ phần trăm.

1.1.6. Nền tảng thực nghiệm

Các kỹ thuật AS hiện có thường được thực nghiệm trong các điều kiện rất khác nhau, do đó khó có được một đánh giá so sánh công bằng. Các nhà nghiên cứu trong lĩnh vực này đã tự xây dựng các nền tảng đánh giá riêng, phù hợp với nhu cầu riêng. Các nền tảng đánh giá này có thể được phân loại thành các chương trình mô phỏng, các bộ dữ liệu mẫu tùy chỉnh và các nhà cung cấp đám mây công cộng. Ngoại trừ các chương trình mô phỏng đơn giản, một tiêu chuẩn ứng dụng có thể mở rộng phải được thực hiện trên nền tảng để thực hiện đánh giá.

Thực nghiệm có thể được thực hiện trong cơ sở hạ tầng đưa ra, hoặc từ các nhà cung cấp đám mây thực tế hoặc trong một đám mây riêng. Ưu điểm chính của việc sử dụng nền tảng đám mây thực tế là đề xuất có thể được kiểm tra trong các tình huống thực tế, do đó chứng minh cải tiến phù hợp. Tuy nhiên, có hạn chế là: đối với mỗi thực nghiệm, toàn bộ ngữ cảnh cần phải được thiết lập. Trong trường hợp sử dụng nhà cung cấp dịch vụ đám mây công cộng, cơ sở hạ tầng đã được cung cấp, nhưng nhà nghiên cứu vẫn cần cấu hình hệ thống giám sát và AS, triển khai tiêu chuẩn ứng dụng và bộ sinh tải trên một nhóm VM, tìm hiểu cách trích xuất thông tin và lưu trữ nó để xử lý sau. Có thể, mỗi lần thực thi sẽ phải trả chi phí theo các khoản phí do nhà cung cấp đám mây thiết lập.

Để giảm chi phí thực nghiệm và có một môi trường được kiểm soát nhiều hơn, một bộ dữ liệu mẫu tùy chỉnh có thể được thiết lập. Phải trả chi phí để cấu hình hệ thống (ngoài việc mua phần cứng, nếu không có sẵn). Bước đầu tiên, bao gồm cài đặt phần mềm ảo hóa để quản lý việc tạo VM (không đơn giản), hỗ trợ điều chỉnh,…. Ảo hóa có thể được áp dụng ở mức PM, mức hệ điều hành hoặc ở mức ứng dụng. Đối với các bộ dữ liệu mẫu tùy chỉnh, một môi trường ảo hóa mức PM là cần thiết, thường được gọi là Hypervisor hoặc Virtual Machine Monitor (VMM). Một số hypervisor phổ biến là Xen, VMWare ESXi và Kernel Virtual Machine (KVM). Có một số nền

tảng để triển khai các đám mây tùy chỉnh, bao gồm các lựa chọn mã nguồn mở như OpenStack và Eucalyptus. OpenStack là một mã nguồn mở đầu tiên được hỗ trợ bởi nhiều công ty liên quan đến đám mây như RackSpace, HP, Intel và AMD, với một lượng khách hàng lớn. Eucalyptus cho phép tạo ra cơ sở hạ tầng tại chỗ như một đám mây dịch vụ, với sự hỗ trợ cho Xen, KVM và ESXi và API của Amazon EC2. VCloud Director là nền tảng thương mại được phát triển bởi VMWare.

Một phương án khác thay thế cho cơ sở hạ tầng thực tế (bộ dữ liệu mẫu hay nhà cung cấp dịch vụ đám mây công cộng), các công cụ phần mềm có thể được sử dụng để mô phỏng hoạt động của nền tảng đám mây, bao gồm việc cấp phát và thu hồi tài nguyên, thực thi VM, giám sát và các nhiệm vụ quản lý đám mây còn lại. Các nhà nghiên cứu có thể chọn một bộ mô phỏng hiện có và phù hợp với nhu cầu của mình, hoặc thực hiện một công cụ mô phỏng mới từ đầu. Rõ ràng, bằng cách sử dụng chương trình mô phỏng nhằm chỉ ra một sự cố gắng ban đầu để thích nghi hoặc phát triển phần mềm nhưng ngược lại có nhiều lợi thế. Quá trình đánh giá được rút ngắn theo nhiều cách. Mô phỏng cho phép thử nghiệm nhiều thuật toán, tránh cấu hình lại cấu trúc cơ sở hạ tầng. Bên cạnh đó, mô phỏng tách biệt thực nghiệm không bị ảnh hưởng bởi các yếu tố bên ngoài, các hệ số không kiểm soát được (ví dụ: ảnh hưởng với các ứng dụng khác, quá trình hợp nhất VM do nhà cung cấp tạo ra,…), điều này không thể thực hiện được trong cơ sở hạ tầng đám mây thực tế. Các thực nghiệm được thực hiện trong cơ sở hạ tầng thực tế có thể kéo dài hàng giờ, trong khi trong một mô phỏng dựa trên sự kiện, quá trình này có thể mất vài phút. Mô phỏng được cấu hình cao và cho phép nhà nghiên cứu thu thập bất kỳ thông tin nào về trạng thái hệ thống hoặc các số đo hiệu năng. Mặc dù có những ưu điểm như vậy, các môi trường mô phỏng vẫn là trừu tượng về các cụm PM, do đó độ tin cậy của các kết quả sẽ phụ thuộc vào mức chi tiết thực hiện được xem xét trong quá trình phát triển. Một số công cụ mô phỏng đám mây dùng cho hướng nghiên cứu là CloudSim [15], GreenCloud [59] và GroudSim [82].

1.1.7. Tải công việc

Tải công việc đầu vào cho ứng dụng có thể là dữ liệu (tải công việc) tổng hợp, được tạo bằng các chương trình cụ thể hoặc thu được từ người dùng thực tế.

Như mô tả ở trên, tải công việc biểu diễn cho các yếu tố đầu vào được xử lý bởi các tiêu chuẩn ứng dụng. Chúng có thể được tạo bằng cách sử dụng một số mẫu hoặc được thu thập từ các ứng dụng đám mây thực tế (và thường được lưu trữ trong các file bộ dữ liệu).

Hệ thống dựa trên đám mây xử lý hai loại tải công việc chính: lô (batch) và giao tác. Tải công việc theo lô bao gồm các công việc tùy ý, kéo dài, các công việc đòi hỏi nhiều tài nguyên, như các chương trình khoa học hoặc chuyển mã video. Ví dụ về tải công việc giao tác là các ứng dụng web được xây dựng để phục vụ các khách hàng HTTP trực tuyến. Các hệ thống này thường phục vụ các loại nội dung như các trang HTML, hình ảnh hoặc luồng video. Tất cả các nội dung này có thể được lưu trữ tĩnh hoặc được hiển thị tự động bởi các máy chủ.

1.1.7.1. Tải công việc tổng hợp

Tải công việc tổng hợp có thể được tạo dựa vào các mẫu khác nhau. Theo Mao và Humphrey [72] và Calzarossa và đồng nghiệp [17], có bốn mẫu tải công việc đại diện trong môi trường đám mây: Ổn định (stable), phát triển (Growing), chu kỳ/bùng nổ (Cycle/Bursting) và Bật-và-tắt (On-and-off). Trong số đó, mỗi mẫu đại diện cho một ứng dụng hoặc trường hợp cụ thể. Tải công việc ổn định được đặc trưng bởi số lượng yêu cầu cố định trong mỗi phút. Ngược lại, một mẫu phát triển cho thấy tải nhanh chóng tăng lên, ví dụ: một mẩu tin tức đột nhiên trở nên phổ biến. Mẫu thứ ba được gọi là cyclic/bursting bởi vì nó có thể chỉ ra các khoảng thời gian định kỳ (ví dụ: ban ngày có tải công việc nhiều hơn đêm) hoặc các bùng nổ tải ở các thời điểm trong ngày (ví dụ: có chính sách ưu đãi đặc biệt). Mẫu điển hình cuối cùng là tải công việc On-and-off, có thể VM xử lý theo lô hoặc phân tích dữ liệu được thực hiện hàng ngày.

Có một loạt các bộ sinh tải công việc, có thể tạo ra các yêu cầu đơn giản hoặc thậm chí các phiên HTTP thực tế, kết hợp các hành động khác nhau (ví dụ: đăng nhập hoặc duyệt) và mô phỏng thời gian suy nghĩ của người dùng.

1.1.7.2. Tải công việc thực tế

Theo thực tế, không có các bộ dữ liệu thực tế có sẵn công khai từ các ứng dụng được lưu trữ trên đám mây và đây là hạn chế rõ ràng cho nghiên cứu trên đám mây. Trong nghiên cứu, một số tác giả đã tạo ra bộ dữ liệu riêng của họ, chạy các tiêu chuẩn hoặc ứng dụng thực tế trong nền tảng đám mây. Ngoài ra còn có một số tham chiếu đến bộ dữ liệu từ những đám mây riêng chưa được công bố. Tuy nhiên, hầu hết các tác giả đã sử dụng các bộ dữ liệu từ các máy chủ Internet, chẳng hạn như bộ dữ liệu ClarkNet trace[8], bộ dữ liệu World Cup 98 trace[9] và các bộ dữ liệu truy cập Wikipedia[10].

1.2. TỔNG QUAN VỀ CÁC CÔNG TRÌNH LIÊN QUAN

Phần này sẽ trình bày tổng quan về các công trình liên quan đến các các kỹ thuật tự động điều chỉnh trong CC, phân tích những ưu điểm và hạn chế của chúng nhằm đề xuất hướng nghiên cứu tiếp theo.

1.2.1. Kỹ thuật sử dụng luật dựa vào ngưỡng

Các luật dựa trên các ngưỡng tạo thành một cơ chế triển khai và sử dụng dễ dàng để quản lý lượng tài nguyên được gán cho một ứng dụng được lưu trữ trong CC, AS tài nguyên tạo ra nhu cầu đầu vào. Tuy nhiên, việc tạo ra các luật đòi hỏi sự cố gắng từ bộ phận quản lý ứng dụng (khách hàng), cần chọn số đo hiệu năng phù hợp hoặc kết hợp logic giữa các số đo và cũng như giá trị của một số tham số, chủ yếu là các ngưỡng. Các thực nghiệm trong [60] cho thấy các số đo cụ thể cho ứng dụng, như thời gian chờ trung bình trong hàng đợi, có được hiệu năng tốt hơn so với các số đo hệ thống cụ thể, như tải CPU. Bộ phận quản lý ứng dụng cũng cần được thiết lập các ngưỡng trên (ví dụ: 70%) và ngưỡng dưới (ví dụ: 30%) cho biến số đo hiệu năng (ví dụ: tải CPU). Các ngưỡng là chìa khóa cho hoạt động chính xác của các luật. Đặc biệt, Dutreilh và đồng nghiệp [27] cho thấy về các ngưỡng cần phải được điều chỉnh cẩn thận để tránh vấn đề dao động trong hệ thống (ví dụ như số lượng VM hoặc số lượng CPU được gán).

---

[8] *ClarkNet HTTP Trace: http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html.*
[9] *World Cup 98 Trace: http://ita.ee.lbl.gov/html/contrib/WorldCup.html.*
[10] *Wikipedia traces: http://www.wikibench.eu/?page_id=60/.*

Hầu hết các tác giả và nhà cung cấp dịch vụ đám mây chỉ sử dụng hai ngưỡng cho mỗi số đo hiệu năng. Tuy nhiên, Hasan và đồng nghiệp [42] đã xem xét sử dụng một bộ bốn ngưỡng và hai khoảng thời gian: ThrU (ngưỡng trên); ThrbU (ngưỡng thấp hơn ngưỡng trên một ít); ThrL (ngưỡng dưới); ThroL (ngưỡng cao hơn ngưỡng dưới một chút). Được sử dụng kết hợp, nó có thể xác định xu hướng của số đo hiệu năng (ví dụ: xu hướng tăng lên hoặc giảm xuống) và sau đó thực hiện các quyết định AS tốt hơn.

Điều kiện trong các luật thường dựa trên một hoặc nhiều nhất là hai số đo hiệu năng, phổ biến nhất là tải CPU trung bình của các VM, thời gian đáp ứng hoặc tốc độ yêu cầu đầu vào. Dutreilh và đồng nghiệp [27] và Han và đồng nghiệp [41] đều sử dụng thời gian đáp ứng trung bình của ứng dụng. Ngược lại, Hasan và đồng nghiệp [42] sử dụng các số đo hiệu năng từ nhiều khía cạnh (tính toán, lưu trữ và mạng) hoặc thậm chí là sự tương quan của một số khía cạnh.

Trong hầu hết các trường hợp, các luật sử dụng các số đo trực tiếp thu được từ bộ giám sát, do đó hành động theo cách hoàn toàn bị động. Tuy nhiên, có thể tiến hành phân tích dữ liệu được giám sát trước đó để dự đoán hành vi tương lai của hệ thống và thực hiện các luật theo cách chủ động [56]. Vấn đề này sẽ được đề cập trong phần kỹ thuật sử dụng phân tích chuỗi thời gian.

Thuật toán AS của RightScale đề xuất kết hợp các luật bị động thường xuyên với quá trình đề cử. Nếu phần lớn các VM đồng ý rằng chúng cần điều chỉnh mở rộng/thu nhỏ hoặc điều chỉnh giảm xuống/tăng lên, hành động đó được thực hiện; ngược lại, không có hành động nào được lên kế hoạch. Mỗi VM đề cử để điều chỉnh mở rộng/thu nhỏ hoặc điều chỉnh giảm xuống/tăng lên dựa trên một tập luật được đánh giá riêng. Sau mỗi hành động điều chỉnh, RightScale đề xuất thời gian ổn định 15 phút vì các VM mới thường mất từ 5 đến 10 phút để hoạt động. Kỹ thuật AS này đã được một số tác giả sử dụng. Chieu và đồng nghiệp [22] đề xuất một tập các luật bị động dựa trên số phiên hoạt động, nhưng công trình này đã được mở rộng trong Chieu và đồng nghiệp [21] theo phương pháp RightScale: nếu tất cả các VM có các phiên hoạt động vượt quá mức ngưỡng trên cho trước, một VM mới được cung cấp;

nếu các VM có các phiên hoạt động thấp hơn mức ngưỡng dưới cho trước và có ít nhất một VM không có phiên hoạt động, thì VM đó sẽ ngừng hoạt động.

RightScale[11] đề xuất một biến thể quan trọng khác của phương pháp dựa trên luật đơn giản. Ý tưởng cốt lõi của nó là để cho mỗi VM quyết định có thể thu nhỏ hay mở rộng theo các luật được xác định trước và sau đó sử dụng phương pháp đề cử theo số đông để đưa ra quyết định cuối cùng. Hạn chế chính của RightScale là kỹ thuật phụ thuộc nhiều vào các giá trị ngưỡng do bộ quản lý xác định, và cả về đặc điểm của tải công việc đầu vào.

Để tiết kiệm chi phí, Kupferman và đồng nghiệp [63] và các tác giả khác [19] đưa ra ý tưởng gọi là smart kill. Nhiều nhà cung cấp CC tính chi phí theo giờ, tức là sử dụng một phần cũng được tính tròn là một giờ. Do đó, không nên kết thúc VM trước khi quá một giờ, ngay cả khi tải ít. Ngoài việc giảm chi phí, smart kill cũng có thể cải thiện hiệu năng hệ thống: trong trường hợp tải công việc đầu vào dao động, tránh được các chi phí liên tục cho việc tắt và khởi động VM, cải thiện độ trễ khởi động và giảm các vi phạm SLA.

Sự phổ biến của các luật cũng như phương pháp AS có thể là do sự đơn giản và thực tế là chúng dễ hiểu đối với khách hàng. Tuy nhiên, loại kỹ thuật này cho thấy hai vấn đề chính: tính chất bị động và khó khăn trong việc chọn chính xác tập số đo hiệu năng và các ngưỡng. Hiệu quả của các ngưỡng đó phụ thuộc nhiều vào các thay đổi về tải công việc và có thể yêu cầu điều chỉnh thường xuyên. Để giải quyết vấn đề các ngưỡng tĩnh, Lorido-Botran và đồng nghiệp [69] đưa ra khái niệm về ngưỡng động: các giá trị ban đầu được thiết lập, nhưng chúng tự động sửa đổi theo kết quả của các vi phạm SLA được quan sát. Các luật meta bao gồm việc xác định các ngưỡng được sử dụng như thế nào trong các luật điều chỉnh có thể thay đổi để thích nghi tốt hơn với tải công việc.

Tóm lại, các luật dựa vào ngưỡng có thể được sử dụng để dễ dàng tự động hóa AS của từng ứng dụng cụ thể, đặc biệt trong trường hợp các ứng dụng phổ biến, có thể

[11] *https://support.rightscale.com/12-Guides/RightScale_101/System_Architecture/RightScale_Alert_System/Alerts_based_on_Voting_Tags/Understanding_the_Voting_Process/. Truy cập 09/10/2018.*

dự đoán được. Tuy nhiên, trong trường hợp tải công việc tăng đột biến, kỹ thuật luật dựa vào ngưỡng hoạt động kém hiệu quả, cần xem xét sử dụng các kỹ thuật khác.

1.2.2. Kỹ thuật sử dụng học tăng cường

Học tăng cường nhằm mục đích để cho hệ thống phần mềm học cách thích nghi đáp ứng trong môi trường cụ thể để tối đa hóa lợi ích hoặc phần thưởng nhận được. Kỹ thuật này phù hợp để giải quyết các bài toán điều khiển tự động như AS [7, 13, 26, 27, 29, 37, 45, 65, 98, 111, 114]. Đối với bài toán AS, mục tiêu của thuật toán học là tạo một bảng chỉ định hành động cung cấp hoặc thu hồi tốt nhất theo từng trạng thái. Quá trình học tương tự như phương pháp thử sai. Thuật toán học chọn một hành động cụ thể và sau đó quan sát kết quả. Nếu kết quả là dương, thì bộ AS sẽ có nhiều khả năng thực hiện hành động tương tự vào lần tiếp theo khi nó đối mặt với một tình huống tương tự.

Thuật toán học tăng cường được sử dụng nhiều nhất trong AS là thuật toán học Q. Chi tiết thuật toán học Q và các biến thể của chúng trong AS có thể tham khảo công trình khảo sát của Lorido-Botran và đồng nghiệp [68].

Ba yếu tố cơ bản phải được xác định để áp dụng học tăng cường cho AS: tập hợp hành động A, không gian trạng thái S và hàm phần thưởng R. Hai yếu tố đầu tiên phụ thuộc vào loại điều chỉnh: điều chỉnh theo chiều ngang hay điều chỉnh theo chiều dọc, trong khi đó hàm phần thưởng thường tính đến chi phí tài nguyên được mua (thuê VM, băng thông,...) và chi phí phạt do vi phạm SLA. Trong trường hợp điều chỉnh theo chiều ngang, trạng thái chủ yếu được xác định theo tải công việc đầu vào và số lượng VM. Ví dụ, Tesauro và đồng nghiệp [99] đề xuất sử dụng (w, u_{t-1}, u_t), trong đó w là tổng số yêu cầu người dùng được quan sát trong một khoảng thời gian; u_t và u_{t-1} là số lượng VM được cấp phát cho ứng dụng ở thời điểm hiện tại (t) và ở thời điểm trước đó $(t-1)$; Dutreilh và đồng nghiệp [26] định nghĩa trạng thái (w, u, p), trong đó p là hiệu năng theo thời gian đáp ứng trung bình cho các yêu cầu, được giới hạn bởi giá trị P_{max} được chọn từ các quan sát thực nghiệm. Tập hợp các hành động cho việc điều chỉnh theo chiều ngang thường là ba hành động: thêm một VM mới, loại bỏ một VM hiện có và không làm gì cả.

Kỹ thuật học Q đã đưa ra một số bài toán và được nghiên cứu, giải quyết theo các cách khác nhau [7, 26, 27, 89, 99]:

- Hiệu năng ban đầu kém và thời gian huấn luyện lớn. Hiệu năng thu được trong quá trình huấn luyện trực tuyến có thể thấp không được chấp nhận, cả lúc ban đầu lẫn trong một thời gian huấn luyện dài, trước khi tìm thấy một giải pháp đủ tốt. Thuật toán cần khám phá các trạng thái và hành động khác nhau.

- Không gian trạng thái lớn. Số lượng trạng thái tăng theo cấp số nhân với số biến trạng thái, dẫn đến các vấn đề về khả năng mở rộng. Trong dạng đơn giản nhất, một bảng tra cứu được sử dụng để lưu giá trị riêng biệt cho mỗi cặp trạng thái - hành động có thể. Khi kích thước của bảng này tăng lên, mọi truy cập vào bảng sẽ mất nhiều thời gian hơn, độ trễ cập nhật bảng và lựa chọn hành động.

- Hành động thăm dò và khả năng thích nghi với những thay đổi môi trường. Ngay cả khi giả định rằng một chính sách tối ưu được tìm thấy, các điều kiện môi trường (ví dụ: mô hình tải công việc) có thể thay đổi và chính sách phải được điều chỉnh lại. Với mục đích này, thuật toán học tăng cường thực thi một số hành động thăm dò nhất định, được cho là tối ưu cục bộ. Điều này có thể dẫn đến hiệu năng xấu không mong muốn, nhưng nó là điều cần thiết để thích nghi với chính sách hiện hành. Theo phương pháp này, các thuật toán học tăng cường có thể đối phó với những thay đổi tương đối làm mịn theo hành vi của ứng dụng, nhưng không thể tăng đột ngột theo tải công việc đầu vào.

Vấn đề về hiệu năng thấp là do thiếu thông tin ban đầu và bản chất ngẫu nhiên của chính sách thăm dò. Dutreilh và đồng nghiệp [27] đã sử dụng một phương pháp heuristic tùy chỉnh để hướng dẫn thăm dò không gian trạng thái, và quá trình học kéo dài cho tới 60 giờ. Cùng các tác giả [26] tiếp tục đề xuất một xấp xỉ ban đầu của hàm Q để cập nhật giá trị cho tất cả các trạng thái tại mỗi lần lặp, và cũng làm tăng tốc độ hội tụ đến chính sách tối ưu. Giảm thời gian huấn luyện này cũng có thể được giải quyết bằng chính sách truy cập một số trạng thái ở mỗi bước [89] hoặc sử dụng tác nhân học song song [7]. Trong phần sau, mỗi tác nhân không cần phải truy cập vào mọi trạng thái và hành động; thay vào đó, nó có thể học giá trị của các trạng thái không

được truy cập từ các tác nhân lân cận. Một cách tiếp cận hoàn toàn khác để tránh hiệu năng ban đầu thấp do huấn luyện trực tuyến bao gồm việc sử dụng một mô hình thay thế (ví dụ như một mô hình hàng đợi) để điều khiển hệ thống, trong khi đó mô hình học tăng cường được huấn luyện off-line trên dữ liệu đã thu thập [99].

Học tăng cường có thể được sử dụng kết hợp với các phương pháp khác như lý thuyết điều khiển. Zhu và Agrawal [114] kết hợp một bộ điều khiển PID với một tác nhân học tăng cường chịu trách nhiệm ước lượng thành phần đạo hàm. Bộ điều khiển hướng dẫn việc điều chỉnh tham số của các ứng dụng (ví dụ: kích thước hình ảnh) để đáp ứng SLA. Sau đó, các tài nguyên ảo: CPU và bộ nhớ, được cấp pháp tự động theo sự thay đổi của các tham số thích nghi.

Điều quan trọng là phải biết khả năng nổi bật của các thuật toán học tăng cường để có được chính sách quản lý tốt nhất cho mục tiêu, mà không có bất kỳ tri thức nào. Khách hàng không cần xác định một mô hình cụ thể cho ứng dụng; thay vào đó, nó được học trực tuyến và thích nghi nếu các điều kiện của ứng dụng, tải công việc hoặc thay đổi hệ thống. Như vậy, các kỹ thuật học tăng cường có thể là một phương pháp đầy hứa hẹn để giải quyết nhiệm vụ AS với các ứng dụng tổng quát. Trong lĩnh vực nghiên cứu mở này, các nỗ lực cần được giải quyết nhằm cung cấp khả năng thích nghi phù hợp cho việc tăng đột ngột của tải công việc đầu vào, và cũng để đối phó với không gian trạng thái và hành động liên tục.

Trong luận án này sử dụng kỹ thuật học tăng cường và kết hợp với một số kỹ thuật khác để đề xuất giải pháp AS hiệu quả trong CC nhằm giải quyết những hạn chế trên. Chi tiết được trình bày trong Chương 4.

1.2.3. Kỹ thuật sử dụng lý thuyết hàng đợi

Trong lĩnh vực nghiên cứu, cả hai mô hình hàng đợi đơn giản và các mạng hàng đợi phức tạp hơn đã được sử dụng rộng rãi để phân tích hiệu năng của các ứng dụng và hệ thống máy tính.

Ali-Eldin và đồng nghiệp [1, 2] mô hình một ứng dụng lưu trữ trên đám mây dưới dạng một hàng đợi G/G/n, trong đó n là số lượng máy chủ. Mô hình có thể được giải quyết để tính toán, ví dụ: các tài nguyên cần thiết để xử lý tải công việc đầu vào đã cho

λ hoặc thời gian đáp ứng trung bình cho các yêu cầu, được cung cấp cấu hình máy chủ. Công trình của Bi và đồng nghiệp [10] đã đề xuất một mô hình kết hợp, trong đó tầng đầu tiên được mô hình hóa như một hàng đợi M/M/n trong khi các tầng khác được mô hình hóa là hàng đợi M/M/1. Khác với lý thuyết hàng đợi truyền thống, Salah và đồng nghiệp [93] sử dụng một phương pháp chuỗi Markov nhúng để mô hình hóa hệ thống hàng đợi.

Các mạng hàng đợi cũng có thể được sử dụng để mô hình hóa các ứng dụng đàn hồi, biểu diễn mỗi VM (máy chủ) là một hàng đợi riêng biệt. Ví dụ: Urgaonkar và đồng nghiệp [101] sử dụng một mạng các hàng đợi G/G/1 (một hàng đợi cho một VM). Họ sử dụng biểu đồ để dự đoán tải công việc lúc cao điểm. Dựa trên giá trị này và mô hình hàng đợi, số lượng VM (máy chủ) trên mỗi tầng ứng dụng được tính toán. Con số này có thể được điều chỉnh bằng cách sử dụng phương pháp bị động. Hạn chế của mô hình này là cung cấp cho tải lúc cao điểm dẫn đến mức sử dụng tài nguyên cao lên.

Các ứng dụng đa tầng có thể được nghiên cứu bằng cách sử dụng một hoặc nhiều hàng đợi trên mỗi tầng. Zhang và đồng nghiệp [112] xem xét số lượng người dùng hạn chế, và do đó, họ đã sử dụng hệ thống mạng hàng đợi đóng; mô hình này có thể được giải quyết hiệu quả bằng cách sử dụng MVA. Han và đồng nghiệp [40] đã mô hình ứng dụng đa tầng là một hệ thống mạng hàng đợi G/G/n mở (một hàng đợi cho mỗi tầng); mô hình này được sử dụng để ước tính số lượng VM cần được thêm vào hoặc bị loại bỏ khỏi tầng bị nút cổ chai và chi phí liên quan (mức sử dụng VM được tính theo mỗi phút, thay vì mô hình chi phí tính theo mỗi giờ). Cuối cùng, Bacigalupo và đồng nghiệp [5] xem xét một mạng hàng đợi ba tầng, giải quyết từng tầng để tính toán thời gian đáp ứng trung bình.

Các phương pháp đã thảo luận được sử dụng như một phần của giai đoạn phân tích trong vòng lặp điều khiển MAPE. Nhiều kỹ thuật khác nhau có thể được sử dụng để thực hiện giai đoạn lập kế hoạch, chẳng hạn như bộ điều khiển dự đoán [2] hoặc thuật toán tối ưu hóa (ví dụ: phân bố các máy chủ cho các ứng dụng khác nhau, trong khi đó tối đa hóa doanh thu [104]). Thông tin cần thiết cho mô hình hàng đợi, như tải công việc đầu vào (số lượng yêu cầu) hoặc thời gian phục vụ có thể thu được bằng

cách theo dõi/giám sát trực tuyến [40, 101] hoặc ước lượng bằng các phương pháp khác nhau. Ví dụ: Zhang và đồng nghiệp [112] đã sử dụng phép tính xấp xỉ dựa trên hồi quy để ước lượng nhu cầu CPU, dựa trên số lượng và loại các yêu cầu của khách hàng (duyệt, đặt hàng hoặc mua sắm).

Các mô hình hàng đợi đã được sử dụng rộng rãi để mô hình hóa các ứng dụng và hệ thống. Chúng thường áp đặt một kiến trúc cố định, và bất kỳ thay đổi nào trong cấu trúc hoặc các tham số đều cần phải giải quyết mô hình (với các công cụ phân tích hoặc dựa trên mô phỏng). Vì lý do này, chúng không dễ dàng khi được sử dụng với các ứng dụng đàn hồi (biến động) mà phải đối phó/xử lý với tải công việc đầu vào thay đổi và một nhóm tài nguyên khác nhau. Ngoài ra, một hệ thống hàng đợi là một công cụ phân tích, đòi hỏi các thành phần thêm vào để thực hiện một bộ AS hoàn chỉnh. Các mô hình hàng đợi có thể hữu ích cho một số trường hợp ứng dụng cụ thể, ví dụ: khi mối quan hệ giữa số lượng yêu cầu và tài nguyên cần thiết gần tuyến tính. Mặc dù có nhiều cố gắng để mô hình hóa các ứng dụng đa tầng phức tạp hơn, nhưng lý thuyết hàng đợi có thể không phải là lựa chọn tốt nhất khi cố gắng thiết kế một hệ thống AS tổng quát/đa năng.

Mặc dù có những hạn chế như trên, nhưng việc sử mô hình hàng đợi hay mạng hàng đợi để mô hình hóa môi trường CC cũng mang lại một số hiệu quả nhất định. Trong luận án này, đã nghiên cứu và đề xuất một mô hình mạng hàng đợi cho CC. Chi tiết được trình bày trong Chương 3.

1.2.4. Kỹ thuật sử dụng lý thuyết điều khiển

Các bộ điều khiển khuếch đại cố định, bao gồm PID (Proportional Integral Derivate), PI (Proportional Integral) và I (Integral), là các kiểu bộ điều khiển đơn giản nhất và đã được sử dụng rộng rãi trong nghiên cứu. Lim và đồng nghiệp [66, 67] sử dụng bộ điều khiển I để điều chỉnh số lượng VM dựa trên mức sử dụng CPU trung bình, trong khi Park và Humphrey [84] áp dụng bộ điều khiển PI để quản lý tài nguyên theo yêu cầu của công việc hàng loạt, dựa trên tiến trình thực hiện của họ. Ví dụ, trong [84] một mô hình được xây dựng để ước lượng tiến độ của một công việc đối với các tài nguyên được cung cấp. Zhu và Agrawal [114] dựa vào một tác nhân học

tăng cường để ước lượng thành phần đạo hàm của một bộ điều khiển PID. Với việc huấn luyện thử sai, tác nhân học tăng cường học cách tối thiểu tổng số lỗi bình phương của các biến điều khiển (các tham số thích nghi) mà không vi phạm các ràng buộc về thời gian và ngân sách thời gian.

Kỹ thuật điều khiển thích nghi cũng khá phổ biến. Ví dụ: Ali-Eldin và đồng nghiệp [2] đề xuất kết hợp hai bộ điều khiển chủ động, thích nghi để điều chỉnh giảm xuống, sử dụng các tham số khuếch đại động dựa vào tải công việc đầu vào. Một phương pháp bị động được sử dụng để điều chỉnh tăng lên. Cùng các tác giả đề xuất trong [1] một bộ điều khiển thích nghi, điều chỉnh, sử dụng phương pháp chủ động cho cả điều chỉnh tăng lên và điều chỉnh giảm xuống, và có tính đến thời gian khởi động VM. Như trình bày ở trên, kỹ thuật điều khiển thích nghi dựa vào việc sử dụng các mô hình hiệu năng. Padala và đồng nghiệp [83] đề xuất một bộ điều khiển thích nghi MIMO sử dụng ARMA bậc hai để mô hình hóa mối quan hệ phi tuyến tính và việc thay đổi thời gian giữa việc cấp phát tài nguyên và hiệu năng chuẩn hóa của nó. Bộ điều khiển có thể điều chỉnh mức sử dụng CPU và đĩa I/O. Bodík và đồng nghiệp [11] kết hợp các phương pháp splines làm mịn (được sử dụng để ánh xạ tải công việc và số lượng máy chủ đến hiệu năng ứng dụng) với bộ điều khiển thích nghi lập lịch khuếch đại. Kalyvianaki và đồng nghiệp [55] đã thiết kế các bộ điều khiển SISO và MIMO khác nhau để xác định việc cấp phát CPU cho VM, dựa vào bộ lọc Kalman, trong khi đó [33] sử dụng mô hình Kriging để dự đoán thời gian hoàn thành công việc như một hàm của số VM, số lượng các yêu cầu đến và các công việc chờ trong hàng đợi tại nút chủ.

Các mô hình mờ đã được sử dụng như mô hình hiệu năng trong các hệ thống điều khiển để liên kết tải công việc (biến đầu vào) và các tài nguyên cần thiết (biến đầu ra). Đầu tiên, cả các biến đầu vào và đầu ra của hệ thống đều được ánh xạ thành các tập mờ. Ánh xạ này được xác định bởi hàm thành viên xác định một giá trị nằm trong khoảng $[0,1]$. Một mô hình mờ được dựa trên một tập hợp các luật, có liên quan đến các biến đầu vào (điều kiện của luật), với các biến đầu ra (kết quả của luật). Quá trình chuyển các giá trị đầu vào thành một hay nhiều tập mờ được gọi là mờ hóa. Giải

mờ là phép biến đổi ngược thành một giá trị số duy nhất biểu diễn tốt nhất cho các giá trị mờ của các biến đầu ra. Một hệ thống điều khiển phụ thuộc vào mô hình mờ dựa trên luật được gọi là bộ điều khiển mờ. Thông thường, các tập luật và hàm thành viên của mô hình mờ là cố định khi thiết kế, và do đó, bộ điều khiển không thể thích nghi với tải công việc thay đổi nhiều. Một phương pháp thích nghi có thể được sử dụng, trong đó mô hình mờ được cập nhật liên tục dựa trên thông tin được giám sát trực tuyến [106, 110]. Xu và đồng nghiệp [110] đã áp dụng bộ điều khiển mờ thích nghi cho tầng logic nghiệp vụ của ứng dụng web và ước lượng tải CPU cần thiết cho tải công việc đầu vào. Một phương pháp tương tự trong [106], nhưng ở đây các tác giả tập trung vào tầng cơ sở dữ liệu. Họ đề xuất sử dụng mô hình mờ để dự đoán nhu cầu tài nguyên trong tương lai; tuy nhiên, họ sử dụng tải công việc của bước thời gian hiện tại t, để tính toán tài nguyên cần thiết r_{t+1} của bước thời gian $t + 1$, dựa trên giả sử là không có thay đổi đột ngột nào xảy ra trong khoảng thời gian của một bước thời gian.

Một cải tiến tiếp theo là bộ điều khiển mờ nơron, sử dụng mạng nơron bốn lớp để biểu diễn mô hình mờ. Mỗi nút trong lớp đầu tiên tương ứng với một biến đầu vào. Lớp thứ hai xác định hàm thành viên của mỗi biến đầu vào tạo ra tập mờ (quá trình mờ). Mỗi nút trong lớp ba thể hiện phần điều kiện tiên quyết của một luật logic mờ. Cuối cùng, lớp đầu ra hoạt động như một bộ giải mờ, nó chuyển đổi các kết quả mờ từ lớp ba thành đầu ra (kết quả) là giá trị số để điều chỉnh tài nguyên. Ở các bước ban đầu, mạng nơron chỉ chứa các lớp đầu vào và đầu ra. Các hàm thành viên và các luật được tạo tự động thông qua việc học cấu trúc và các tham số. Lama và Zhou [64] dựa trên bộ điều khiển mờ nơron, có khả năng tự xây dựng cấu trúc của nó (cả luật mờ và hàm thành viên) và điều chỉnh các tham số của nó thông qua việc học trực tuyến nhanh (bộ điều khiển cấu hình lại).

Bộ điều khiển cuối cùng theo phương pháp chủ động là MPC. Roy và đồng nghiệp [90] đã kết hợp với mô hình ARMA để dự báo tải công việc, với bộ điều khiển look-ahead để tối ưu hóa bài toán cấp phát tài nguyên. Các mô hình mờ cũng có thể được sử dụng để tạo ra một bộ điều khiển dự báo mô hình mờ [105].

Sự phù hợp của các bộ điều khiển cho nhiệm vụ AS phụ thuộc rất lớn vào loại bộ điều khiển và tính động của hệ thống đích. Ý tưởng về việc có một bộ điều khiển tự động hóa quá trình thêm/loại bỏ tài nguyên là rất hấp dẫn, nhưng vấn đề vẫn là cách tạo ra một mô hình hiệu năng đáng tin cậy để ánh xạ các biến đầu vào và đầu ra. Bộ điều khiển bị động đơn giản có thể được sử dụng cho các ứng dụng dễ dàng dự đoán nhu cầu. Tuy nhiên, có vẻ như cần tập trung cố gắng vào cả mô hình thích nghi và MPCs để có thể thích nghi với mô hình ứng dụng và có thể phù hợp hơn để tạo ra giải pháp AS tổng quát.

Trong luận án này, đã sử dụng bộ điều khiển mờ kết hợp với kỹ thuật học tăng cường để thiết kế giải pháp AS tài nguyên trong CC hiệu quả được trình bày trong Chương 4.

1.2.5. Kỹ thuật sử dụng phân tích chuỗi thời gian

Trong bối cảnh các ứng dụng đàn hồi, phân tích chuỗi thời gian đã được áp dụng chủ yếu để dự đoán tải công việc hoặc mức sử dụng tài nguyên. Mô hình trung bình di chuyển (MA) đơn giản có thể được sử dụng cho mục đích này [38]. Phương pháp tự hồi quy (AR) cũng được sử dụng để dự báo tài nguyên hoặc tải công việc. Roy và đồng nghiệp [90] đã áp dụng mô hình AR để dự đoán tải công việc, dựa trên ba quan sát cuối cùng. Giá trị dự đoán sau đó được sử dụng để ước lượng thời gian đáp ứng. Một bộ điều khiển tối ưu có thời gian đáp ứng này như là một đầu vào và tính toán cấp phát tài nguyền tốt nhất, có tính đến chi phí của các vi phạm SLA, cho thuê tài nguyên và cấu hình lại. Các mô hình ARMA có thể nắm giữ các đặc trưng của một chuỗi thời gian như tải công việc đầu vào hoặc mức sử dụng CPU. Fang và đồng nghiệp [30] thấy sự hữu ích khi dự đoán mức sử dụng CPU trong tương lai của các VM. Tuy nhiên, họ chú ý đến chi phí tính toán của kỹ thuật này, bao gồm lựa chọn giá trị p và q, ước lượng các hệ số của từng thành phần và các tham số khác.

Hầu hết các kỹ thuật phân tích chuỗi thời gian đã được áp dụng riêng biệt cho điều chỉnh theo chiều dọc hoặc điều chỉnh theo chiều ngang. Dự báo chuỗi thời gian (liên quan đến việc ra quyết định chủ động) có thể được kết hợp với các kỹ thuật bị động. Ví dụ, Iqbal và đồng nghiệp [46] đã đề xuất một kỹ thuật điều chỉnh lai (kết

hợp) sử dụng các luật bị động để điều chỉnh tăng lên (dựa vào mức sử dụng CPU) và một phương pháp dựa vào hồi quy để điều chỉnh giảm xuống. Sau đó cố định số khoảng thời gian trong đó thời gian đáp ứng được thỏa mãn, họ tính toán số lượng yêu cầu cho các VM của tầng ứng dụng và của tầng cơ sở dữ liệu bằng cách sử dụng hồi quy đa thức (bậc hai).

Các kỹ thuật phân tích chuỗi thời gian rất hấp dẫn để triển khai các bộ AS, vì chúng có thể dự đoán các nhu cầu trong tương lai cho các ứng dụng đàn hồi. Luận án không đề cập đến kỹ thuật tự động điều chỉnh này.

KẾT LUẬN CHƯƠNG 1:

Chương 1 đã trình bày về cơ sở lý thuyết được sử dụng trong luận án như khái niệm về AS, vòng lặp điều khiển MAPE, phương pháp điều chỉnh trong CC, các kỹ thuật AS hiện có, kiến trúc ứng dụng đa tầng, nền tảng thực nghiệm và tải công việc. Các công trình liên quan đến luận án được trình bày theo từng kỹ thuật AS. Trong đó, luận án tập trung vào các kỹ thuật sử dụng lý thuyết hàng đợi, lý thuyết điều khiển và học tăng cường. Trong chương tiếp theo luận án sẽ nghiên cứu và tìm hiểu các kỹ thuật căn bản trong CC có ảnh hưởng đến hoạt động dịch vụ AS, như kỹ thuật cân bằng tải và di trú trong CC.

CHƯƠNG 2 : CÁC ĐỀ XUẤT CƠ CHẾ CƠ BẢN CÓ ẢNH HƯỞNG ĐẾN TỰ ĐỘNG ĐIỀU CHỈNH TÀI NGUYÊN TRONG ĐIỆN TOÁN ĐÁM MÂY

Chương này, luận án trình bày về các kỹ thuật cơ bản trong CC có thể ảnh hưởng đến chất lượng dịch vụ AS, như các cơ chế cân bằng tải, di trú và đề xuất một kỹ thuật cân bằng tải động hiệu quả và một kỹ thuật di trú hiệu quả trong môi trường CC. Cuối cùng là phần đánh giá ảnh hưởng giữa các kỹ thuật cân bằng tải với cơ chế AS tài nguyên trong CC để duy trì hiệu năng CC và khách hàng có được dịch vụ tốt nhất. Lựa chọn kỹ thuật cân bằng tải tốt sẽ giúp cho cơ chế AS trong CC hoạt động hiệu quả, tránh việc cung cấp quá mức hoặc cung cấp quá ít. Ngoài ra, để hệ thống có thể cân bằng tải tốt hơn thì kỹ thuật di trú trong CC sẽ giúp cải thiện vấn đề này. Chương này được tổng hợp từ các công trình TCNN1, TCTN1, HNQT1, HNQT2.

2.1. ĐẶT VẤN ĐỀ

CC là môi trường tính toán phân tán bao gồm nhiều đơn vị tính toán vật lý chia sẻ tải với nhau. Ngay từ những ngày đầu của CC, vấn đề điều phối tải công bằng giữa các đơn vị tính toán vật lý đã được đặt ra và các cơ chế cân bằng tải đã được đề xuất nhằm giải quyết vấn đề này. Cân bằng tải đóng vai trò là bộ phận tiếp nhận và điều phối tải ở lối vào của hệ thống, mà muốn điều phối tốt cần phải nhận biết chính xác trạng thái của các thành phần tính toán vật lý bên trong cụm tính toán. Đây là thách thức lớn đối với nhiệm vụ cân bằng tải ở lối vào. Mục 2.2 sẽ đề xuất một kỹ thuật cân bằng tải động hiệu quả để đáp ứng các yêu cầu trên và được công bố trong công trình HNQT1.

Khi bộ cân bằng tải thực hiện nhiệm vụ phân bố tải vẫn chưa được công bằng, khi đó rất cần một cơ chế hỗ trợ từ bên trong và chiến lược di trú có thể đóng vai trò này. Mục tiêu của các kỹ thuật di trú để quản lý tài nguyên động, tiết kiệm điện năng và duy trì hệ thống hoạt động cân bằng hơn, tránh tình trạng mất cân bằng trong hệ thống CC. Một kỹ thuật di trú hiệu quả được đề xuất trong mục 2.3, và được công bố trong công trình TCTN1 và HNQT2.

Trong khi cân bằng tải và di trú thực hiện điều phối tải để sử dụng tài nguyên sẵn có hiệu quả thì cơ chế AS tài nguyên thực hiện bổ sung tài nguyên để chống quá

tải nhằm bảo đảm QoS đã cam kết. Như vậy, trong môi trường CC có hỗ trợ cơ chế AS tài nguyên lại có thêm một cơ chế tác động đến hệ thống bên cạnh các cơ chế cân bằng tải và di trú. Câu hỏi đặt ra là các cơ chế này có bị chi phối lẫn nhau hay không. Để thực hiện AS tài nguyên hiệu quả cũng đòi hỏi cơ chế AS phải nhận thức rõ tình trạng của các đơn vị tính toán vật lý để dự báo chính xác và đưa ra quyết định đúng. Tuy nhiên trạng thái của các đơn vị tính toán vật lý liên tục thay đổi bởi tác động của cơ chế cân bằng tải. Vì vậy ảnh hưởng của cân bằng tải đến tính hiệu quả của AS tài nguyên cần phải được xem xét và chỉ lựa chọn cơ chế cân bằng tải phù hợp với AS tài nguyên. Việc sử dụng kỹ thuật cân bằng tải tốt, phù hợp trong môi trường CC có cơ chế AS sẽ giúp hệ thống hoạt động hiệu quả hơn và tiết kiệm chi phí, vấn đề này được đánh giá trong mục 2.4 và được công bố trong công trình TCNN1.

Phần tiếp theo sẽ trình bày chi tiết từng vấn đề nêu trên.

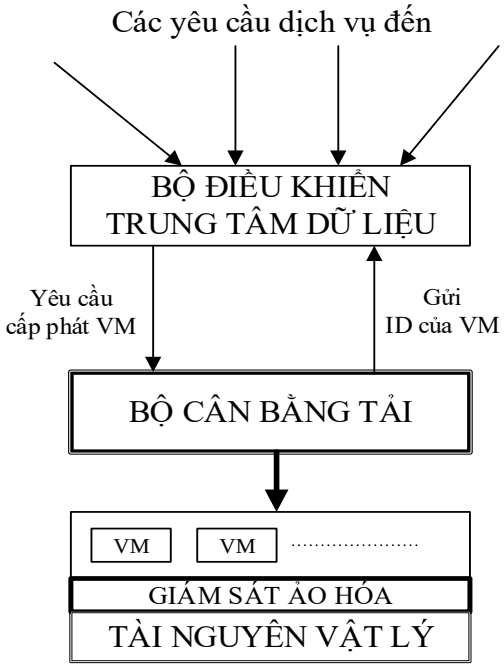
2.2. ĐỀ XUẤT MỘT KỸ THUẬT CÂN BẰNG TẢI ĐỘNG HIỆU QUẢ TRONG ĐIỆN TOÁN ĐÁM MÂY

Những yếu tố quan trọng khi xem xét phát triển các kỹ thuật cân bằng tải động là ước lượng tải, so sánh tải, khả năng ổn định của hệ thống khác nhau, hiệu suất của hệ thống, tương tác giữa các nút, tính chất của công việc được chuyển giao. Tải có thể được xem xét trong các thuật ngữ của tải CPU, số lượng bộ nhớ sử dụng, độ trễ hoặc tải trên mạng. Khi một khối lượng tải cho trước được gửi đến cho bất kỳ cụm các nút yêu cầu được thực thi. Cần phải có một cơ chế để lựa chọn các nút có các nguồn tài nguyên phù hợp với các yêu cầu đến. Lập lịch là một thành phần hay cơ chế chịu trách nhiệm chọn một nút hay cụm nút. Cơ chế này sẽ xem xét trạng thái cân bằng tải. Vì vậy, lập lịch cần các thuật toán cân bằng tải để giải quyết vấn đề như vậy.

Hình 2.1 là một mô hình cân bằng tải tổng quát. Khi yêu cầu (tác vụ) đến, bộ cân bằng tải xác định VM thích hợp, gửi ID của VM tới bộ điều khiển trung tâm dữ liệu. Bộ điều khiển trung tâm dữ liệu phân bố tải đến VM đó để xử lý.

Mục đích của kỹ thuật cân bằng tải là cải thiện hiệu năng của toàn bộ hệ thống một cách cơ bản; giảm thiểu thời gian chờ của công việc; có một kế hoạch cung cấp trong trường hợp hệ thống bị lỗi thậm chí một phần; duy trì sự ổn định và thích nghi

sự biến đổi trong tương lai của hệ thống; đảm bảo những công việc nhỏ không bị thiếu trong thời gian dài; đồng thời cũng phải đảm bảo hạn chế tình trạng một nút có tải quá nhiều trong khi những nút khác chỉ có ít tải [62, 77]. Vì vậy, nhiều kỹ thuật cân bằng tải đã được phát triển trong nhiều năm qua nhưng không có một kỹ thuật nào là thích hợp cho tất cả các ứng dụng, các hệ thống tính toán phân tán. Việc lựa chọn một kỹ thuật cân bằng tải tương ứng phụ thuộc tham số các ứng dụng cũng như các tham số phần cứng.



Hình 2.1: Mô hình cân bằng tải

Kỹ thuật cân bằng tải còn phụ thuộc vào mức áp dụng, đó là ở mức PM hay là mức VM. Trong phần tiếp theo sẽ tìm hiểu về các mức này.

2.2.1. Cơ chế chia sẻ tài nguyên trong điện toán đám mây

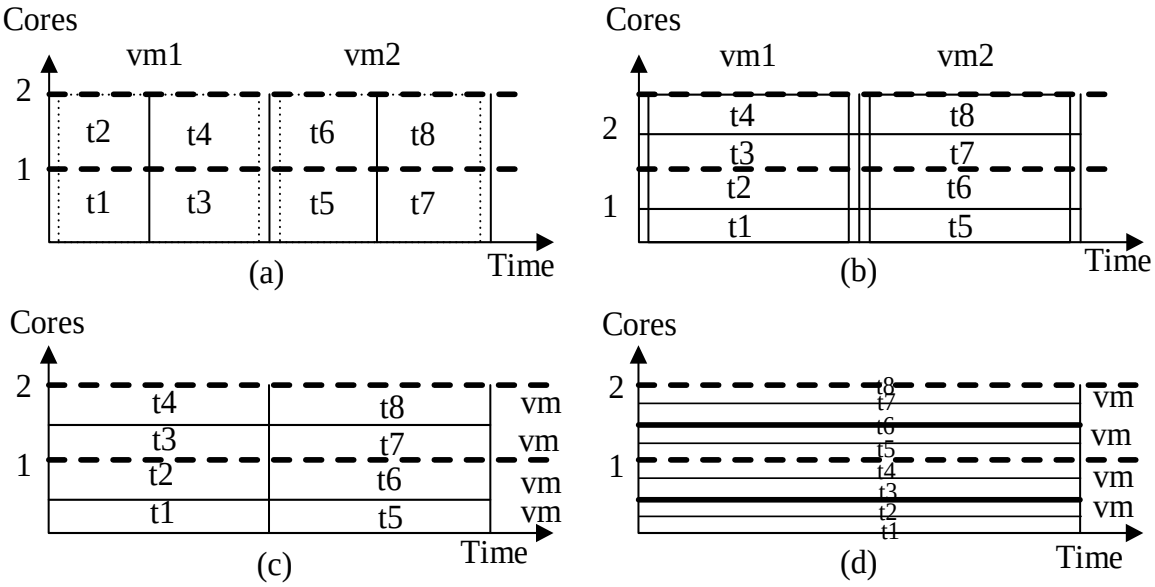
Kỹ thuật cân bằng tải trong CC có thể áp dụng ở những mức khác nhau, nó phụ thuộc vào việc muốn cân bằng tải những gì. Điều này có thể được nghiên cứu và phát hiện ở các mức độ khác nhau như: giả sử có một cơ sở hạ tầng SaaS, trong đó có nhiều yêu cầu về dịch vụ được tiếp nhận. Các yêu cầu này phải được phân phối tới các VM sẵn có để xử lý. Vấn đề là làm thế nào phân phối các yêu cầu đó sao cho thời gian đáp ứng yêu cầu là nhỏ nhất, làm thế nào để quyết định số lượng và đặc điểm của các VM xử lý cho các yêu cầu này. Mặt khác, cơ sở hạ tầng CC để xử lý các yêu

cầu có thể được chia sẻ giữa các trung tâm dữ liệu khác nhau của cùng hoặc khác nhà cung cấp dịch vụ. Do đó, làm thế nào để phân phối các yêu cầu giữa chúng để thời gian thực thi được tối thiểu. Trong trường hợp này, các nhà cung cấp dịch vụ khác nhau có thể tính phí khác nhau cho các nguồn tài nguyên và để giải quyết vấn đề này cần phải được xem xét với cùng kỹ thuật cân bằng tải.

Do đó, để thực hiện các chính sách cân bằng tải phụ thuộc vào cách người dùng muốn thực hiện ở mức nào. Chẳng hạn như, nếu người dùng muốn trung tâm dữ liệu cân bằng tải các PM liên quan đến VM, tức là chính sách PM cung cấp các VM, gọi là cân bằng tải ở mức PM. Nếu người dùng muốn cân bằng tải của các VM đang chạy ứng dụng, tức là mỗi VM chia sẻ các nguồn tài nguyên nhận được từ PM cho các tác vụ (yêu cầu) hay dịch vụ ứng dụng đang chạy bên trong chúng, cân bằng tải ở mức này gọi là mức VM. Có hai chính sách lập lịch áp dụng ở mức PM để thực hiện cung cấp các VM là Time-shared (TS) hoặc Space-share (SS) và cũng có hai chính sách lập lịch áp dụng ở mức VM để thực hiện cung cấp cho các tác vụ là TS hoặc SS.

Để minh họa sự khác nhau giữa các chính sách này và hiệu quả của chúng về mặt hiệu suất dịch vụ ứng dụng, trong Hình 2.2 trình bày một trường hợp cung cấp VM đơn giản. Trong ví dụ này, một PM có hai core CPU nhận yêu cầu lưu giữ hai VM. Mỗi VM yêu cầu hai core và có kế hoạch lưu giữ bốn đơn vị tác vụ. Cụ thể, các tác vụ t1, t2, t3, t4 được lưu giữ trong VM_1, và các tác vụ t5, t6, t7 và t8 được lưu giữ trong VM_2. Mỗi tác vụ yêu cầu một core xử lý.

Hình 2.2(a) trình bày một trường hợp cung cấp, trong đó chính sách SS được áp dụng cho việc phân bổ cho cả VM và các đơn vị tác vụ. Vì mỗi VM yêu cầu hai core, với chính sách SS chỉ một VM có thể chạy ở một thời gian nhất định. Vì vậy, VM_2 chỉ có thể được cấp phát core để xử lý khi VM_1 hoàn thành thực thi các đơn vị tác vụ của nó. Tương tự xảy ra đối với các tác vụ cung cấp bên trong mỗi VM. Vì mỗi đơn vị tác vụ chỉ yêu cầu một core, do đó hai trong số các tác vụ của mỗi VM có thể chạy đồng thời. Trong suốt thời gian này các tác vụ còn lại phải chờ đợi trong hàng đợi thực thi. Do đó, sử dụng chính sách SS, thời gian hoàn thành dự kiến của một tác vụ p được quản lý bởi VM i được tính theo công thức (2.2).



Hình 2.2: Ảnh hưởng của các chính sách khác nhau đối với việc thực thi các tác vụ [15]

Trong Hình 2.2(b), chính sách SS được áp dụng để phân bổ VM tới các PM và chính sách TS để phân bổ các đơn vị tác vụ tới core xử lý bên trong một VM. Do đó, trong suốt thời gian sống của VM, tất cả các tác vụ được giao tới nó được chuyển theo bối cảnh động. Bằng cách sử dụng chính sách TS, thời gian hoàn thành dự kiến của một tác vụ p được quản lý bởi một VM i được tính theo công thức (2.3).

Trong Hình 2.2 (c), VM sử dụng chính sách cung cấp TS, trong khi các tác vụ được cung cấp dựa trên chính sách SS. Trong trường hợp này, mỗi VM nhận một khe thời gian trên mỗi core xử lý, VM sau đó phân phối các khe thời gian giữa các đơn vị tác vụ trên cơ sở SS. Vì các core được chia sẻ, số lượng năng lực xử lý sẵn sàng để sử dụng cho một VM có thể thay đổi. Điều này xác định các VM thích hợp hoạt động trên một PM. Cũng như các đơn vị tác vụ được gán dựa trên chính sách SS, có nghĩa là tại bất kỳ thời gian cụ thể chỉ một tác vụ có thể được sử dụng core xử lý.

Cuối cùng Hình 2.2(d), chính sách phân bổ TS áp dụng cho cả VM và tác vụ. Do đó, công suất xử lý được chia sẻ đồng thời bởi các VM và việc nhận chia sẻ này của mỗi VM được chia sẻ giữa các tác vụ của nó. Trường hợp này, không có trễ xếp hàng liên quan tới các tác vụ.

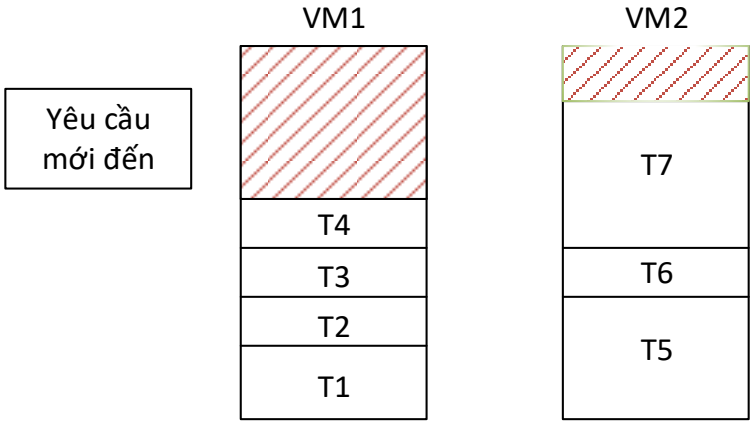
2.2.2. Phân tích kỹ thuật cân bằng tải động được đề xuất

Trong công trình [49] đưa ra thuật toán cân bằng tải giám sát tích cực (gọi tắt là: AMLB-Old). Cách thức hoạt động của AMLB-Old là nắm bắt thông tin về mỗi VM và số lượng yêu cầu hiện đang được cấp phát cho mỗi VM. Khi có một yêu cầu (tác vụ) mới yêu cầu cấp phát đến VM để xử lý, AMLB-Old sẽ tìm VM có số lượng yêu cầu đang xử lý ít nhất để cấp phát. Nếu có nhiều VM thỏa mãn, thì AMLB-Old sẽ chọn VM đầu tiên thỏa mãn và AMLB-Old trả về ID của VM cho bộ điều khiển trung tâm dữ liệu. Bộ điều khiển trung tâm dữ liệu gửi yêu cầu tới VM có ID đó để xử lý. AMLB-Old báo cho bộ điều khiển trung tâm việc cấp phát mới và các yêu cầu được gửi tới VM đó.

Tuy nhiên, thuật toán AMLB-Old luôn tìm thấy VM có số lượng yêu cầu (tác vụ) đang xử lý là ít nhất để gán yêu cầu mới đến, nhưng thuật toán không kiểm tra xem khả năng còn lại của VM đó có xử lý tiếp được yêu cầu mới gửi đến không và không xem xét đến kích thước của mỗi yêu cầu cần được xử lý (vì vậy một số VM có tải nặng, tức là kích thước mỗi yêu cầu lớn mặc dù số lượng yêu cầu bền trong VM đó là ít nhất). Hiện cũng có một số cải tiến cho thuật toán AMLB-Old như [25, 36, 61]. Nhưng các công trình này mới chỉ đề xuất một thuật toán cân bằng tải hiệu quả nhằm cải thiện thời gian đáp ứng và sử dụng hiệu quả các tài nguyên sẵn có, mà chưa xem xét đến kích thước của tải.

Do thuật toán AMLB-Old mới chỉ dựa trên công suất xử lý đã có của VM và số lượng yêu cầu được gán hiện tại, chưa xem xét đến công suất còn lại thực tế của VM và kích thước yêu cầu xử lý. Điều này có thể dẫn đến việc phân bố các yêu cầu cho VM không phù hợp, không xử lý được yêu cầu mới gửi đến. Một VM với công suất xử lý mạnh nhất được lựa chọn trước tiên và có số yêu cầu được giao ít nhất có thể không phải là VM có thể xử lý được các yêu cầu tiếp theo. Như đã đề cập ở trên, công suất xử lý còn lại của VM tùy thuộc vào chính sách lập lịch ở hai mức. Vì vậy, số lượng các yêu cầu hiện đang được cấp phát cho VM không phản ánh sức mạnh còn lại thực sự của VM đó.

Xem ví dụ Hình 2.3, với hai VM: VM_1, VM_2. Hiện tại, VM_1 chứa 4 yêu cầu T1, T2, T3, T4 và VM2 chứa 3 yêu cầu T5, T6, T7. VM1 chứa số yêu cầu nhiều hơn VM_2, nhưng công suất xử lý còn lại của VM_2 lại ít hơn VM_1. Khi một yêu cầu mới đến, theo thuật toán AMLB-Old, yêu cầu mới này sẽ được cấp phát cho VM_2 để xử lý. Do đó, VM_2 có thể bị quá tải, mặc dù số lượng yêu cầu của nó là ít hơn.



Hình 2.3: Ví dụ có 2 VM đang hoạt động như trên

Do vậy, luận án đã đề xuất một thuật toán mới, gọi tắt là AMLB-New, để khắc phục hạn chế trên bằng cách xem xét đến kích thước của các yêu cầu và khả năng xử lý còn lại của mỗi VM để cấp phát yêu cầu tiếp theo, cũng như làm giảm thiểu thời gian đáp ứng và thời gian xử lý trong hệ thống.

Sức mạnh xử lý phụ thuộc vào chính sách lập lịch ở hai mức: mức PM và mức VM. Lập lịch các yêu cầu được thực hiện sau khi các tài nguyên được cấp phát cho tất cả các thực thể đám mây. Thứ nhất, ở mức PM có thể xác định toàn bộ sức mạnh xử lý của mỗi core sẽ được gán cho mỗi VM, gọi là chính sách lập lịch VM. Thứ hai, ở mức VM được gán một lượng sức mạnh xử lý sẵn có cố định cho các yêu cầu ứng dụng riêng (đơn vị tác vụ) được lưu trữ trong máy thực thi của nó, gọi là chính sách lập lịch tác vụ. Với hai loại chính sách lập lịch: TS và SS cho cả hai mức PM và mức VM, sẽ có bốn trường hợp sau đây [15]:

(1) Ở mức PM và mức VM được cung cấp theo chính sách lập lịch SS, gọi là SS-SS.

(2) Ở mức PM, chính sách lập lịch VM là TS và ở mức VM, chính sách lập lịch tác vụ là SS, gọi là TS-SS.

(3) Ở mức PM, chính sách lập lịch cho VM là SS và ở mức VM, chính sách lập lịch tác vụ là TS, gọi là SS-TS.

(4) Ở mức PMvà mức VM được cung cấp theo chính sách lập lịch TS, gọi là TS-TS.

Thuật toán AMLB-New có tính đến cả sức mạnh xử lý hiện tại của VM và kích thước của các yêu cầu được giao. Về yêu cầu cấp phát tiếp theo, AMLB-New phải ước tính thời gian mà tất cả các yêu cầu đang xếp hàng chờ và yêu cầu tiếp theo được thực hiện hoàn toàn trong mỗi VM. VM nào có thời gian hoàn thành ước tính sớm nhất sẽ được chọn để phân bố yêu cầu đến.

Để xác định sức mạnh xử lý tức thì thực tế của VM là rất phức tạp. Sức mạnh xử lý thay đổi tùy theo chính sách lập lịch ở hai mức trong CC. Ở đây sử dụng cách tính thời gian hoàn thành trong [15] và hoàn thành yêu cầu với các công thức tính toán sức mạnh xử lý của core ảo tùy thuộc vào các trường hợp lập lịch khác nhau. Tính hiệu quả của việc cân bằng tải phụ thuộc vào nhiều yếu tố: tải và công suất. Tải là chỉ số hàng đợi CPU và mức sử dụng CPU.

Thời gian đáp ứng dự kiến có thể được xác định theo công thức sau: [15]:

$$\text{Response Time} = \text{Fin}_t - \text{Arr}_t + \text{TDelay}, \tag{2.1}$$

trong đó, Arr_t là thời điểm đến của yêu cầu; Fin_t là thời điểm hoàn thành yêu cầu (tác vụ) người dùng và TDelay là độ trễ truyền tải.

+ Nếu chính sách lập lịch là SS-SS hoặc TS-SS, Fin_t có thể tính như sau:

$$\text{Fin}_t = \text{est(p)} + \frac{\text{rl}}{\text{Capacity} \times \text{cores(p)}}. \tag{2.2}$$

+ Nếu chính sách lập lịch là SS-TS hoặc TS-TS, Fin_t có thể tính như sau:

$$\text{Fin}_t = \text{ct} + \frac{\text{rl}}{\text{Capacity} \times \text{cores(p)}}. \tag{2.3}$$

Thời gian thực hiện của một tác vụ p có thể tính như sau:

$$\text{Execution Time} = \frac{\text{rl}}{\text{Capacity} \times \text{Cores(p)}}. \quad (2.4)$$

trong đó, est(p) là thời gian mà tác vụ p được bắt đầu; ct là thời gian mô phỏng hiện tại; rl là tổng số chỉ thị của tác vụ p (đơn vị tính MI); Cores(p) là số core hoặc các thành phần xử lý theo yêu cầu của tác vụ (đơn vị là MIPS); Capacity là công suất xử lý trung bình (tính theo MIPS) của một core cho tác vụ.

Tham số Capacity xác định công suất thực sự cho việc xử lý các yêu cầu (tác vụ) trên mỗi VM. Rõ ràng, Capacity phụ thuộc vào chính sách lập lịch trên các hệ thống ảo. Tổng dung lượng xử lý trên PM là không đổi và phụ thuộc vào số lượng core vật lý và sức mạnh xử lý của mỗi core. Tuy nhiên, khi các tài nguyên này được chia sẻ cho nhiều yêu cầu/tác vụ cùng lúc, mỗi tác vụ đòi hỏi một số core nhất định. Nếu tổng số core yêu cầu lớn hơn tổng số core vật lý, khi đó xuất hiện khái niệm về core ảo, và sức mạnh xử lý của mỗi core ảo có thể nhỏ hơn so với core vật lý. Do đó, Capacity bản chất là sức mạnh xử lý trung bình của một core ảo. Từ phân tích, luận án đã phát triển công thức tính toán Capacity theo bốn trường hợp lập lịch như đã đề cập ở trên. Ở đây, đề xuất các công thức để tính toán Capacity trong từng trường hợp để sử dụng trong thuật toán AMLB-New:

(1) Chính sách lập lịch VM là SS, chính sách lập lịch tác vụ là SS:

$$\text{Capacity} = \sum_{\text{i}=1}^{\text{np}} \frac{\text{Cap(i)}}{\text{np}}, \quad (2.5)$$

trong đó, Cap(i) là sức mạnh xử lý của core thứ i, np là số lượng core thực có của PM đã được xem xét.

(2) Chính sách lập lịch VM là SS, chính sách lập lịch tác vụ là TS:

$$\text{Capacity} = \frac{\sum_{\text{i}=1}^{\text{np}} \text{Cap(i)}}{\text{Max}(\sum_{\text{j}=1}^{\alpha} \text{Cores(j)}, \text{np})}, \quad (2.6)$$

trong đó, Cores(j) là số core mà tác vụ j yêu cầu. α là tổng số tác vụ trong VM có chứa tác vụ.

(3) Chính sách lập lịch VM là TS, chính sách lập lịch tác vụ là SS:

$$\text{Capacity} = \frac{\sum_{i=1}^{np} \text{Cap(i)}}{\text{Max}(\sum_{k=1}^{\beta} \sum_{j=1}^{\gamma} \text{Cores(j)}, \text{np})}, \quad (2.7)$$

trong đó, β là số VM trong PM hiện tại. γ là số lượng các tác vụ đang chạy đồng thời trong VM thứ k.

Bảng 2.1: Thuật toán cân bằng tải đề xuất AMLB-New

```
1  /*Đầu vào:
2  *vms: danh sách các VM được tạo ra trong trung tâm dữ liệu đám mây
3  *requests: các yêu cầu/tác vụ của khách hàng đến bộ điều khiển đám mây
4  *vmState<vmId, state>: chứa thông tin trạng thái của VM.
5  *state = 0 là chưa xử lý xong request,
6  *state = 1 đã xử lý xong request.
7  *submitedRequestToVM<request, vmState>: bảng chưa thông tin trạng thái,
8  *cho biết request được gán cho VM nào và cho biết trạng thái đã xử lý xong chưa
9  *Đầu ra: Các request đã được gán cho các VM */
10 //Khởi tạo bảng submitedRequestToVM rỗng.
11 //Xem xét các yêu cầu đến.
12 while requests is not empty
13     //Lấy yêu cầu đầu tiên trong danh sách
14     request = requests.get(0)
15         //Phân tích bảng submitedRequestToVM để ước lượng tổng thời gian
16         //hoàn thành các yêu cầu trong mỗi VM cộng với thời gian dự kiến
17         //hoàn thành yêu cầu mới đến dựa vào các công thức đã đề xuất.
18         //VM nào có thời gian dự kiến hoàn thành sớm nhất sẽ được chọn
19         //để gán yêu cầu mới đến. Nếu có nhiều VM thỏa mãn thì chọn VM đầu tiên.
20         // VM được chọn có chỉ số vmId.
21                 //Gán yêu cầu mới đến (request) cho VM có chỉ số vmId.
22                 //Cập nhật trạng thái cho VM
23                 vmState.vmId=vmId
24                 vmState.state = 0 //Chưa xử lý xong yêu cầu request
25                 //Cập nhật bảng trạng thái submitedRequestToVM
26                 submitedRequestToVM.put(request, vmState)
27     //Khi có request được xử lý xong.
28     //Lấy VM có vmId đã xử lý xong request
29     vmState.vmId = submitedRequestToVM.get(request).vmId
30     vmState.state = 1 // Đã xử lý xong yêu cầu request
31     //Cập nhật lại bảng trạng thái submitedRequestToVM
32     submitedRequestToVM.put(request, vmState)
```

Trong một VM cũng có thể có nhiều tác vụ chạy đồng thời theo lập lịch SS. Đây là trường hợp số core trong VM lớn hơn số core được yêu cầu bởi một tác vụ. Xem

ví dụ Hình 2.2, mỗi VM có 2 core, mỗi tác vụ chỉ yêu cầu 1 core, như vậy sẽ có hai tác vụ/yêu cầu chạy xử lý cùng một lúc.

(4) Chính sách lập lịch VM là TS, chính sách lập lịch tác vụ là TS:

$$\text{Capacity} = \frac{\sum_{i=1}^{np} \text{Cap(i)}}{\text{Max}(\sum_{j=1}^{\delta} \text{Cores(j)}, \text{np})}, \quad (2.8)$$

trong đó, δ là tổng số tác vụ của PM được xem xét.

Ở đây chỉ xét các thuật toán cân bằng tải thực hiện trong một trong tâm dữ liệu, nên tham số độ trễ truyền dữ liệu có thể bỏ qua (TDelay $=$ 0).

2.2.3. Thực nghiệm và đánh giá

2.2.3.1. Mô tả ngữ cảnh thực nghiệm

Trong phần này tiến hành các thực nghiệm so sánh ba thuật toán (AMLB-Old) trong [49], Thuật toán cân bằng tải Throttled (TLB) [86] và Thuật toán đề xuất (AMLB-New được trình bày trong Bảng 2.1). Việc đánh giá dựa trên hai số đo: thời gian đáp ứng trung bình và thời gian xử lý trung bình. Các thuật toán được lập trình bằng ngôn ngữ Java sử dụng công cụ mô phỏng môi trường CC CloudSim [15].

Bảng 2.2: Thiết lập các tham số cho đám mây

Type	Parameter	Value
Trung tâm dữ liệu	Số trung tâm dữ liệu	1
	Số máy chủ	5
Máy chủ (PM)	Số core trong một PM	1-4
	Tốc độ xử lý 1 core (MIPS)	10000 – 40000 MIPS
	Bộ nhớ (RAM)	3072-12288
	Ổ đĩa	5000001000000
	Băng thông (BW)	1000000 MB
VM	Số VM	5
	RAM	1024–4096
	Băng thông	10000 MB
Tác vụ (Cloudlet)	Số tác vụ	10 100
	Độ dài của tác vụ	1246-9660MI
	Số core yêu cầu	1-2

Mô phỏng bao gồm một Datacenter và năm VM (Bảng 2.3) đang chạy trên PM. Các tham số được đưa ra trong Bảng 2.2, Bảng 2.3 và việc thay đổi trường hợp bằng cách gửi số lượng tác vụ tăng từ 10 đến 100 nhiệm vụ để làm rõ sự biến thiên của thời gian đáp ứng và thời gian xử lý dữ liệu của ba thuật toán.

Thực nghiệm được tiến hành dựa trên bốn trường đã được đề cập ở trên.

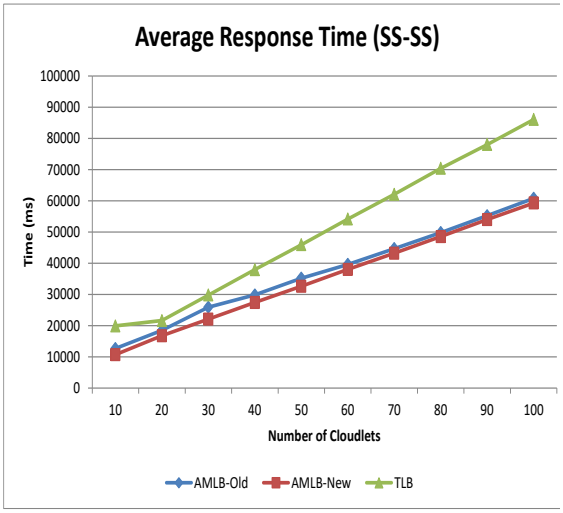
Bảng 2.3: Các tham số cho VM

vmID	RAM (MB)	Bandwidth (MB)	PEs	PE rate (MIPS)
0	2048	10000	2	1000
1	2048	10000	1	4000
2	1024	10000	1	1500
3	4096	10000	2	4000
4	1024	10000	1	1000

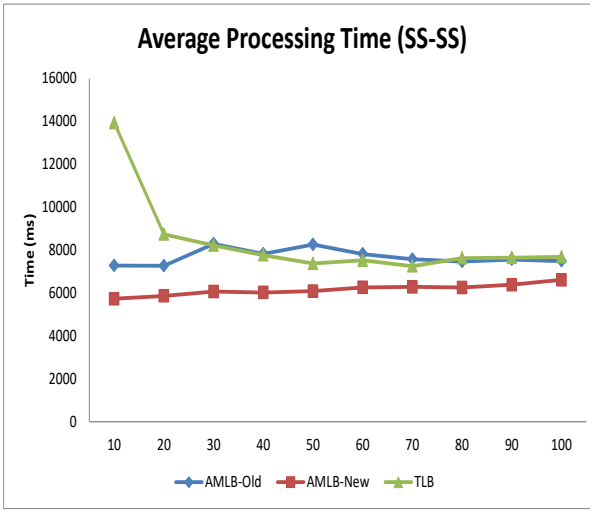
2.2.3.2. Kết quả thực nghiệm

a. Trường hợp 1: Ở mức PM và mức VM được cung cấp theo chính sách lập lịch SS, gọi là SS-SS.

Sử dụng công thức (2.2) để tính toán thời gian hoàn thành ước tính của VM và công thức (2.5) tính toán sức mạnh xử lý trung bình của mỗi core ảo (PE).



Hình 2.4: Thời gian đáp ứng trung bình theo trường hợp 1

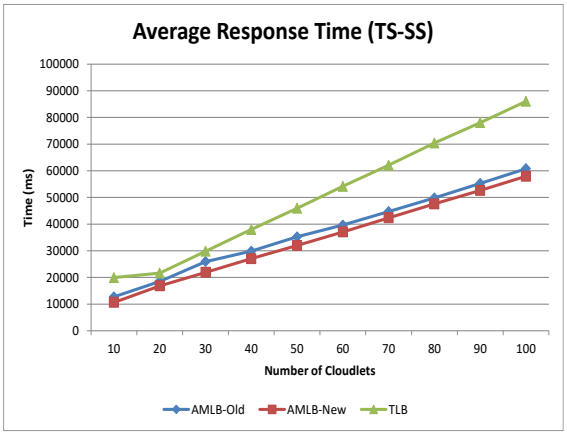


Hình 2.5: Thời gian xử lý trung bình theo trường hợp1

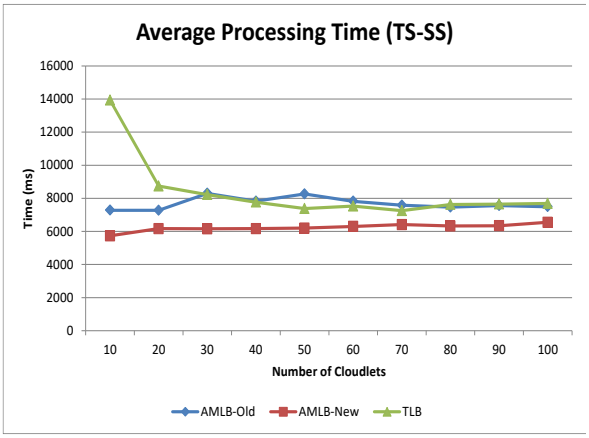
Như trong Hình 2.4 và Hình 2.5, cả thời gian đáp ứng trung bình và thời gian xử lý trung bình trong mỗi trường hợp, thuật toán đề xuất (AMLB-New) cho kết quả tốt nhất. Với số lượng nhiệm vụ thấp hơn 30 (như Hình 2.5), thời gian xử lý trung

bình của thuật toán AMLB-Old tốt hơn so với thuật toán TLB. Tuy nhiên, với số lượng tác vụ càng lớn, thì thời gian xử lý trung bình của hai thuật toán AMLB-Old và TLB hầu như giống nhau.

b. Trường hợp 2: Ở mức PM, chính sách lập lịch VM là TS và ở mức VM, chính sách lập lịch tác vụ là SS, gọi là TS-SS.



Hình 2.6: Thời gian đáp ứng trung bình theo trường hợp 2



Hình 2.7: Thời gian xử lý trung bình theo trường hợp 2

Thời gian hoàn thành ước tính của một tác vụ (cloudlet) p được quản lý bởi VM i và được tính theo công thức (2.2) và sức mạnh xử lý trung bình của mỗi core ảo (PE) được tính bằng công thức (2.7).

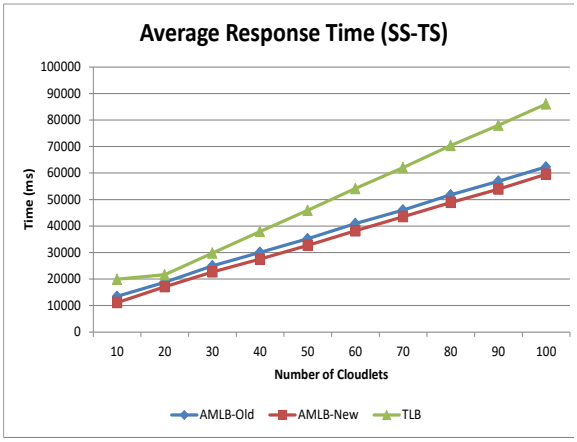
Hình 2.6 và Hình 2.7 cho thấy rằng thời gian đáp ứng trung bình và thời gian xử lý trung bình của thuật toán AMLB-New là tốt nhất.

c. Trường hợp 3: Ở mức PM, chính sách lập lịch cho VM là SS và ở mức VM, chính sách lập lịch tác vụ là TS, gọi là SS-TS.

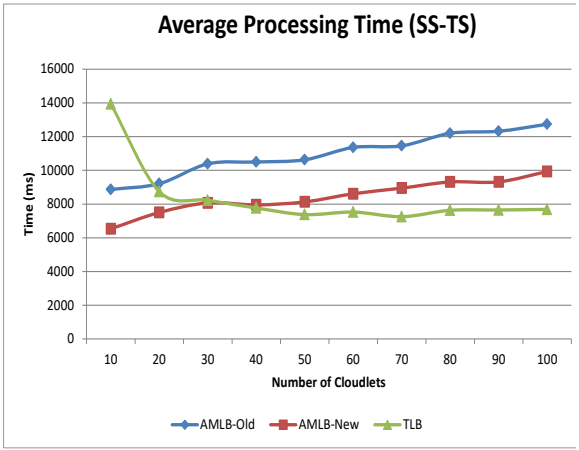
Trong suốt chu kỳ sống của VM, tất cả các tác vụ được gán cho nó sẽ được loại bỏ trong bối cảnh hành động. Bằng cách sử dụng chính sách lập lịch TS, thời gian hoàn thành ước tính của một tác vụ được quản lý bởi VM được tính bằng công thức (2.3). Trong chính sách lập lịch TS, nhiều tác vụ có thể có nhiều nhiệm vụ bên trong một VM. Trong trường hợp này, sức mạnh xử lý trung bình của mỗi core ảo được tính bằng công thức (2.6).

Hình 2.8 cho thấy thời gian đáp ứng trung bình của thuật toán AMLB-New là tốt nhất. Trong Hình 2.9, với số lượng tác vụ dưới 30, thời gian xử lý trung bình của

AMLB-New là tốt nhất. Với nhiều hơn 30 tác vụ, thời gian xử lý trung bình của thuật toán AMLB-New nhỏ hơn thuật toán TLB, nhưng vẫn tốt hơn thuật toán AMLB-Old.



Hình 2.8: Thời gian đáp ứng trung bình theo trường hợp 3



Hình 2.9: Thời gian xử lý trung bình theo trường hợp 3

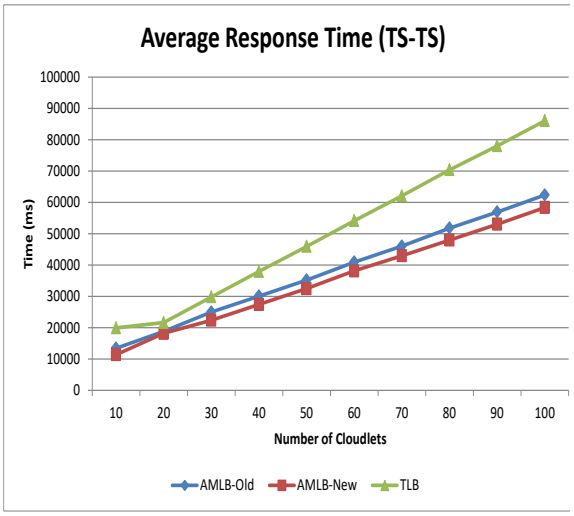
d. Trường hợp 4: Ở mức PMvà mức VM được cung cấp theo chính sách lập lịch TS, gọi là TS-TS.

Sức mạnh xử lý được chia sẻ đồng thời bởi VM và nhận được chia sẻ của mỗi VM để được chia sẻ giữa các tác vụ của nó. Trong trường hợp này, không có độ trễ hàng đợi liên quan đến tác vụ. Chính sách lập lịch TS ước tính thời gian hoàn thành của một tác vụ được quản lý bởi một VM được tính bằng công thức (2.3). Trong chính sách lập lịch TS, nhiều tác vụ có thể xử lý bên trong một VM. Trong trường hợp này, sức mạnh xử lý trung bình của mỗi core ảo được tính bằng công thức (2.8).

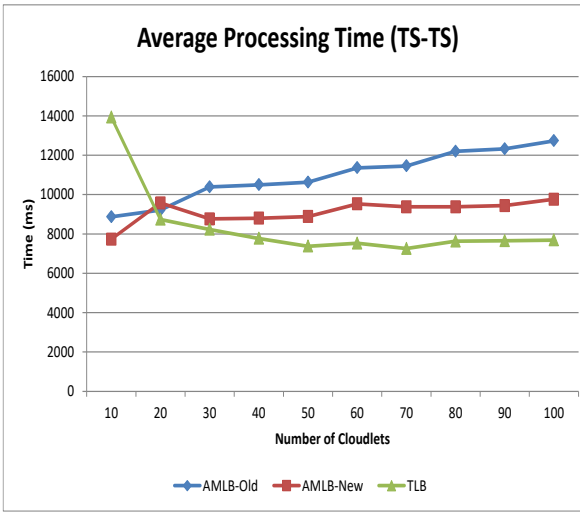
Hình 2.10 cho thấy thời gian đáp ứng trung bình trong thuật toán AMLB-New là tốt nhất. Trong Hình 2.11, với số lượng các tác vụ nhỏ hơn 20, thời gian xử lý trung bình của thuật toán AMLB-New là tốt nhất. Với số lượng tác vụ trên 20, thời gian xử lý trung bình của thuật toán TLB là tốt nhất, thời gian xử lý trung bình của thuật toán AMLB-Old là xấu nhất.

Tóm lại, trong cả bốn trường hợp trên, thời gian đáp ứng trung bình của thuật toán AMLB-New tốt nhất so với các thuật toán AMLB-Old và TLB. Về thời gian xử lý trung bình, thuật toán AMLB-New là tốt nhất cho hai trường hợp SS-SS và TS-SS. Trong trường hợp SS-TS và TS-TS, thuật toán AMLB-New vượt trội hơn thuật toán AMLB-Old nhưng không tốt bằng thuật toán TLB.

Ngoài ra, Hình 2.12 và Hình 2.13 chỉ ra các kết quả tương ứng khi so sánh thời gian đáp ứng trung bình và thời gian xử lý trung bình, theo bốn chính sách lập lịch của ba thuật toán AMLB-Old, AMLB-New và TLB.

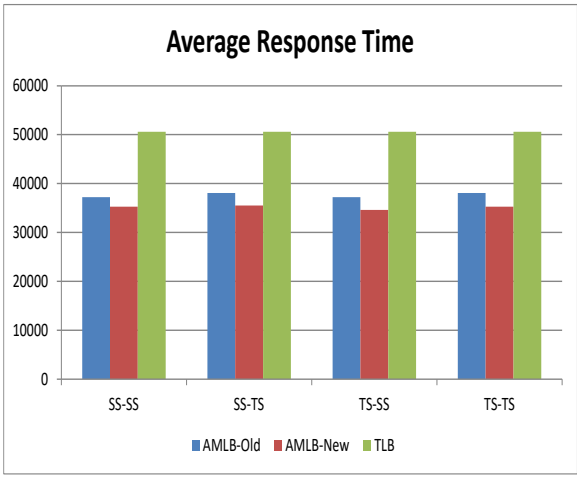


Hình 2.10: Thời gian đáp ứng trung bình theo trường hợp 4

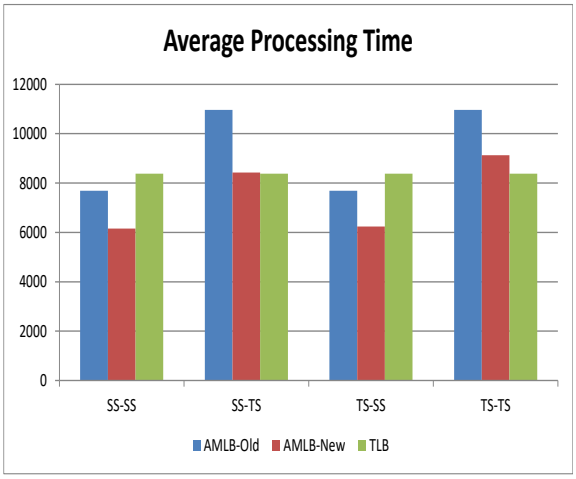


Hình 2.11: Thời gian xử lý trung bình theo trường hợp 4

Hình 2.12, cho thấy thời gian đáp ứng trung bình của thuật toán AMLB-New là tốt nhất trong cả bốn trường hợp. Thuật toán TLB cho thời gian đáp ứng trung bình thấp nhất.



Hình 2.12:Thời gian đáp ứng trung bình cho các trường hợp lập lịch



Hình 2.13: Thời gian xử lý trung bình cho các trường hợp lập lịch

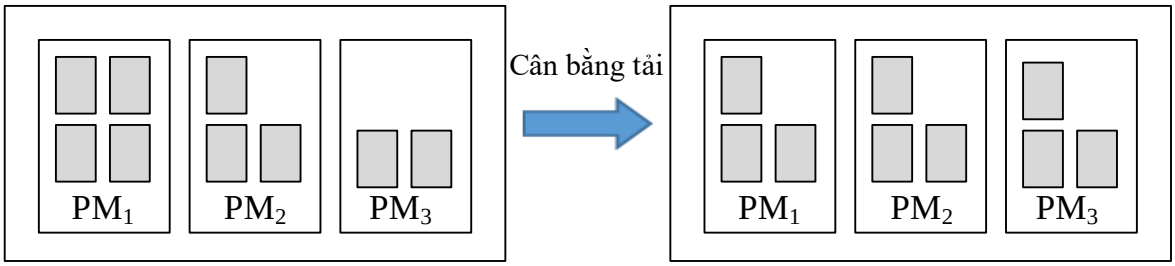
Hình 2.13 cho thấy rằng thời gian xử lý trung bình của thuật toán AMLB-New tốt nhất trong hai trường hợp SS-SS và TS-SS. Trong các trường hợp SS-TS và TS-TS, thời gian đáp ứng trung bình của thuật toán AMLB-New không tốt như thuật toán TLB, nhưng vẫn tốt hơn thuật toán AMLB-Old.

Trong phần tiếp theo, luận án sẽ trình bày về đề xuất kỹ thuật di trú hiệu quả trong CC được tổng hợp từ các công trình TCTN1 và HNQT2.

2.3. ĐỀ XUẤT MỘT KỸ THUẬT DI TRÚ HIỆU QUẢ TRONG ĐIỆN TOÁN ĐÁM MÂY

Các kỹ thuật di trú sẽ giúp hệ thống giảm sự mất cân bằng trong hệ thống về mức độ sử dụng tài nguyên trên tất cả các PM trong trung tâm dữ liệu, tránh được tình trạng một số PM bị quá tải, một số PM khác hoạt động ít. Các kỹ thuật di trú sẽ di trú một số tải trong PM đang bị quá tải sang các PM có ít tải hơn, giúp duy trì hiệu năng hệ thống và hoạt động ở mức ổn định hơn [78]. Như Hình 2.14 là một ví dụ. Kỹ thuật di trú sẽ thực hiện di trú tải ở PM_1 có tải nhiều sang PM_3 có ít tải hơn, tạo ra sự cân bằng trong hệ thống, hệ thống có thể hoạt động ổn định hơn.

Quá trình di trú có thể chia thành hai giai đoạn: giai đoạn đưa ra quyết định di trú và giai đoạn thực hiện di trú. Giai đoạn đưa ra quyết định di trú bao gồm xác định điều kiện và những VM nào cần thực hiện di trú, di trú chúng đến PM nào. Trong giai đoạn thứ hai, giai đoạn này áp dụng các kỹ thuật khác nhau để di chuyển trạng thái VM đang hoạt động tới PM đích.



Hình 2.14. Minh hoạ kỹ thuật di trú giúp cân bằng hoạt động của hệ thống [78].

Công trình [78] đã khảo sát các kỹ thuật di trú khác nhau dựa trên phương pháp heuristic, trong đó bài toán lựa chọn VM và xác định vị trí VM đã được giải quyết với ngưỡng khác nhau được xác định từ tri thức hoặc kinh nghiệm. Công trình [9] đề xuất kỹ thuật di trú và hợp nhất VM phục vụ cho tối ưu năng lượng trong một trung tâm dữ liệu. Trong [9] đã đề xuất thuật toán: thuật toán ra quyết định di trú tổng quát và thuật toán chọn các VM cần di trú từ các PM bị quá tải: (1) chính sách thời gian di trú nhỏ nhất để hoàn thành việc di trú đến một PM khác; (2) chính sách chọn ngẫu nhiên: chọn một VM di trú theo phân bố đều; và (3) chính sách có mối tương quan

lớn nhất: chọn các VM được di trú phải có mối tương quan về mức sử dụng CPU lớn nhất so với các VM khác.

Để xác định khi nào và VM nào cần được di trú, công trình [8] đã đưa ra ba chính sách chọn VM hai ngưỡng: (1) chính sách di trú ít nhất (MM): chọn số VM cần di trú là ít nhất để đưa PM về mức hoạt động bình thường; (2) chính sách khả năng tăng trưởng cao nhất (HPG), khi PM quá tải, chính sách HPG di trú các VM có mức sử dụng CPU thấp nhất có liên quan đến khả năng của CPU được xác định từ các tham số của VM để tối thiểu khả năng làm tăng mức sử dụng của PM và ngăn chặn vi phạm SLA; (3) chính sách chọn ngẫu nhiên (RC) phụ thuộc vào việc chọn ngẫu nhiên số VM cần thiết để giảm mức sử dụng CPU của PM xuống dưới ngưỡng trên. Công trình [8] đã dựa trên ý tưởng cơ bản là thiết lập ngưỡng mức sử dụng trên (UT) và ngưỡng dưới (LT) cho các PM và duy trì tổng mức sử dụng CPU của các VM được cấp phát trong PM đó nằm giữa các ngưỡng này. Nếu mức sử dụng CPU của PM thấp hơn LT, tất cả các VM phải được di trú sang PM khác và PM này phải được chuyển sang chế độ nghỉ để loại bỏ tiêu thụ năng lượng nhàn rỗi. Nếu mức sử dụng của PM vượt quá ngưỡng UT, một số VM phải được di trú để giảm mức sử dụng PM này xuống dưới ngưỡng UT. Công trình này đã tiến hành thực nghiệm và cho kết quả đánh giá khả quan về các phương pháp đã đề xuất. Tuy nhiên, đối với phương pháp lựa chọn VM với số lần di trú ít nhất còn chưa hiệu quả trong trường hợp phải di trú nhiều VM về khía cạnh duy trì mức sử dụng của PM ở mức tối đa có thể, vấn đề này sẽ trình bày cụ thể ở phần sau.

Ở đây, luận án tập trung giải pháp áp dụng cho giai đoạn thứ nhất của quá trình di trú, sau đó đánh giá kỹ thuật di trú được đề xuất trên môi trường CC bằng cách sử dụng công cụ mô phỏng CloudSim [15].

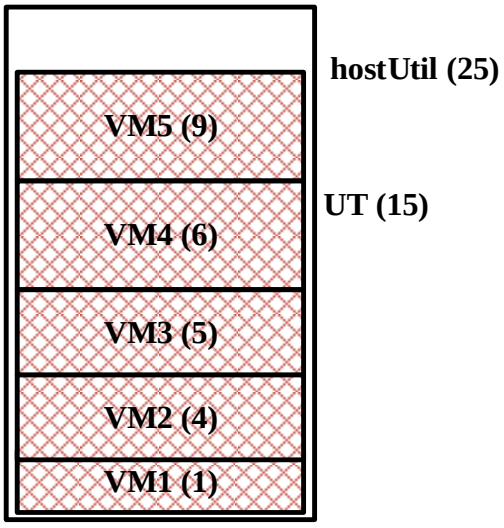
2.3.1. Phân tích kỹ thuật di trú được đề xuất

Thuật toán lựa chọn VM để di trú với số lần di trú là ít nhất trong công trình [8], gọi tắt là MM-Old, sắp xếp danh sách các VM trong một PM theo mức sử dụng CPU giảm dần. Sau đó, duyệt qua danh sách các VM và tìm VM tốt nhất để di trú. VM tốt nhất phải thỏa mãn hai điều kiện. Trong trường hợp phải di trú một VM: *Thứ nhất*, MM-Old sẽ tìm VM có mức sử dụng cao hơn mức sử dụng chênh lệch (diffUtil) giữa

mức sử dụng toàn bộ PM (hostUtil) và ngưỡng sử dụng trên (UT). *Thứ hai*, nếu VM được di trú, thì mức chênh lệch giữa UT và mức sử dụng CPU mới của PM là nhỏ nhất. Trong trường hợp phải di trú nhiều hơn một VM: MM-Old sẽ chọn VM có mức sử dụng cao nhất để di trú, loại bỏ VM này ra khỏi danh sách $vmList$ và tiếp tục vòng lặp mới. Thuật toán MM-Old dừng lại khi mức sử dụng CPU mới của PM nằm dưới UT.

Thuật toán MM-Old có ưu điểm là số lần di trú là ít nhất, dẫn đến ít xảy ra các vi phạm SLA do quá trình thực hiện di trú gây ra, đồng thời cũng giảm thiểu được việc tiêu thụ hiệu năng CPU cũng như băng thông mạng phục vụ cho quá trình di trú. Bên cạnh đó, danh sách VM được sắp xếp trước khi thực hiện chọn VM để di trú, trong trường hợp chỉ cần di trú một VM, thì mức CPU khả dụng sau khi di trú VM đi là nhỏ nhất. Tức là khai thác được tối đa tài nguyên nhất.

Tuy nhiên, trong trường hợp phải di trú nhiều hơn một VM, thì thuật toán MM-Old không khai thác được tối đa tài nguyên của PM (duy trì PM hoạt động với khả năng cao nhất có thể) trong một số trường hợp. Hãy xét ví dụ Hình 2.15.



Hình 2.15: Ví dụ cho trường hợp phải di trú nhiều VM

Hình 2.15 mức sử dụng hiện tại của PM là $\text{hostUtil} = 25$, ngưỡng trên là $\text{UT} = 15$. PM này được cấp phát 5 VM với các mức sử dụng tương ứng như trên. Như vậy, PM này đang hoạt động quá tải, độ chênh lệch mức sử dụng vượt là $\text{diffUtil} = 25 - 15 = 10$.

Theo thuật toán MM-Old, thì VM có mức sử dụng lớn nhất sẽ được chọn đầu tiên để di trú là VM5 có mức sử dụng là 9. Khi VM5 được di trú, PM vẫn bị quá tải và vượt mức còn lại là $\text{diffUtil} = 10 - 9 = 1$. Tiếp theo, VM2 (4) sẽ được chọn tiếp theo để di trú và sẽ đưa PM về mức sử dụng xuống dưới ngưỡng UT. Do đó, sau khi di trú hai máy ảo VM5(9) và VM2(4), mức sử dụng CPU của PM là: $25 - 13 = 12$.

Tuy nhiên, có thể thấy nếu chọn hai máy ảo VM4 (6) và VM3(5) để di trú thì vẫn thỏa mãn số lần di trú ít nhất đồng thời có thể duy trì mức hoạt động của PM cao nhất có thể. Mức sử dụng của PM khi di trú VM4 và VM5 là: $25 - 11 = 14$.

Như vậy, phương án lựa chọn mới này vẫn đảm bảo số lần di trú là ít nhất và vẫn duy trì mức sử dụng PM ở mức cao nhất có thể.

Từ việc phân tích ở trên, chính sách di trú với số lần ít nhất (MM-New) có thể phát biểu lại như sau:

Cho V_j là một tập hợp các VM được cấp phát cho PM j. Gọi $P(V_j)$ là tập công suất (mức sử dụng) của V_j. Chính sách MM-New tìm một tập $R \in V_j$ được định nghĩa như sau:

$$R = \begin{cases} \left\{ \begin{array}{l} S | S \subset V_j, \text{hostUtil}_j - \sum_{v \in S, u_a(v) \in P(V_j)} u_a(v) < UT, \\ |S| \to \min, \sum_{v \in S, u_a(v) \in P(V_j)} u_a(v) \to \min \end{array} \right\}, \text{nếu hostUtil}_j > \text{UT}; \\ V_j \quad , \text{nếu hostUtil}_j < \text{LT}; \\ \emptyset, \text{còn lại} \end{cases} \tag{2.9}$$

trong đó UT là ngưỡng sử dụng trên; LT là ngưỡng sử dụng dưới; hostUtil_j là mức sử dụng CPU hiện tại của PM j; và $u_a(v)$ là một phần mức sử dụng CPU được cấp phát cho VM v.

Thuật toán MM-New được trình bày trong Bảng 2.4, khác thuật toán MM-Old ở chỗ, trong trường hợp phải di trú nhiều hơn một VM, thuật toán MM-New sẽ xác định số lượng VM cần di trú ít nhất trước, được chỉ ra từ dòng 17-21. Sau đó, thuật toán MM-New tìm các VM cần di trú để thỏa mãn số lượng VM cần di trú như trên đồng thời có thể duy trì công suất hoạt động của PM cao nhất có thể, từ dòng 26-33.

Bảng 2.4: Thuật toán đề xuất MM-New

```
/* MM-New
*Đầu vào:
* hosts: danh sách máy chủ
* Đầu ra:
* migrations: danh sách các VM cần di trú*/
for h in hosts
    vms = h.getVMs()
    vms.sortDescendingUltilization()
    hostUtil = h.getUtil()
    bestFitUtil = MAX
    //Số VM cần di trú
    numberVMMigrate = 0
    //Tổng mức sử dụng của VM
    sumVMUtil = 0
    while hostUtil > UT
        for vm in vms
            bestFitVM = NULL
            if vm.getUtil() > hostUtil – UT
                t = vm.getUtil() – hostUtil + UT
                if t < bestFitUtil
                    bestFitUtil = t
                    bestFitVM = vm
                else if bestFitUtil = MAX
                    numberVMMigrate++
                    sumVMUtil += vm.getUtil()
                    if sumVMUtil > hostUtil – UT then
                        break;
            if bestFitVM != NULL
                hostUtil = hostUtil – bestFitVM.getUtil()
                migrations.add(bestFitVM)
                vms.remove(bestFitVM)
        if numberVMMigrate > 1
            int totalVMs = vms.size()
            int result[ ] = new int [numberVMMigrate + 1]
            //Sử dụng thuật toán quay lui để tìm phương án di trú
            Try(1, result, totalVMs, numberVMMigrate, vms)
    if hostUtil < LT
        migrations.add(bestFitVM)
        vms.remove(bestFitVM)
return migrations
```

2.3.2. Thực nghiệm và đánh giá

2.3.2.1. Thiết lập các tham số thực nghiệm

Trong phần này tiến hành thực nghiệm trên môi trường CC không đồng nhất và có một trung tâm dữ liệu với các thông tin trong Bảng 2.5. Các thuật toán được thực hiện thông qua công cụ mô phỏng môi trường CC CloudSim [15]. Ngôn ngữ Java được sử dụng cho việc phát triển và thực hiện các thuật toán. Thực nghiệm sẽ tiến hành so sánh thuật toán đề xuất MM-New với thuật toán MM-Old trong [8].

Trong mô phỏng có sử dụng hai loại PM của HP [9] được cho trong Bảng 2.6. Mức tiêu thụ năng lượng tương ứng với CPU như bảng 2.7.

Việc lựa chọn mô phỏng hai PM có cấu hình thấp đóng vai trò quan trọng trong quá trình mô phỏng, vì với những loại PM có cấu hình thấp, PM dễ rơi vào trạng thái quá tải, hay ít tải, do đó dễ dàng đánh giá các giải thuật hơn.

Bảng 2.5: Thông tin về trung tâm dữ liệu

Hệ điều hành	**Linux**
Kiến trúc	X86
VMM	Xen
Múi giờ	7
Di trú VM	Cho phép
Chi phí sử dụng hệ thống/mỗi giây	3
Chi phí sử dụng 1 đơn vị Ram	0.05
Chi phí mỗi đơn vị lưu trữ	0.001
Chí phí sử dụng băng thông	0

Bảng 2.6: Bảng số liệu máy chủ trong mô phỏng

Loại máy chủ	**HP Proliant G4**	**HP Proliant G5**
Số lượng	110	110
Host Mips	1860	2660
Số lượng Pe	2	2
Ram	4Gb	4Gb
Băng thông	10Gbps	10Gbps
HDD	100000 Tbs	100Tb

Bảng 2.7: Tiêu thụ năng lượng của hai loại máy chủ

Server	**0%**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
G5	93.7	97	101	105	110	116	121	125	129	133	135

Trong môi trường mô phỏng có một Broker yêu cầu 2001 VM như bảng 2.8.

Bảng 2.8: Thông tin VM trong sử dụng trong mô phỏng

Tên VM	Mips	Ram	Chính sách	Số lượng
low	500	128	Time Shared	1001
hight	850	512	Time Shared	1000

Các VM đều yêu cầu số lượng ổ cứng cũng như băng thông là như nhau. Các VM được xem như đang chạy các dịch vụ game, dịch vụ trực tuyến với thông số tác vụ (Cloudlet): 2500 Mips, $\text{OutputFile} = \text{InputFile} = 300$ byte.

Các ngưỡng trên UT và ngưỡng dưới LT sử dụng trong chương trình mô phỏng lần lượt là 70% và 20%. Thời gian mô phỏng trong trường hợp là 1 ngày = 60 * 24 = 1440 phút. Khoảng thời gian để hệ thống tiến hành xem xét việc di trú là 350 giây.

Thực nghiệm được thực hiện với 2 cặp thuật toán sau:

Cặp 1: Chọn VM có số lần di trú ít nhất theo thuật toán đề xuất MM-New và Chọn PM có mức tiêu thụ năng lượng tốt nhất.

Cặp 2: Chọn VM có số lần di trú là ít nhất theo thuật toán MM-Old và Chọn PM có mức tiêu thụ năng lượng tốt nhất.

2.3.2.2. Kết quả thực nghiệm

Kết quả thực nghiệm được chỉ ra ở bảng 2.9 và hình 2.16. Hình 2.16 mô tả kết quả so sánh hai thuật toán MM-Old và MM-New theo ba tham số: Mức sử dụng (Utilization), Số PM hoạt động (Number of Host Active) và Số VM di trú (Number of VM Migration), đường màu xanh là kết quả khi áp dụng thuật toán MM-Old, đường màu đỏ là kết quả của thuật toán MM-New. Như vậy, từ bảng 2.9 và hình 2.16 cho thấy thuật toán MM-New tốt hơn thuật toán MM-Old ở nhiều tiêu chí đánh giá.

Bảng 2.9: Kết quả một số thông số khi chạy thực nghiệm

Thông số	MM-Old	MM-New
Tiêu thụ năng lượng (Kwh)	375.68	**366.75**
Số VM di trú	34783	**27820**
Số PM tắt	150	**143**
SLA suy giảm do di trú	5.20	**3.88**
Số lần vi phạm SLA (%)	**45.06**	45.95
Thời gian chọn VM cần để di trú	**2119**	1330901
Thời gian chọn PM đích	137268	**93880**
Tổng thời gian di trú	**139387**	1424781

Về lý thuyết, thuật toán MM-New sẽ cho mức độ khai thác PM tối đa hơn, cụ thể là mức sử dụng CPU sau khi di trú VM sẽ gần với ngưỡng trên hơn, dẫn đến chỉ số Utilization cao hơn (hình 2.16, phần Utilization là đường màu đỏ nằm trên đường màu xanh). Có một số giai đoạn, chỉ số Utilization của hai thuật toán tương đối cân bằng, đó là khi chỉ cần di trú một VM là có thể đưa PM về ngưỡng chấp nhận.

Hình 2.16: Kết quả so sánh hài thuật toán MM-Old và MM-New

Về mặt năng lượng tiêu thụ, thuật toán MM-New cũng cho kết quả tốt hơn (bảng 2.9). Trong thuật toán MM-New, số lượng PM hoạt động duy trì ở mức thấp hơn (hình 2.16), do vậy năng lượng tiêu thụ của PM và hệ thống làm mát sẽ ít hơn. Bảng 2.9 cho thấy hệ thống duy trì hoạt động ở mức tiêu thụ CPU gần với ngưỡng trên, do thuật toán MM-New cần ít số lần di trú hơn, cũng làm giảm tỉ lệ vi phạm SLA. Mặt khác, thuật toán MM-New chọn những VM có mức tiêu thụ CPU thấp hơn, do đó thời gian di trú của những VM này sẽ ít hơn thời gian di trú những VM của thuật toán MM-Old. Điều này làm giảm các vi phạm cam kết dịch vụ, nhà cung cấp dịch vụ

giảm được chi phí tiền phạt do việc vi phạm cam kết dịch vụ tạo ra và thời gian cần cho việc chọn PM đích giảm xuống (bảng 2.9).

Ở các phần trên đã nghiên cứu và đề xuất được một kỹ thuật cân bằng tải hiệu quả và một kỹ thuật di trú hiệu quả trong môi trường CC. Trong phần tiếp theo sẽ nghiên cứu ảnh hưởng của các kỹ thuật cân bằng tải với cơ chế AS trong môi trường CC. Nội dung sau sẽ được tổng hợp từ công trình TCNN1.

2.4. ẢNH HƯỞNG CỦA CÂN BẰNG TẢI ĐẾN CƠ CHẾ TỰ ĐỘNG ĐIỀU CHỈNH TÀI NGUYÊN

Trong thực tế, khi các yêu cầu người sử dụng tăng lên và vượt quá sức mạnh thực hiện được giao, hệ thống thuê sẽ rơi vào tình trạng quá tải và QoS bị suy giảm đáng kể. Để khắc phục tình hình này, các nhà cung cấp dịch vụ đám mây ngày nay đã cung cấp cho khách hàng một dịch vụ đặc biệt được gọi là AS. Dịch vụ mới được thực hiện bởi một cơ chế AS cho phép thêm các nguồn tài nguyên tạm thời vào các gói dịch vụ đã đăng ký trong trường hợp các tài nguyên sắp cạn kiệt. Cơ chế AS là kỹ thuật mới được bổ sung vào môi trường đám mây đã ổn định với các cơ chế hệ thống vốn có. Câu hỏi đặt ra là liệu có vấn đề phát sinh hay không nếu đám mây vẫn chạy với phiên bản hiện tại của cơ chế hệ thống. Khi sử dụng các dịch vụ, các kỹ thuật cân bằng tải là một mục mà các thuê bao cần phải chọn để xác định cấu hình dịch vụ AS trong đám mây EC2 [87]. Điều này cho thấy các kỹ thuật cân bằng tải có vai trò quan trọng trong việc cung cấp dịch vụ AS cho thuê bao đám mây và có tác vụ hiện tại cần thực hiện với cân bằng tải truyền thống để thích nghi với việc AS. Để làm rõ vấn đề, trong phần này sẽ tìm hiểu những gì sẽ xảy ra nếu thay đổi các kỹ thuật cân bằng tải trong môi trường CC có AS và cũng tìm hiểu những gì làm cho hiệu năng của đám mây trở nên xấu hơn.

Trước tiên, luận án sẽ nghiên cứu một mô hình đám mây bao gồm cả bộ cân bằng tải và bộ AS. Bộ cân bằng tải được thực hiện bằng cách sử dụng các thuật toán cân bằng tải điển hình như Round Robin (RR) và kỹ thuật cân bằng tải giám sát tích cực được đề xuất trong phần 2.2.2 (AMLB-New). Bộ AS được áp dụng kỹ thuật AS

dựa trên ngưỡng. Tiếp theo, luận án sẽ tiến hành mô phỏng các trường hợp khác nhau trên mô hình CC và quan sát kết quả thu được.

2.4.1. Một số vấn đề liên quan

Như trên đã cho thấy cân bằng tải có vai trò quan trọng trong CC. Có rất nhiều thách thức để phát triển bộ cân bằng tải hiệu quả. Các lược đồ cân bằng tải phải kiểm soát tải đến để tất cả các core xử lý trong hệ thống hoặc mỗi PM trong đám mây thực hiện một số tải công việc tương đương vào bất kỳ thời điểm nào [15]. Nghiên cứu trong [49] đã đề xuất các thuật toán dựa trên giám sát tích cực. Các thuật toán theo dõi tải hiện tại trên tất cả các máy trong đám mây. Tuy nhiên, trong chiến lược cân bằng tải của môi trường đám mây nên được cân nhắc trong hai mức độ lập lịch, lập lịch ở mức PM và mức VM. Mục 2.2.2 của Chương này đã trình bày một kỹ thuật cân bằng tải động hiệu quả (AMLB-New) và được công bố trong công trình [$HNQT1$].

Tương tự như cân bằng tải, AS phát sinh nhiều thách thức cho nỗ lực đạt được các mục tiêu đã đề ra. Do đó, chủ đề này cũng thu hút nhiều sự quan tâm của các nhà nghiên cứu, như trong [3, 68]. Vấn đề đầu tiên của AS là làm thế nào để thuê đúng số lượng các tài nguyên và phải trả tiền cho những gì được sử dụng. Hơn nữa, làm thế nào để thu hồi số tài nguyên đã cho thuê đúng lúc để giữ cho chi phí thấp là vấn đề thứ hai. Cho đến bây giờ, để giải quyết cả hai vấn đề này không phải là một nhiệm vụ dễ dàng. Khi nghiên cứu các vấn đề này, trong [70], các tác giả đã đề xuất một cơ chế AS mới gọi là BATS, đáp ứng được giới hạn ngân sách đồng thời giảm thiểu độ trễ của dịch vụ. Ngoài ra, nghiên cứu [81] đề xuất một hệ thống AS dựa trên thuật toán dự đoán chuỗi thời gian. Để đạt được độ chính xác dự đoán của hệ thống AS, một thuật toán chính quan trọng gọi là bộ dự đoán tự thích nghi, nó có thể tự động lựa chọn thuật toán dự đoán phù hợp nhất dựa trên mô hình tải công việc đi vào. Phần nghiên cứu và đề xuất giải pháp AS sẽ được trình bày trong Chương 4. Ở đây, luận án sẽ xem xét mối quan hệ giữa các kỹ thuật cân bằng tải và AS được đề cập trong [18]. Công trình [18] mô tả cách các hệ thống CC nổi tiếng như EC2, Azure và RackSpace cung cấp cho người dùng với dịch vụ AS. Đặc biệt, EC2 sử dụng một bộ

giám sát gọi là Cloudwatch để phát ra thông báo về số đo để bắt đầu hoặc ngừng các VM. Liên quan đến cả cân bằng tải và AS, công trình [4] cũng đưa ra một so sánh giữa các nhà cung cấp đám mây khác nhau về các tính năng của cân bằng tải và AS. Tuy nhiên, các công trình này cũng chưa làm rõ giữa cân bằng tải và AS ảnh hưởng đến nhau như thế nào trong môi trường CC. Phần tiếp theo sẽ trình bày về mô hình CC có cả bộ cân bằng tải và AS, đề xuất phương pháp đánh giá ảnh hưởng giữa cân bằng tải với hệ thống CC có AS.

2.4.2. Mô hình trung tâm dữ liệu có bộ điều chỉnh tài nguyên tự động

Cho trung tâm dữ liệu (DC) có N VM: $VM_1, VM_2, \ldots, VM_N$. Có thể biểu diễn trung tâm dữ liệu như sau: DC $(VM_1, VM_2, \ldots, VM_N)$. Trong đó, mỗi VM có sức mạnh xử lý P_i và tải công việc $L_i(t)$ tại thời điểm t.

$$VM_1(L_1(t),\ P_1)$$
$$VM_2(L_2(t),\ P_2)$$
$$\ldots\ldots$$
$$VM_N(L_N(t),\ P_N)$$

Giả sử trung tâm dữ liệu áp dụng chính sách lập lịch chia sẻ thời gian (TS) cho các mức lập lịch ở mức PM và ở mức VM. Ngoài ra, trung tâm dữ liệu chỉ tạo ra một số lượng VM hạn chế trên mỗi PM để đảm bảo đủ sức mạnh xử lý cho VM.

Gọi $l(t)$ là tải công việc mới (yêu cầu mới) đến tại thời điểm t được tạo ra bởi yêu cầu gửi đến bộ cân bằng tải và bộ cân bằng tải phân bố nó tới một VM nhất định dựa trên thuật toán cân bằng tải cụ thể. Vì vậy, gọi $L'_i(t)$ là tải công việc của VM_i sau khi bộ cân bằng tải đã phân bố tải $l(t)$ đến, như vậy:

$$L'_i(t) = L_i(t) + a_i(t).l(t);\ i = 1..N \qquad (2.10)$$

với $a_i(t)$ lấy giá trị 1 hoặc 0, tùy thuộc vào việc liệu VM_i có phù hợp với điều kiện đúng của thuật toán cân bằng tải hay không. Nó có giá trị 1 nếu điều kiện là đúng và ngược lại có giá trị là 0.

$$a_i(t) = \begin{cases} 0 \text{ nếu } VM_i \text{ không được gán tải,} \\ 1 \text{ nếu } VM_i \text{ được gán.} \end{cases} \qquad (2.11)$$

Nếu bộ cân bằng tải thực hiện thuật toán Round Robin, các tải công việc đến được gán cho mỗi VM theo thứ tự xoay vòng luân phiên. Với thuật toán cân bằng tải giám sát tích cực (ALMB-New), bộ cân bằng tải phải tìm VM có thời gian hoàn thành tải công việc dự kiến sớm nhất sẽ được chọn để phân bổ tải mới. Trong trường hợp này, $a_i(t)$ như sau:

$$a_i(t) = \begin{cases} 1 \text{ nếu } VM_i \text{ có thời gian hoàn thành} \\ \text{dự kiến sớm nhất được gán tải,} \\ 0 \text{ nếu ngược lại.} \end{cases} \quad (2.12)$$

Mục đích chính của bộ cân bằng tải là cấp phát tải công việc công bằng giữa các VM bên trong một trung tâm dữ liệu. Không có VM phải chịu tải nhiều hơn những VM khác. Do đó, ở đây cũng xem xét việc cấp phát hợp lý là một trong những tiêu chí để so sánh giữa các bộ cân bằng tải. Bộ cân bằng tải có độ lệch tải công việc nhỏ nhất giữa các VM trong trung tâm dữ liệu là bộ cân bằng tải tốt nhất. Việc này cũng dẫn đến độ trễ tối thiểu khi độ lệch tải công việc đạt được giá trị tối thiểu. Ở đây chú ý đến thời gian phục vụ còn lại của VM, điều này có nghĩa là lượng thời gian cần thiết để thực hiện tải công việc hiện tại hoàn thành trong VM. Thời gian phục vụ còn lại của VM, được gọi là $T_{left,i}(t)$, được tính như sau:

$$T_{left,i}(t) = \frac{L_i(t)}{P_i} \quad (2.13)$$

Gọi $Dev(t)$ là sự khác biệt giữa thời gian phục vụ còn lại nhỏ nhất và thời gian phục vụ còn lại lớn nhất của các VM bên trong hệ thống được xem xét. Thành phần này sẽ được kiểm tra trong các trường hợp mô phỏng khác nhau ở phần 2.4.3 và được tính như sau.

$$Dev(t) = \left(\frac{L_i(t)}{P_i}\right)_{max} - \left(\frac{L_i(t)}{P_i}\right)_{min} \quad (2.14)$$

$$\text{Lý tưởng là: } Dev(t) \rightarrow 0 \text{ khi } t \rightarrow \infty$$

Nói chung, bộ AS theo dõi thường xuyên trung tâm dữ liệu và thu thập thông tin về tình trạng của hệ thống. Khi hệ thống rơi vào tình trạng quá tải, nó sẽ cung cấp công suất bằng cách thêm một máy mới vào hệ thống. Sử dụng các ngưỡng cài

sẵn để ra quyết định là phương pháp phổ biến được áp dụng trong bộ AS hiện nay. Số đo hệ thống được thiết lập trong các ngưỡng có thể biểu diễn trạng thái của hệ thống. Các số đo là sự khác nhau giữa các thuật toán AS. Ở đây, sử dụng thời gian phục vụ còn lại trung bình trong hệ thống thuê bao là một số đo cho AS. Gọi $T_{av}(t)$ thời gian phục vụ còn lại trung bình tại thời điểm t, được tính như sau:

$$T_{av}(t) = \frac{\sum_i^N L_i(t)}{\sum_i^N P_i} \tag{2.15}$$

Nếu gọi $N'(t)$ là số lượng máy trong hệ thống được xem xét tại thời điểm t thì $N'(t)$ được tính theo công thức sau:

$$N'(t) = N(t) + n(t) \tag{2.16}$$

$n(t)$ là số máy tạm thời được bổ sung vào hệ thống được xem xét tại thời điểm t. Bộ AS tuân theo quy tắc sau:

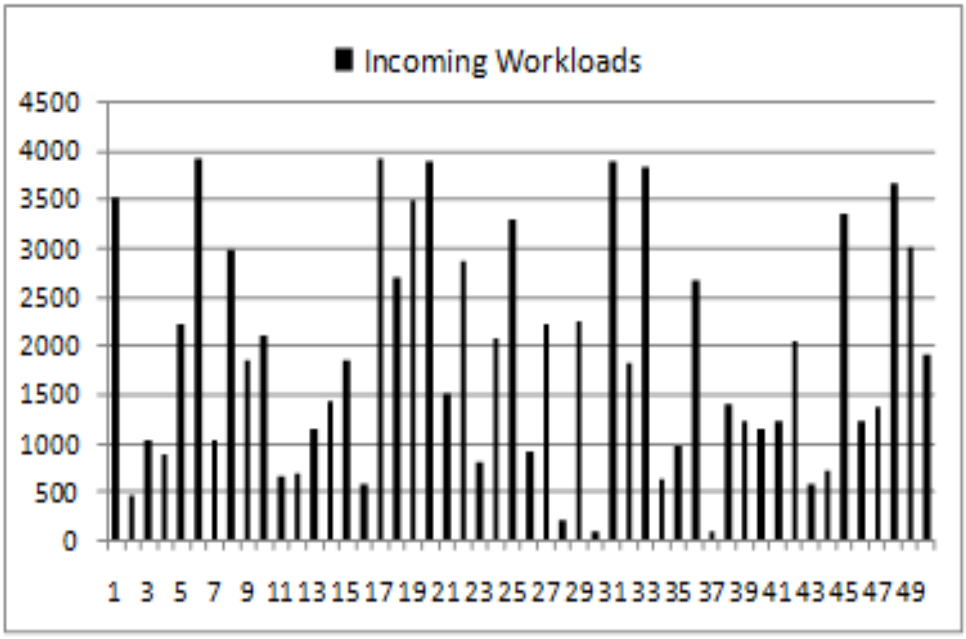
$$n(t) = \begin{cases} 0 \ Threshold_{low} \le T_{av}(t) \le Threshold_{high}, \\ 1 \ T_{av}(t) > Threshold_{high}, \\ -1 \ T_{av}(t) < Threshold_{low}. \end{cases} \tag{2.17}$$

Thông thường, bộ cân bằng tải làm việc với một số máy cố định trong một hệ thống đã đăng ký nhưng nó sẽ phải làm việc với sự thay đổi số lượng máy trong hệ thống có bộ AS. Làm thế nào để bộ cân bằng tải biết đúng số máy là một vấn đề trong ngữ cảnh này. Điều này làm phát sinh vấn đề là số lượng máy có thể thay đổi nhanh do sự biến động của tải công việc. Mặc dù vấn đề này có thể xảy ra, phần này vẫn xem xét tất cả các ảnh hưởng về hiệu năng của bộ cân bằng tải và hiệu năng của hệ thống CC.

2.4.3. Thực nghiệm và đánh giá

Thực nghiệm được phát triển từ một mô phỏng đám mây dựa trên mô hình trong phần 2.4.2. Chương trình mô phỏng cũng bao gồm tất cả các thành phần chính của hệ thống đám mây như PM, VM, bộ cân bằng tải, bộ AS,... Trong mô phỏng này sử dụng các thuật toán cân bằng tải: Round Robin và kỹ thuật cân bằng tải giám sát tích cực AMLB-New. Việc mô phỏng bao gồm một bộ AS hoạt động dựa trên ngưỡng.

Trong mô phỏng này xây dựng một hệ thống CC thuê bao với bốn VM có 500MIPS CPU, RAM 2Gbyte và băng thông 1Gbps. Bốn VM này là số VM tối thiểu trong hệ thống được xem xét là thuộc về gói dịch vụ đám mây của khách hàng. Bộ AS sẽ bổ sung các VM tạm thời khi hệ thống đám mây bị quá tải và nó cũng loại bỏ các VM bổ sung khi hệ thống đám mây trở lại trạng thái bình thường. Vì vậy, bộ AS làm thay đổi số lượng VM lớn hơn bốn và trở lại trong trạng thái bình thường. Trong mô phỏng, thiết lập ngưỡng trên bằng 6 giây và ngưỡng dưới là 2 giây cho thời gian phục vụ. Sử dụng tải công việc như hình 2.17, bộ cân bằng tải là thuật toán Round robin và thuật toán giám sát tích cực AMLB-New. Hình 2.17 cho thấy tải công việc đã dao động nhiều hơn bình thường. Trước khi đạt đến đỉnh điểm xấp xỉ 4000MI, đã có tải công việc tăng lên nhưng sau đó tải công việc lại giảm đáng kể vào thời gian sau đó.

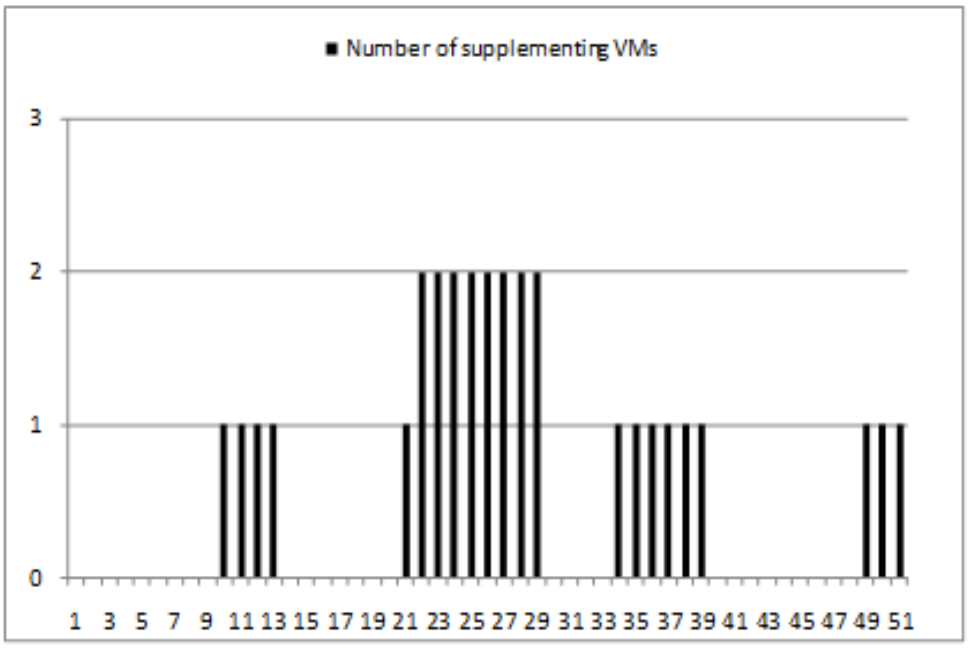


Hình 2.17: Phấn bố tải công việc đến trong ClarkNet trace[12]

Thứ nhất, để xem xét cách thức bộ AS bổ sung các VM để đăng ký với hệ thống CC có bộ cân bằng tải sử dụng Round Robin và AMLB-New. Các kết quả chạy với bộ cân bằng tải sử dụng Round Robin trong hình 2.18 và các kết quả chạy với bộ cân bằng tải sử dụng AMLB-New được chỉ ra trong hình 2.19.

[12] *HTTP, ClarkNet ClarkNet HTTP Trace, truy cập 9/7/2018, from http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html.*

Hình 2.18 cho thấy bộ AS đã phải bổ sung các VM vào hệ thống đám mây nhiều lần, từ 1 đến 2 VM, và các VM thực hiện ở lại lâu trong hệ thống. Trong khi đó, bộ AS sử dụng AMLB-New chỉ cần một thời gian ngắn để thêm một VM vào hệ thống, như hình 2.19. Hơn nữa, VM tạm thời sẽ không ở lại lâu trong hệ thống đám mây đã đăng ký.

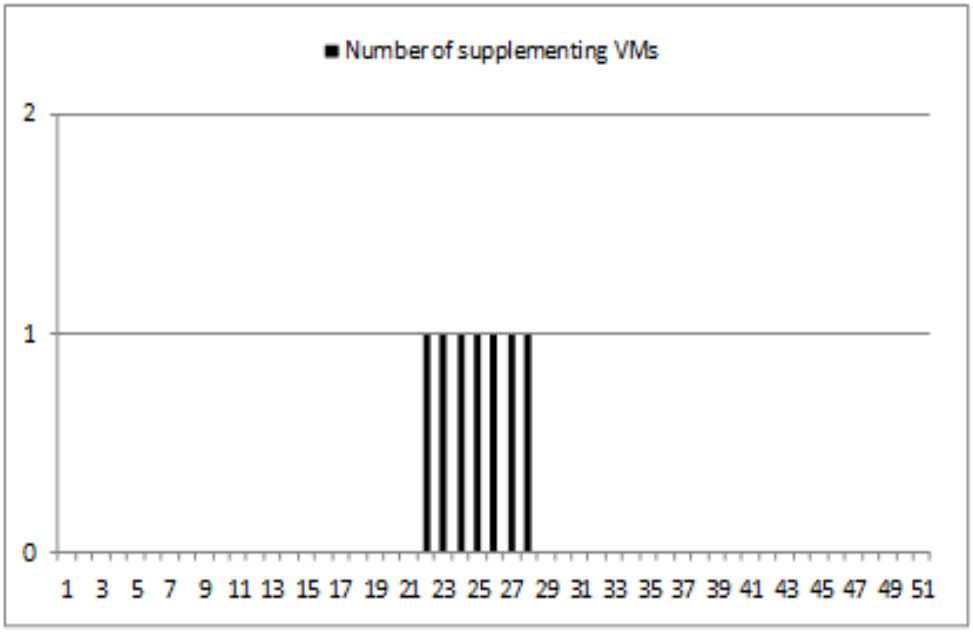


Hình 2.18: Số lượng VM được bổ sung khi sử dụng Round Robin

Thứ hai, tập trung vào sự khác biệt theo độ lệch $Dev(t)$ giữa các bộ cân bằng tải sử dụng Round Robin và AMLB-New. Các kết quả từ cả hai trường hợp được mô tả trong hình 2.20. Hình 2.20 cho thấy độ lệch từ bộ cân bằng tải sử dụng Round Robin thường lớn hơn so với độ lệch của bộ cân bằng tải giám sát tích cực AMLB-New. Các độ lệch có thể đạt được giá trị là 25.2 với thuật toán Round Robin, trong khi giá trị lớn nhất trong thuật toán AMLB-New là khoảng 15.5. Ngoài ra, hình 2.20 cũng cho thấy sự gia tăng theo độ lệch tương ứng với mỗi lần bổ sung các VM vào hệ thống đám mây. Hình 2.20 cũng cho thấy sự gia tăng từ bộ cân bằng tải sử dụng AMLB-New không lớn hơn giá trị trung bình, trong khi đó các giá trị này khá cao và dài hơn so với bộ cân bằng tải sử dụng Round Robin.

Như vậy, cân bằng tải trong các môi trường đám mây có cơ chế AS đã được xem xét. Có nhiều loại cân bằng tải có thể làm cho bộ AS thêm các VM tạm thời vào

hệ thống thuê bao nhiều hơn các bộ cân bằng tải khác. Ví du: bộ cân bằng tải Round Robin. Điều này dẫn đến việc sử dụng tài nguyên đám mây không hiệu quả và các thuê bao phải trả thêm tiền. Nghiên cứu này cũng chỉ ra rằng các bộ cân bằng tải áp dụng các thuật toán dựa trên tải công việc hiện tại của VM trong hệ thống đám mây đã thích nghi tốt với môi trường đám mây có AS. Ví dụ, bộ cân bằng tải sử dụng AMLB-New không chỉ giúp giữ mức mất cân bằng tải giữa các VM thấp mà còn hỗ trợ AS hiệu quả. Vì vậy, cần cân nhắc lựa chọn các cân bằng tải dựa trên tải công việc của VM trong môi trường đám mây cho phép AS, đặc biệt là các bộ AS sử dụng các thuật toán dựa trên ngưỡng.



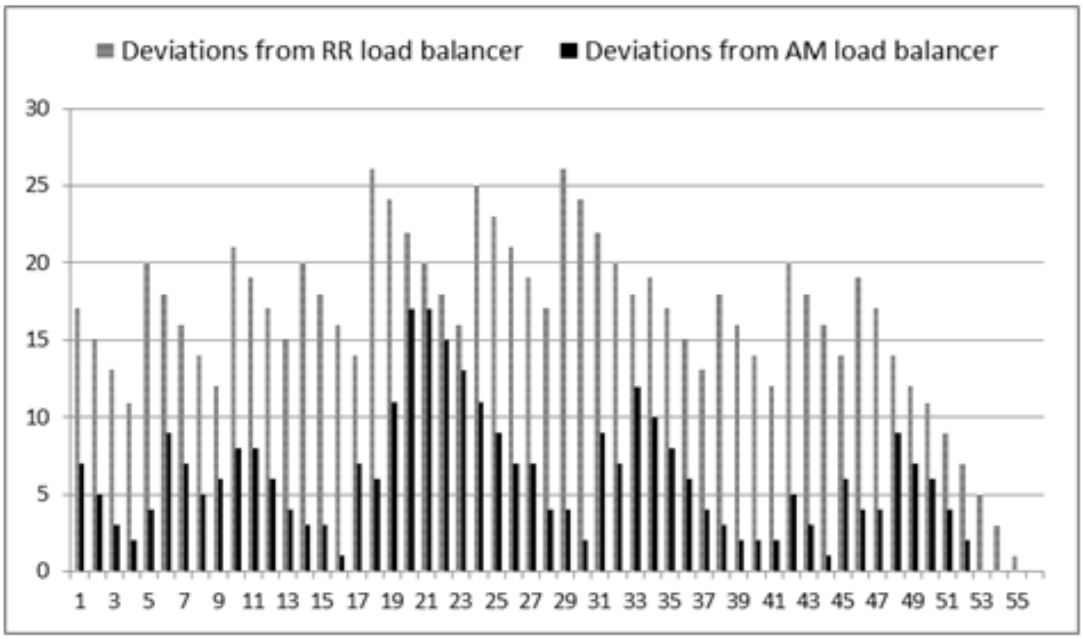
Hình 2.19: Số lượng VM được bổ sung khi sử dụng AMLB-New

KẾT LUẬN CHƯƠNG 2:

Chương này, luận án đã nghiên cứu về các cơ chế căn bản trong CC đó là kỹ thuật cân bằng tải và kỹ thuật di trú, và đã đề xuất một kỹ thuật cân bằng tải động hiệu quả và một kỹ thuật di trú hiệu quả trong CC. Cuối cùng là đánh giá ảnh hưởng của các kỹ thuật cân bằng tải trong CC có cơ chế AS tài nguyên, các kết quả thực hiện và mô phỏng đã chỉ ra tính hiệu quả của các kỹ thuật được đề xuất và đánh giá được vai trò, sự ảnh hưởng của các kỹ thuật cân bằng tải trong một trường CC có cơ chế AS, điều đó cho thấy, việc lựa chọn kỹ thuật cân bằng tải phù hợp trong môi

trường CC có cơ chế AS sẽ giúp cho hệ thống hoạt động tốt hơn, tận dụng được tài nguyên và tối thiểu được chi phí.

Trong phần tiếp theo, luận án sẽ nghiên cứu và đề xuất một mô hình CC sử dụng mô hình hàng đợi để xem xét và đánh giá một số thông số hiệu năng trong CC như: thời gian đáp ứng trung bình, thời gian chờ trung bình, thông lượng,… Các tham số đầu ra từ mô hình là các tham số đầu vào quan trọng cho các giải pháp AS trong môi trường CC, như thời gian đáp ứng trung bình, số lượng VM,…



Hình 2.20: Độ lệch Dev(t) theo bộ cân bằng tải sử dụng Round Robin và sử dụng AMLB-New

CHƯƠNG 3 : ĐỀ XUẤT MỘT MÔ HÌNH MẠNG HÀNG ĐỢI CHO HỆ THỐNG ĐIỆN TOÁN ĐÁM MÂY

Chương này nghiên cứu các mô hình CC sử dụng mô hình hàng đợi và mạng hàng đợi. Đã có nhiều công trình sử dụng mô hình hàng đợi đơn để mô hình CC, nhưng các mô hình này không còn phù hợp với độ phức tạp ngày càng phát triển của môi trường CC. Do đó, cần phải nghiên cứu mô hình mạng hàng đợi để có thể mô hình được môi trường CC phức tạp và không đồng nhất. Mô hình mạng hàng đợi được đề xuất sử dụng cho các ứng dụng đa tầng trên nền tảng CC. Các số đo hiệu năng của mô hình là dữ liệu đầu vào quan trọng cho các bài toán AS trong CC như: thời gian đáp ứng, số lượng VM được cấp phát/thu hồi, mức sử dụng hệ thống,… Nội dung chương này được tổng hợp từ công trình TCNN3.

3.1. ĐẶT VẤN ĐỀ

Các công trình [57, 58, 100, 101, 107, 109] sử dụng các mô hình hàng đợi phổ biến (như M/M/1, M/G/1, M/M/m, M/G/m/K) để tìm hiểu và phân tích hành vi các dịch vụ đàn hồi. Chưa có mô hình nào tập trung vào việc xác định số lượng VM cần thiết để đáp ứng tiêu chí hiệu năng SLO cho các dịch vụ dựa trên đám mây. Ngoài ra, các mô hình này không đạt được các hành vi thực sự của các dịch vụ đàn hồi. Cụ thể, tất cả các mô hình này bỏ qua vai trò của bộ cân bằng tải. Vì bộ cân bằng tải đóng một vai trò quan trọng trong việc phân bổ, giám sát và theo dõi tính sẵn sàng của các VM tính toán tại trung tâm dữ liệu đám mây. Thời gian xử lý này tại bộ cân bằng tải có thể là đáng kể. Do đó, bất kỳ mô hình phân tích nào cũng phải tính đến vai trò của bộ cân bằng tải để mô hình chính xác hành vi và hiệu năng.

Một trung tâm dữ liệu cung cấp dịch vụ CC thường có môi trường không đồng nhất vì đa phần nó chứa nhiều thế hệ PM có cấu hình phần cứng khác nhau, đặc biệt là dung lượng và tốc độ của bộ vi xử lý. Các PM này thường được thêm vào trung tâm dữ liệu dần dần và được cung cấp để thay thế các máy hiện tại (hoặc "cũ") đã có sẵn [24, 74]. Tính không đồng nhất của các nền tảng PM này sẽ làm ảnh hưởng đến hiệu năng của các trung tâm dữ liệu. Bên cạnh đó, các ứng dụng triển khai trên CC ngày nay được phát triển theo hướng dịch vụ. Theo đó, một ứng dụng được triển khai trên một

tập các dịch vụ triển khai độc lập. Điều này không giống như các ứng dụng nguyên khối bao gồm các mô-đun kết hợp chặt chẽ, các ứng dụng dựa trên kiến trúc hướng dịch vụ rất phù hợp cho các cơ sở hạ tầng CC [91].

Sự biến động của tải công việc thường xuyên xảy ra trong môi trường CC. Biến động này gây ảnh hưởng đến chất lượng cung cấp cho người dùng mặc dù kiến trúc CC có tính co giãn - khả năng tự động cấp phát và thu hồi tài nguyên dựa vào nhu cầu tải công việc hiện tại. Do vậy, vấn đề đảm bảo QoS là một thành phần tiêu chuẩn của SLA được thiết lập giữa khách hàng và nhà cung cấp dịch vụ đám mây, đây là một trong những vấn đề quan trọng đối với nhà cung cấp CC [31, 102]. Để đánh giá QoS cho CC có nhiều công trình nghiên cứu sử dụng các số đo hiệu năng quan trọng của hệ thống như thời gian đáp ứng trung bình, thời gian chờ trung bình, số thông lượng công việc trung bình, khả năng bị từ chối phục vụ [103]. Các số đo này có thể được phân tích và mô hình dựa trên lý thuyết hàng đợi. Các ứng dụng triển khai trên CC thường được xây dựng bằng các mô hình hàng đợi đơn giản như một hàng đợi G/G/N, trong đó có N máy chủ là thay đổi [2]. Mô hình được sử dụng để ước lượng các tham số như các tài nguyên cần thiết cho một tải công việc đầu vào nhất định hoặc thời gian đáp ứng trung bình cho các yêu cầu. Sau đó, thông tin này được chuyển vào bộ dự đoán, bộ điều khiển hoặc để giải quyết bài toán tối ưu hóa. Tuy nhiên, khi kiến trúc của ứng dụng trên CC phát triển và ngày càng trở nên phức tạp thì việc sử dụng mô hình hàng đợi đơn trở nên khó khăn. Do đó, mô hình mạng hàng đợi được sử dụng để tạo một tầng ứng dụng đơn gồm N máy chủ ứng dụng [99, 104] hoặc xem xét một hàng đợi cho một máy chủ [101], hoặc chỉ là một hàng đợi cho mỗi tầng [6, 112]. Đa số các nghiên cứu hiện tại ít khi xem xét tính không đồng nhất của dịch vụ cơ sở hạ tầng của CC chẳng hạn như một ứng dụng được triển khai trên đám mây nhiều VM với các thế hệ khác nhau có tốc độ CPU khác nhau và khả năng bộ xử lý khác nhau. Thay vào đó, các nghiên cứu thường giả định rằng mỗi nút hoạt động ở cùng tốc độ.

Đánh giá hiệu năng của các trung tâm CC đã được một số nghiên cứu tập trung gần đây. Tuy nhiên, do sự phức tạp và đa dạng của các yêu cầu người dùng của trung

tâm CC, hiệu năng của nó rất khó để mô hình hóa và phân tích. Dịch vụ đám mây khác với dịch vụ lưu trữ truyền thống theo ba khía cạnh chính: Thứ nhất, nó được cung cấp theo nhu cầu; Thứ hai, các trung tâm đám mây ảo hóa cho phép nhiều VM chạy trên cùng một PM; và Thứ ba, di trú trực tuyến VM được sử dụng rộng rãi để cải thiện độ tin cậy của hệ thống.

Trong phần này tập trung vào khía cạnh QoS dành cho khách hàng. Nghiên cứu và đề xuất một mô hình hàng đợi để phân tích, đánh giá các số đo hiệu năng cho một môi trường CC không đồng nhất nhằm thỏa mãn các nhu cầu về QoS cho khách hàng. Ở đây, khách hàng là người sử dụng dịch vụ cơ sở hạ tầng CC để triển khai ứng dụng web đa tầng trên các VM có cấu hình khác nhau (không đồng nhất) để phục vụ cho người dùng đầu cuối. Do đó, luận án mô hình hệ thống ứng dụng đa tầng trên nền tảng CC dựa vào lý thuyết mạng hàng đợi, cụ thể là mạng Jackson mở [103] để xác định và đo các thông số QoS của đám mây như: thời gian đáp ứng, thời gian đợi trung bình của hệ thống. Sau đó, phân tích mô hình theo các tham số khác nhau như: tỉ lệ yêu cầu đến của các dịch vụ khách hàng, số lượng và tốc độ phục vụ của các máy chủ xử lý và những tham số khác.

Mô hình đề xuất là tổng quát có thể áp dụng để mô tả và nắm bắt hành vi của các hệ thống CC. Thứ nhất, mô hình có thể được sử dụng trong thuật toán AS, trong đó công suất và số lượng VM tính toán được xác định dựa vào thời gian đáp ứng được giám sát, mức sử dụng tài nguyên của hệ thống,… Hiện tại, Amazon AWS sử dụng AS dựa vào việc thiết lập các ngưỡng dưới và ngưỡng trên cho mức sử dụng CPU tổng thể. Mức sử dụng CPU có thể gây ra sai sót vì có thể bị ảnh hưởng từ các tác nhân đang chạy, các tác vụ hoặc các quá trình không liên quan đến tải công việc của ứng dụng. Thứ hai là đối với các ứng dụng đa tầng, mô hình đề xuất có thể sử dụng để tính toán độ trễ mong đợi ở mỗi tầng. Độ trễ tích hợp ở mỗi tầng có thể được sử dụng để tính toán thời gian đáp ứng đầu cuối. Thứ ba, mô hình đề xuất có thể tính toán yêu cầu các VM và ước lượng thời gian đáp ứng dịch vụ và yêu cầu băng thông mạng. Vì các VM liên tục được cấp phát và thu hồi (do AS, di trú và kết thúc), thời gian lưu trú trung bình cho một VM có thể được ước tính. Mô hình đề xuất có thể sử

dụng để xác định các số đo hiệu năng chính, bao gồm: mức sử dụng tài nguyên, băng thông mạng, thời gian phục vụ, thông lượng,…

3.2. MÔ HÌNH HÓA ỨNG DỤNG ĐA TẦNG TRÊN ĐIỆN TOÁN ĐÁM MÂY

3.2.1. Mô hình ứng dụng đa tầng

Một ứng dụng đa tầng ảo hóa trong môi trường CC được triển khai trên nhiều VM và mỗi tầng cung cấp một chức năng nhất định. Ở đây, xem xét một ứng dụng bao gồm n tầng, được biểu thị bằng $T_1, T_2, \ldots, T_n$.

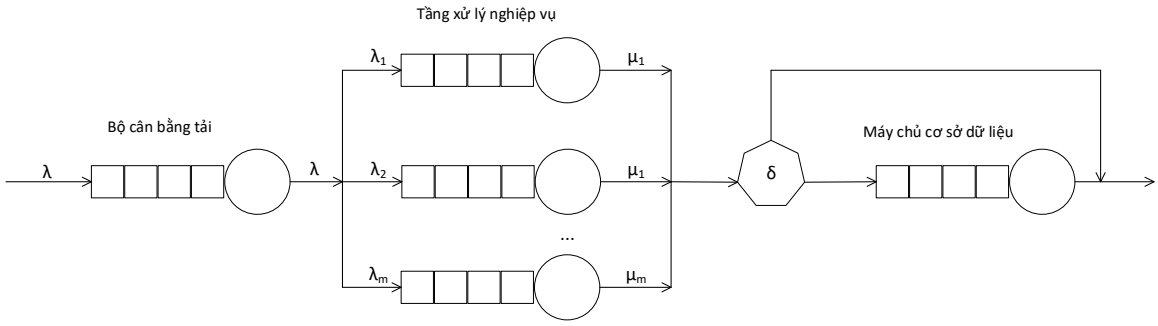
Giả sử có c các VM trong mỗi tầng của môi trường thực thi ứng dụng đa tầng và các yêu cầu của tất cả các phiên làm việc đến nhập vào một hàng đợi chung được duy trì bởi bộ cân bằng tải ở tầng thứ nhất, chờ các tài nguyên sẵn sàng. Bộ cân bằng tải chịu trách nhiệm phân phối các yêu cầu này. Sau đó, mỗi yêu cầu được quyết định xử lý tại các VM của tầng tiếp theo trong ứng dụng ảo hóa.

Mô hình hàng đợi nhiều VM của các tầng khác có trách nhiệm cung cấp tài nguyên động theo các yêu cầu của tầng đó. Mỗi yêu cầu ở tầng T_i được chuyển tiếp tới tầng tiếp theo T_{i+1} của nó để xử lý tiếp hoặc có thể rời khỏi hệ thống. Khi kết quả được xử lý bởi tầng cuối cùng T_n, kết quả được gửi phản hồi cho khách hàng. Ví dụ, một yêu cầu trang web tĩnh được xử lý hoàn toàn bởi tầng Web và sẽ không được chuyển tiếp tới các tầng sau. Mặt khác, tìm kiếm từ khóa ở tầng Web có thể kích hoạt nhiều yêu cầu đến tầng tiếp theo. Giả sử tầng cơ sở dữ liệu với kiến trúc chia sẻ mọi thứ, có thể được nhóm lại và nhân bản theo nhu cầu, truy cập cơ sở dữ liệu với xác suất δ. Thông qua việc mô hình hóa tất cả các tầng và các tương tác của chúng, mô hình đa tầng cho phép tích hợp các quyết định cho tầng đầu tiên vào bộ cân bằng tải và các tầng khác có thể chứa một số VM. Để nắm bắt ứng dụng đa tầng ảo hóa cho việc cung cấp tài nguyên động, việc xác định mô hình đề xuất là một mô hình phân tích kết hợp, phù hợp với môi trường thực. Điều này không chỉ tiết kiệm thời gian truyền trong mạng mà còn cải thiện hiệu quả xử lý các yêu cầu.

3.2.2. Mô hình hàng đợi mở cho ứng dụng đa tầng

Tải công việc trên ứng dụng đa tầng thường là khách hàng dựa trên phiên làm việc, trong đó phiên làm việc khách hàng bao gồm một loạt các yêu cầu. Như trong

Hình 1.6, ứng dụng trên CC có nhiều tầng khác nhau đảm nhận vai trò khác nhau trong ứng dụng cũng như nhu cầu sử dụng các dạng VM khác nhau tại các tầng. Tại một thời điểm, nhiều yêu cầu khách hàng đồng thời tương tác với ứng dụng đa tầng. Hình 3.1 xem xét mô hình mạng hàng đợi mở cho ứng dụng đa tầng gồm có ba tầng để mô hình cho ứng dụng đa tầng như hình 1.6, với cấu trúc các VM có thể khác nhau, mô hình hệ thống phân bố như một hệ thống hàng đợi gồm nhiều hàng đợi M/M/1. Các yêu cầu tại mỗi nút trong mạng được phục vụ theo thứ tự đến trước được phục vụ trước (FCFS). Sự xuất hiện của các yêu cầu ở mỗi tầng được giả định đến theo phân bố Poisson, tức là khoảng thời gian giữa các lần yêu cầu đến có phân bố mũ. Giả thiết này được xác nhận bằng cách phân tích các bộ dữ liệu được lấy từ một trang web ứng dụng đa tầng thương mại điện tử [10]. Bên cạnh đó, thời gian phục vụ trung bình của các yêu cầu cũng được coi là phân bố mũ. Các giả định này cũng được sử dụng phổ biến trong phân tích các hệ thống CC như trong các công trình [1, 57, 58, 92, 100, 101, 107].



Hình 3.1: Mô hình mạng hàng đợi mở cho ứng dụng đa tầng trên CC

Hệ thống mạng hàng đợi trong Hình 3.1 thể hiện cho cụm phục vụ ứng dụng của người dùng cuối: bao gồm tầng chứa bộ cân bằng tải, tầng xử lý nghiệp vụ và tầng xử lý cơ sở dữ liệu. Tầng chứa bộ cân bằng tải là một VM đóng vai trò điều phối yêu cầu đến các VM trong tầng xử lý nghiệp vụ. Tầng xử lý nghiệp vụ gồm nhiều VM không đồng nhất, có khả năng xử lý khác nhau tùy vào cấu hình. Sau đó, yêu cầu có thể được chuyển đến tầng xử lý dữ liệu là một VM hoặc là không tùy vào yêu cầu ứng dụng người dùng có hoặc không truy cập vào cơ sở dữ liệu, sau đó kết quả được chuyển đến người dùng.

Tầng chứa bộ cân bằng tải nhận các yêu cầu đến từ người dùng đầu cuối và sau đó xử lý, điều phối yêu cầu đến các VM ở tầng xử lý nghiệp vụ dựa trên cấu hình của

chúng. Chính sách cấp phát tài nguyên cho yêu cầu dưới dạng chia sẻ không gian, có nghĩa tại một thời điểm VM chỉ phục vụ cho một yêu cầu. Việc điều phối yêu cầu từ bộ cân bằng tải đến các VM ở tầng xử lý nghiệp vụ được tính theo chính sách cân bằng tải Round Robin (Bảng 3.1). Tùy vào trọng số của từng VM trong tầng xử lý nghiệp vụ, tốc độ phục vụ của VM phụ thuộc vào năng lực xử lý của VM đó. Ở đây tập trung vào tính không đồng nhất của VM, giả sử mỗi VM i có c_i core xử lý, tốc độ thực thi của mỗi core là s đo bằng giga lệnh trên mỗi giây (GIPS).

Tốc độ phục vụ của VM i được tính như sau:

$$\mu_i = \frac{s * c_i}{\bar{Z}}, \tag{3.1}$$

trong đó, $\bar{Z}$ là số chỉ thị trung bình của một yêu cầu được thi tại VM i.

Trong tầng xử lý nghiệp vụ, các VM xử lý trong mạng có thể có tải phụ thuộc, có nghĩa là tỉ lệ yêu cầu rời dịch vụ là một hàm của số lượng yêu cầu hiện tại trong VM đó μ_i. Trong tầng lưu trữ dữ liệu, xác suất δ là một yêu cầu từ VM trong tầng xử lý nghiệp vụ đến truy cập vào tầng lưu trữ cơ sở dữ liệu. Khi thực hiện mô hình hóa một hệ thống dựa trên web, không phải tất cả yêu cầu sẽ yêu cầu truy cập vào máy chủ cơ sở dữ liệu.

Chúng tôi xem xét các giả định sau, các giả định này cũng được sử dụng trong [92, 93, 103]:

(i) Các yêu cầu đến từ ngoài hệ thống CC vào một nút i *tuân theo quá trình Poisson;*

(ii) Thời gian phục vụ tại mỗi nút i *trong hệ thống CC là độc lập và tuân theo phân bố mũ và được phục vụ theo nguyên tắc FCFS;*

(iii) Xác suất khi một yêu cầu được phục vụ xong tại một nút i *có thể chuyển qua nút* j *(*$i \neq j$*)* $r_{ij} \geq 0$ *độc lập với trạng thái hệ thống hoặc sẽ rời khỏi hệ thống và không quay lại với xác suất* $1 - \sum_{j=1}^{N} r_{ij}$*, với* N *là số nút mạng trong mạng hàng đợi.*

Các tính chất này thỏa mãn các yêu cầu của một mạng hàng đợi Jackson mở [95]. Như vậy, có thể xem hệ thống CC được mô hình như hình 3.1 là một mạng hàng đợi Jackson mở có N nút mạng.

Bảng 3.1: Thuật toán cân bằng tải Round Robin

```
2  * Đầu vào:
3  * vms[ ]: danh sách các VM
4  * requests: danh sách các yêu cầu
5  * Đầu ra:
6  * gán các yêu cầu cho các VM */
7  map;
8  position = 0;
9  for request in requests
10     position %= vms.size()
11     if vms[position] có thể xử lý request
12         vm=vms[position]
           map.add(request, vm)
13         position++
14 return map
```

3.2.3. Công thức của Little và một số kết quả của hàng đợi M/M/1

3.2.3.1. Công thức của Little

Trước khi đi phân tích mô hình đề xuất, phần này giới thiệu công thức đóng một vai trò quan trọng trong việc suy luận các kết quả tổng quát về các hệ thống lý thuyết hàng đợi. Khi xây dựng một hệ thống, thường làm đơn giản hóa các giả định về nó - quá trình đến hoặc quá trình phục vụ là quá trình Poisson,…. Khi những giả định này không thỏa mãn, công thức kết quả cho hành vi hệ thống trở nên quá khó giải quyết hoặc thậm chí không thể suy luận. Tuy nhiên, có một công thức tổng quát, nó đúng cho tất cả các hệ thống ổn định.

Nếu hệ thống ổn định, nghĩa là nó trở nên trống rỗng trong khoảng thời gian cuối cùng - tốc độ đến bằng với tốc độ đi ra, có công thức của Little [47, 48, 95]:

$$\overline{L} = \lambda * \overline{W} \tag{3.2}$$

trong đó $\overline{L}$ là số khách hàng/yêu cầu trung bình trong hệ thống và $\overline{W}$ là thời gian trung bình mà khách hàng/yêu cầu ở trong hệ thống. Vì tổng số thời gian chờ trong hệ thống bao gồm thời gian chờ trong hàng đợi $\overline{W_Q}$ và thời gian phục vụ $\overline{W_S}$: $\overline{W} = \overline{W_Q} + \overline{W_S}$ (cũng giống như số khách hàng: $\overline{L} = \overline{L_Q} + \overline{L_S}$), công thức của Little cũng có dạng cho hàng đợi và thời gian chờ phục vụ:

$$\overline{L_Q} = \lambda * \overline{W_Q} \quad (3.3)$$

$$\overline{L_S} = \lambda * \overline{W_S} \quad (3.4)$$

Công thức của Little là đúng cho tất cả các loại hệ thống và nó được sử dụng để dẫn xuất cho nhiều kết quả.

3.2.3.2. Hàng đợi M/M/1

Bây giờ, xem xét các thuộc tính của hàng đợi M/M/1-FCFS [95], trong đó các khách hàng đến theo quá trình Poisson với tốc độ λ (hay nói cách khác, các khoảng thời gian liên tiếp theo phân bố mũ), thời gian phục vụ theo mũ phân bố với giá trị trung bình là $\frac{1}{\mu}$, và có một máy chủ phục vụ khách hàng theo chính sách FCFS. Cho $X(t)$ thể hiện số lượng khách hàng trong hệ thống tại thời điểm t. Trong một khoảng thời gian ngắn $(t, t + h]$ một khách hàng mới có thể đến với xác suất: $\lambda h + o(h)$ (với việc chuyển trạng thái từ $i \rightarrow i + 1$) và một khách hàng rời khỏi hệ thống với một xác suất $\mu h + o(h)$ (với việc chuyển trạng thái từ $i \rightarrow i - 1$), trong đó $o(h)$ là một hàm bất kỳ thỏa mãn: $\lim_{h \to 0} o(h)/h = 0$. Do đó $X(t)$ tạo thành một chuỗi Markov và $X(t)$ cũng là một quá trình sinh-tử với không gian trạng thái vô hạn S = {0, 1, 2,...}. Bằng cách tạo ra các phương trình cân bằng tổng quát và một điều kiện chuẩn hóa [95], có thể có được công thức xác suất trạng thái (phân bố cân bằng). Số lượng khách hàng trung bình $E[X]$ trong hệ thống được tính theo công thức (3.9). Phương sai $D^2[X]$ được tính theo (3.10).

$$\rho = \frac{\lambda}{\mu} \quad (3.5)$$

$$\pi_i = \rho^i \pi_0, i = 0, 1, 2, \ldots \quad (3.6)$$

$$\pi_0 = 1 - \rho \quad (3.7)$$

$$E[X] = \frac{\frac{\lambda}{\mu}}{1 - \frac{\lambda}{\mu}} = \frac{\rho}{1 - \rho} \quad (3.8)$$

$$D^2[X] = \frac{\rho}{(1-\rho)^2} \quad (3.9)$$

Nếu sự phân bố thời gian đến hoặc thời gian phục vụ không phải hàm mũ và chính sách lập lịch vẫn là FCFS, kết quả trên sẽ không có. Tuy nhiên, để lập lịch theo PS công thức vẫn giữ nguyên mặc dù phân phối thời gian phục vụ sẽ thay đổi, tức là công thức trên vẫn đúng cho các hàng đợi M/G/1 -PS.

Tổng thời gian trễ D của hệ thống bao gồm cả thời gian chờ đợi W và thời gian phục vụ S: D = W + S. Độ trễ trung bình có thể được tính theo công thức của Little [47, 48] và nó thỏa mãn cho tất cả các công việc – duy trì các chính sách lập lịch (FCFS, PS, LCFS,...). Trong trường hợp hàng đợi M/M/1-FCFS, độ trễ trung bình được tính theo công thức 3.11 và 3.12:

$$E[D] = \frac{E[X]}{\lambda} = \frac{1}{\lambda} * \frac{\rho}{1-\rho} = \frac{1}{\mu - \lambda} \quad (3.10)$$

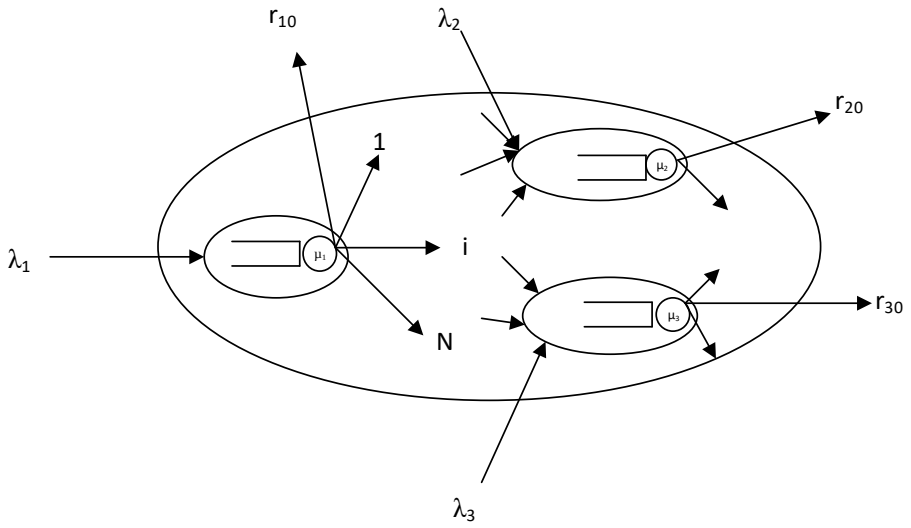
Thời gian chờ của khách hàng là: W = D – S.

$$E[W] = E[D] - E[S] = \frac{1}{\mu} * \frac{1}{1-\rho} - \frac{1}{\mu} = \frac{1}{\mu} * \frac{\rho}{1-\rho} \quad (3.11)$$

3.2.4. Phân tích mô hình đề xuất

Trong phần này phân tích mô hình mạng hàng đợi thể hiện trong Hình 3.1 - đây là một mạng hàng đợi Jackson mở gồm N nút mạng, mỗi nút mạng tương ứng là một VM và mỗi VM được mô hình là một hàng đợi M/M/1.

Tại bộ cân bằng tải sẽ nhận các yêu cầu đến từ người dùng bên ngoài và sau đó xử lý, điều phối yêu cầu đến các VM trong tầng xử lý nghiệp vụ dựa trên cấu hình của chúng. Yêu cầu đến bộ cân bằng tải từ bên ngoài có tốc độ đến theo phân bố Poisson với tham số γ (số lượng yêu cầu trên giây). Khoảng thời gian giữa các lần đến trung bình là $1/\gamma$. Thiết lập hệ số mức sử dụng tài nguyên trung bình cho mỗi VM thứ i (ký hiệu ρ_i) với ngưỡng nằm trong khoảng $[\alpha_l, \alpha_h]$ tùy vào từng loại VM khác nhau với tốc độ phục vụ μ_i. Điều này mô tả tốc độ mà các máy chủ tại mỗi nút xử lý các yêu cầu $1/\mu_i$ là thời gian phục vụ trung bình.



Hình 3.2: Mạng hàng đợi mở tổng quát

Các nút có thể liên kết với nhau theo dạng nối tiếp hoặc song song. Giả sử độ dài của mỗi hàng đợi là vô hạn và nguyên tắc phục vụ của hàng đợi là FCFS. Tại mỗi nút của hệ thống là một hàng đợi M/M/1 và việc phát sinh các yêu cầu HTTP của người dùng là quá trình ngẫu nhiên độc lập. Gọi r_{ij} là xác suất chuyển trạng thái sau khi yêu cầu được xử lý xong tại nút i sẽ chuyển đến nút j. Ta có, tổng các xác suất chuyển trạng thái trong mạng hàng đợi mở là 1 (ở đây: r_{i0} là xác suất chuyển ra ngoài mạng từ nút i):

$$r_{i0} = 1 - \sum_{j=1, j\neq i}^{N} r_{ij}, i \in [1, N] \tag{3.12}$$

Gọi λ_i là tốc độ đến từ bên ngoài vào nút thứ i và Λ_i là tổng tốc độ đến của nút thứ i (bao gồm cả tốc độ đến từ bên ngoài và tốc độ chuyển đến từ các nút bên trong hệ thống mạng đến nút thứ i), khi đó ta có:

$$\Lambda_i = \lambda_i + \sum_{j=1, j\neq i}^{N} \Lambda_j r_{ji}, i \in [1, N] \tag{3.13}$$

Gọi $\Lambda = [\Lambda_1, \Lambda_2, \dots, \Lambda_N]^T$ là vector tổng tốc độ đến các hàng đợi của các nút. Gọi $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_N]^T$ là vector tốc độ đến từ bên ngoài, khi đó công thức (3.13) có thể được viết lại dưới dạng phương trình ma trận sau:

$$\Lambda = \lambda + R^T \Lambda, \tag{3.14}$$

với R là ma trận xác suất chuyển trạng thái (R^T là ma trận chuyển vị của ma trận R).

$$R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1N} \\ r_{21} & r_{12} & \cdots & r_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ r_{N1} & r_{N1} & \cdots & r_{NN} \end{pmatrix} \tag{3.15}$$

Do số lượng yêu cầu ở các VM có thể khác nhau, ta gọi $X_i(t)$ là biến ngẫu nhiên xác định số yêu cầu trong VM thứ i ($i = 1,2,..,N$) tại thời điểm t. Trạng thái của hệ thống tại thời điểm t được ký hiệu là: $X(t) = (X_1(t), X_2(t), \dots, X_N(t))$. Khi đó, ở trạng thái ổn định, xác định phân phối xác suất chung trong thời gian dài như sau: $p_{x_1,x_2,\dots,x_N} \equiv P(x_1, x_2, \dots, x_N) = P_r\{X_1(t) = x_1, X_2(t) = x_2, \dots, X_N(t) = x_N\}$.Vì các VM là độc lập với nhau và $P(x_i)$ là xác suất biên với $X_i(t) = x_i$. Từ xác suất chung, có thể tính xác suất biên của một số lượng yêu cầu cụ thể tại mỗi VM.

Một mạng Jackson mở [95] được xem xét như một chuỗi Markov thời gian liên tục với vector trạng thái:

$$\bar{x} = (x_1, x_2, \dots, x_N), \tag{3.16}$$

trong đó với x_i là số lượng yêu cầu đang có tại VM i. Sử dụng phương trình trạng thái cân bằng dựa trên hệ thống Markov.

Trong Bảng 3.2, gọi $\bar{x}$ biểu diễn trạng thái ổn định của hệ thống; khi có một yêu cầu đến VM i thì hệ thống từ trạng thái $\bar{x}$ sang trạng thái $\bar{x}; i^+$; ngược lại khi có một yêu cầu rời khỏi VM i, hệ thống sẽ chuyển từ trạng thái $\bar{x}$ sang trạng thái $\bar{x}; i^-$; hoặc trạng thái hệ thống từ $\bar{x}$ sang trạng thái $\bar{x}; i^+j^-$ khi có một yêu cầu từ VM j chuyển đến VM i.

Bảng 3.2. Bảng mô tả trạng thái

Trạng thái	Ký hiệu
$x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_N$	$\bar{x}$
$x_1, \dots, x_{i-1}, x_i + 1, x_{i+1}, \dots, x_N$	$\bar{x}; i^+$
$x_1, \dots, x_{i-1}, x_i - 1, x_{i+1}, \dots, x_N$	$\bar{x}; i^-$
$x_1, \dots, x_{i-1}, x_i + 1, x_{i+1}, \dots, x_{j-1}, x_j - 1, x_{j+1}, \dots, x_N$	$\bar{x}; i^+j^-$

Xích Markov này có xác suất chuyển ký hiệu $p_{\cdot,\cdot}$ được xác định như sau:

$$p_{\bar{x};i^+} = \lambda_i$$
$$p_{\bar{x};i^-} = \mu_i r_{i0}$$
$$p_{\bar{x};i^+j^-} = \mu_j r_{ji}$$

Sử dụng nguyên lý cân bằng trạng thái luồng vào trạng thái $\bar{x}$ bằng luồng ra khỏi trạng thái $\bar{x}$, giả sử $x_i \geq 1$ tại mọi VM trong hệ thống, ta có phương trình cân bằng tổng quát sau:

$$\sum_{i=1}^{N} \lambda_i \, p_{\bar{x};i^-} + \sum_{j=1}^{N} \sum_{i=1}^{N} \mu_i r_{ij} p_{\bar{x};i^+j^-} + \sum_{i=1}^{N} \mu_i r_{i0} p_{\bar{x};i^+}$$
$$= \sum_{i=1}^{N} \mu_i (1 - r_{ii}) \, p_{\bar{x}} + \sum_{i=1}^{N} \lambda_i \, p_{\bar{x}} \qquad (3.17)$$

Định lý trong [47, 48]: Phân bố cân bằng cho mạng Jackson mở là:

$$p_{\bar{x}} = C \prod_{i=1}^{N} \rho_i^{x_i}, \text{với } \rho_i = \frac{\Lambda_i}{\mu_i} < 1.$$

Theo [95] cung cấp phân bố chung cho tất cả VM trong hệ thống, giải pháp trạng thái ổn định cho phương trình (3.17) là:

$$\begin{aligned} p_{\bar{x}} \equiv p_{x_1, x_2, \ldots, x_N} &= (1-\rho_1)\rho_1^{x_1}(1-\rho_2)\rho_2^{x_2} \ldots (1-\rho_N)\rho_N^{x_N} \\ &= \prod_{i=1}^{N} (1-\rho_i)\rho_i{}^{x_i}. \end{aligned} \qquad (3.18)$$

$$\text{với } C = \prod_{i=1}^{N} (1 - \rho_i).$$

Số lượng yêu cầu trung bình L_i tại mỗi VM i cho hàng đợi M/M/1 với tổng tốc độ đến Λ_i:

$$L_i = \frac{\rho_i}{1 - \rho_i}, i = 1,2, \ldots, N \qquad (3.19)$$

Tổng số lượng yêu cầu trung bình của toàn mạng được tính như sau:

$$L = L_1 + L_2 + \cdots + L_N = \sum_{i=1}^{N} \frac{\rho_i}{1-\rho_i}. \quad (3.20)$$

Thời gian đợi trung bình của yêu cầu trong mạng với Luật của Little [47, 48]:

$$W = \frac{L}{\beta}, \quad (3.21)$$

trong đó $\beta = \sum_{i=1}^{N} \lambda_i$ là tổng tốc độ đến từ bên ngoài vào mạng. Tất cả yêu cầu của khách hàng đến từ bên ngoài đều phải qua bộ cân bằng tải: $\beta = \gamma$. Thời gian đáp ứng trung bình của một yêu cầu trong mạng tại mỗi VM thứ i ($i = 1,2,\dots,N$) được tính bằng công thức như sau:

$$W_i = \frac{L_i}{\Lambda_i} = \frac{1}{\mu_i(1-\rho_i)}, \quad (3.22)$$

trong đó Λ_i được tính theo công thức (3.4) và $W \neq W_1 + W_2 + \cdots + W_N$.

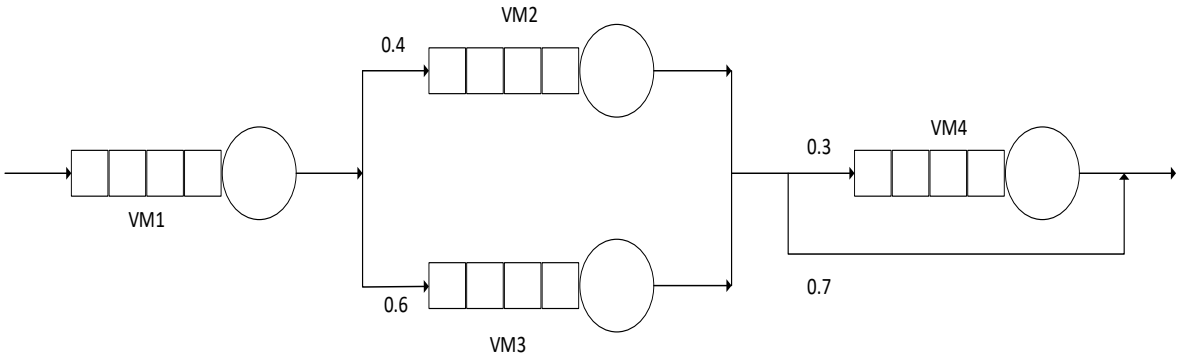
3.2.5. Một ví dụ cho mô hình mạng hàng đợi đề xuất

Hình 3.3 là một ví dụ về mô hình hàng đợi Jackson mở cho hệ thống CC ứng dụng ba tầng gồm bốn nút, mỗi nút mạng là một VM được mô hình sử dụng một hàng đợi M/M/1. Nút 1 là VM_1 đại diện cho tầng thứ nhất là Bộ cân bằng tải có tốc độ yêu cầu đến từ bên ngoài vào là γ, tầng xử lý nghiệp vụ gồm 2 nút là VM_2 và VM_3, tốc độ yêu cầu được chuyển tiếp từ VM_1 đến các VM trong tầng xử lý nghiệp vụ tuân theo xác suất chuyển trạng thái tương ứng là $r_{12} = 0.4$ và $r_{13} = 0.6$. Sau khi một yêu cầu được phục vụ xong ở tầng xử lý nghiệp vụ, có thể chuyển sang truy cập vào tầng lưu trữ cơ sở dữ liệu VM3 với xác suất $r_{34} = r_{24} = \delta = 0.3$ hoặc là không với xác suất là $r_{30} = r_{20} = 0.7$. Sau đó kết quả được phản hồi đến khách hàng. Tốc độ phục vụ tại mỗi VM thứ i theo phân bố mũ có tham số μ_i.

Khi đó, ta có ma trận chuyển trạng thái sau:

$$R = \begin{pmatrix} 0 & 0.4 & 0.6 & 0 \\ 0 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (3.23)$$

trong đó, xác suất rời khỏi hệ thống tại mỗi VM thứ i, $i \in [1,2,3,4]$ tương ứng là: $r_{10} = 0, r_{30} = r_{20} = 0.7, r_{40} = 1$.



Hình 3.3: Ví dụ mô hình mạng hàng đợi mở cho ứng dụng đa tầng trên CC

Các tham số đầu vào: tốc độ đến từ bên ngoài vào bộ cân bằng tải γ = 20 requests/sec. Thời gian phục vụ tại các VM lần lượt là: $\frac{1}{\mu_1} = 0.03\ sec, \frac{1}{\mu_2} = 0.06\ sec, \frac{1}{\mu_3} = 0.05\ \text{sec}\ and\ \frac{1}{\mu_4} = 0.04\ sec$.

Giả sử phải tính xác suất trạng thái ổn định của trạng thái sau:

$$P(n_1, n_2, n_3, n_4) = P(n_1 = 3, n_2 = 2, n_3 = 4, n_4 = 1)$$

Tức là tính xác suất ở trạng thái ổn định tại thời điểm ở bộ cân bằng tải (VM1) có 3 yêu cầu, máy chủ xử lý 1 (VM2) có 2 yêu cầu, máy chủ xử lý 2 (VM3) có 4 yêu cầu và máy chủ cơ cở dữ liệu (VM4) có 1 yêu cầu đang xử lý. Để tính $P(n_1 = 3, n_2 = 2, n_3 = 4, n_4 = 1)$, thực hiện tháo các bước sau:

Bước 1: Tính tốc độ đến cho mỗi nút (VM) theo công thức lưu lượng (4): $\Lambda_1 = \gamma = 20;\ \Lambda_2 = \Lambda_1 . r_{12} = 8; \Lambda_3 = \Lambda_1 . r_{13} = 12;$ và $\Lambda_4 = \Lambda_2 . r_{24} + \Lambda_3 . r_{34} = 6$

Bước 2: Tính các xác suất trạng thái cho mỗi nút (VM) tại thời điểm xét:

Sử dụng công thức tính mức sử dụng tài nguyên trung bình cho mỗi VM: $\rho_i = \frac{\Lambda_i}{\mu_i}$ để biết nhu cầu phục vụ tại mỗi nút:

$\rho_1 = 0.6, \rho_2 = 0.48, \rho_3 = 0.6,$ và $\rho_4 = 0.24$.

Tính xác suất tại mỗi VM, sử dụng công thức:

$P_i(n_i) = P(X_i(t) = n_i) = (1 - \rho_i)\rho_i^{n_i}$ để tính xác suất có n_i yêu cầu trong hàng đợi M/M/1:

$P_1(3) = 0.09, P_2(2) = 0.12, P_3(4) = 0.05,$ và $P_4(1) = 0.18$.

Bước 3: Tính xác suất trạng thái ổn định $P(3,2,4,1)$. Theo công thức (9): $P(3,2,4,1) = P_1(3)P_2(2)P_3(4)P_4(1)$=0.0000972.

Sau đây là một số số đo quan trọng khác cần được xem xét khi tiến hành mô phỏng và thực nghiệm cho mô hình đề xuất: Tính số lượng yêu cầu trung bình trong mỗi VM theo công thức: $L_i = \frac{\rho_i}{1-\rho_i}$, kết quả như sau:

$L_1 = 1.5, L_2 = 0.92, L_3 = 1.5$ và $L_4 = 0.32$.

Tính thời gian đáp ứng trung bình tại mỗi VM theo công thức: $W_i = \frac{L_i}{\Lambda_i}$, kết quả như sau:

$W_1 = 0.075, W_2 = 0.115, W_3 = 0.125,$ và $W_4 = 0.053$.

Tính thời gian đáp ứng trung bình cho toàn hệ thống theo công thức (12):

$$W = \frac{L}{\gamma} = \frac{1}{\gamma}\sum_{i=1}^{4} L_i = 0.212 \neq W_1 + W_2 + W_3 + W_4 = 0.368$$

Như vậy, qua ví dụ trên cho thấy, mô hình đề xuất có thể xác định được xác suất của trạng thái ổn định, xác định các số đo quan trọng như thời gian đáp ứng trung bình, số yêu cầu trung bình,…

3.3. THỰC NGHIỆM VÀ ĐÁNH GIÁ

3.3.1. Thiết lập

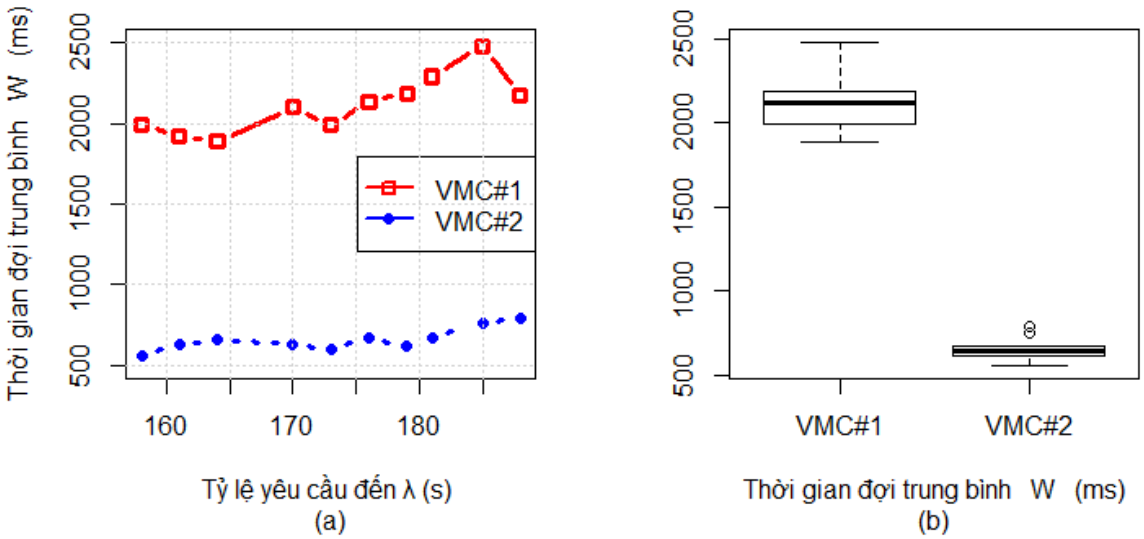
Trong phần này tiến hành thực nghiệm để đánh giá hiệu năng các số đo được lập trình bằng ngôn ngữ Java trên máy tính Laptop có cấu hình Intel® Core™ i7-7500U CPU @ 2.70GHz, 8GB Ram, 1 TB HDD và chạy hệ điều hành Window 10. Thực nghiệm xem xét hệ thống dịch vụ cơ sở hạ tầng CC gồm hai cụm VM phục vụ yêu cầu từ khách hàng sử dụng dịch vụ web. Mỗi cụm VM có 7 VM (ID từ 1 đến 7). Trong mỗi cụm sẽ có một VM đảm nhận công việc của bộ cân bằng tải (VM1), một VM đảm nhận việc lưu trữ cơ sở dữ liệu (VM7) và các máy còn lại đảm nhận công việc ở tầng xử lý nghiệp vụ (vm_i, với $i = 2, 3, \ldots, 6$). Khả năng xử lý của mỗi VM được tính dựa theo GIPS. Hai cụm VM có cấu hình khác nhau theo Bảng 3.3, trong cả 2 cụm các máy ở bộ cân bằng tải và máy lưu trữ cơ sở dữ liệu có khả năng như

nhau. Trong cụm VMC#1, các máy xử lý nghiệp vụ có khả năng đồng nhất với nhau. Ở cụm VMC#2 các máy xử lý nghiệp vụ không đồng nhất với nhau.

Bảng 3.3. Cấu hình cụm VM

VM	Khả năng xử lý (GIPS)	
	VMC#1	VMC#2
1	4	4
2	2	2
3	2	4
4	2	6
5	2	8
6	2	10
7	4	4

3.3.2. Kết quả thực nghiệm

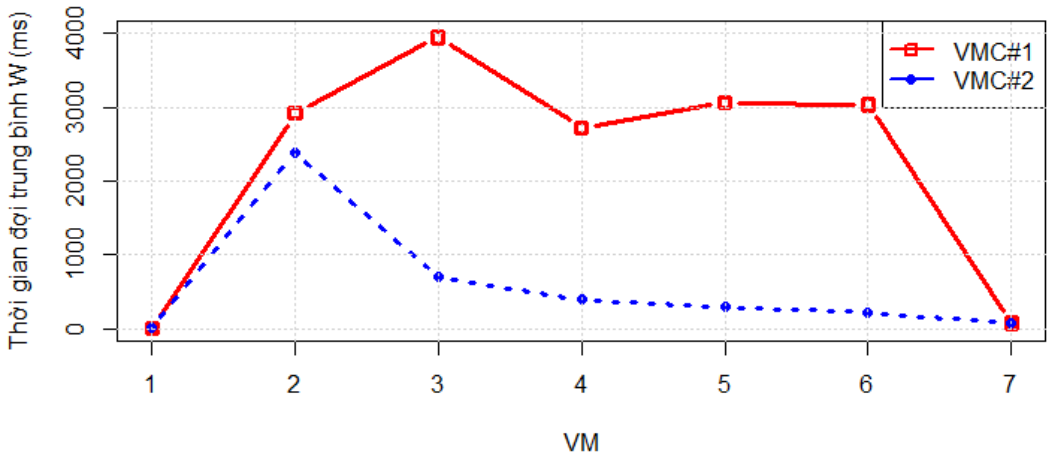


Hình 3.4: Thời gian đợi trung bình của cụm VM cho thực nghiệm 1

Thực nghiệm 1 nhằm đánh giá các tham số trong mô hình, cho số lượng yêu cầu đến là 1000, γ= {158,161,164,167,170,173,176,179,181,185, 188} tốc độ xử lý (phục vụ) của bộ cân bằng tải là $\mu_{LB} = \frac{\gamma}{\rho_{LB}}$. Xác suất chuyển trạng thái đến các VM ở tầng xử lý nghiệp vụ là r_{ij}. Sau 30 lần chạy thực nghiệm, tính được thời gian đợi trung bình $\overline{W}$, thời gian đáp ứng trung bình $\overline{T}$. Sau đó, tính toán độ tin cậy 95% cho dữ liệu thực nghiệm và dữ liệu tính toán từ mô hình.

Kết quả thực nghiệm 1 thể hiện trong hình 3.4, cho thấy khi điều chỉnh việc tăng tốc độ yêu cầu đến từ 158 lên 188 thì thời gian đợi trung bình tăng nhưng biên độ tăng

không lớn. Đối với cụm VMC#1 các máy có cấu hình CPU đồng nhất với nhau nên trong khi đó cụm VMC#2 có cấu hình CPU không đồng nhất với nhau và tốc độ xử lý lớn hơn VMC#1 do vậy thời gian đáp ứng trung bình của VMC#2 nhỏ hơn nhiều so với VMC#1 (Hình 3.4b).



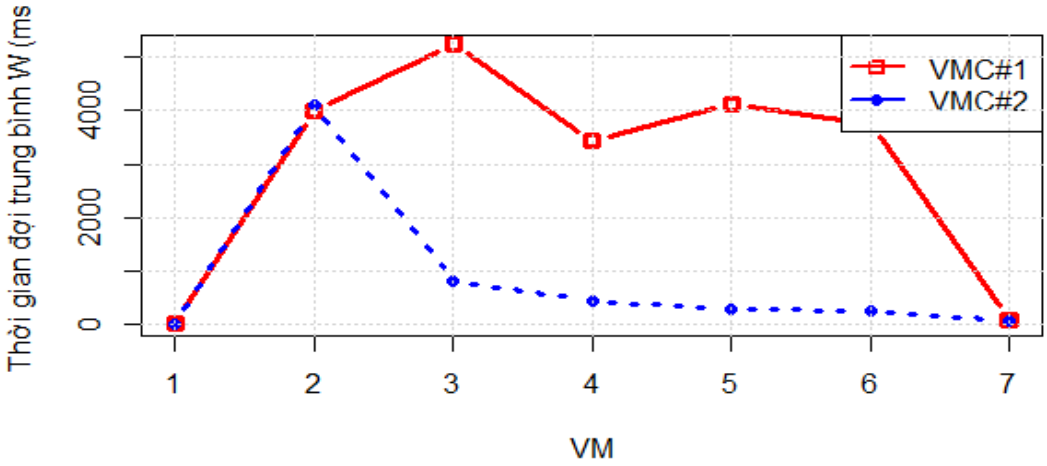
Hình 3.5: Thời gian đợi trung bình của các VM cho thực nghiệm 2

Thực nghiệm 2 sử dụng bộ dữ liệu của Bảng 3.3 với $\lambda = 185$, được thực hiện 30 lần và tính thời gian đợi trung bình $\bar{W}$. Sau đó, tính toán độ tin cậy 95% cho dữ liệu thực nghiệm và dữ liệu tính toán từ mô hình.

Bảng 3.4. Khoảng tin cậy 95% của thời gian đợi trung bình của mỗi VM trong từng cụm

Thời gian đợi trung bình $\bar{W}$					
VM	ρ_i	**VMC**	**Trung bình**	**Cận dưới**	**Cận trên**
1	0.8	VMC#1	6.39069	6.20901	6.57238
		VMC#2	6.37414	6.16965	6.57863
2	0.71	VMC#1	3987.14	3126.29	4848.00
		VMC#2	4115.76	3241.04	4990.49
3	0.73	VMC#1	5256.16	4211.63	6300.7
		VMC#2	803.147	701.051	905.243
4	0.75	VMC#1	3430.45	2538.11	4322.80
		VMC#2	434.463	401.364	467.562
5	0.77	VMC#1	4127.60	3154.68	5100.51
		VMC#2	277.476	249.991	304.960
6	0.79	VMC#1	3756.96	2980.27	4533.66
		VMC#2	236.522	219.641	253.403
7	0.81	VMC#1	73.9026	68.5611	79.2442
		VMC#2	69.6167	65.9044	73.3289

Kết quả ở Hình 3.5 cho thấy thời gian đáp ứng trung bình của VM1 và VM7 là rất nhỏ ở cả hai cụm VMC#1 và VMC#2. Ở VM1 đảm nhận công việc cân bằng tải, thời gian xử lý nhanh. Đối với VM7 đảm nhận công việc lưu trữ cơ sở dữ liệu, với xác suất yêu cầu đến là $\delta = 0.2$. Các VMi (i=2,3, …,6) ở tầng xử lý nghiệp vụ có thời gian đáp ứng trung bình ở hai cụm khác nhau. Theo đó, thời gian đáp ứng trung bình của từng máy trong cụm VMC#1 có biên độ giao động không quá lớn. Trong khi đối với cụm VMC#2 có xu hướng nhỏ dần từ VM2 đến VM6, do cấu hình CPU của VM2 đến VM6 ở cụm VMC#2 tăng dần nên thời gian xử lý ở các máy này giảm dần.



Hình 3.6: Thời gian đợi trung bình của các VM cho thực nghiệm 3

Thực nghiệm 3 nhằm phân tích mối quan hệ của thông số trong mô hình, sử dụng hai cụm VM như trong Bảng 3.3 với tốc độ đến trung bình $\lambda = 185$, điều chỉnh mức độ sử dụng tài nguyên trung bình ρ_i của mỗi VM theo Bảng 3.4 và sau đó phân tích thời gian đáp ứng trung bình của hệ thống đối với từng VM trong cụm. Điều chỉnh mức độ sử dụng tài nguyên trung bình ρ_i của các VM ở tầng xử lý nghiệp vụ của ứng dụng từ 0.71 đến 0.79. Kết quả thực nghiệm được thể hiện trong bảng 3.4 và hình 3.6.

KẾT LUẬN CHƯƠNG 3

Chương này đã đề xuất được một mô hình CC sử dụng mạng Jackson mở, nhằm đánh giá các số đo hiệu năng của hệ thống. Với kết quả thực nghiệm, bước đầu đã đánh giá được sự đúng đắn của mô hình, đưa ra các số đo của hệ thống có thể tin cậy. Các thông số này sẽ làm đầu vào cho giải pháp AS được nghiên cứu trong Chương 4 tiếp theo. Nội dung tiếp theo sẽ trình bày về giải pháp AS hiệu quả trong môi trường CC trong đó có sử dụng bộ điều khiển mờ kết hợp với phương pháp học tăng cường.

CHƯƠNG 4 : GIẢI PHÁP TỰ ĐỘNG ĐIỀU CHỈNH TÀI NGUYÊN CHO ỨNG DỤNG ĐA TẦNG TRÊN ĐIỆN TOÁN ĐÁM MÂY

Chương này trình bày về giải pháp AS tài nguyên trong CC. Luận án xây dựng một bộ AS dựa vào phương pháp học tăng cường và điều khiển mờ, tạo ra một cơ chế tự thích nghi mạnh mẽ, nhằm kết nối kinh nghiệm của con người với cơ chế tiến hóa liên tục. Chương này được tổng hợp từ các công trình TCNN2, HNQT3 và HNTN1.

4.1. ĐẶT VẤN ĐỀ

Các hệ thống ứng dụng dựa trên web thường xuyên gặp phải sự cố về tải. Ví dụ, một ứng dụng Facebook gần đây đã tăng 10 lần số người dùng từ 25.000 lên 250.000 chỉ trong ba ngày với tối đa 20.000 đăng ký mới mỗi giờ trong thời gian cao điểm [50]. Các hệ thống kinh doanh thông thường như vậy phải đáp ứng các SLA nhất định, ví dụ: các giới hạn trên về thời gian đáp ứng nhận được của người dùng. Ngược lại, tải công việc không mong muốn gây ra mức dịch vụ kém làm cho khách hàng không hài lòng. Amazon báo cáo mất 245 triệu đô la để tăng 100ms về thời gian đáp ứng [50]. Để tránh tình trạng như vậy và duy trì QoS, việc quản lý tự động các ứng dụng như vậy là cần thiết. Đã có một số nghiên cứu và quan tâm thực tế trong việc cung cấp tài nguyên tự động cho các ứng dụng như vậy, đó là các giải pháp AS.

Điều quan trọng của bộ AS là làm thế nào, thời điểm và số lượng tài nguyên cần thực hiện điều chỉnh để đáp ứng các SLA. Giảm chi phí ứng dụng và đáp ứng các SLA là hai yếu tố quan trọng trong việc thiết kế bộ điều khiển tự động. Các ứng dụng trên môi trường CC thường có tải công việc biến động theo thời gian, do đó việc quản lý tài nguyên và sử dụng cần linh hoạt tùy thuộc theo nhu cầu của khách hàng. Để tránh việc cung cấp tài nguyên quá ít hay quá nhiều trong khi đó vẫn duy trì được QoS là thách thức cần được nghiên cứu và giải quyết. Hệ thống CC có thể AS mở rộng hay thu hồi tài nguyên để đảm bảo tính sẵn sàng cho các ứng dụng. Để giải quyết thách thức này, nhiều hướng tiếp cận khác nhau [18, 68, 80] gọi là AS, đã được đề xuất. Trạng thái hiện tại phụ thuộc các luật dựa vào ngưỡng và nhờ vào tính đơn giản và trực quan, chúng chủ yếu được cung cấp bởi nhiều nhà cung cấp/nền tảng CC thương mại như Amazon EC2, Microsoft Azure, OpenStack. Thực tế, việc quản lý,

thường là tập các luật điều chỉnh nhỏ và dễ hiểu, giả định sự phụ thuộc tuyến tính và liên tục giữa cấp phát tài nguyên và cải tiến hiệu năng, trong khi ở các ứng dụng điều chỉnh trên Internet, độ phức tạp của kiến trúc ứng dụng, xung đột giữa các thành phần, tần suất do xuất hiện lỗi phần cứng và phần mềm phát sinh thường làm mất hiệu lực các giả định này [32].

Các xu hướng gần đây dựa trên bộ điều khiển tự tổ chức đã chứng minh là có thể phù hợp tốt hơn cho độ phức tạp của bộ điều khiển đám mây [28, 32]. Tuy nhiên, một thách thức thực tế vẫn chưa được giải quyết, đó là sự phụ thuộc vào người sử dụng để xác định bộ điều khiển đám mây. Có một số vấn đề từ phía nhà cung cấp CC, các chi tiết về ứng dụng không nhìn thấy được, làm khó khăn cho việc đưa ra một cách chính xác tập luật tối ưu. Do đó, khó khăn của việc xác định tập luật như vậy lại giao cho phía người sử dụng CC, nhưng họ lại không có đủ tri thức về tải công việc, cơ sở hạ tầng hoặc xây dựng mô hình hiệu năng. Vì vậy, bộ điều khiển đám mây trong thực tế vẫn phụ thuộc vào người sử dụng/người vận hành khi xác định các tham 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/người vận hành. Như trong công trình [50] đã sử dụng logic mờ để khám phá tri thức trực quan của con người, chuyển tri thức này thành một tập luật **NẾU-THÌ** để thực hiện AS. Tuy nhiên, do tập luật này phải được xác định ngay từ khi bắt đầu xây dựng bộ AS nên có thể gặp những vấn đề sau: tri thức có thể không có sẵn, người dùng không thể tạo ra bất kỳ luật nào; tri thức có thể có sẵn nhưng có một phần, người dùng chỉ có thể xác định các luật một phần cho một số trường hợp; tri thức không phải luôn tối ưu, người dùng có thể xác định các luật nhưng chúng không có hiệu quả, ví dụ: các luật dư thừa; tri thức có thể được chính xác đối với một số luật nhưng có thể ít chính xác đối với một số luật khác, ví dụ: các luật chứa sự không chắc chắn, tùy thuộc vào mức độ tri thức ưu tiên; tri thức có thể cần phải thay đổi trong thời gian thực hiện, các luật có thể được chính xác tại thời gian thiết kế nhưng có thể thay đổi trong thời gian chạy. Kết quả là, người sử dụng các luật đã được định nghĩa trước có thể dẫn đến các quyết định điều chỉnh tối ưu cục bộ và tốn tiền trả cho các nhà cung cấp ứng dụng CC.

Do vậy, phương pháp sử dụng cơ chế học tăng cường có thể giúp giải quyết vấn đề trên. Kỹ thuật sử dụng học tăng cường [97] là một loại phương pháp ra quyết định tự động đã được sử dụng trong bộ AS. Không có bất kỳ tri thức ưu tiên nào, các kỹ thuật học tăng cường có thể xác định hành động điều chỉnh tốt nhất để thực hiện cho mọi trạng thái của ứng dụng, với tải công việc đầu vào. Tuy nhiên, để khắc phục một số hạn chế của phương pháp học tăng cường, như: không gian trạng thái lớn, hiệu năng ban đầu kém, thời gian huấn luyện dài,…, đã cố một số phương pháp được đề xuất như mạng nơron, tính toán tiến hóa [32, 54].

Tuy nhiên, với mạng nơron phù hợp cho việc xấp xỉ hàm, phân loại và nhận dạng mẫu. Một nhược điểm của mạng nơron là đòi hỏi phải có số lượng lớn các tập huấn luyện chuẩn bị trước khi đưa vào hoạt động thực tế, có thể là hạn chế về máy chủ để thử nghiệm huấn luyện trong trường hợp AS tài nguyên trên CC rất tốn kém và mất nhiều thời gian [32]. Theo [54] cho thấy phương pháp kết hợp giữa hai quá trình mạng nơron và học tăng cường cho tốc độ học rất chậm.

Với thuật toán tính toán tiến hoá, như thuật toán di truyền và thuật toán tối ưu hóa bầy đàn, dựa vào việc tạo ra một cách ngẫu nhiên và so sánh sự phát triển của nhiều gen và nhiều cá thể, mỗi cá thể thể hiện một cấu hình khác nhau của bộ điều khiển đám mây. Tuy nhiên, việc thiếu tính tổng quát trong định nghĩa của bộ điều khiển đám mây dẫn đến giai đoạn tối ưu phải lặp lại nhiều lần để tìm ra cấu hình tối ưu trong không gian dữ liệu lớn, nhất là trong môi trường CC.

Cuối cùng, luận án lựa chọn học tăng cường với những lý do sau:

- Tải công việc cho các ứng dụng dựa trên đám mây là không thể dự đoán trước và để có được một bộ dữ liệu huấn luyện thực tế làm đại diện của tất cả các tình huống thời gian chạy sẽ là một nhiệm vụ không khả thi. Không giống như hướng tiếp cận học có giám sát khác (ví dụ: mạng nơron), trong học tăng cường không cần một tập dữ liệu huấn luyện.

- Do không dự đoán trước được tải công việc và độ phức tạp của ứng dụng dựa trên đám mây, các nhà cung cấp không có tri thức đầy đủ để thực hiện hành động

điều chỉnh thích hợp. Kỹ thuật học tăng cường không yêu cầu biết tri thức về ứng dụng và tri thức của hệ thống sẽ được cập nhật dựa vào các quan sát gần nhất [97].

Ở đây, luận án sử dụng kết hợp giữa điều khiển mờ và học Q mờ 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 [51]. Tuy nhiên, trong [51] 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 CC lại có rất nhiều ứng dụng khác nhau thực thi cùng lúc tại một thời điểm. Bên cạnh đó, công trình [51] chỉ xem xét đến thời gian đáp ứng và tải của hệ thống làm tham số đầu vào cho bộ AS. Công trình [53] sử dụng tải công việc và số lượng VM làm tham số đầu vào cho bộ AS.

Trong Chương này, luận án xem xét bài toán AS tài nguyên trên CC, các yêu cầu của ứng dụng được xử lý trong một cụm VM không đồng nhất, giả sử mỗi yêu cầu được xử lý bởi một VM. Nhu cầu của ứng dụng thay đổi theo thời gian sẽ ảnh hưởng tới số lượng VM trong cụm dùng để xử lý các yêu cầu của ứng dụng đó. Ở đây, luận án sử dụng các thông số như tài nguyên khả dụng của mỗi VM, tài nguyên khả dụng trung bình của cụm VM đang phục vụ ứng dụng đó và thời gian đáp ứng trung bình để làm tham số điều chỉnh cho bộ điều khiển AS. Các tham số tài nguyên khả dụng trung bình của cụm VM và phương sai của nó được tính toán theo mô hình trong phần 4.3.1. Còn tham số thời gian đáp ứng trung bình được tính theo mô hình trong Chương 3.

Ở đây giả định tất cả sự ngẫu nhiên trong hệ thống có thể được xử lý tốt bởi một quá trình Markovian. Số đo dịch vụ cho ứng dụng là thời gian đáp ứng (rt) cho các yêu cầu của người dùng. Mỗi VM phải chịu một khoảng chi phí nào đó cho mỗi đơn vị thời gian. Nếu thời gian đáp ứng cao hơn SLA thì hệ thống phải chịu một hình phạt nào đó. Trong một hành động điều chỉnh, bộ AS có thể thêm hoặc loại bỏ một số VM nhất định. Giả sử không có chi phí cho hành động điều chỉnh. Bộ AS thực hiện điều chỉnh thích hợp nhằm đạt được lợi nhuận cao nhất trong thời gian dài mà không cần có tri thức trước. Bộ AS có thể thực hiện điều chỉnh tài nguyên mà không cần có tri

thức ưu tiên, có nghĩa là có thể hoạt động với một cơ sở tri thức rỗng và bộ AS sẽ có được tri thức trong thời gian thực hiện, thông qua cơ chế tiến hóa tri thức.

4.2. KIẾN TRÚC ỨNG DỤNG ĐA TẦNG

Phần này nghiên cứu kiến trúc ứng dụng đa tầng có dạng như trong hình 1.6 đã được trình bày trong mục 1.1.5 của Chương 1. Các tài nguyên ảo hóa được nhà cung cấp dịch vụ đám mây IaaS cung cấp dưới dạng VM. Các VM được sử dụng để lưu trữ các ứng dụng thông qua cổng tự phục vụ. Thêm vào đó, các VM được gom nhóm lại thành các cụm xử lý các công việc chuyên biệt như tối ưu tính toán, tối ưu hóa bộ nhớ, tối ưu hóa lưu trữ, GPU... Người sử dụng có thể chọn lựa linh hoạt các VM thích hợp cho các ứng dụng của họ. Ví dụ: loại VM tối ưu CPU dành cho các máy chủ web và loại tối ưu bộ nhớ cho cơ sở dữ liệu hiệu năng cao. Các loại VM khác nhau bao gồm các kết hợp khác nhau về khả năng CPU, bộ nhớ, dung lượng lưu trữ, mạng và tính sẵn sàng với mô hình giá khác nhau. Một ứng dụng có thể được triển khai trên một cụm VM, trong đó các VM có thể có cấu hình không đồng nhất.

Các ứng dụng, được xem xét trong nghiên cứu này là ứng dụng máy chủ web [16]. Một ứng dụng web là một ứng dụng tương tác cung cấp nội dung như các trang web bằng cách sử dụng giao thức truyền siêu văn bản (HTTP) qua Internet và liên quan đến các ràng buộc thời gian đáp ứng nhỏ. Ứng dụng này thực hiện một số loại hành động hoặc chức năng nhất định cho khách hàng và có thể được mô hình hoá như là một tập hợp các yêu cầu độc lập được thực hiện bởi khách hàng với các thể hiện ứng dụng. Một ứng dụng được biểu diễn thành một chuỗi n yêu cầu độc lập $\{r_1, r_2, \ldots, r_n\}$ nhận được từ các thể hiện ứng dụng tại các thời điểm $\{t_1, t_2, \ldots, t_n\}$. Vì các yêu cầu là độc lập với nhau, không có phụ thuộc giao tiếp hay phụ thuộc kiểm soát giữa các yêu cầu. Việc triển khai các ứng dụng này trên đám mây được giả định thực hiện bằng các cụm VM lưu trữ trong các PM.

Để cung cấp hiệu năng chấp nhận được cho khách hàng, các nhà cung cấp dịch vụ thường hướng đến mục tiêu QoS nhất định, ví dụ: thời gian đáp ứng, thông lượng, độ trễ, thời gian thực hiện và thời gian giao dịch. Kiểm soát chất lượng phù hợp tạo ra các dịch vụ có uy tín dẫn đến có được sự hài lòng của khách hàng và từ đó tăng cơ

sở khách hàng và doanh thu. Công trình này hướng đến mục tiêu thời gian đáp ứng trung bình là một số đo QoS được sử dụng rộng rãi và có ảnh hưởng trực tiếp đến hiệu năng của hệ thống CC và có tính đến tài nguyên khả dụng, tài nguyên còn lại có thể đáp ứng yêu cầu của khách hàng. Hình 1.1 mô tả vòng lặp kiểm soát hiệu năng được sử dụng rộng rãi trong môi trường CC. Vòng lặp MAPE giám sát tải công việc ứng dụng và tự động bổ sung hoặc thu hồi các VM phù hợp với nhu cầu tải công việc đến để đạt được các mục tiêu QoS cần thiết. Mô hình giá dựa trên mô hình thanh toán theo phút với thời hạn thanh toán tối thiểu 10 phút dựa trên Google Cloud[13]. Ví dụ, nếu một VM được chạy trong 2 phút, nó sẽ được tính hóa đơn cho 10 phút sử dụng. Tuy nhiên, sau 10 phút, các VM được tính phí theo từng phút một lần, được làm tròn đến phút gần nhất.

AS tài nguyên trên CC có thể được xây dựng dựa trên hệ thống quản lý tự trị theo vòng lặp MAPE được trình bày trong phần 1.1.2. Việc lập kế hoạch tuân theo các luật có thể đơn giản như một chính sách sự kiện - điều kiện - hành động (ECA), 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 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.

4.3. ĐỀ XUẤT MÔ HÌNH TỰ ĐỘNG ĐIỀU CHỈNH

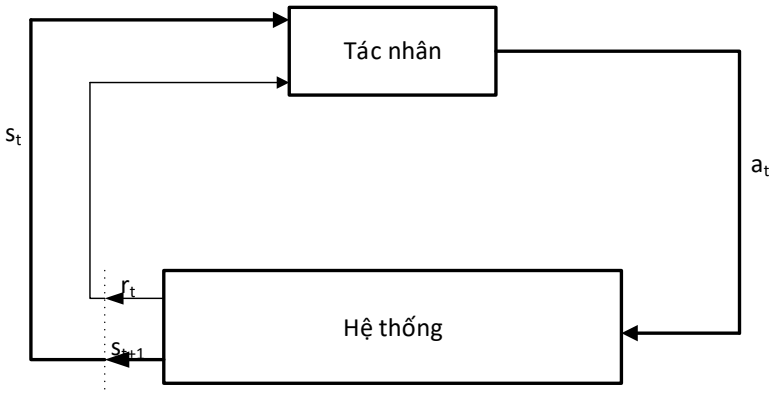
4.3.1. Phân tích mô hình

Bộ giám sát thu thập dữ liệu về tình trạng sử dụng tài nguyên và thời gian đáp ứng trung bình của hệ thống để chuyển đến bộ phận AS đưa ra các hành động điều chỉnh tài nguyên thích hợp cho từng yêu cầu của công việc. Cụ thể, bộ AS sẽ đưa ra quyết định tăng/giảm số lượng VM phục vụ công việc cụ thể. Hành động điều chỉnh VM hướng đến mục đích: đảm bảo QoS, tối đa hóa việc sử dụng tài nguyên của hệ thống và tiết kiệm chi phí sử dụng tài nguyên.

Như phân tích ở trên, việc xây dựng bộ AS là một thách thức lớn đối với việc quản lý và kiểm soát hệ thống. Thêm vào đó, hệ thống ngày càng phức tạp về kiến

[13] https://cloud.google.com/compute/pricing

trúc cũng như về mức độ, cường độ sử dụng. Bên cạnh đó, bộ AS khi triển khai thực tế vẫn phụ thuộc vào người sử dụng/người vận hành, đặc biệt là trong việc xác định các tham số điều khiển cũng như tập luật dùng để điều khiển. Do đó, việc ra quyết định của bộ AS có thể được xem xét như một quá trình quyết định Markov (MDP) – cung cấp nền tảng toán học cho việc mô hình hóa việc ra quyết định trong tình huống mà kết quả là một phần ngẫu nhiên và một phần dưới sự điều khiển của một người ra quyết định. Chiến lược điều khiển tối ưu trong bài toán quyết định Markov dẫn tới việc xây dựng phương trình tối ưu Bellman. Thông thường bài toán này được giải quyết bằng phương pháp quy hoạch động và học tăng cường. Theo đó, phương pháp giải bằng qui hoạch động được sử dụng khi biết được các đặc tính thống kê của hệ thống. Trong khi đó, phương pháp học tăng cường tìm kiếm giải pháp tối ưu thông qua các giá trị phản hồi hay phần thưởng thu nhận được trong quá trình tương tác với hệ thống và trạng thái của hệ thống [23, 54]. Việc xây dựng bộ AS theo hướng tiếp cận mô hình học tăng cường sẽ mang lại một hứa hẹn cho cơ chế kiểm soát có khả năng thích nghi. Ban đầu với ít hoặc không có tri thức về đặc điểm hệ thống, bộ điều khiển bắt đầu thực hiện hành động và tự học thông qua chất lượng phản hồi quan sát từ hệ thống.



Hình 4.1: Mô hình tương tác giữa tác nhân và hệ thống

MDP biểu diễn sự tương tác giữa hệ thống và tác nhân. Trong đó, tác nhân (bộ AS) đóng vai trò ra quyết định khi nhận được thông tin phản hồi từ hệ thống.

Tại thời điểm t, ta có:

- Trạng thái hệ thống được ký hiệu là $s_t \in S$ với S là tập trạng thái của hệ thống có thể có tại thời điểm khác nhau,

- Thông qua trạng thái hiện tại s_t của hệ thống, tác nhân có thể ra quyết định thực hiện một hành động $a_t \in A(s_t)$ với xác suất π_t,

- Tác nhân nhận giá trị phần thưởng $r_t \in R$ tại thời điểm $t+1$ với hệ thống chuyển sang trạng thái s_{t+1}.

Ánh xạ $\pi_t: S \to A$ gọi là chiến lược của tác nhân, sau N bước tác nhân thực hiện được một dãy các hành động $\pi = \{\pi_0(s_0), \pi_1(s_1), \dots, \pi_{N-1}(s_{N-1})\}$ và hệ thống nhận được giá trị phản hồi R_t tổng cộng. Giá trị phản hồi được biểu diễn dưới dạng hàm số đối với giá trị phần thưởng. Hàm giá trị trạng thái của hệ thống được tính bằng kỳ vọng toán học của hàm phản hồi R_t theo thời gian. Hàm giá trị V thể hiện mức độ thích hợp của chiến lược điều khiển π đối với tác nhân khi hệ thống đang ở trạng thái s:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} \quad (4.1)$$

Lời giải của MDP là một chính sách thể hiện hành động mà tác nhân nên thực hiện tại mỗi trạng thái s_t sao cho hàm giá trị của trạng thái hệ thống đạt giá trị cực đại. Chính sách như vậy gọi là chính sách tối ưu, kí hiệu là π^* và V^* là giá trị tối ưu.

$$\pi^* = \arg\max_{\pi}\{V^{\pi}(s)\} \quad (4.2)$$

$$V^*(s) = \max_{\pi}\{V^{\pi}(s)\} \quad (4.3)$$

Đối với bài toán quyết định Markov có số bước vô hạn phương trình tối ưu Bellman ở trên tồn tại nghiệm là một chiến lược tối ưu dừng tương ứng với hàm π ánh xạ trạng thái trong S vào không gian hành động A.

Giả sử có n ứng dụng. Mỗi ứng dụng j được thực thi trên một cụm k VM không đồng nhất. Mỗi VM i được biểu diễn dưới dạng vector $v_i = \{c_i, r_i, d_i\}\ \forall\ i \in [1, k]$, trong đó c_i là tham số CPU đã được sử dụng, r_i là lượng bộ nhớ RAM đã được sử dụng và d_i là dung lượng ổ đĩa đã được sử dụng của VM thứ *i* trong cụm thứ *j* tại thời điểm xét. Giả sử các tham số của VM được thu thập dưới dạng phần trăm. Theo Siar và đồng nghiệp [96] đã dùng các số đo khả năng tính toán được sử dụng trung bình và giá trị phương sai của khả năng tính toán được sử dụng để cân bằng tải trong hệ thống phân tán dựa vào lý thuyết trò chơi. Công trình [108] sử dụng số đo khả năng tính toán của một nút trong cụm là một hàm có trọng số của tải CPU, RAM và ổ đĩa

để thực hiện cân bằng tải trong cụm tính toán. Ngoài ra, Branco và đồng nghiệp [12] đề xuất hai chỉ số hiệu năng để đánh giá các hệ thống không đồng nhất. Dựa vào các công trình [12, 96, 108], trong phần này sử dụng khái niệm tài nguyên khả dụng của VM, tài nguyên khả dụng trung bình của cụm VM và phương sai của tài nguyên khả dụng trong cụm VM.

Tài nguyên khả dụng của VM là tài nguyên sẵn sàng của VM có thể phục vụ cho các yêu cầu. Như vậy, tài nguyên khả dụng của mỗi VM *i* trong cụm thứ *j* là w_j^i được tính theo công thức sau:

$$w_j^i = \alpha_1(1 - c_i) + \alpha_2(1 - r_i) + \alpha_3(1 - d_i), \quad (4.4)$$

trong đó, α_1, α_2 và α_3 là các hằng số sao cho $\alpha_1 + \alpha_2 + \alpha_3 = 1$.

Tài nguyên khả dụng trung bình của cụm k VM dành cho ứng dụng j được tính như sau:

$$\overline{w} = \frac{1}{k}\sum_{i=1}^{k} w_j^i \quad (4.5)$$

Như vậy phương sai của k giá trị tài nguyên khả dụng của từng cụm k VM là:

$$var = \frac{\sum_{i=1}^{k}(w_j^i - \overline{w})^2}{k} \quad (4.6)$$

Xem xét dựa trên hai yếu tố để bổ sung hay thu hồi tài nguyên: Giá trị phương sai và giá trị trung bình về tài nguyên khả dụng còn lại.

- Nếu giá trị tài nguyên khả dụng trung bình nhỏ, số lượng tài nguyên còn lại của cụm k VM ít và giá trị phương sai cũng nhỏ, do đó cần bổ sung thêm một số VM nào đó, còn nếu giá trị phương sai lớn, cũng cần bổ sung thêm VM, nhưng số lượng VM bổ sung trong trường hợp này ít hơn.

- Nếu giá trị tài nguyên khả dụng trung bình lớn, tài nguyên sẵn sàng phục vụ nhiều và phương sai nhỏ, do đó cần thu hồi một số VM nào đó, còn nếu phương sai lớn, do đó cần thu hồi một số VM, số lượng VM cần thu hồi trong trường hợp này nhiều hơn.

Ngoài ra, sử dụng tham số thời gian đáp ứng trung bình của hệ thống để đánh giá QoS của hệ thống. Thời gian đáp ứng trung bình (rt) của hệ thống được tính theo Chương 3.

Như vậy, hệ thống điều khiển AS có ba tham số đầu vào là: tài nguyên khả dụng trung bình ($\overline{w}$), phương sai (var) của tài nguyên khả dụng trong một cụm k VM và thời gian đáp ứng trung bình (rt) của hệ thống. Các tham số này sẽ được mờ hóa thành các biến ngôn ngữ theo logic mờ để biểu diễn trạng thái của hệ thống. Tương ứng với các giá trị của các tham số này, hệ thống điều khiển AS sẽ quyết định điều chỉnh thêm vào hay loại bỏ một số VM, số lượng VM được thêm vào hoặc loại bỏ nằm trong tập hành động A= $\{a_1, a_2, .., a_e\}$ $\forall$ a $\in$ N. Chi tiết mờ hóa các tham số này được trình bày ở dưới.

Trong quá trình vận hành hệ thống, chi phí cũng là một trong các tiêu chí để đánh giá chất lượng hệ thống. Chi phí thuê dịch vụ được tính toán dựa trên số lượng tài nguyên đã được sử dụng theo thời gian. Với từng loại VM khác nhau sẽ có đơn giá tính chi phí khác nhau theo thời gian. Ở đây, giả sử đơn giá thuê của từng loại VM phụ thuộc vào số lượng CPU tương ứng của VM. Và đơn giá thuê được tính linh hoạt theo khả năng cấp phát CPU cho VM của máy vật lý theo từng thời điểm khác nhau. Theo đó, nếu lượng tài nguyên được sử dụng nhiều hơn thì khách hàng phải trả chi phí nhiều hơn. Do vậy, đơn giá tỷ lệ thuận với việc sử dụng tài nguyên và thời gian để hoàn thành công việc [20]. Gọi ω_t là số lượng CPU đã được cấp phát. Vì vậy, đơn giá thuê ω_t tài nguyên tại thời điểm t được biểu diễn như sau:

$$p(\omega_t) = a\omega_t + b \ (0 < \omega_t < c), \tag{4.7}$$

trong đó, a, b là hằng số, c là tổng số CPU của hệ thống. Chi phí sử dụng VM của ứng dụng j sử dụng trong thời gian t được tính như sau:

$$U_j(t) = 1 - \frac{vm_j(t) * p(\omega_t)}{cost_{max}}, \tag{4.8}$$

với $vm_j(t)$ là số lượng VM thuê trong thời gian t của ứng dụng j, $cost_{max}$ là chi phí tối đa của ứng dụng j. Chi phí sử dụng $U_j(t)$ sẽ là cơ sở cho việc tính toán giá trị phần thưởng trong thuật toán học tăng cường – học Q mờ.

4.3.2. Bộ tự động điều chỉnh

Bộ điều khiển thích nghi mờ có miền tham số điều chỉnh lớn có thể điều chỉnh các tham số như hàm thành viên, các luật hợp thành, các phép toán OR, AND, NOT, nguyên lý giải mờ,... cho phép điều chỉnh phù hợp với đối tượng chưa biết rõ đã đưa hệ thích nghi mờ trở thành một hệ điều khiển thông minh hơn. Bộ điều khiển thích nghi mờ có khả năng điều chỉnh các tham số của các tập mờ (các hàm thành viên) hay tự thay đổi cấu trúc là bộ điều khiển mờ có khả năng điều chỉnh lại các luật điều khiển. Giúp cho hệ thống bắt đầu làm việc mà chưa cần có đầy đủ các luật. Ở đây, xây dựng bộ AS mờ thích nghi trên nền tảng toán học MDP. Theo đó, một quyết định Markov là một quá trình điều khiển ngẫu nhiên thời gian rời rạc trong khi đó miền xác định của các biến tác động và tham số điều khiển (các tham số giám sát từ hệ thống) như trên là liên tục. Để giải quyết vấn đề này có thể dùng các phương pháp xấp xỉ hàm như phương pháp mạng nơron, hệ thống suy diễn mờ,… để rời rạc hóa không gian đầu vào. Phương pháp sử dụng là kết hợp hệ suy diễn mờ với phương pháp học tăng cường, cụ thể là thuật toán học Q với luật suy diễn nhiều đầu vào và một đầu ra (MISO) để xây dựng bộ điều khiển AS mờ thích nghi. Bằng cách sử dụng phương pháp suy diễn mờ Takagi-Sugeno để biểu diễn xấp xỉ hàm giá trị. Hệ thống suy diễn mờ Takagi-Sugeno sẽ gán cho mỗi miền vào một hàm rõ của các biến đầu vào và cung cấp cơ chế xấp xỉ phi tuyến với các phần luật là tuyến tính. Thuật toán học Q đảm nhận việc biểu diễn hàm giá trị theo bảng trong không gian trạng thái – hành động rời rạc và tính các giá trị phản hồi từ hệ thống và ước lượng hàm giá trị. Quá trình ước lượng được thực hiện đối với từng hành động tại từng trạng thái và lựa chọn hành động tốt nhất tương ứng với giá trị phần thưởng theo thời gian. Việc xấp xỉ hàm giá trị trạng thái-hành động tối ưu bằng thuật toán học Q sẽ hội tụ nếu mỗi quỹ đạo chuyển tiếp trạng thái của hệ thống bao gồm một số vô hạn trạng thái tương

ứng với một cặp trạng thái - hành động [23]. Với thời gian rời rạc, khái niệm vô hạn có nghĩa là ta không giới hạn số lần chuyển trạng thái của tác nhân.

4.3.2.1. Hệ suy diễn mờ

a. Luật mờ

Một dạng của luật **NẾU - THÌ** mờ, được đề xuất bởi Takagi và Sugeno [52], có các tập mờ tham gia vào chỉ trong phần giả thiết. Trong khía cạnh này, logic mờ bắt chước khả năng quyết định của tri thức con người để tổng hợp dữ liệu và tập trung vào thông tin quyết định có liên quan.

Một luật mờ là một biểu thức **NẾU - THÌ** được phát biểu ở dạng ngôn ngữ tự nhiên thể hiện sự phụ thuộc nhân quả giữa các biến. Trong phần này chỉ xét các luật **NẾU - THÌ** có dạng các tập mờ tham gia phần giả thiết của luật, phần kết luận của luật là giá trị rõ, tức là V là không gian tập rõ, B sẽ là một giá trị rõ.

Cho $U_i \neq \emptyset$, i = 1.. n là không gian nền của biến vào x_i, i = 1.. n. Gọi $F(U_i)$ là bộ các tập mờ trên U_i. $F(U_i) = \{\mu_{Ai}(u_i)$, - tập mờ trên $U_i\}$. Cho $V \neq \emptyset$ là không gian nền của biến ra y.

Như vậy luật mờ được phát biểu như sau:

Cho n biến vào x_1.. x_n, một biến ra y. Luật mờ R có dạng:

NẾU (x_1 là A_1) ∧ … ∧ (x_i là A_i) ∧ … ∧ (x_n là A_n) **THÌ** y là B

Ở đây $A_i \in F(U_i)$, i = 1.. n, $B \in V$

** Ví dụ:*

NẾU (tài nguyên khả dụng trung bình là High) ∧ (Phương sai của tài nguyên khả dụng là Low) ∧ (Thời gian đáp ứng trung bình là Nomal) **THÌ** Cần loại bỏ 1 VM.

b. Hệ mờ trên cơ sở các luật mờ

Hệ suy diễn mờ cũng được xem như hệ mờ dựa trên cơ sở các luật mờ, mô hình mờ, bộ nhớ tương tự mờ hoặc các điều khiển mờ khi sử dụng trong điều khiển.

Tư tưởng cơ bản của điều khiển dựa vào logic mờ là đưa các kinh nghiệm chuyên gia của những người vận hành giỏi hệ thống vào trong thiết kế các bộ điều khiển các quá trình trong đó quan hệ vào - ra được cho bởi một tập các luật điều khiển mờ (dạng luật **NẾU - THÌ**).

Hệ mờ trên cơ sở các luật mờ được phát biểu như sau:

Cho U_i, i = 1.. n là không gian nền của biến vào x_i, i = 1.. n, cho V là không gian nền của biến ra y. Hệ mờ MISO được xác định bởi bộ m luật mờ $\{R_1, \ldots, R_m\}$. Trong đó luật R_k có dạng:

NẾU (x_1 là A_{k1}) ∧ … ∧ (x_i là A_{ki}) ∧ … ∧ (x_n là A_{kn}) **THÌ** y là B_k

Ở đây: $A_{ki} \in F(U_i)$, i = 1.. n, k = 1..m, $B_k \in V$.

c. Kiến trúc cơ bản của hệ suy diễn mờ

Hệ suy diễn mờ có cấu trúc cơ bản bao gồm 5 khối chức năng (xem Hình 4.2):

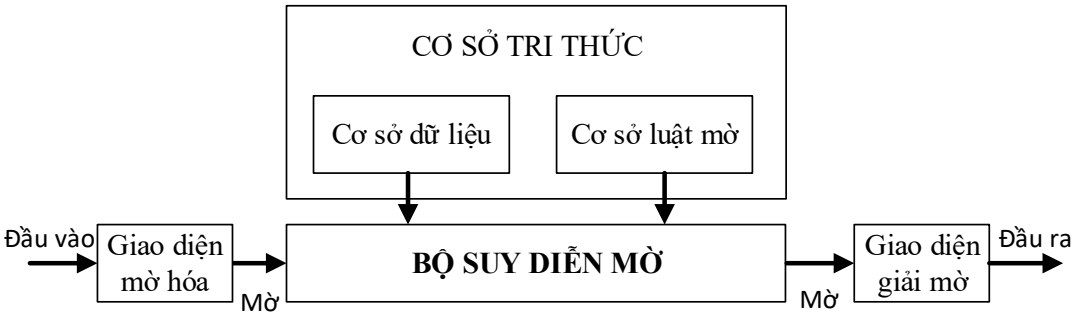
- *Cơ sở luật mờ:* bao gồm một số các luật mờ **NẾU - THÌ**.

- *Cơ sở dữ liệu:* định nghĩa các hàm thành viên của các tập mờ được sử dụng trong các luật mờ;

- *Bộ suy diễn mờ*: nhiệm vụ là kết hợp các luật trong cơ sở luật mờ, áp dụng vào tập mờ đầu vào theo các phương pháp suy diễn mờ để xác định tập mờ đầu ra;

- *Giao diện mờ hóa:* chuyển đổi các lớp đầu vào vào các biên độ phù hợp với các giá trị ngôn ngữ;

- *Giao diện giải mờ*: chuyển đổi các giá trị kết quả mờ của hệ suy diễn ra các lớp đầu ra.



Hình 4.2: Cấu trúc cơ bản của hệ suy diễn mờ

Thông thường, cơ sở luật mờ và cơ sở dữ liệu là suy diễn liên kết như là một cơ sở tri thức. Với hệ mờ trên cơ sở các luật mờ, các bước lập luận mờ (hoạt động suy diễn tương ứng theo các luật mờ **NẾU - THÌ**) được thực hiện bởi các hệ suy diễn mờ.

4.3.2.2. Quá trình mờ hóa

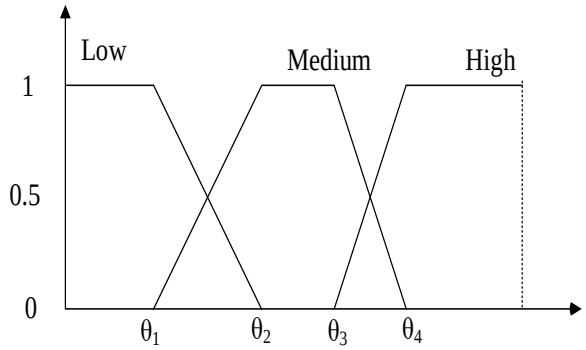
Mờ hoá các biến vào: Vì nhiều luật cho dưới dạng dùng các biến ngôn ngữ với các từ thông thường. Như vậy với những giá trị (rõ) quan sát được, đo được cụ thể, để có thể tham gia vào quá trình suy diễn thì cần thiết phải mờ hoá.

Có thể định nghĩa, mờ hoá là một ánh xạ từ không gian các giá trị quan sát được (rõ) vào không gian của các từ (tập mờ) trên không gian nền của các biến ngôn ngữ.

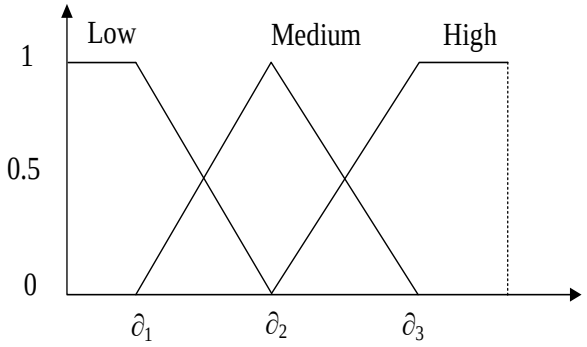
Suy luận mờ là quá trình ánh xạ một bộ điều khiển đầu vào cho một tập hợp các đầu ra điều khiển thông qua các luật hợp thành mờ. Tiềm năng của logic mờ là khả năng của ước tính phi tuyến tính bằng cách biểu diễn tri thức tương tự với suy luận của con người [50].

Bộ giám sát hệ thống sẽ quan sát mức độ tiêu thụ tài nguyên của hệ thống, sau đó tính được tài nguyên khả dụng trung bình của hệ thống và phương sai của tài nguyên khả dụng, đồng thời cho biết thời gian đáp ứng trung bình của hệ thống làm dữ liệu đầu vào cho bộ điều điều khiển AS và đầu ra là hành động điều chỉnh số lượng VM cần thêm vào hay loại bỏ.

Các giá trị đầu vào cho bộ điều khiển AS: tài nguyên khả dụng trung bình ($\overline{w}$), phương sai của tài nguyên khả dụng (var) và thời gian đáp ứng trung bình (rt) được mờ hóa theo các hàm thành viên sau đây:



Hình 4.3: Hàm thành viên mờ cho tài nguyên khả dụng trung bình



Hình 4.4: Hàm thành viên mờ cho phương sai của tài nguyên khả dụng

a. Tài nguyên khả dụng trung bình ($\overline{w}$)

Công thức mờ hóa với tài nguyên khả dụng trung bình $\overline{w}$ có giá trị Low:

$$\mu_{\text{low}}(\bar{w}) = \begin{cases} 1 \text{ nếu } \bar{w} < \theta_1, \\ \dfrac{\theta_2 - \bar{w}}{\theta_2 - \theta_1} \text{ nếu } \theta_1 \le \bar{w} \le \theta_2, \\ 0 \text{ nếu } \bar{w} > \theta_2. \end{cases} \quad (4.9)$$

Công thức mờ hóa với tài nguyên khả dụng trung bình $\bar{w}$ có giá trị Medium:

$$\mu_{\text{medium}}(\bar{w}) = \begin{cases} 0 \text{ nếu } \bar{w} < \theta_1 \text{ hoặc } \bar{w} > \theta_4, \\ \dfrac{\bar{w} - \theta_1}{\theta_2 - \theta_1} \text{ nếu } \theta_1 \le \bar{w} \le \theta_2, \\ 1 \text{ nếu } \bar{w} > \theta_2 \text{ và } \bar{w} < \theta_3, \\ \dfrac{\theta_4 - \bar{w}}{\theta_4 - \theta_3} \text{ nếu } \theta_3 \le \bar{w} \le \theta_4. \end{cases} \quad (4.10)$$

Công thức mờ hóa với tài nguyên khả dụng trung bình $\bar{w}$ có giá trị High:

$$\mu_{\text{high}}(\bar{w}) = \begin{cases} 1 \text{ nếu } \bar{w} > \theta_4, \\ \dfrac{\bar{w} - \theta_3}{\theta_4 - \theta_3} \text{ nếu } \theta_3 \le \bar{w} \le \theta_4, \\ 0 \text{ nếu } \bar{w} < \theta_3. \end{cases} \quad (4.11)$$

b. Phương sai của tài nguyên khả dụng (var**)**

Công thức mờ hóa với phương sai của tài nguyên khả dụng var có giá trị Low:

$$\mu_{\text{low}}(\text{var}) = \begin{cases} 1 \text{ nếu var} < \partial_1, \\ \dfrac{\partial_2 - \text{var}}{\partial_2 - \partial_1} \text{ nếu } \varphi_1 \le \text{var} \le \partial_2, \\ 0 \text{ nếu var} > \partial_2. \end{cases} \quad (4.12)$$

Công thức mờ hóa với phương sai của tài nguyên khả dụng var có giá trị Medium:

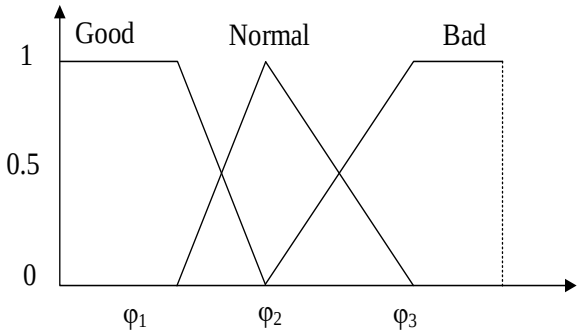
$$\mu_{\text{medium}}(\text{var}) = \begin{cases} 0 \text{ nếu var} < \partial_1 \text{ hoặc var} > \partial_3, \\ \dfrac{\text{var} - \partial_1}{\partial_2 - \partial_1} \text{ nếu } \partial_1 \le \text{var} \le \partial_2, \\ \dfrac{\partial_3 - \text{var}}{\partial_3 - \partial_2} \text{ nếu } \partial_2 \le \text{var} \le \partial_3. \end{cases} \quad (4.13)$$

Công thức mờ hóa với phương sai của tài nguyên khả dụng var có giá trị High:

$$\mu_{\text{high}}(\text{var}) = \begin{cases} 1 \text{ nếu var} > \partial_3, \\ \dfrac{\text{var} - \partial_2}{\partial_3 - \partial_2} \text{ nếu } \partial_2 \le \text{var} \le \partial_3, \\ 0 \text{ nếu var} < \partial_2. \end{cases} \quad (4.14)$$

c. Thời gian đáp ứng (rt)

Thời gian đáp ứng (rt) là chỉ số dùng để đánh giá khả năng phản hồi của hệ thống với yêu cầu dịch vụ. Thời gian đáp ứng được tính bằng khoảng thời gian từ lúc nhận yêu cầu cho tới khi hệ thống phân tích, xử lý yêu cầu, kết xuất xong kết quả và trả về phản hồi cho yêu cầu đó. Giả sử ba tập mờ được sử dụng để mô tả các giá trị của thời gian đáp ứng: Bad, Normal, Good. Những giá trị của tham số đầu vào rt sẽ quyết định giá trị của biến mờ, và phụ thuộc vào tình trạng hệ thống mà giá trị của biến mờ sẽ thay đổi từ 0 đến 1.



Hình 4.5: Hàm thành viên mờ cho thời gian đáp ứng trung bình

Công thức mờ hóa với thời gian đáp ứng trung bình (rt) có giá trị Good:

$$\mu_{\text{good}}(\text{rt}) = \begin{cases} 1 \text{ nếu rt} < \varphi_1, \\ \dfrac{\varphi_2 - \text{rt}}{\varphi_2 - \varphi_1} \text{ nếu } \varphi_1 \le \text{rt} \le \varphi_2, \\ 0 \text{ nếu rt} > \varphi_2. \end{cases} \quad (4.15)$$

Công thức mờ hóa với thời gian đáp ứng trung bình (rt) có giá trị Normal:

$$\mu_{\text{normal}}(\text{rt}) = \begin{cases} 0 \text{ nếu rt} < \varphi_1 \text{ hoặc rt} > \varphi_3, \\ \dfrac{\text{rt} - \varphi_1}{\varphi_2 - \varphi_1} \text{ nếu } \varphi_1 \le \text{rt} \le \varphi_2, \\ \dfrac{\varphi_3 - \text{rt}}{\varphi_3 - \varphi_2} \text{ nếu } \varphi_2 \le \text{rt} \le \varphi_3. \end{cases} \quad (4.16)$$

Công thức mờ hóa với thời gian đáp ứng trung bình (rt) có giá trị Bad:

$$\mu_{bad}(rt) = \begin{cases} 1 \text{ nếu } rt > \varphi_3, \\ \dfrac{rt - \varphi_2}{\varphi_3 - \varphi_2} \text{ nếu } \varphi_2 \le rt \le \varphi_3, \\ 0 \text{ nếu } rt < \varphi_2. \end{cases} \quad (4.17)$$

4.3.2.3. Quá trình lập luận xấp xỉ

Kết hợp hệ thống suy diễn mờ Takagi-Sugeno và thuật toán học Q, tập luật suy diễn mờ và cơ chế xấp xỉ đầu ra như sau:

Luật^1: **NẾU** x_1 là s_1 $\wedge$ ... $\wedge$ x_n là s_n **THÌ**

$y_1 = a_1^1$ với q[1,1] là mức độ phù hợp của lựa chọn đầu ra hoặc

$y_1 = a_2^1$ với q[1,2] là mức độ phù hợp của lựa chọn đầu ra hoặc

.....

$y_1 = a_m^1$ với q[1, m] là mức độ phù hợp của lựa chọn đầu ra,

.....

Luật^i: **NẾU** x_1 là s_1 $\wedge$ ... $\wedge$ x_n là s_n **THÌ**

$y_i = a_1^i$ với q[i, 1] là mức độ phù hợp của lựa chọn đầu ra hoặc

$y_i = a_2^i$ với q[i, 2] là mức độ phù hợp của lựa chọn đầu ra hoặc

.....

$y_i = a_m^i$ với q[i, m] là mức độ phù hợp của lựa chọn đầu ra.

trong đó,

- $x_j, j = 1, .., n$ là các biến đầu vào của hệ thống suy diễn mờ được định nghĩa trên không gian S,

- a_m^i là hành động được tác nhân lựa chọn cho tập luật thứ i được định nghĩa trên không gian hành động A được rời rạc hóa thành m đoạn,

- q[i, m] là độ thích hợp khi lựa chọn giá trị đầu ra (hành động cục bộ) a_m^i khi luật Rule^i được lựa chọn.

Thành phần học sử dụng kỹ thuật học Q. Tại mỗi vòng lặp, bộ điều khiển sẽ đưa ra một hành động a dựa vào trạng thái s được ký hiệu $Q(s, a) = q[i, m]$. Giá trị phản hồi này được biểu diễn dưới dạng hàm số đối với giá trị phần thưởng. Giả thiết rằng hàm phản hồi có tính chất cộng tính.

Thành phần điều khiển sẽ chọn hành động mà có giá trị phần thưởng tốt trong tương lai.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (4.18)$$

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

$$r_t = U(t) - U(t-1), \quad (4.19)$$

với $U(t)$ được tính theo công thức (4.7).

4.3.2.4. Quá trình giải mờ

Giải mờ kết quả tìm được cho ta một số rõ: Đây là khâu thực hiện quá trình xác định một giá trị rõ có thể chấp nhận được làm đầu ra từ hàm thành viên của giá trị mờ đầu ra.

Chính sách $\pi(x, a)$ là xác suất chọn hành động a từ trạng thái x, hàm hành động – giá trị Q(x,a) được tính theo công thức sau:

$$Q^{\pi} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right\} \quad (4.20)$$

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.

Theo cơ chế suy diễn của hệ thống Takagi-Sugeno [54], tác động đầu ra $a(x)$ đối với véctơ đầu vào $x = (x_1, x_2, \dots, x_N)$, được xấp xỉ bằng công thức sau:

$$a(x) = \frac{\sum_i \alpha_i(x) a_{j*}^i}{\sum_i \alpha_i(x)}, \quad (4.21)$$

trong đó,

$\alpha_i(x) = \min(\mu_1^i(x_1), \mu_2^i(x_2), \dots, \mu_N^i(x_N))$ là giá trị chân lý của mệnh đề $y = y^i$

a_{j*}^i tác động đầu ra được lựa chọn của luật $Rule^i$,

Quá trình học trong hệ thống là xấp xỉ các độ thích hợp với lựa chọn $q[i, j]$ của mỗi giá trị đầu ra a_j^i với từng véctơ đầu vào x. Do vậy, giá trị tương ứng của hàm giá trị trạng thái - hành động được xấp xỉ bằng:

$$Q(x, a) = \frac{\sum_i \alpha_i(x) q[i, j^*]}{\sum_i \alpha_i(x)}. \quad (4.22)$$

Hàm giá trị tại trạng thái x của hệ thống được tính như sau:

$$V(x) = \max_a Q(x, a) = \frac{\sum_i \alpha_i(x) \max_j q[i, j]}{\sum_i \alpha_i(x)}. \quad (4.23)$$

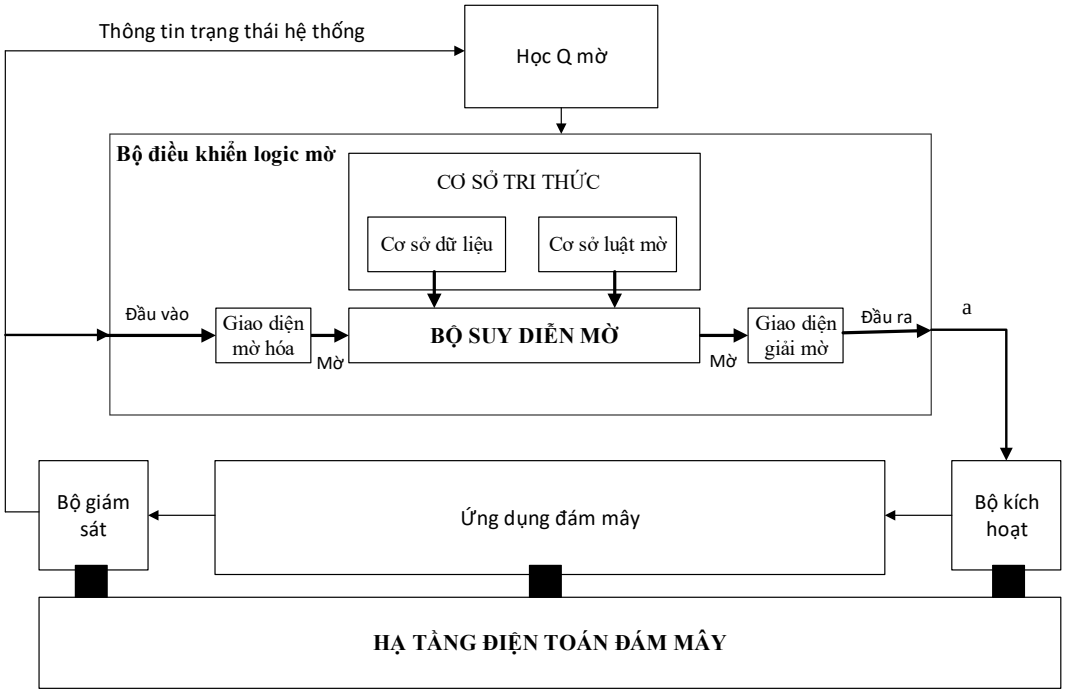
Trong quá trình học, hàm giá trị trạng thái-hành động Q và các giá trị của độ thích hợp của các luật $q[i, j^*]$ được hiệu chỉnh theo hệ thức lặp sau:

$$Q_t(x_t, a) \leftarrow Q_t(x_t, a) + \eta \Delta Q_t, \quad (4.24)$$

trong đó, η là tốc độ học và $\Delta Q_t = r_{t+1} + \gamma \max_{a'} Q(x_{t+1}, a_t) - Q(x_t, a_t)$.

4.3.4. Giải pháp tự động điều chỉnh

Sơ đồ bộ AS đề xuất được thể hiện trong hình 4.6. Bộ AS này sẽ hỗ trợ công việc vận hành và quản lý chất lượng của kiến trúc đám mây bằng cách tự động điều chỉnh tài nguyên tính toán theo nhu cầu của khách hàng, duy trì hiệu năng phục vụ của đám mây và giảm thiểu chi phí tài nguyên. Bộ AS này có khả năng tự học và sửa đổi các luật mờ trong thời gian chạy, quan sát và giám sát liên tục việc tiêu thụ tài nguyên của các ứng dụng và thực hiện cung cấp theo kế hoạch. Việc tính toán tài nguyên để cung cấp cho ứng dụng bằng cách giám sát việc sử dụng tài nguyên của các VM trong cụm đang hoạt động xử lý ứng dụng hiện hành, đó là mức tiêu thụ tài nguyên CPU, bộ nhớ và ổ đĩa trên mỗi VM, từ đó tính được các tham số tài nguyên khả dụng trung bình ($\overline{w}$) và phương sai (var), và thời gian đáp ứng trung bình của ứng dụng (rt) và số lượng VM đang hoạt động.



Hình 4.6: Bộ tự động điều chỉnh đề xuất

Bảng 4.1: Thuật toán tự động điều chỉnh (AS AFQL)

1	*/*AFQL*
2	** Đầu vào:*
3	** η*
4	** γ*
5	** Đầu ra:*
6	**Hành động a */*
7	*//khởi tạo giá trị q-values:*
8	for (application app in applications)
9	$q[i,k] = 0;$
10	*//sinh luật điều khiển mờ*
11	generateFuzzyRules()
12	while(q chưa hội tụ)
13	*//xấp xĩ giá trị Q cho từng ứng dụng ở trạng thái hiện tại x_t và hành động a*
14	for (application app in applications)
15	$Q(s(t),a) = \frac{\sum_i \alpha_i(x_t) q[i,j^*]}{\sum_i \alpha_i(x_t)}$
16	*//triển khai hành động a*
17	for (application app in applications)
18	applyAction(app, a)
19	*//tính các giá trị tại thời điểm t+1*
20	$\Delta Q_t = r_{t+1} + \gamma \max_{a'} Q(x_{t+1}, a_t) - Q(x_t, a_t)$
21	$q[i,j] = q[i,j] + \eta \Delta Q_t \frac{\alpha_i(x_t)}{\sum_i \alpha_i(x_t)}$

Trong đó, tài nguyên khả dụng trung bình ($\overline{w}$) và phương sai (var) được tính theo công thức (4.5) và (4.6). Thời gian đáp ứng trung bình có thể được lấy ở Chương 3.

Mục tiêu của hệ thống là chi phí, SLA, thời gian đáp ứng,… được tham chiếu trong hàm phần thưởng (công thức (4.19)).

Các số đo hiệu năng của hệ thống thể hiện cho trạng thái hiện tại: $\overline{w}, \mathrm{var}, \mathrm{rt}$, số lượng VM,… được bộ giám sát cung cấp cho cả bộ điều khiển mờ và học Q mờ. Bộ kích hoạt hành động thực hiện các lệnh từ bộ điều khiển theo mỗi chu kỳ điều khiển ở nền tảng đám mây.

Bộ điều khiển đám mây là bộ điều khiển mờ thực hiện quan sát dữ liệu và thực hiện các hành động điều chỉnh. Thành phần học Q mờ liên tục được cập nhật cơ sở tri thức từ việc học các luật thích hợp được mô tả trong phần 4.3.2.

Hiệu quả của bộ điều khiển AS tài nguyên sẽ phụ thuộc và tốc độ học η, hệ số suy giảm γ, sự phân hoạch mờ không gian các tham số đầu vào,…

Đối với giải pháp AS ở đây được thiết kế trên cấu trúc hệ suy diễn mờ cố định với số lượng các luật suy diễn cố định (ở đây là 27 luật), phần kết luận của luật là các hành động cố định nằm trong khoảng $[-2, 2]$ và mỗi không gian đầu vào được phân hoạch thành một số cố định các tập mờ (tài nguyên khả dụng trung bình ($\overline{w}$) được phân hoạch thành 3 phân đoạn mờ, phương sai (var) được phân hoạch thành 3 phân đoạn mờ và thời gian đáp ứng trung bình (rt) được phân hoạch thành 3 phân đoạn mờ).

Hoạt động của thuật toán AFQL (Bảng 4.1). Bước 1: khởi tạo các giá trị q cho tất cả các trạng thái ban đầu bằng 0. Bước 2: sinh luật điều khiển mờ: liên quan đến việc chọn lựa hành động. Để học từ hệ thống, bộ AS cần khám phá những tri thức đạt được, sử dụng chiến lược thăm dò/khai thác ε-greedy [97]. Như vậy, hành động có phần thưởng tốt nhất sẽ được chọn với xác suất (1-ε) hoặc một hành động ngẫu nhiên sẽ được chọn với xác suất thấp ε để thăm dò các hành động chưa được chọn. Bước 3: tính toán hành động điều khiển được suy ra từ bộ AS. Đầu ra được tính theo công thức (4.21). Bước 4: Xấp xỉ hàm Q từ các giá trị hiện tại và mức độ sử dụng các luật, được tính theo công thức (4.22). Hàm Q(s(t),a) cho biết mức độ mong muốn đạt được

trạng thái *s* tại thời điểm t khi thực hiện hành động a bằng cách cho phép thực hiện hành động *a* này nhiều lần và quan sát giá trị trả về (Bước 5). Bước 6: tính toán các giá trị tại thời điểm *t+1* và cập nhật giá trị *q* sau mỗi lần lặp theo công thức:

$$q[i,j] = q[i,j] + \eta \Delta Q_t \frac{\alpha_i(x_t)}{\sum_i \alpha_i(x_t)}, \quad (4.25)$$

trong đó η là tốc độ học có giá trị từ 0 đến 1. Giá trị η nhỏ có nghĩa là gần các giá trị trước một chút với mọi cập nhật và giá trị η lớn hơn sẽ ảnh hưởng đến phần thưởng gần nhất.

Trong trường hợp bộ AS đề xuất, không gian trạng thái là hữu hạn (tức là có 27 trạng thái là kết hợp của 3 x 3 x 3 hàm thành viên cho các biến tài nguyên khả dụng trung bình ($\overline{w}$), phương sai của tài nguyên khả dụng (var) và thời gian đáp ứng trung bình (rt)) và bộ AS phải lựa chọn một trong năm hành động có thể {-2, -1, 0, 1, 2}. Tuy nhiên phương pháp thiết kế này là tổng quát có thể áp dụng cho những không gian trạng thái và hành động khác.

4.4. THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.4.1. Thiết lập hệ thống

Trong phần này, chúng tôi tiến hành thực nghiệm để đánh giá hiệu năng các số đo được lập trình bằng ngôn ngữ Java trên máy tính Laptop có cấu hình Intel® Core™ i7-7500U CPU @ 2.70GHz, 8GB Ram, 1 TB HDD và chạy hệ điều hành Window 10.

Thực nghiệm được tiến hành trên các VM có cấu hình trong Bảng 4.2.

Bảng 4.2: Cấu hình các VM

	Số CPU (c)	**Tốc độ xử lý của 1 CPU (s)**	**RAM**
VM1	4	1000	2GB
VM2	2	1000	1GB
VM3	4	1000	2GB
VM4	4	1000	2GB
VM5	2	1000	1GB
VM6	2	1000	1GB
VM7	4	1000	2GB
VM8	6	1000	2GB
VM9	4	1000	1GB
VM10	6	1000	2GB

Theo công trình [53], Luận án đã sử dụng phương pháp chuẩn hóa dữ liệu đầu vào cho tài nguyên khả dụng trung bình ($\overline{w}$), phương sai (var) và thời gian đáp ứng trung bình (rt) vào khoảng $[0,1]$ theo công thức sau:

$$T_{normalize} = a + \frac{\overline{T} - T_{min}}{T_{max} - T_{min}} * (b - a),$$

trong đó, T có thể là tài nguyên khả dụng trung bình ($\overline{w}$), phương sai (var) và thời gian đáp ứng trung bình (rt). a, b là hằng số: a=0.1 và b=0.99.

Với dữ liệu về tài nguyên khả dụng trung bình ($\overline{w}$), phương sai (var) và thời gian đáp ứng trung bình (rt) được cho trong các Bảng 4.3, Bảng 4.4 và Bảng 4.5.

Bảng 4.3: Các phân hoạch của giá trị tài nguyên khả dụng trung bình

	Biến ngôn ngữ	**Miền xác định**	
		Giá trị bắt đầu	**Giá trị kết thúc**
Tài nguyên khả dụng trung bình ($\overline{w}$)	Low	0.00	0.40
	Medium	0.20	0.80
	High	0.60	1.00

Bảng 4.4: Các phân hoạch của giá trị phương sai

	Biến ngôn ngữ	**Miền xác định**	
		Giá trị bắt đầu	**Giá trị kết thúc**
phương sai (var)	Low	0.00	0.50
	Medium	0.25	0.75
	High	0.50	1.00

Bảng 4.5: Các phân hoạch của thời gian đáp ứng

	Biến ngôn ngữ	**Miền xác định**	
		Giá trị bắt đầu	**Giá trị kết thúc**
Thời gian đáp ứng (rt)	Good	0.00	0.50
	Normal	0.25	0.75
	Bad	0.50	1.00

Dựa vào phương pháp sử dụng trong công trình [50], Luận án đã xây dựng tập luật cho trong Bảng 4.6, các giá trị hành động nằm trong khoảng $[-2, 2]$, tương ứng với việc loại bỏ 2 VM, loại bỏ 1 VM, không làm gì cả, thêm 1 VM và thêm 2 VM. Với việc loại bỏ các VM, sẽ chọn VM nào có mức khả dụng trung bình nhiều nhất.

Việc thêm VM, sẽ thực hiện thêm VM có cấu hình thấp nhất trong số VM đang hoạt động để đảm bảo chi phí thuê là nhỏ nhất.

Bảng 4.6: Tập luật được xây dựng dựa theo tri thức chuyên gia

Luật	**Tiền đề của luật**			**Hành động**
	Giá trị tài nguyên khả dụng trung bình ($\overline{w}$)	**Thời gian đáp ứng (rt)**	**Phương sai của tài nguyên khả dụng (var)**	$[-2,2]$
1	Low	Good	Low	0
2	Low	Good	Medium	0
3	Low	Good	High	1
4	Low	Normal	Low	0
5	Low	Normal	Medium	0
6	Low	Normal	High	1
7	Low	Bad	Low	2
8	Low	Bad	Medium	2
9	Low	Bad	High	2
10	Medium	Good	Low	-1
11	Medium	Good	Medium	-1
12	Medium	Good	High	-2
13	Medium	Normal	Low	0
14	Medium	Normal	Medium	0
15	Medium	Normal	High	0
16	Medium	Bad	Low	1
17	Medium	Bad	Medium	1
18	Medium	Bad	High	2
19	High	Good	Low	-2
20	High	Good	Medium	-2
21	High	Good	High	-2
22	High	Normal	Low	-1
23	High	Normal	Medium	-1
24	High	Normal	High	-2
25	High	Bad	Low	-2
26	High	Bad	Medium	-2
27	High	Bad	High	-2

4.4.2. Kết quả thực nghiệm

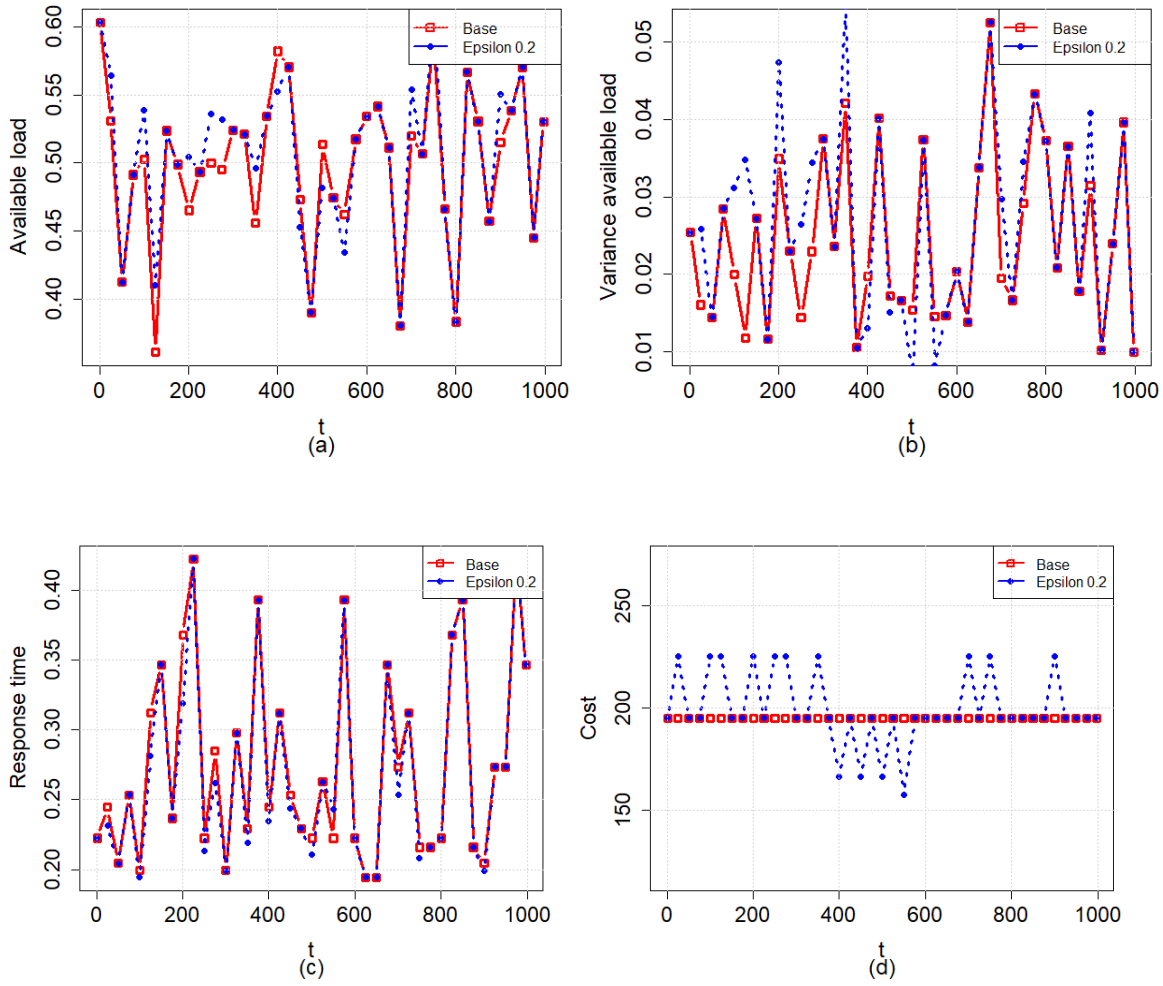
Trong phần này sẽ tiến hành xem xét sự hội tụ của các giá trị q[i, j] khi tiến hành điều chỉnh các tham số thăm dò/khai thác (ε), hệ số học (η) và hệ số suy giảm trong hàm phần thưởng (γ). Sau đó sẽ đánh giá mô hình đề xuất với hệ thống AS chỉ sử dụng logic mờ (sử dụng tập luật trong Bảng 4.6). Các thực nghiệm được thực hiện nhiều lần trên bộ dữ liệu sinh ngẫu nhiên. Thiết lập tất cả các giá trị trong ma trận Q bằng 0, tức là không có tri thức ban đầu.

4.4.2.1. Đánh giá việc lựa chọn tham số ε.

Khi bộ AS phải thực hiện một hành động trong mỗi vòng lặp kiểm soát, cần phải cố gắng chọn những hành động đã được thực hiện trước đó để tạo ra phần thưởng tốt hơn (phần thưởng tích lũy được tính theo công thức (4.18)). Tuy nhiên, nếu bộ AS chỉ thực hiện hành động dựa trên các hành động đã thực hiện nó sẽ đưa đến một tri thức tối ưu cục bộ. Do đó, tồn tại một sự cân bằng giữa việc lựa chọn các hành động đã thực hiện trước đó (gọi là khai thác) và các hành động mới có thể dẫn đến phần thưởng trong tương lai tốt hơn (gọi là thăm dò). Sự cân bằng giữa thăm dò và khai thác là rất quan trọng đối với việc học tăng cường. Như vậy, ở đây sử dụng phương pháp chọn hành động là ε-greedy, tức là bộ AS sẽ chọn hành động đã có có giá trị q tốt nhất (chiến lược khai thác) với xác suất là 1-ε. Ngược lại, sẽ chọn một hành động ngẫu nhiên (hành động mới – chiến lược thăm dò) trong tập các hành động, ở đây là {-2, -1, 0, 1, 2} với xác suất là ε. Các giá trị ε có thể tăng lên hoặc giảm xuống để ưu tiên chiến lược thăm dò so với chiến lược khai thác.

Bộ AS đề xuất bắt đầu với việc cấp phát tài nguyên mà không có tri thức. Sau khi thăm dò đủ, kết quả của luật có thể được xác định bằng cách chọn những hành động tương ứng với giá trị q cao nhất trong mỗi hàng của ma trận Q.

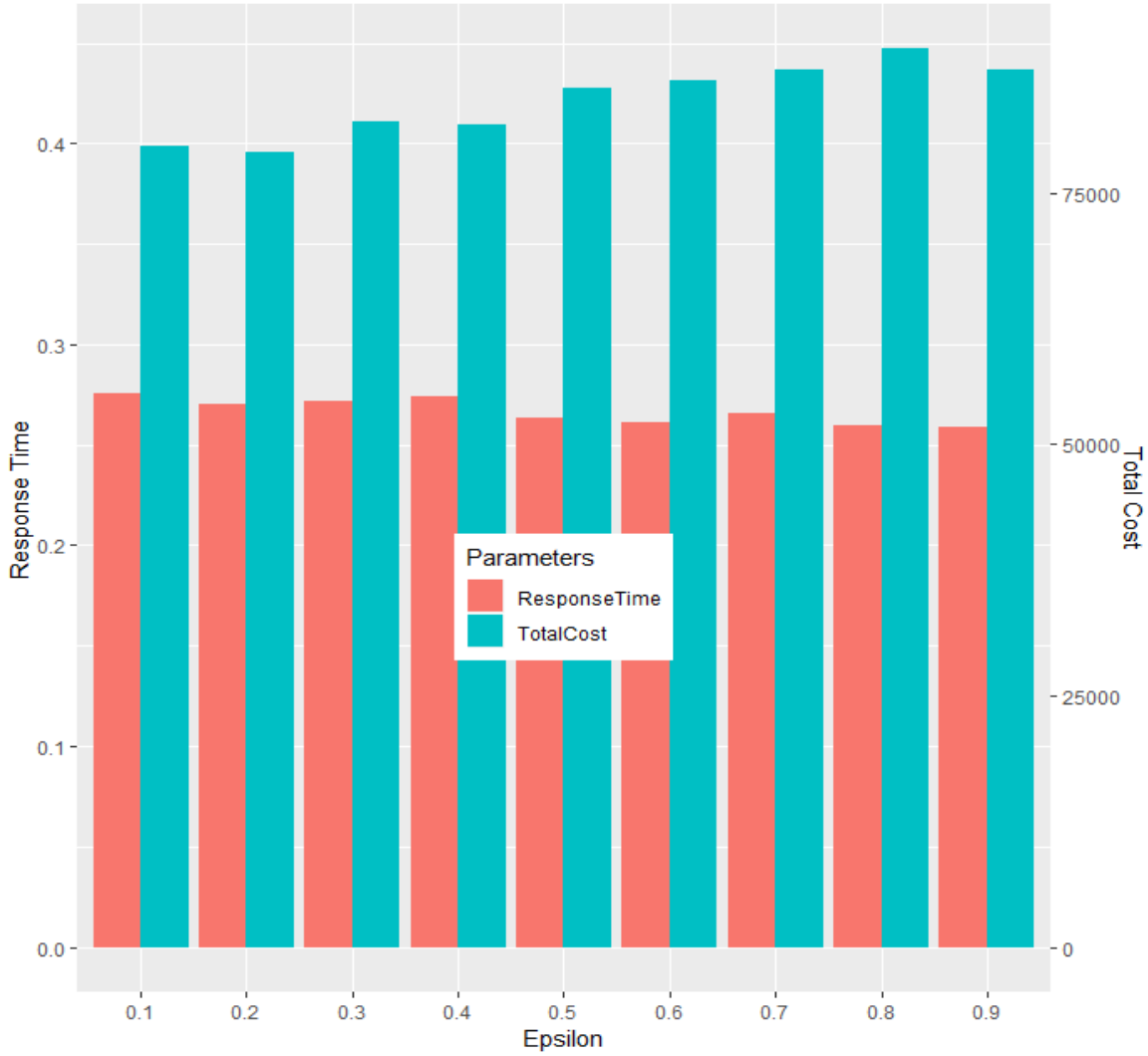
Để đánh giá chiến lược lựa chọn hành động của bộ AS, ở đây đã xem xét các chiến lược khai thác/thăm dò khác nhau: ε = 0.1; ε = 0.2; ε = 0.3; ε = 0.4; ε=0.5; ε = 0.6; ε = 0.7; ε = 0.8 và ε = 0.9. Trong các thực nghiệm thiết lập tham số tốc độ học $\eta = 0.5$ và hệ số suy giảm $\gamma = 0.5$. Ứng với mỗi giá trị của ε, thực nghiệm sẽ đánh giá kết quả sau khi áp dụng bộ AS so với cơ sở dữ liệu ban đầu. Chẳng hạn như, Hình 4.7 là kết quả thực nghiệm so sánh tài nguyên khả dụng trung bình (a), phương sai của tài nguyên khả dụng (b), thời gian đáp ứng trung bình (c) và chi phí (d) sau khi áp dụng bộ AS với tham số (ε=0.2, $\gamma = 0.5$, $\eta = 0.5$) với giá trị ban đầu. Đường màu đỏ là biểu diễn cơ sở dữ liệu đầu vào, đường nét đứt màu xanh thể hiện kết quả sau khi áp dụng bộ AS.



Hình 4.7: Kết quả thực nghiệm áp dụng bộ AS với tham số ε=0.2, $\gamma = 0.5, \eta = 0.5$.

Tương tự với các giá trị ε khác, cuối cùng tổng hợp được kết quả như hình 4.8, thể hiện các thông tin thời gian đáp ứng trung bình và tổng chi phí theo từng giá trị của ε. Hình 4.8 cho thấy, giá trị thời gian đáp ứng càng cao thì chi phí giảm, và ngược lại, thời gian đáp ứng thấp thì chi phí lại cao. Ví dụ, với ε = 0.8 cho thời gian đáp ứng trung bình gần như lớn nhất, nhưng chi phí lại ít nhất. Với với giá trị của ε càng lớn, việc lựa chọn hành động của bộ AS thiên về thăm dò các hành động mới (hành động ngẫu nhiên) hơn là sử dụng các hành động (khai thác) đã có và sẽ ít phụ thuộc vào các giá trị q hơn, dẫn đến mất cân bằng trong việc khai thác/thăm dò các hành động. Với giá trị ε nhỏ có thể tạo ra sự cân bằng giữa khai thác và thăm dò các hành động có thể làm tăng tốc độ học và hạn chế tác động tiêu cực của việc thăm dò ngẫu nhiên.

Do vậy, ở đây có thể chọn ε = 0.1 cho chi phí gần nhỏ nhất và thời gian đáp ứng trung bình cao nhất, hoặc có thể chọn ε = 0.2 với chi phí nhỏ nhất và thời gian đáp ứng trung bình gần cao nhất.



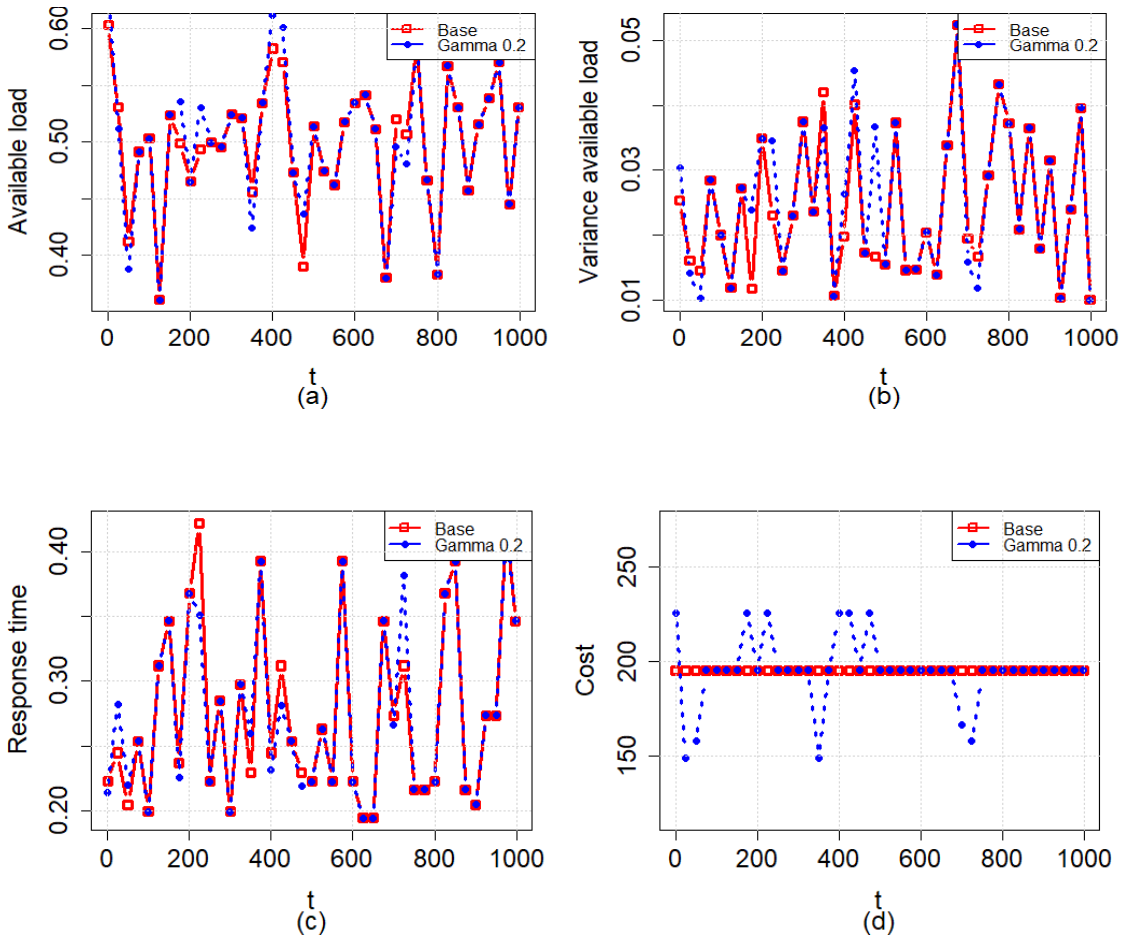
Hình 4.8: Xác định tham số thăm dò/khai thác (ε)

4.4.2.2. Đánh giá việc lựa chọn hệ số suy giảm (γ).

Trong phần này lấy hệ số thăm dò/khai thác hành động ε=0.2 và tốc độ học η = 0.5. Điều chỉnh hệ số suy giảm trong hàm phần thưởng để xem xét tính hiệu quả của mô hình đề xuất. Hệ số suy giảm γ xác định tầm quan trọng của các phần thưởng trong tương lai, cho phép các phần thưởng từ các khoảng thời gian tiếp theo được tính vào tổng lợi nhuận, làm tối đa lợi nhuận (phần thưởng) tích lũy (công thức (4.18)). Ở

đây đã xem xét các giá trị: γ = 0.1; γ = 0.2; γ = 0.3; γ = 0.4; γ=0.5; γ = 0.6; γ = 0.7; γ = 0.8 và γ = 0.9.

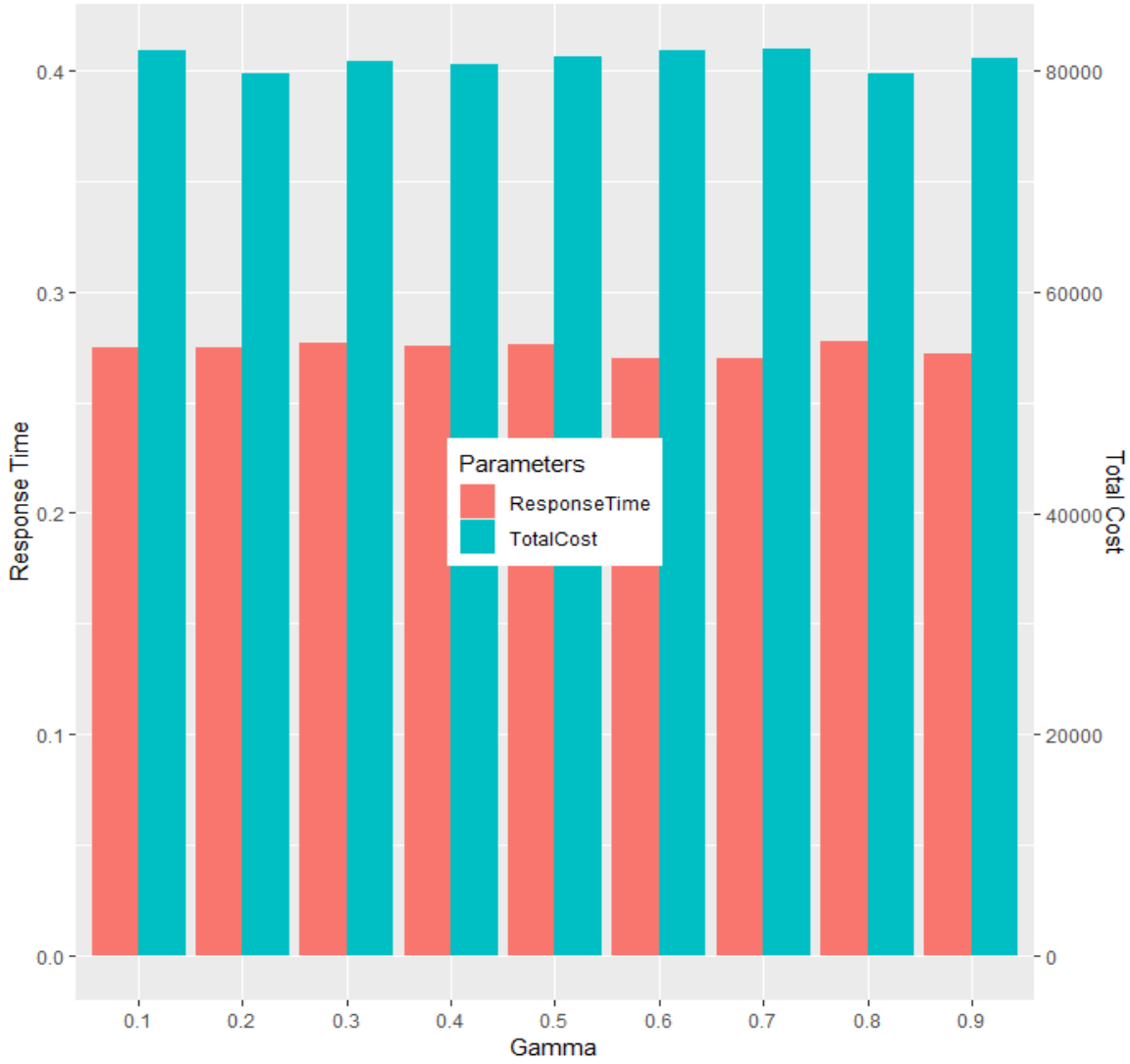
Tương tự như trên, với mỗi giá trị của γ, cũng tiến hành so sánh kết quả sau khi áp dụng bộ AS với các tham số thiết lập so với dữ liệu đầu vào. Chẳng hạn với γ = 0.2, Hình 4.9 là kết quả thực nghiệm so sánh các giá tài nguyên khả dụng trung bình (a), phương sai của tài nguyên khả dụng (b), thời gian đáp ứng trung bình (c) và chi phí (d) sau khi áp dụng bộ AS với tham số ($\varepsilon = 0.2, \gamma = 0.2, \eta = 0.5$) với giá trị ban đầu. Trong đó, đương màu đỏ thể hiện dữ liệu đầu vào cho bộ AS, đường nét đứt màu xanh thể hiện kết quả sau khi áp dụng bộ AS với các tham số $\varepsilon = 0.2, \gamma = 0.2, \eta = 0.5$.



Hình 4.9: Kết quả thực nghiệm áp dụng bộ AS với bộ tham số $\varepsilon = 0.2, \gamma = 0.2, \eta = 0.5$

Sau khi thực nghiệm cho các giá trị của γ, kết quả tổng hợp thời gian đáp ứng trung bình và chi phí theo các giá trị của γ được thể hiện trong Hình 4.10. Theo Hình

4.10, việc thay đổi giá trị của thời gian đáp ứng trung bình và chi phí không nhiều, cũng có xu hướng thời gian đáp ứng thấp thì chi phí cao. Theo như kết quả hình 4.10 có thể chọn tham số γ = 0.8 cho thời gian đáp ứng trung bình cao nhất và chi phí nhỏ nhất.



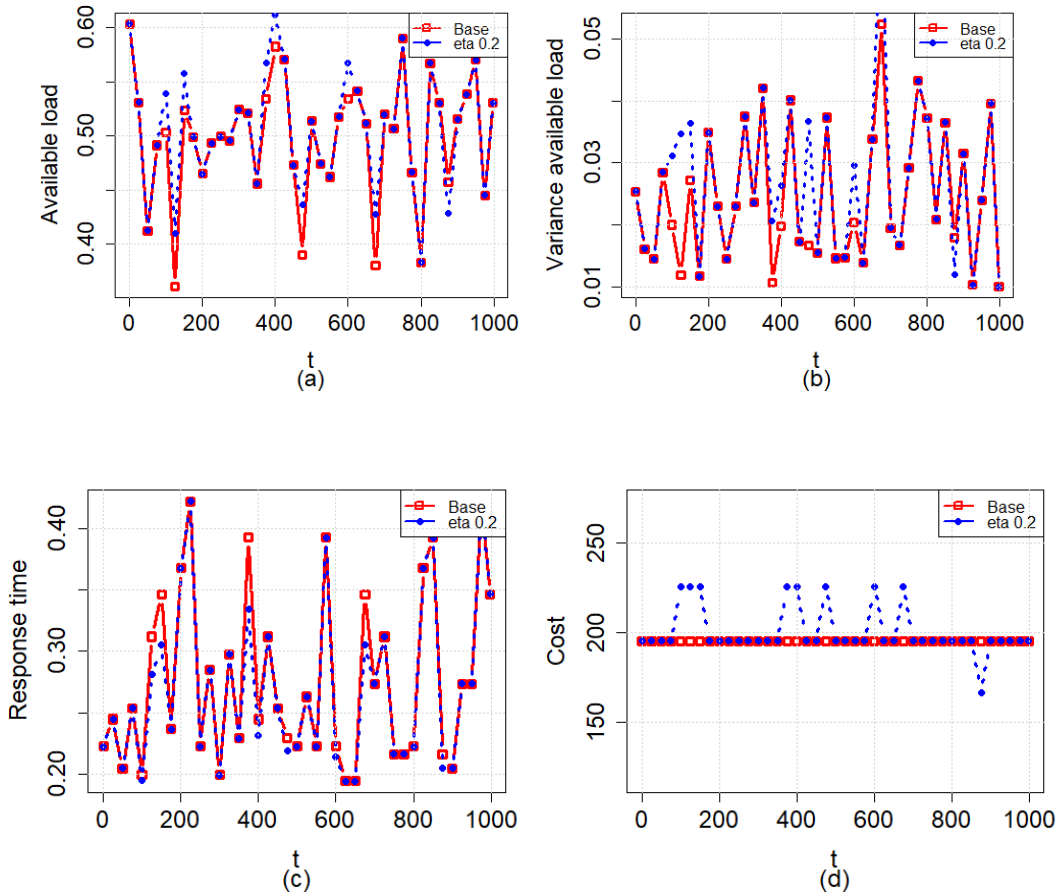
Hình 4.10: Xác định hệ số suy giảm (γ)

4.4.2.3. Đánh giá việc lựa chọn tham số tốc độ học η.

Trong phần này, thiết lập với các giá trị ε=0.2, γ=0.8 và xem xét các giá trị: η = 0.1; η = 0.2; η = 0.3; η = 0.4; η=0.5; η = 0.6; η = 0.7; η = 0.8 và η = 0.9.

Tương ứng với mỗi giá trị của η, sẽ được đánh giá so sánh kết quả thực hiện bộ AS với dữ liệu ban đầu. Chẳng hạn với η = 0.2, Hình 4.11 là kết quả thực nghiệm so sánh các giá tài nguyên khả dụng trung bình (a), phương sai của tài nguyên khả dụng

(b), thời gian đáp ứng trung bình (c) và chi phí (d) sau khi áp dụng bộ AS với tham số ($\varepsilon = 0.2, \gamma = 0.8$ và $\eta = 0.2$) với giá trị ban đầu, đường màu đỏ biểu diễn dữ liệu đầu vào cho bộ AS, đường nét đứt màu xanh thể hiện kết quả sau khi áp dụng bộ AS với các tham số $\varepsilon = 0.2, \gamma = 0.8$ và $\eta = 0.2$.



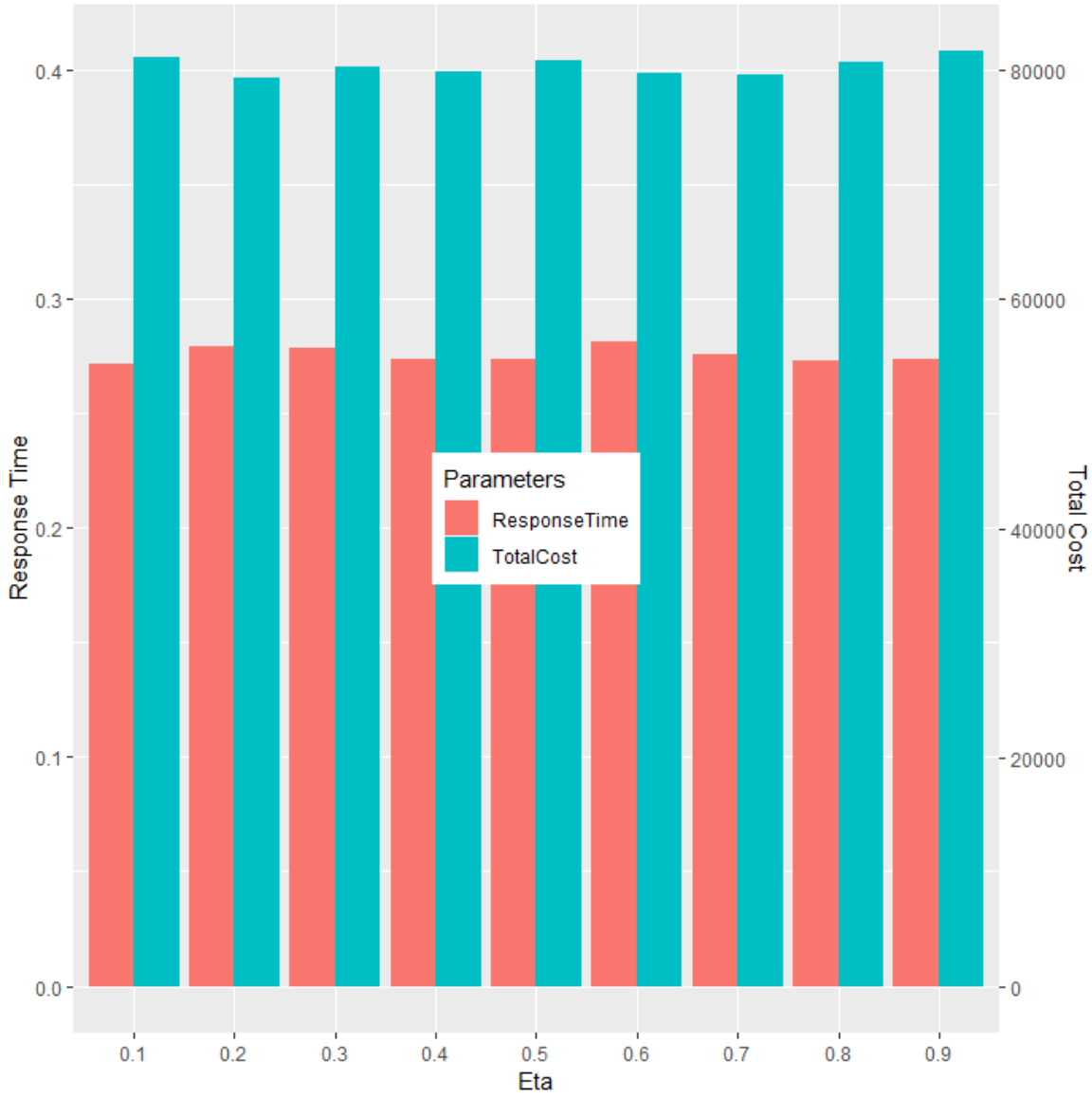
Hình 4.11: Kết quả thực nghiệm áp dụng bộ AS với bộ tham số $\varepsilon = 0.2, \gamma = 0.8, \eta = 0.2$

Hình 4.12 là kết quả đánh giá thời gian đáp ứng trung bình và chi phí theo từng trường hợp của η. Với dữ liệu đầu vào như vậy, các giá trị này cũng ít thay đổi. Như kết quả Hình 4.12, có thể chọn η = 0.2 cho kết quả thời gian đáp ứng trung bình tương đối cao và chi phí nhỏ nhất.

Như vậy, thông qua thực nghiệm, với kết quả thu được có thể chọn các tham số cho bộ AS là: tốc độ học η = 0.2, hệ số thăm dò/khai thác hành động cho các luật ε=0.2 và hệ số suy giảm trong hàm phần thưởng γ=0.8.

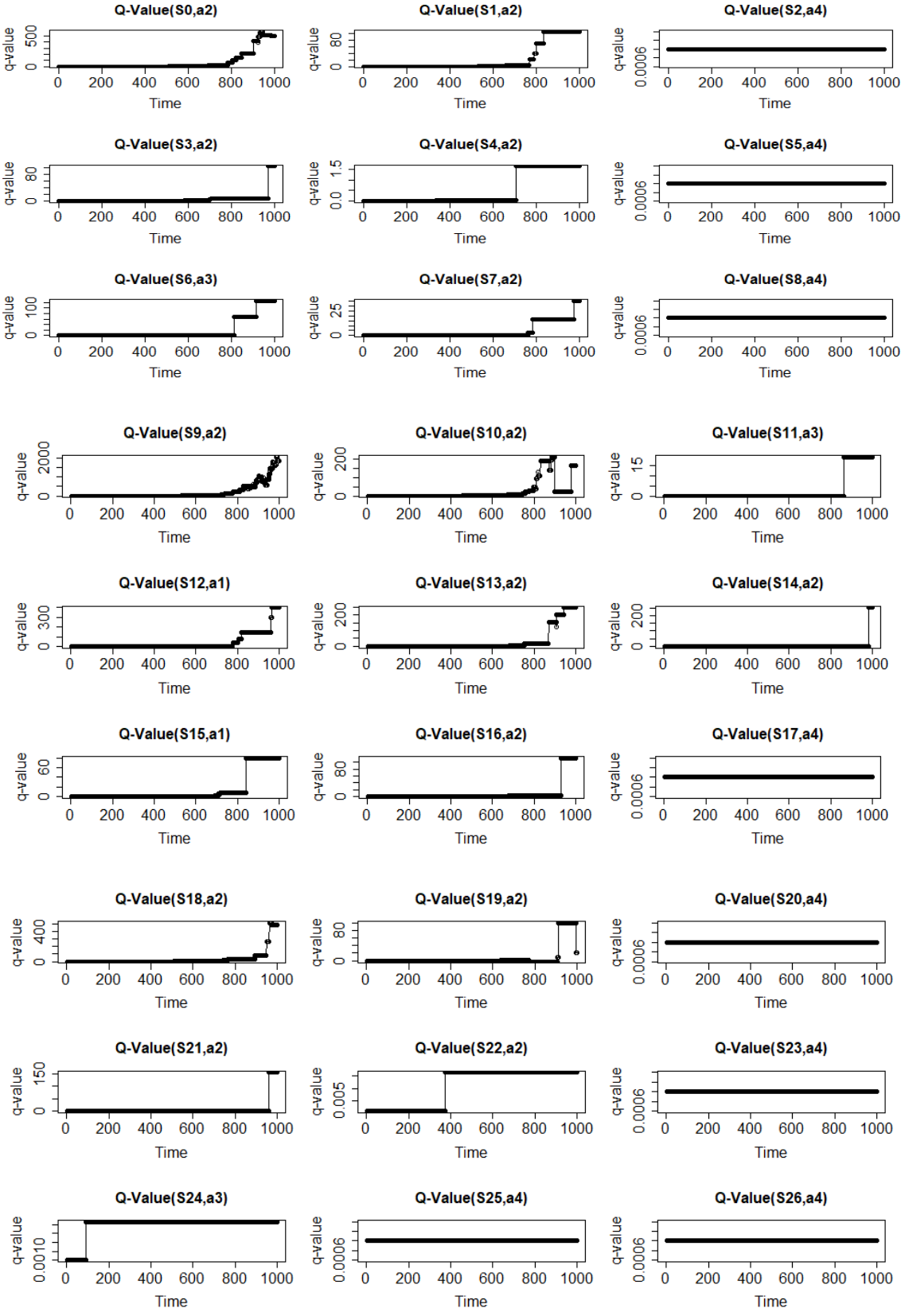
4.4.2.4. Đánh giá sự tiến hóa của giá trị q.

Không có tri thức, ban đầu các giá trị q được thiết lập bằng 0. Sau khi tiến hành thực nghiệm với bộ tham số ε=0.2, γ=0.8 và η=0.2, kết quả như Hình 4.13.

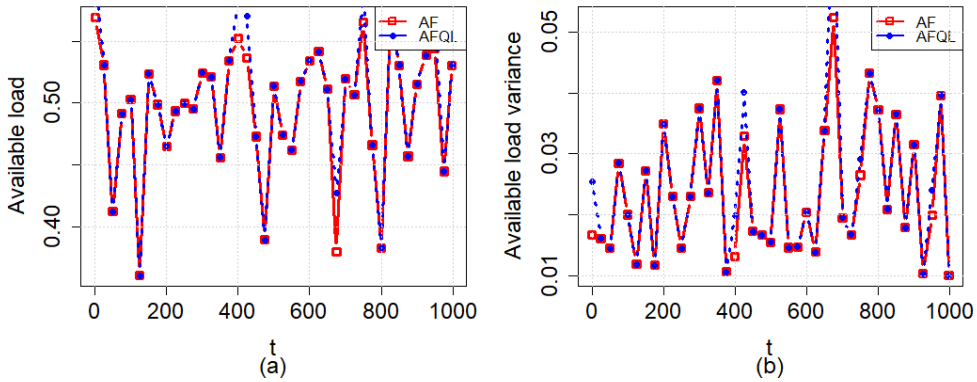


Hình 4.12: Xác định tham số tốc độ học (η)

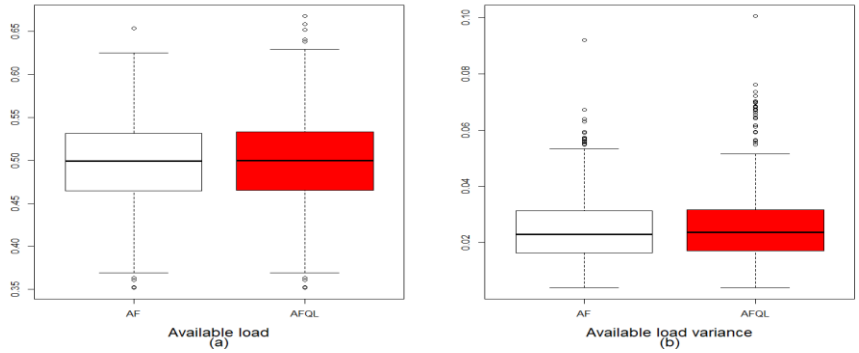
Có 27 trạng thái tương ứng với các giá trị s0, s1,…, s26 và có 5 giá trị hành động {-2, -1, 0, 1, 2} tương ứng là a0=-2, a1=-1, a2=0, a3=1, a4=2. Các giá trị q được tạo ra khi luật tương ứng được kích hoạt. Kết quả tối ưu cho mỗi luật là giá trị q lớn nhất sau quá trình học. Ví dụ, hành động a2=0 là kết quả tốt nhất cho trạng thái (luật) thứ 23 (s22) theo chiến lược ở trên.



Hình 4.13: Sự tiến hóa của các giá trị q với các tham số ε=0.2, γ=0.8 và η=0.2.



Hình 4.14: So sánh dựa vào tài nguyên khả dụng trung bình và phương sai



Hình 4.15: Biểu đồ hộp so sánh giữa tài nguyên khả dụng trung bình và phương sai

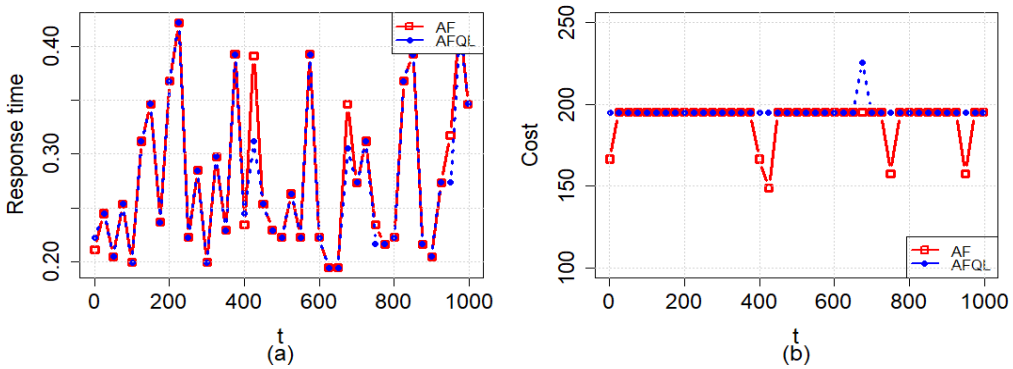
4.4.2.5. Đánh giá bộ tự động điều chỉnh đề xuất với logic mờ sử dụng tập luật chuyên gia

Ở đây, sử dụng kết quả học theo các giá trị tham số ε=0.2, γ=0.8 và η=0.2 được một tập luật sau khi học. Sau đó áp dụng tập luật này cho bộ AS để so sánh với tập luật chuyên gia ở trên.

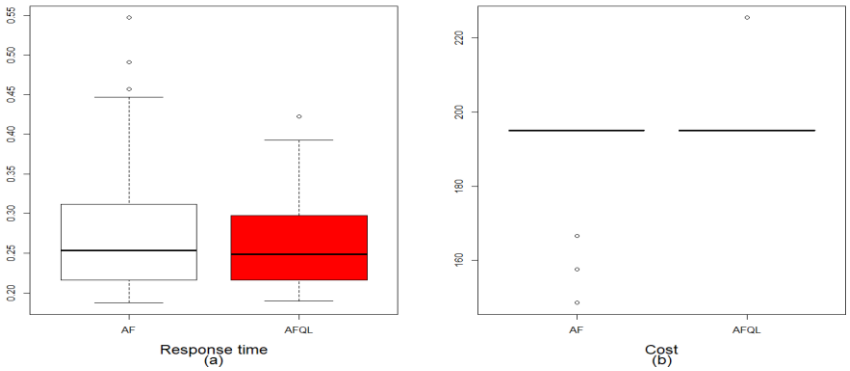
Từ kết quả Hình 4.14 và Hình 4.15 cho thấy, giá trị trung bình của tài nguyên khả dụng và phương sai là như nhau cho cả hai phương pháp học tăng cường và logic mờ. Tuy nhiên, đối với phương pháp học tăng cường mờ, bắt đầu là không có tri thức, tri thức được tích lũy trong quá trình học, do vậy nó có nhiều điểm bất thường hơn phương pháp logic mờ sử dụng tập luật chuyên gia (có tri thức sẵn) (xem Hình 4.15).

Từ Hình 4.16 và 4.17 cho thấy, giá trị trung bình giá trị thời gian đáp ứng và chi phí cho cả hai phương pháp là như nhau. Tuy nhiên, về biên độ dao động giá trị thời gian đáp ứng trung bình, đối với phương pháp logic mờ sử dụng tri thức chuyên gia có biên độ lớn hơn (Hình 4.17(a)). Về chi phí là gần như nhau đối với cả hai phương

pháp, nhưng phương pháp logic mờ lại có nhiều điểm bất thường (outlier) hơn. Hình 4.14, 4.15, 4.16 và 4.17 cho thấy kết quả so sánh tổng thể giữa bộ AS sử dụng logic mờ (AF) và bộ AS sử dụng kỹ thuật học tăng cường kết hợp với logic mờ (AFQL). AFQL cho kết quả thời gian đáp ứng trung bình tốt hơn so với AF. Tuy nhiên, tỉ lệ khác biệt về mặt chi phí thấp hơn tỉ lệ khác biệt về mặt thời gian đáp ứng.



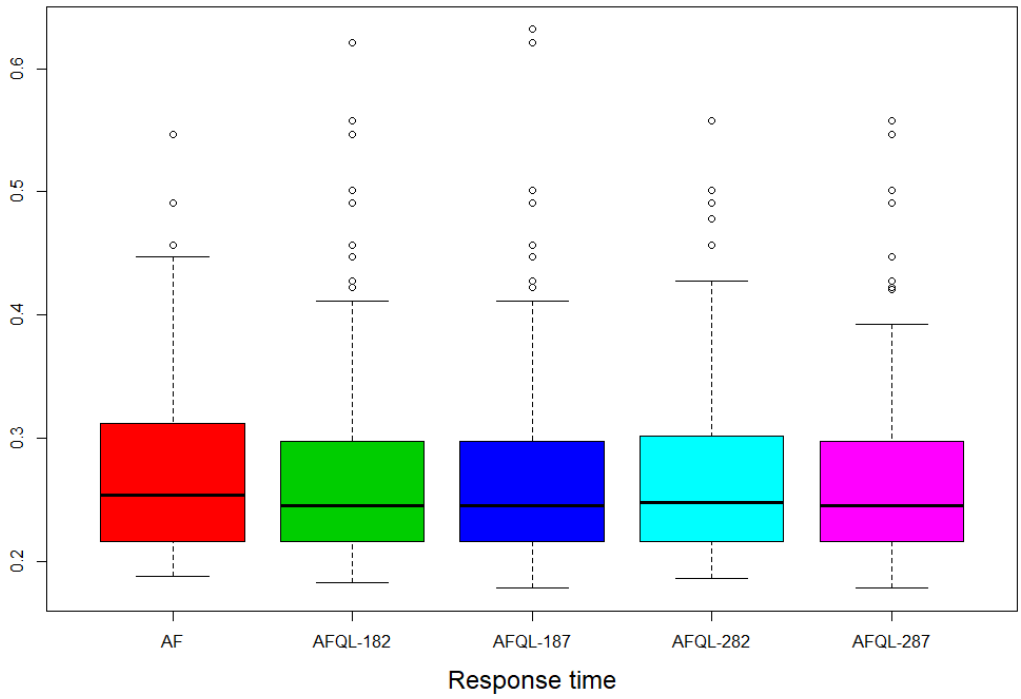
Hình 4.16: So sánh dựa vào thời gian đáp ứng trung bình và chi phí



Hình 4.17: Biểu đồ hộp so sánh giữa thời gian đáp ứng trung bình và chi phí

Ngoài ra, luận án còn tiến hành thử nghiệm với bốn bộ tham số: AFQL-182(ε = 0.1, γ = 0.8, η = 0.2); AFQL-187(ε = 0.1, γ = 0.8, η = 0.7); AFQL-282(ε = 0.2, γ = 0.8, η = 0.2) và AFQL-287(ε = 0.2, γ = 0.8, η = 0.7) để so sánh về thời gian đáp ứng trung bình (Response time), kết quả thể hiện trong Hình 4.18. Từ Hình 4.18, cho thấy bốn bộ tham số được lựa chọn có thể tốt hơn AF về thời gian đáp ứng trung bình , trong đó bộ tham số AFQL-282 cho kết quả xấu nhất trong bốn bộ tham số lựa chọn. Điều này cho thấy cần phải tiến hành nhiều thực nghiệm với nhiều bộ dữ liệu khác

nhau để tìm ra được bộ tham số tốt nhất cho bộ AS, vấn đề này là một hướng phát triển tiếp theo của luận án.



Hình 4.18: So sánh thời gian đáp ứng trung bình đối với bốn bộ tham số khác nhau

Tóm lại, kết quả tập luật thu được sau khi học (AFQL) với tập luật theo chuyên gia (AF) là gần tương đương nhau, điều này chứng tỏ rằng, bộ AS có thể tự học để tìm ra được tập luật tốt mà không cần có tri thức ban đầu. Do đó, bộ AS AFQL có thể tự thích nghi tốt hơn bộ AS AF, bộ AS đề xuất có thể hỗ trợ tốt hơn tập luật theo chuyên gia, tập luật theo chuyên gia còn phụ thuộc vào ý kiến chủ quan của các chuyên gia. Bộ AS đề xuất có thể thay thế con người tìm ra được tập luật tối ưu hoặc gần tối ưu để thực hiện điều chỉnh tài nguyên theo nhu cầu tải công việc.

4.4.2.6. Đánh giá bộ tự động điều chỉnh đề xuất với các bộ dữ liệu khác

Để đánh giá hiệu quả của bộ AS đề xuất, luận án đã tiến hành thực nghiệm trên các bộ dữ liệu khác nhau. Dựa trên các bộ dữ liệu được đánh giá từ công trình của Nikolas [43], luận án đã tạo ra ba bộ dữ liệu theo hướng của bộ dữ liệu ClarNet, bộ dữ liệu Wiki và bộ dữ liệu được sinh ngẫu nhiên. Kết quả thực nghiệm được đánh giá dựa vào ba bộ dữ liệu trên thông qua các tham số đầu vào gồm tài nguyên khả dụng

trung bình (Available load), phương sai của tài nguyên khả dụng (Variance Available load), thời gian đáp ứng trung bình (Response time) và sau đó so sánh kết quả thu được theo tổng chi phí và thời gian đáp ứng trung bình theo các bộ tự động điều chỉnh khác nhau.

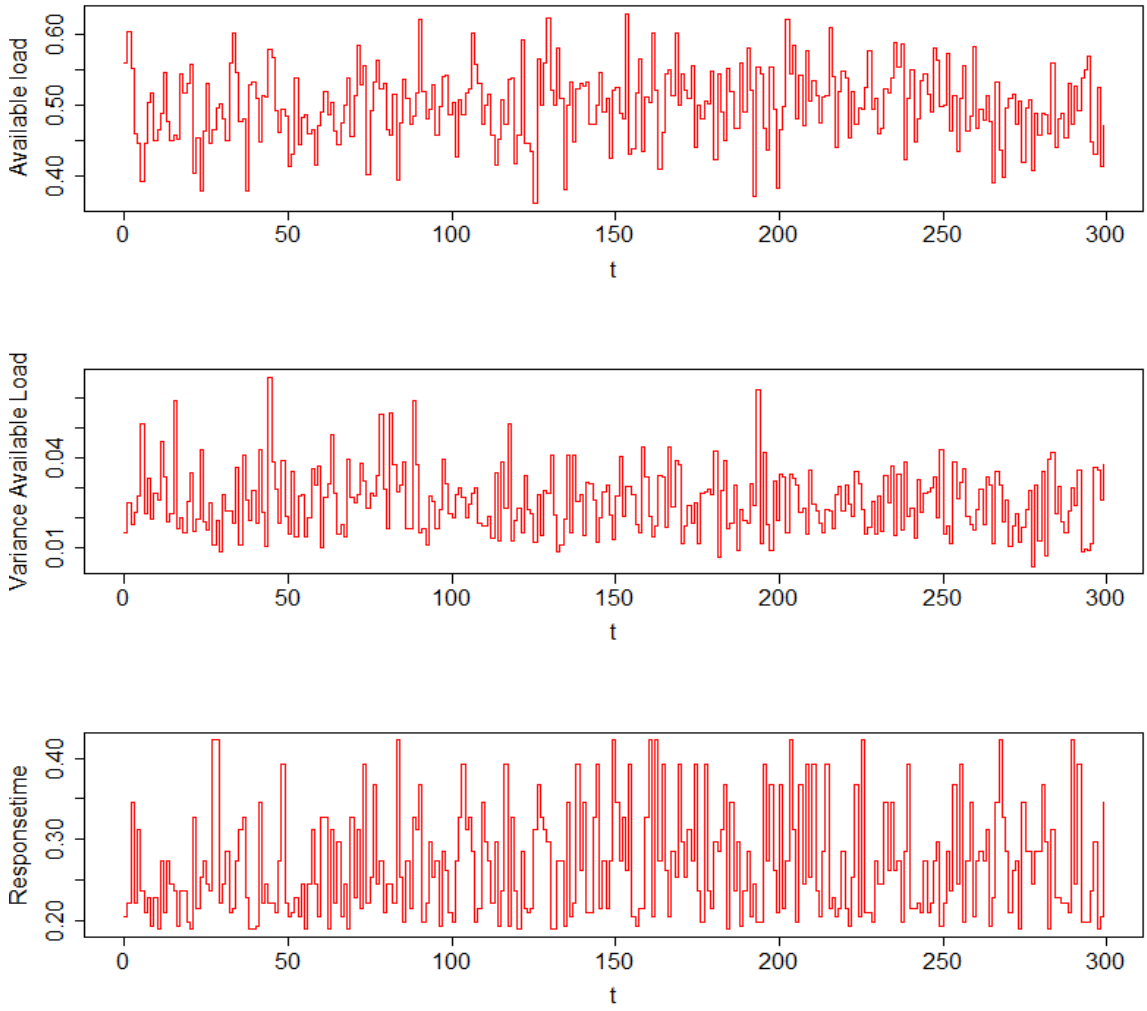
Luận án thực hiện so sánh, đánh giá kết quả bộ AS AFQL với các giá trị tham số ε=0.2, γ=0.8 và η=0.2 như trong phần trên đã nghiên cứu với thuật toán điều khiển mờ AF sử dụng tập luật của chuyên gia (Bảng 4.6). Thêm vào đó, luận án thu được kết quả một tập luật điều khiển mờ từ kết quả học (Bảng 4.7).

Bảng 4.7: Tập luật thu được sau khi học xong với bộ tham số ε=0.2, γ=0.8 và η=0.2

Luật	**Tiền đề của luật**			**Hành động**
	Giá trị tài nguyên khả dụng trung bình ($\overline{w}$)	**Thời gian đáp ứng (rt)**	**Phương sai của tài nguyên khả dụng (var)**	$[-2,2]$
1	Low	Good	Low	1
2	Low	Good	Medium	1
3	Low	Good	High	2
4	Low	Normal	Low	1
5	Low	Normal	Medium	1
6	Low	Normal	High	2
7	Low	Bad	Low	1
8	Low	Bad	Medium	-1
9	Low	Bad	High	2
10	Medium	Good	Low	1
11	Medium	Good	Medium	1
12	Medium	Good	High	1
13	Medium	Normal	Low	-1
14	Medium	Normal	Medium	-1
15	Medium	Normal	High	-1
16	Medium	Bad	Low	-1
17	Medium	Bad	Medium	1
18	Medium	Bad	High	2
19	High	Good	Low	0
20	High	Good	Medium	1
21	High	Good	High	1
22	High	Normal	Low	1
23	High	Normal	Medium	-1
24	High	Normal	High	2
25	High	Bad	Low	-1
26	High	Bad	Medium	2
27	High	Bad	High	2

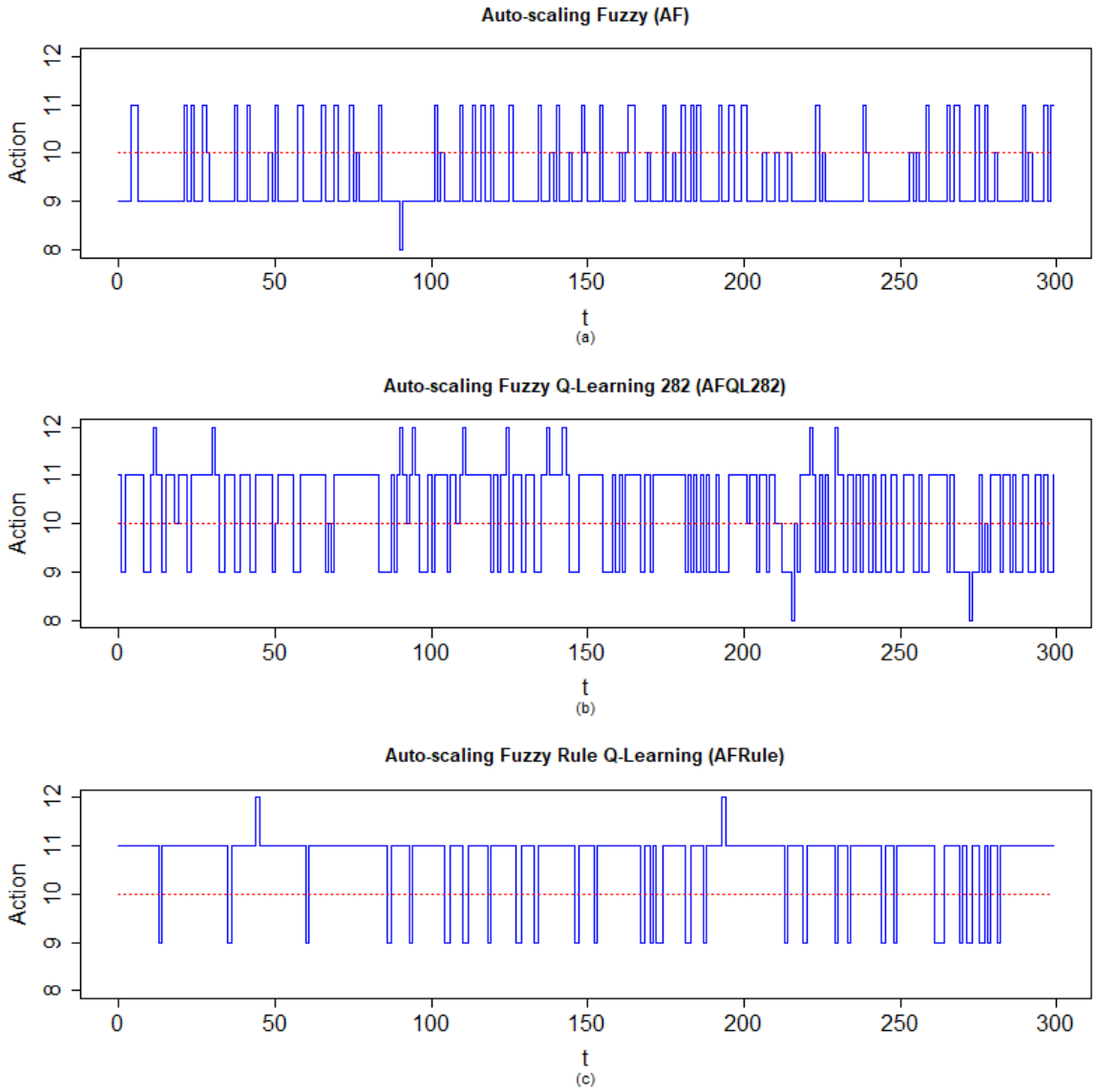
a. Với bộ dữ liệu ClarkNet

Dữ liệu đầu vào ClarkNet với các tham số tài nguyên khả dụng trung bình (Available load), phương sai (Variance Available load) và thời gian đáp ứng trung bình (Response time) được thể hiện như trong Hình 4.19.



Hình 4.19: Dữ liệu đầu vào của bộ dữ liệu ClarkNet

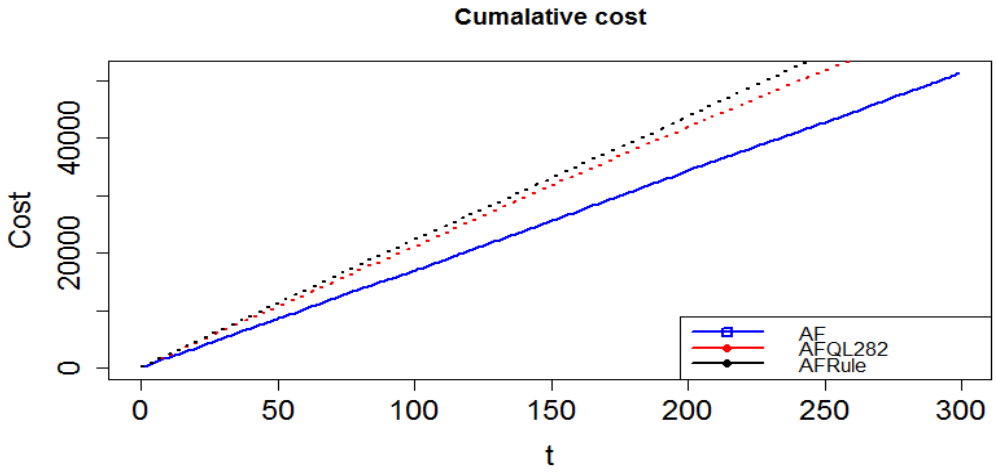
Sau khi chạy với bộ dữ liệu trên, kết quả việc điều chỉnh hành động được thể hiện như hình 4.20. Hình 4.20 (a) là kết quả của bộ AS sử dụng logic mờ với tập luật chuyên gia (Bảng 4.6), gọi là AF. Hình 4.20(b) là kết quả của bộ AS sử dụng học tăng cường kết hợp với bộ điều khiển mờ, gọi là AFQL282. Hình 4.20(c) là kết quả của bộ AS sử dụng logic mờ với tập luật đã được học theo bộ tham số (ε=0.2, γ=0.8 và η=0.2) cho trong Bảng 4.7, gọi là AFRule.



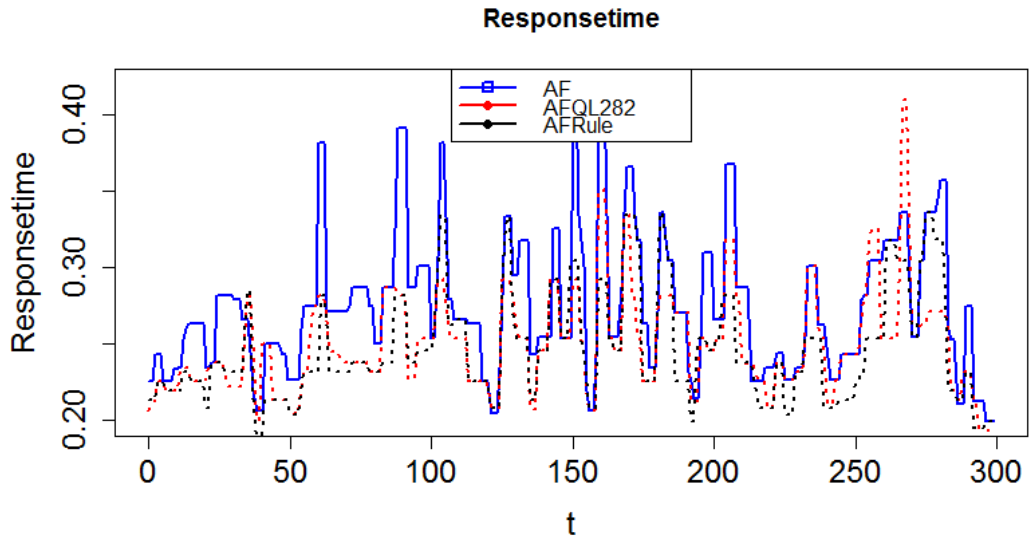
Hình 4.20: Kết quả thực hiện điều chỉnh trên bộ dữ liệu ClarkNet

Hình 4.21 và Hình 4.22 là so sánh tổng chi phí và thời gian đáp ứng trung bình thực hiện điều chỉnh đối với 3 trường hợp ở trên.

Như vậy, từ Hình 4.21 cho thấy tổng chi phí của bộ AS sử dụng tập luật chuyên gia (Bảng 4.6) AF là nhỏ nhất, tổng chi phí của bộ AS sử dụng tập luật sau khi học xong (Bảng 4.7) AFRule là nhiều nhất. Ngược lại, trong Hình 4.22, cho thấy thời gian đáp ứng trung bình của AFRule nhỏ hơn của AF, tức là với tập luật đã được học cho thời gian đáp ứng tốt hơn, tuy nhiên phải trả chi phí nhiều hơn.



Hình 4.21: So sánh tổng chi phí đối với 3 trường hợp trên dữ liệu ClarkNet



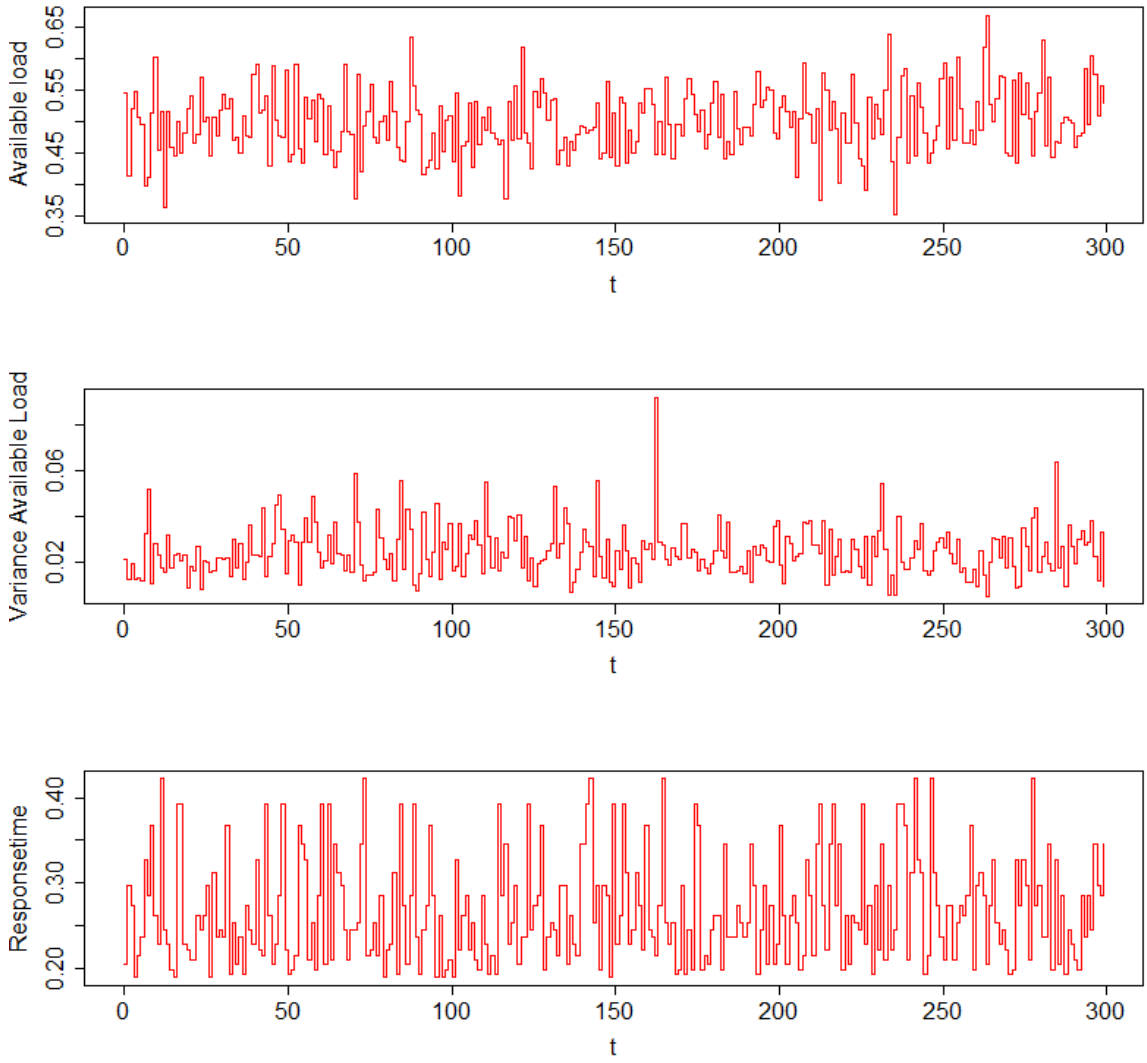
Hình 4.22: So sánh thời gian đáp ứng trung bình đối với 3 trường hợp trên bộ dữ liệu ClarkNet

b. Với bộ dữ liệu Wiki

Dữ liệu đầu vào Wiki với các tham số tài nguyên khả dụng trung bình (Available load), phương sai (Variance Available load) và thời gian đáp ứng trung bình (Response time) được thể hiện như trong hình 4.23.

Sau khi chạy với bộ dữ liệu Wiki, kết quả việc điều chỉnh hành động được thể hiện như hình 4.24. Hình 4.24 (a) là kết quả của bộ AS sử dụng logic mờ với tập luật chuyên gia (bảng 4.6), gọi là AF. Hình 4.24(b) là kết quả của bộ AS sử dụng học tăng

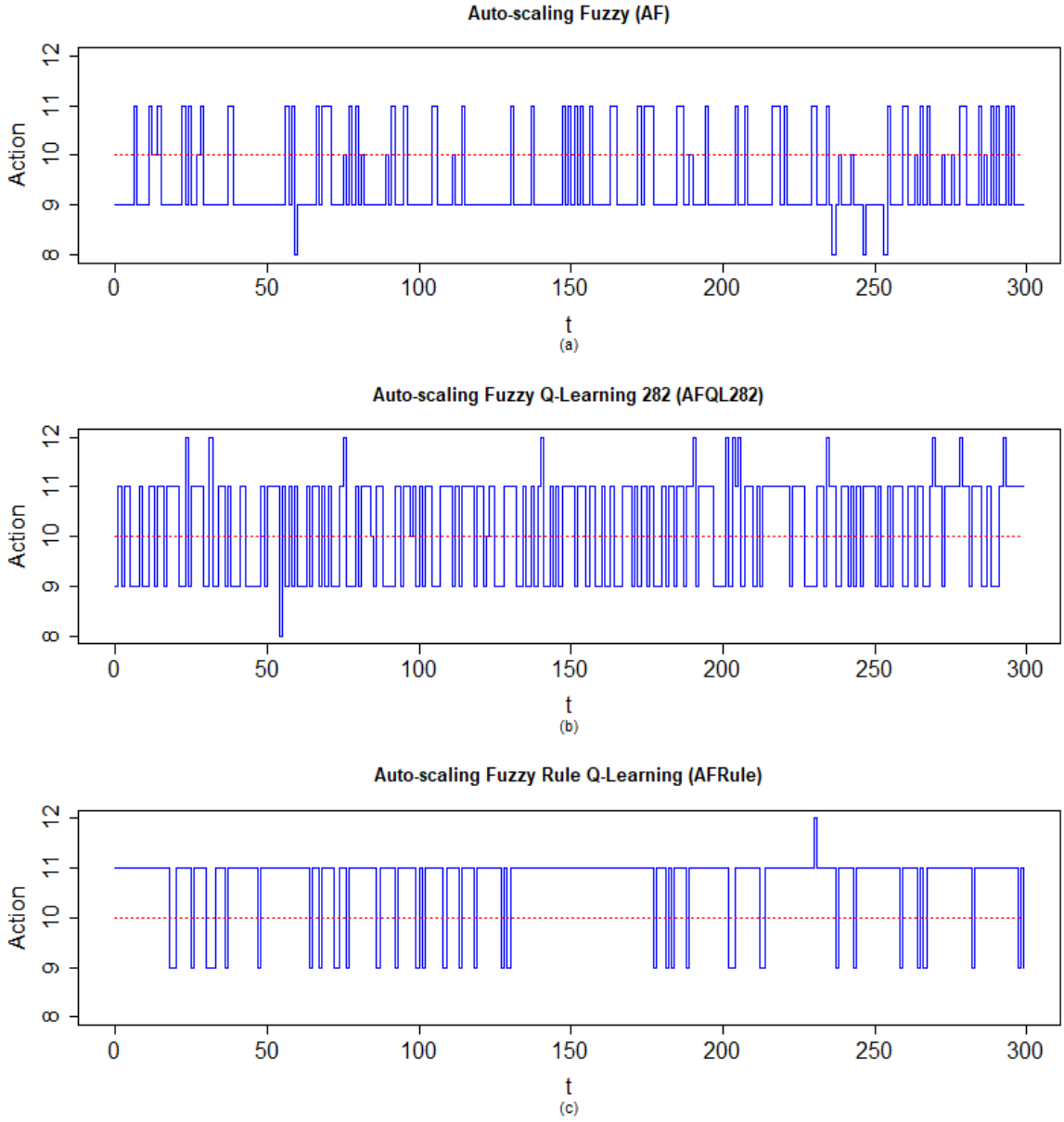
cường kết hợp với bộ điều khiển mờ, gọi là AFQL282. Hình 4.24(c) là kết quả của bộ AS sử dụng logic mờ với tập luật đã được học theo cho trong bảng 4.7, gọi là AFRule.



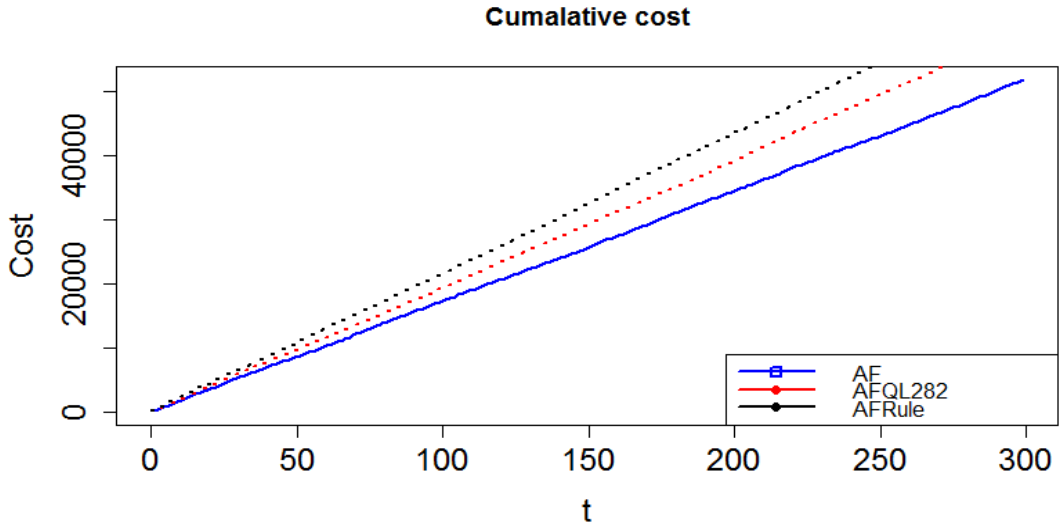
Hình 4.23: Dữ liệu đầu vào của bộ dữ liệu Wiki

Hình 4.25 và Hình 4.26 là so sánh chi phí và thời gian đáp ứng trung bình thực hiện điều chỉnh đối với 3 trường hợp ở trên.

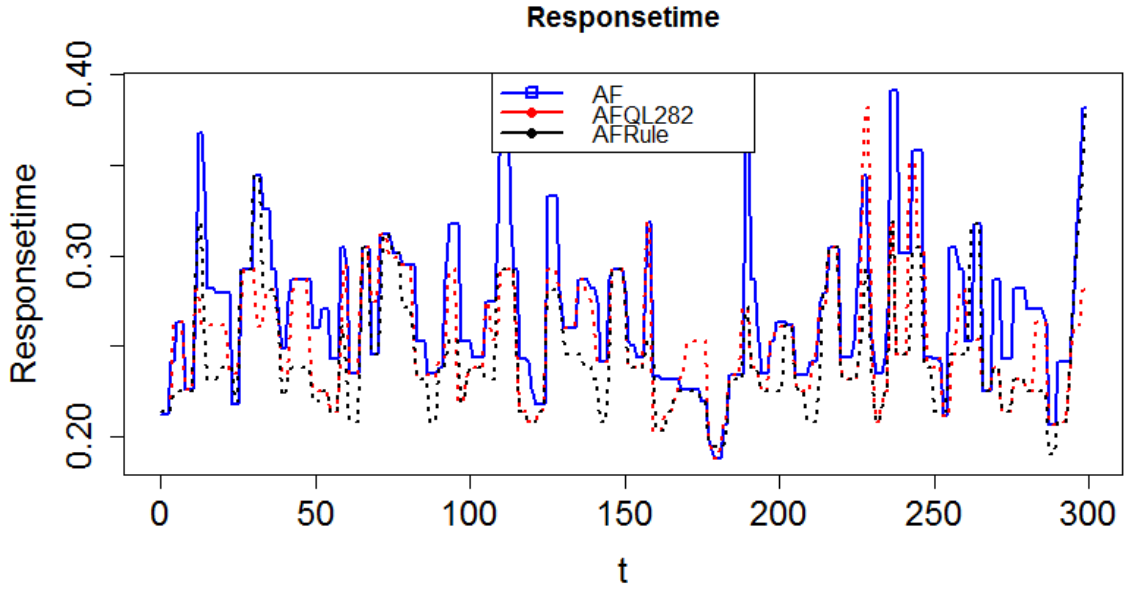
Tương tự như đối với bộ dữ liệu ClarkNet, đối với bộ AS sử dụng tập luật đã được học sẽ đáp ứng tốt hơn (cho thời gian đáp ứng trung bình nhỏ) bộ AS sử dụng tập luật chuyên gia (Bảng 4.6). Nhưng phải tốn nhiều chi phí hơn xem Hình 4.25.



Hình 4.24: Kết quả thực hiện điều chỉnh với bộ dữ liệu Wiki



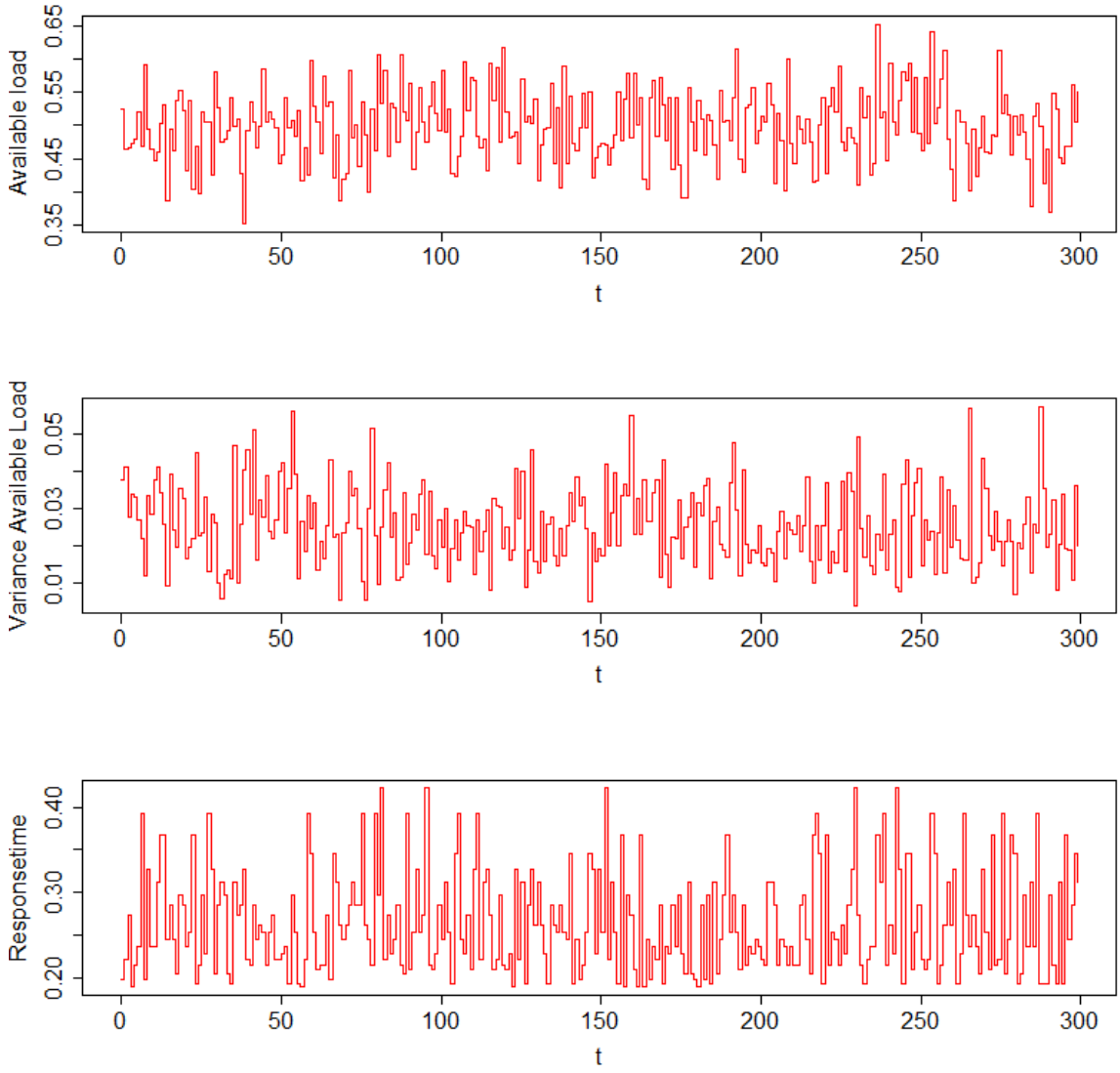
Hình 4.25: So sánh tổng chi phí đối với 3 trường hợp trên dữ liệu Wiki



Hình 4.26: So sánh thời gian đáp ứng trung bình đối với 3 trường hợp trên bộ dữ liệu Wiki

c. Với bộ dữ liệu sinh ngẫu nhiên

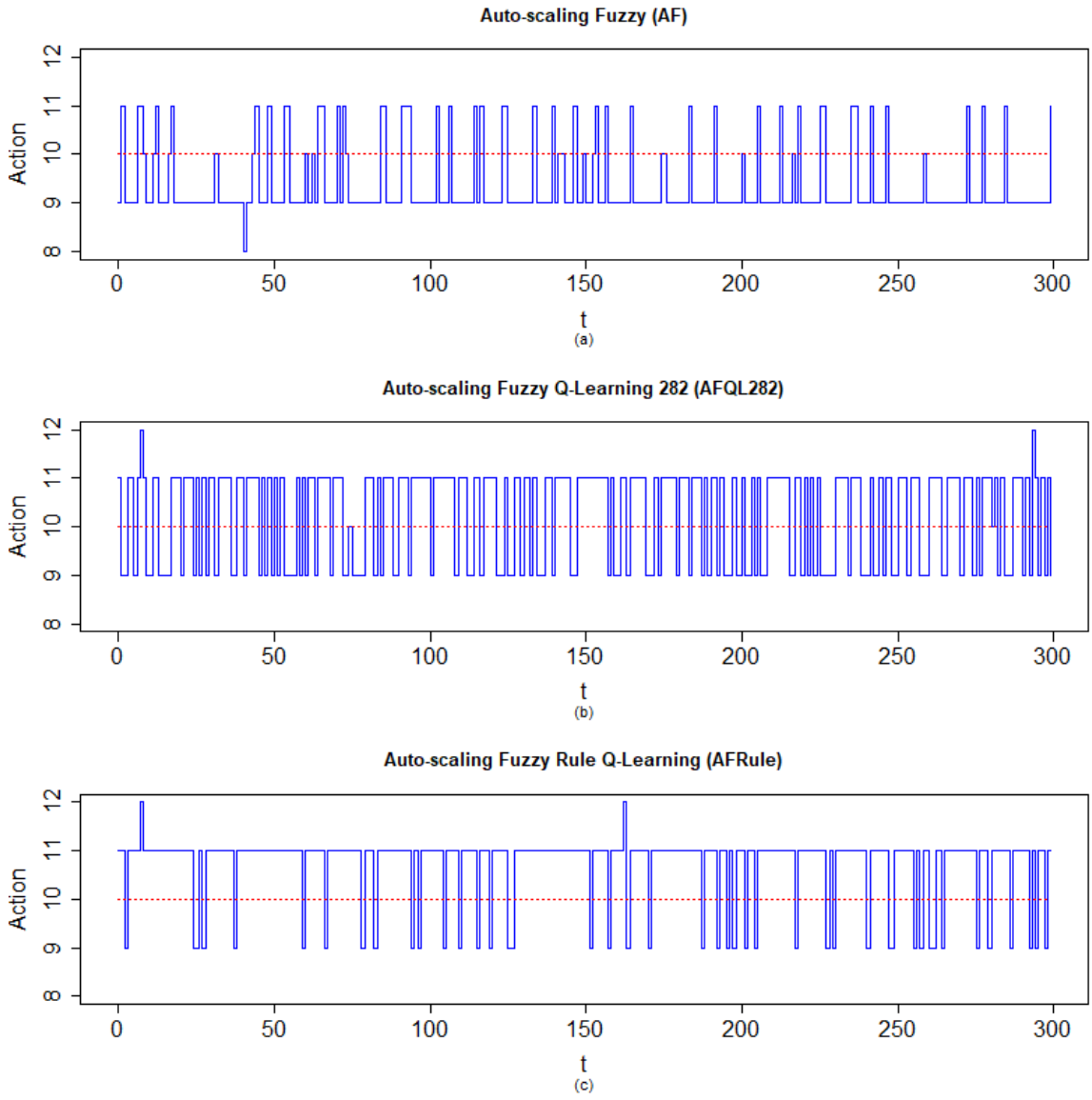
Dữ liệu đầu vào Wiki với các tham số tài nguyên khả dụng trung bình (Available load), phương sai (Variance Available load) và thời gian đáp ứng trung bình (Response time) được thể hiện như trong Hình 4.27.



Hình 4.27: Dữ liệu đầu vào của bộ dữ liệu sinh ngẫu nhiên

Sau khi chạy với bộ dữ liệu sinh ngẫu nhiên, kết quả việc điều chỉnh hành động được thể hiện như hình 4.28. Hình 4.28 (a) là kết quả của bộ AS sử dụng logic mờ với tập luật chuyên gia (Bảng 4.6), gọi là AF. Hình 4.28(b) là kết quả của bộ AS sử dụng học tăng cường kết hợp với bộ điều khiển mờ, gọi là AFQL282. Hình 4.28(c) là kết quả của bộ AS sử dụng logic mờ với tập luật đã được học cho trong Bảng 4.7, gọi là AFRule.

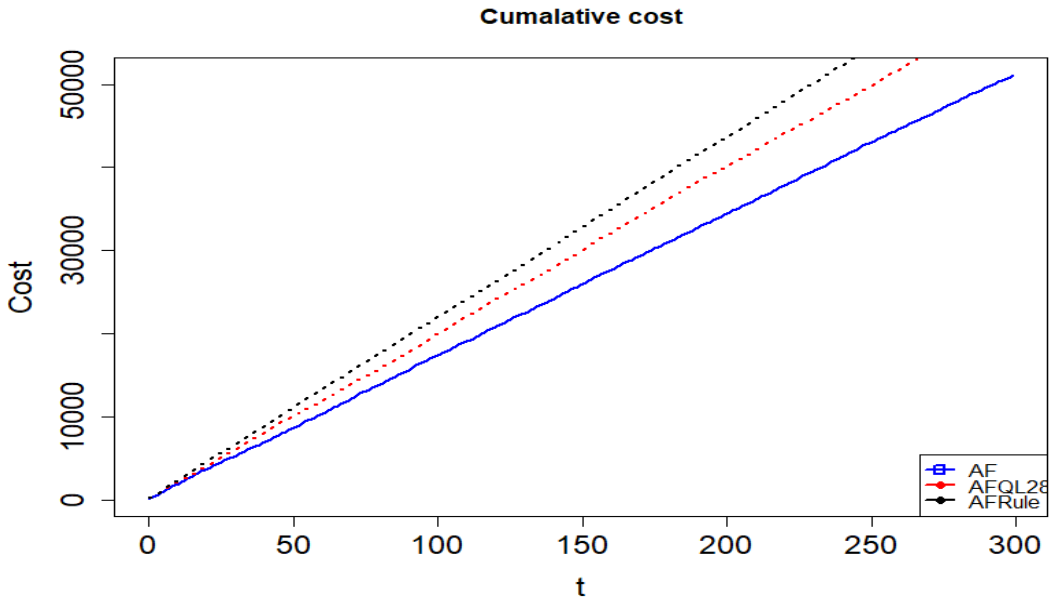
Hình 4.29 và Hình 4.30 là so sánh chi phí và thời gian đáp ứng trung bình thực hiện điều chỉnh đối với 3 trường hợp ở trên.



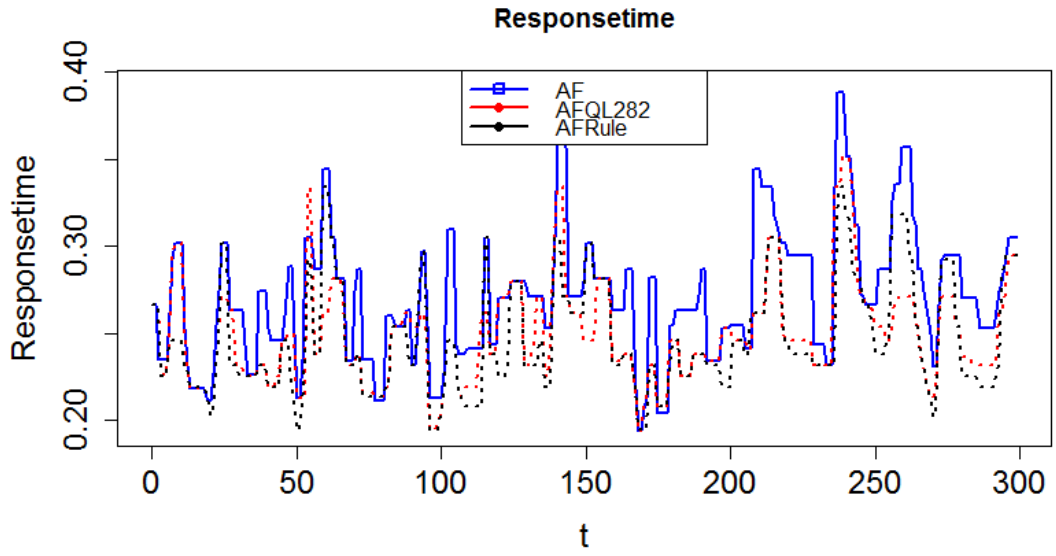
Hình 4.28: Kết quả thực hiện điều chỉnh với bộ dữ liệu sinh ngẫu nhiên

Đối với bộ dữ liệu sinh ngẫu nhiên này, với bộ AS sử dụng học tăng cường kết hợp với điều khiển mờ cho thấy biến động về thời gian đáp ứng trung bình rất lớn (đường nét đứt màu đỏ Hình 4.30), điều này cho thấy việc khám phá luật xảy ra nhiều trong quá trình học luật. Tuy nhiên, đối với bộ AS sử dụng tập luật được học (Bảng

4.7) vẫn cho thời gian đáp ứng nhỏ nhất (xem Hình 4.30) và tổng chi phí lại nhiều hơn (xem Hình 4.29).



Hình 4.29: So sánh tổng chi phí đối với 3 trường hợp trên dữ liệu ngẫu nhiên



Hình 4.30: So sánh thời gian đáp ứng trung bình đối với 3 trường hợp trên bộ dữ liệu ngẫu nhiên

Tóm lại, với kết quả thực nghiệm trên ba bộ dữ liệu trên cho thấy, tập luật được tạo ra sau khi học tăng cường kết hợp với điều khiển mờ (Bảng 4.7) khi sử dụng bộ

tham số (ε=0.2, γ=0.8 và η=0.2) đều cho lợi thế về thời gian đáp ứng ít nhất. Tuy nhiên phải chịu tổng chi phí cao hơn.

KẾT LUẬN CHƯƠNG 4

Chương này đã xây dựng một bộ điều khiển AS hiệu quả. Bộ AS được xây dựng để điều chỉnh tài nguyên tự động bằng cách sử dụng học tăng cường kết hợp với logic mờ để tăng khả năng đáp ứng tức thời yêu cầu tài nguyên trong việc cấp phát/thu hồi tài nguyên tự động dựa vào tài nguyên khả dụng trung bình, phương sai của tài nguyên khả dụng của của cụm VM và thời gian đáp ứng trung bình của hệ thống. Bên cạnh đó, bộ điều khiển AS còn cho thấy hiệu quả về mặt chi phí. Bộ AS đề xuất không cần có tri thức ban đầu mà vẫn hoạt động hiệu quả, các tri thức có được thông qua việc được tích lũy dần dần trong quá trình học. Phần thực nghiệm, đã đánh giá để lựa chọn các tham số thăm dò/khai thác các hành động (ε), hệ số suy giảm trong hàm phần thưởng (γ) để làm tăng tính hội tụ và tính hiệu quả của mô hình đề xuất, đánh giá các hệ số học (η). Sau đó, lựa chọn bộ tham số (ε, γ, η) thích hợp để tiến hành so sánh hai mô hình AS sử dụng học tăng cường kết hợp với logic mờ (AFQL) và mô hình AS sử dụng logic mờ (AF). Kết quả cho thấy bộ AS AFQL hiệu quả và không cần có tri thức ban đầu, bộ AS AFQL vẫn có thể hoạt động hiệu quả, các tri thức có được trong quá trình hoạt động.

Tuy nhiên, hiệu quả của bộ AS đề xuất còn phụ thuộc vào nhiều yếu tố khác, như số đoạn phân hoạch các giá trị đầu vào, hệ thống suy luận mờ, các tham số ε, γ, η… Đây là hướng phát triển tiếp theo của luận án để có thể xây dựng được bộ AS hiệu quả hơn và có thể áp dụng hiệu quả vào trong thực tế.

KẾT LUẬN

Như vậy, luận án đã nghiên cứu được các nội dung đề ra ở phần mục tiêu ban đầu. Đầu tiên, luận án nghiên cứu những cơ sở lý thuyết được sử dụng trong luận án, như: khái niệm về AS, các phương pháp điều chỉnh trong CC, một số kỹ thuật AS hiện có và một số vấn đề liên quan. Nghiên cứu tổng quan các công trình liên quan đến nghiên cứu trong luận án, chỉ ra những ưu điểm và hạn chế, sau đó đề ra hướng nghiên cứu của luận án, những nội dung này được trình bày trong Chương 1.

Chương 2 của luận án đã nghiên cứu và đề xuất một kỹ thuật cân bằng tải, một kỹ thuật di trú hiệu quả trong môi trường CC, sau đó là đánh giá ảnh hưởng giữa các kỹ thuật cân bằng tải với AS trong CC. Các kết quả thực nghiệm cũng đã chỉ ra tính hiệu quả và tính đúng đắn của giải pháp đề xuất, giúp cho người dùng CC có định hướng lựa chọn dịch vụ và kỹ thuật hiệu quả trong thực tế.

Sang Chương 3 của luận án là nghiên cứu và đề xuất mô hình mạng hàng đợi cho hệ thống CC phức tạp và không đồng nhất nhằm đánh giá một số số đo hiệu năng của hệ thống CC, kết quả thực nghiệm cũng chỉ ra kết quả bước đầu của mô hình là có thể tin cậy, đánh giá được các thông số mong muốn, những thông số này là đầu vào quan trong cho các bộ AS.

Cuối cùng là giải pháp AS tài nguyên hiệu quả trong CC được trình bày trong Chương 4. Trong chương này, luận án đã xây dựng một bộ AS kết hợp giữa bộ điều khiển mờ và học tăng cường, sử dụng ba tham số đầu vào là tài nguyên khả dụng trung bình của cụm VM ($\overline{w}$), phương sai của tài nguyên khả dụng (var) và thời gian đáp ứng trung bình của hệ thống. Kết quả bước đầu được so sánh với phương pháp logic mờ được xây dựng bằng phương pháp chuyên gia, cho thấy tính hiệu quả của phương pháp đề xuất. Phương pháp đề xuất có thể hoạt động mà không cần tri thức ban đầu, kết quả hoạt động có thể bằng và hơn phương pháp logic mờ truyền thống, được thể hiện qua các kết quả thử nghiệm. Tuy nhiên phương pháp đề xuất vẫn còn những hạn chế nhất định và được chỉ ra ở phần hướng phát triển của luận án.

1) Những kết quả chính của luận án:

(1) Luận án đã nghiên cứu và đề xuất được kỹ thuật cân bằng tải và kỹ thuật di trú hiệu quả trong CC tạo điều kiện thuận lợi cho việc điều chỉnh tài nguyên giúp cải thiện hiệu năng của các trung tâm CC. Tiếp theo, Luận án đã đánh giá sự ảnh hưởng của các kỹ thuật cân bằng tải trong môi trường CC có AS tài nguyên, giúp cho việc lựa chọn kỹ thuật cân bằng tải hiệu quả.

(2) Luận án đã mô hình hóa môi trường CC phức tạp và không đồng nhất sử dụng mô hình mạng hàng đợi – mạng Jackson mở, làm cơ sở để đánh giá các số đo hiệu năng của trung tâm CC.

(3) Luận án đã xây dựng được một bộ AS hiệu quả trong môi trường CC. Bộ AS sử dụng kết hợp giữa học tăng cường và logic mờ.

2) Hướng phát triển của luận án:

(1) Luận án có thể được phát triển theo hướng xây dựng mô hình cơ sở dựa vào các mô hình hàng đợi khác nhau để đo lường và đánh giá hiệu năng của hệ thống điện toán đám mây. Từ đó có được mô hình lý thuyết đầy đủ hỗ trợ hoạt động nghiên cứu và triển khai hệ thống điện toán đám mây.

(2) Ngoài ra, luận án có thể được phát triển theo hướng cấu hình tùy biến bộ tự động điều chỉnh theo trạng thái của hệ thống, trên cơ sở nghiên cứu phân hoạch giá trị đầu vào và sử dụng các hệ suy luận mờ khác nhau.

DANH MỤC CÁC CÔNG TRÌNH CÔNG BỐ

TẠP CHÍ QUỐC TẾ

TCNN1. Nguyen Hong Son, **Nguyen Khac Chien** (2017), "Load Balancing in Auto Scaling-Enabled Cloud Environments", *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*. Vol. 7, No. 5, October 2017, pp. 15-22.

TCNN2. **Chien Nguyen Khac**, Khiet Bui Thanh, Hung Ho Dac, Vu Pham Tran, Hung Tran Cong, Son Nguyen Hong (2018). *"An auto-scaling approach for infrastructure as a service cloud computing based on Fuzzy Q-Learning".* ***Đang phản biện.*** Concurrency and Computation: Practice and Experience Journal. **Special Issue On**: Context-Aware Systems and Applications.

TCNN3. **Chien Nguyen Khac**, Khiet Bui Thanh, Hung Ho Dac, Son Nguyen Hong, Vu Pham Tran, Hung Tran Cong (2018). "*An open jackson network model for heterogeneous infrastructure as a service on cloud computing"*. International Journal of Computer Networks & Communications (IJCNC) (2018). http://airccse.org/journal/ijcnc.html ***(Đã chấp nhận)***

TẠP CHÍ TRONG NƯỚC

TCTN1. **Nguyễn Khắc Chiến,** Nguyễn Hồng Sơn, Hồ Đắc Lộc, Nguyễn Văn Vịnh (2016), "Nghiên cứu một số thuật toán ra quyết định di trú máy ảo trong điện toán đám mây", *Tạp chí Khoa học Giáo dục Kỹ thuật trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh*. 35B (3/2016), pp. 74-81.

HỘI NGHỊ QUỐC TẾ

HNQT1. **Nguyen Khac Chien,** Nguyen Hong Son, Ho Dac Loc (2016), "Load balancing algorithm based on estimating finish time of services in cloud computing", *International Conference on Advanced Communication Technology (ICACT) at the Phoenix Park, PyeongChang, Korea*, pp. 228-233, Jan. 31-Feb. 3, 2016.

HNQT2. **Nguyen Khac Chien,** Vo Sy Giang Dong, Nguyen Hong Son, and Ho Dac Loc. (2016), "An Efficient Virtual Machine Migration Algorithm based on

Minimization of Migration in Cloud Computing", *International Conference on Nature of Computation and Communication (ICTCC) in Rach Gia city of Vietnam*, Springer, pp. 62-71.

HNQT3. Khiet Bui Thanh, Lam Mai Nguyen Xuan, **Chien Nguyen Khac,** Hung Ho Dac, Vu Pham Tran, Hung Tran Cong (2018), "An auto-scaling VM game approach for multi-tier application with Particle swarm optimization algorithm in Cloud computing". *International Conference on Advanced Technologies for Communication (ATC) October 18-20, 2018, Ho Chi Minh City, Vietnam.*

HỘI NGHỊ TRONG NƯỚC

HNTN1. **Nguyễn Khắc Chiến,** Bùi Thanh Khiết, Hồ Đắc Hưng, Nguyễn Hồng Sơn, Hồ Đắc Lộc (2017), "Một mô hình học tăng cường cho vấn đề điều chỉnh tự động tài nguyên trong CC dựa trên fuzzy q-learning", *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*, pp. 535-543.

TÀI LIỆU THAM KHẢO

[1]. Ali-Eldin, Ahmed, et al. (2012), Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control, *Proceedings of the 3rd workshop on Scientific Cloud Computing*, ACM, pp. 31-40.

[2]. Ali-Eldin, Ahmed, Tordsson, Johan, and Elmroth, Erik (2012), An adaptive hybrid elasticity controller for cloud infrastructures, *Network Operations and Management Symposium (NOMS), 2012 IEEE*, IEEE, pp. 204-212.

[3]. Alipour, Hanieh, Liu, Yan, and Hamou-Lhadj, Abdelwahab (2014), Analyzing auto-scaling issues in cloud environments, *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*, IBM Corp., pp. 75-89.

[4]. Ashalatha, R and Agarkhed, Jayashree (2015), "Evaluation of auto scaling and load balancing features in cloud", *International Journal of Computer Applications*. 117(6).

[5]. Bacigalupo, David A, et al. (2010), "Resource management of enterprise cloud systems using layered queuing and historical performance models".

[6]. Bai, Wei-Hua, et al. (2015), "Performance analysis of heterogeneous data centers in cloud computing using a complex queuing model", *Mathematical Problems in Engineering*. 2015.

[7]. Barrett, Enda, Howley, Enda, and Duggan, Jim (2013), "Applying reinforcement learning towards automating resource allocation and application scalability in the cloud", *Concurrency and Computation: Practice and Experience*. 25(12), pp. 1656-1674.

[8]. Beloglazov, Anton, Abawajy, Jemal, and Buyya, Rajkumar (2012), "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing", *Future generation computer systems*. 28(5), pp. 755-768.

[9]. Beloglazov, Anton and Buyya, Rajkumar (2012), "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers", *Concurrency and Computation: Practice and Experience*. 24(13), pp. 1397-1420.

[10]. Bi, Jing, et al. (2010), Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center, *Cloud Computing (CLOUD), 2010 IEEE 3rd international conference on*, IEEE, pp. 370-377.

[11]. Bodík, Peter, et al. (2009), "Statistical machine learning makes automatic control practical for internet datacenters".

[12]. Branco, Kalinka RLJC, et al. (2006), Piv and wpiv: Performance index for heterogeneous systems evaluation, *Industrial Electronics, 2006 IEEE International Symposium on*, IEEE, pp. 323-328.

[13]. Bu, Xiangping, Rao, Jia, and Xu, Cheng-Zhong (2013), "Coordinated self-configuration of virtual machines and appliances using a model-free learning approach", *IEEE transactions on parallel and distributed systems*. 24(4), pp. 681-690.

[14]. Calcavecchia, Nicolo M, et al. (2012), "DEPAS: a decentralized probabilistic algorithm for auto-scaling", *Computing*. 94(8-10), pp. 701-730.

[15]. Calheiros, Rodrigo N, et al. (2011), "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", *Software: Practice and experience*. 41(1), pp. 23-50.

[16]. Calheiros, Rodrigo N, Ranjan, Rajiv, and Buyya, Rajkumar (2011), Virtual machine provisioning based on analytical performance and QoS in cloud computing environments, *Parallel processing (ICPP), 2011 international conference on*, IEEE, pp. 295-304.

[17]. Calzarossa, Maria Carla, et al. (2016), "Workloads in the Clouds", *Principles of Performance and Reliability Modeling and Evaluation*, Springer, pp. 525-550.

[18]. Caron, Eddy, et al. (2012), *Auto-scaling, load balancing and monitoring in commercial and open-source clouds*, INRIA.

[19]. Casalicchio, Emiliano and Silvestri, Luca (2013), "Autonomic management of cloud-based systems: the service provider perspective", *Computer and Information Sciences III*, Springer, pp. 39-47.

[20]. Chi, Ruiqing, Qian, Zhuzhong, and Lu, Sanglu (2012), A game theoretical method for auto-scaling of multi-tiers web applications in cloud, *Proceedings of the Fourth Asia-Pacific Symposium on Internetware*, ACM, p. 3.

[21]. Chieu, Trieu C, Mohindra, Ajay, and Karve, Alexei A (2011), Scalability and performance of web applications in a compute cloud, *e-Business Engineering (ICEBE), 2011 IEEE 8th International Conference on*, IEEE, pp. 317-323.

[22]. Chieu, Trieu C, et al. (2009), Dynamic scaling of web applications in a virtualized cloud computing environment, *E-Business Engineering, 2009. ICEBE'09. IEEE International Conference on*, IEEE, pp. 281-286.

[23]. Dayan, Peter and Watkins, CJCH (1992), "Q-learning", *Machine learning*. 8(3), pp. 279-292.

[24]. Delimitrou, Christina and Kozyrakis, Christos (2013), Paragon: QoS-aware scheduling for heterogeneous datacenters, *ACM SIGPLAN Notices*, ACM, pp. 77-88.

[25]. Domanal, Shridhar G and Reddy, G Ram Mohana (2014), Optimal load balancing in cloud computing by efficient utilization of virtual machines, *Communication Systems and Networks (COMSNETS), 2014 Sixth International Conference on*, IEEE, pp. 1-4.

[26]. Dutreilh, Xavier, et al. (2011), Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow, *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems*, pp. 67-74.

[27]. Dutreilh, Xavier, et al. (2010), From data center resource allocation to control theory and back, *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, IEEE, pp. 410-417.

[28]. Esfahani, Naeem, Elkhodary, Ahmed, and Malek, Sam (2013), "A learning-based framework for engineering feature-oriented self-adaptive software systems", *IEEE transactions on software engineering*. 39(11), pp. 1467-1493.

[29]. Fallah, Monireh, Arani, Mostafa Ghobaei, and Maeen, Mehrdad (2015), "NASLA: Novel auto scaling approach based on learning automata for web application in cloud computing environment", *International Journal of Computer Applications*. 113(2).

[30]. Fang, Wei, et al. (2012), RPPS: A novel resource prediction and provisioning scheme in cloud data center, *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, IEEE, pp. 609-616.

[31]. Foster, Ian, et al. (2008), Cloud computing and grid computing 360-degree compared, *Grid Computing Environments Workshop, 2008. GCE'08*, Ieee, pp. 1-10.

[32]. Gambi, Alessio, Pezze, Mauro, and Toffetti, Giovanni (2016), "Kriging-based self-adaptive cloud controllers", *IEEE Transactions on Services Computing*. 9(3), pp. 368-381.

[33]. Gambi, Alessio and Toffetti, Giovanni (2012), Modeling cloud performance with kriging, *Software Engineering (ICSE), 2012 34th International Conference on*, IEEE, pp. 1439-1440.

[34]. Gandhi, Anshul, et al. (2014), Adaptive, Model-driven Autoscaling for Cloud Applications, *ICAC*, pp. 57-64.

[35]. Gandhi, Anshul, et al. (2012), "Autoscale: Dynamic, robust capacity management for multi-tier data centers", *ACM Transactions on Computer Systems (TOCS)*. 30(4), p. 14.

[36]. Garg, Shikha, Gupta, DV, and Dwivedi, Rakesh Kumar (2016), Enhanced Active Monitoring Load Balancing algorithm for Virtual Machines in cloud computing, *System Modeling & Advancement in Research Trends (SMART), International Conference*, IEEE, pp. 339-344.

[37]. Ghobaei-Arani, Mostafa, Jabbehdari, Sam, and Pourmina, Mohammad Ali (2018), "An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach", *Future Generation Computer Systems*. 78, pp. 191-210.

[38]. Gong, Zhenhuan, Gu, Xiaohui, and Wilkes, John (2010), Press: Predictive elastic resource scaling for cloud systems, *Network and Service Management (CNSM), 2010 International Conference on*, Ieee, pp. 9-16.

[39]. Grozev, Nikolay and Buyya, Rajkumar (2017), "Dynamic Selection of Virtual Machines for Application Servers in Cloud Environments", *Research Advances in Cloud Computing*, Springer, pp. 187-210.

[40]. Han, Rui, et al. (2014), "Enabling cost-aware and adaptive elasticity of multi-tier cloud applications", *Future Generation Computer Systems*. 32, pp. 82-98.

[41]. Han, Rui, et al. (2012), Lightweight resource scaling for cloud applications, *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, IEEE, pp. 644-651.

[42]. Hasan, Masum Z, et al. (2012), Integrated and autonomic cloud resource scaling, *Network Operations and Management Symposium (NOMS), 2012 IEEE*, IEEE, pp. 1327-1334.

[43]. Herbst, Nikolas Roman (2018), *Methods and Benchmarks for Auto-Scaling Mechanisms in Elastic Cloud Environments*, Universität Würzburg.

[44]. Herbst, Nikolas Roman, Kounev, Samuel, and Reussner, Ralf H (2013), Elasticity in Cloud Computing: What It Is, and What It Is Not, *ICAC*, pp. 23-27.

[45]. Iqbal, Waheed, Dailey, Mathew N, and Carrera, David (2016), "Unsupervised learning of dynamic resource provisioning policies for cloud-hosted multitier web applications", *IEEE Systems Journal*. 10(4), pp. 1435-1446.

[46]. Iqbal, Waheed, et al. (2011), "Adaptive resource provisioning for read intensive multi-tier applications in the cloud", *Future Generation Computer Systems*. 27(6), pp. 871-879.

[47]. Jackson, James R (1957), "Networks of waiting lines", *Operations research*. 5(4), pp. 518-521.

[48]. Jackson, James R (1963), "Jobshop-Like Queueing Systems. Mgmt. Sci. 10, 131-142", *Jackson13110Mgmt. Sci*.

[49]. James, Jasmin and Verma, Bhupendra (2012), "Efficient VM load balancing algorithm for a cloud computing environment", *International Journal on Computer Science and Engineering*. 4(9), p. 1658.

[50]. Jamshidi, Pooyan, Ahmad, Aakash, and Pahl, Claus (2014), Autonomic resource provisioning for cloud-based software, *Proceedings of the 9th international symposium on software engineering for adaptive and self-managing systems*, ACM, pp. 95-104.

[51]. Jamshidi, Pooyan, et al. (2015), "Self-learning cloud controllers: Fuzzy q-learning for knowledge evolution", *arXiv preprint arXiv:1507.00567*.

[52]. Jang, J-SR (1993), "ANFIS: adaptive-network-based fuzzy inference system", *IEEE transactions on systems, man, and cybernetics*. 23(3), pp. 665-685.

[53]. Jin, Yue, et al. (2018), Model-free resource management of cloud-based applications using reinforcement learning, *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, IEEE.

[54]. Jouffe, PY Glorennec and L (1997), Fuzzy Q-learning, *Proceedings of 6th International Fuzzy Systems Conference*, IEEE.

[55]. Kalyvianaki, Evangelia, Charalambous, Themistoklis, and Hand, Steven (2009), Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters, *Proceedings of the 6th international conference on Autonomic computing*, ACM, pp. 117-126.

[56]. Khatua, Sunirmal, Ghosh, Anirban, and Mukherjee, Nandini (2010), Optimizing the utilization of virtual resources in Cloud environment, *Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2010 IEEE International Conference on*, IEEE, pp. 82-87.

[57]. Khazaei, Hamzeh, Misic, Jelena, and Misic, Vojislav B (2012), "Performance analysis of cloud computing centers using m/g/m/m+ r queuing systems", *IEEE Transactions on parallel and distributed systems*. 23(5), pp. 936-943.

[58]. Kikuchi, Shinji and Matsumoto, Yasuhide (2011), Performance modeling of concurrent live migration operations in cloud computing systems using prism probabilistic model checker, *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, IEEE, pp. 49-56.

[59]. Kliazovich, Dzmitry, Bouvry, Pascal, and Khan, Samee Ullah (2012), "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers", *The Journal of Supercomputing*. 62(3), pp. 1263-1283.

[60]. Koperek, Paweł and Funika, Włodzimierz (2011), Dynamic business metrics-driven resource provisioning in cloud environments, *International Conference on Parallel Processing and Applied Mathematics*, Springer, pp. 171-180.

[61]. Kumar, Ankit and Kalra, Mala (2016), Load balancing in cloud data center using modified active monitoring load balancer, *Advances in Computing, Communication, & Automation (ICACCA)(Spring), International Conference on*, IEEE, pp. 1-5.

[62]. Kumar, Y Ranjit, Madhu Priya, M, and Shahu Chatrapati, K (2013), "Effective distributed dynamic load balancing for the clouds", *International Journal of Engineering Research & Technology (IJERT)*. 2(2), pp. 1-6.

[63]. Kupferman, Jonathan, et al. (2009), "Scaling into the cloud", *CS270-advanced operating systems*, p. 39.

[64]. Lama, Palden and Zhou, Xiaobo (2010), Autonomic provisioning with self-adaptive neural fuzzy control for end-to-end delay guarantee, *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on*, IEEE, pp. 151-160.

[65]. Li, Han and Venugopal, Srikumar (2011), Using reinforcement learning for controlling an elastic web application hosting platform, *Proceedings of the 8th ACM international conference on Autonomic computing*, ACM, pp. 205-208.

[66]. Lim, Harold C, Babu, Shivnath, and Chase, Jeffrey S (2010), Automated control for elastic storage, *Proceedings of the 7th international conference on Autonomic computing*, ACM, pp. 1-10.

[67]. Lim, Harold C, et al. (2009), Automated control in cloud computing: challenges and opportunities, *Proceedings of the 1st workshop on Automated control for datacenters and clouds*, ACM, pp. 13-18.

[68]. Lorido-Botran, Tania, Miguel-Alonso, Jose, and Lozano, Jose Antonio (2014), "A review of auto-scaling techniques for elastic applications in cloud environments", *Journal of grid computing*. 12(4), pp. 559-592.

[69]. Lorido-Botrcn, Tania, Miguel-Alonso, Josщ, and Lozano, Jose Antonio (2013), "Comparison of auto-scaling techniques for cloud environments".

[70]. Mahmud, A Hasan, He, Yuxiong, and Ren, Shaolei (2014), BATS: budget-constrained autoscaling for cloud performance optimization, *ACM SIGMETRICS Performance Evaluation Review*, ACM, pp. 563-564.

[71]. Maji, Amiya K, et al. (2014), Mitigating interference in cloud services by middleware reconfiguration, *Proceedings of the 15th International Middleware Conference*, ACM, pp. 277-288.

[72]. Mao, Ming and Humphrey, Marty (2011), Auto-scaling to minimize cost and meet application deadlines in cloud workflows, *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ACM, p. 49.

[73]. Mao, Ming and Humphrey, Marty (2012), A performance study on the VM startup time in the cloud, *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, IEEE, pp. 423-430.

[74]. Mars, Jason, Tang, Lingjia, and Hundt, Robert (2011), "Heterogeneity in "homogeneous" warehouse-scale computers: A performance opportunity", *IEEE Computer Architecture Letters*. 10(2), pp. 29-32.

[75]. Mell, Peter and Grance, Tim (2011), "The NIST definition of cloud computing".

[76]. Menasce, Daniel A, et al. (2004), *Performance by design: computer capacity planning by example*, Prentice Hall Professional.

[77]. Mishra, Ratan and Jaiswal, Anant (2012), "Ant colony optimization: A solution of load balancing in cloud", *International Journal of Web & Semantic Technology*. 3(2), p. 33.

[78]. Mohan, Anju and Shine, S (2013), "Survey on live vm migration techniques", *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*. 2(1), pp. pp: 155-157.

[79]. Nathuji, Ripal, Kansal, Aman, and Ghaffarkhah, Alireza (2010), Q-clouds: managing performance interference effects for qos-aware clouds, *Proceedings of the 5th European conference on Computer systems*, ACM, pp. 237-250.

[80]. Netto, Marco AS, et al. (2014), Evaluating auto-scaling strategies for cloud computing environments, *Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2014 IEEE 22nd International Symposium on*, IEEE, pp. 187-196.

[81]. Nikravesh, Ali Yadavar, Ajila, Samuel A, and Lung, Chung-Horng (2017), "An autonomic prediction suite for cloud resource provisioning", *Journal of Cloud Computing*. 6(1), p. 3.

[82]. Ostermann, Simon, et al. (2010), GroudSim: an event-based simulation framework for computational grids and clouds, *European Conference on Parallel Processing*, Springer, pp. 305-313.

[83]. Padala, Pradeep, et al. (2009), Automated control of multiple virtualized resources, *Proceedings of the 4th ACM European conference on Computer systems*, ACM, pp. 13-26.

[84]. Park, Sang-Min and Humphrey, Marty (2009), Self-tuning virtual machines for predictable escience, *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, IEEE Computer Society, pp. 356-363.

[85]. Patikirikorala, Tharindu and Colman, Alan (2010), Feedback controllers in the cloud, *Proceedings of APSEC*, sn, p. 39.

[86]. Pattanaik, Priyadarashini Adyasha, Roy, Sharmistha, and Pattnaik, Prasant Kumar (2015), Performance study of some dynamic load balancing algorithms in cloud computing environment, *Signal processing and integrated networks (SPIN), 2015 2nd International Conference on*, IEEE, pp. 619-624.

[87]. Pleshakov, Michael (2017), Load Balancing AWS Auto Scaling Groups With NGINX Plus, Editor^Editors.

[88]. Qu, Chenhao, Calheiros, Rodrigo N, and Buyya, Rajkumar (2016), "Auto-scaling web applications in clouds: A taxonomy and survey", *arXiv preprint arXiv:1609.09224*.

[89]. Rao, Jia, et al. (2009), VCONF: a reinforcement learning approach to virtual machines auto-configuration, *Proceedings of the 6th international conference on Autonomic computing*, ACM, pp. 137-146.

[90]. Roy, Nilabja, Dubey, Abhishek, and Gokhale, Aniruddha (2011), Efficient autoscaling in the cloud using predictive models for workload forecasting, *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, IEEE, pp. 500-507.

[91]. Sahni, Jyoti and Vidyarthi, Deo Prakash (2017), "Heterogeneity-aware adaptive auto-scaling heuristic for improved QoS and resource usage in cloud environments", *Computing*. 99(4), pp. 351-381.

[92]. Salah, Khaled (2013), "A queuing model to achieve proper elasticity for cloud cluster jobs", *International Journal of Cloud Computing*. 1, pp. 53-64.

[93]. Salah, Khaled, Elbadawi, Khalid, and Boutaba, Raouf (2016), "An analytical model for estimating cloud resources of elastic services", *Journal of Network and Systems Management*. 24(2), pp. 285-308.

[94]. Sharma, Upendra, et al. (2011), A cost-aware elasticity provisioning system for the cloud, *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, IEEE, pp. 559-570.

[95]. Shortle, John F, et al. (2018), *Fundamentals of queueing theory*, Vol. 399, John Wiley & Sons.

[96]. Siar, Hajar, Kiani, Kourosh, and Chronopoulos, Anthony T (2015), "An effective game theoretic static load balancing applied to distributed computing", *Cluster Computing*. 18(4), pp. 1609-1623.

[97]. Sutton, Richard S and Barto, Andrew G (1998), *Reinforcement learning: An introduction*, MIT press.

[98]. Tesauro, Gerald (2005), Online resource allocation using decompositional reinforcement learning, *AAAI*, pp. 886-891.

[99]. Tesauro, Gerald, et al. (2006), A hybrid reinforcement learning approach to autonomic resource allocation, *Autonomic Computing, 2006. ICAC'06. IEEE International Conference on*, IEEE, pp. 65-73.

[100]. Urgaonkar, Bhuvan, et al. (2005), An analytical model for multi-tier internet services and its applications, *ACM SIGMETRICS Performance Evaluation Review*, ACM, pp. 291-302.

[101]. Urgaonkar, Bhuvan, et al. (2008), "Agile dynamic provisioning of multi-tier internet applications", *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*. 3(1), p. 1.

[102]. Vecchiola, Christian, Pandey, Suraj, and Buyya, Rajkumar (2009), High-performance cloud computing: A view of scientific applications, *Pervasive*

Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on, IEEE, pp. 4-16.

[103]. Vilaplana, Jordi, et al. (2014), "A queuing theory model for cloud computing", *The Journal of Supercomputing*. 69(1), pp. 492-507.

[104]. Villela, Daniel, Pradhan, Prashant, and Rubenstein, Dan (2007), "Provisioning servers in the application tier for e-commerce systems", *ACM Transactions on Internet Technology (TOIT)*. 7(1), p. 7.

[105]. Wang, Lixi, et al. (2011), Adaptive virtual resource management with fuzzy model predictive control, *Proceedings of the 8th ACM international conference on Autonomic computing*, ACM, pp. 191-192.

[106]. Wang, Lixi, et al. (2011), Fuzzy modeling based resource management for virtualized database systems, *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*, IEEE, pp. 32-42.

[107]. Wang, Zhikui, et al. (2009), "AppRAISE: application-level performance management in virtualized server environments", *IEEE Transactions on Network and Service Management*. 6(4), pp. 240-254.

[108]. Weihua, Gong and Yuanzhen, Wang (2006), "A new load balancing scheme on heterogeneous database cluster", *Geo-spatial Information Science*. 9(3), pp. 216-222.

[109]. Xiong, Kaiqi and Perros, Harry (2009), Service performance and analysis in cloud computing, *Services-I, 2009 World Conference on*, IEEE, pp. 693-700.

[110]. Xu, Jing, et al. (2007), On the use of fuzzy modeling in virtualized data center management, *Autonomic Computing, 2007. ICAC'07. Fourth International Conference on*, IEEE, pp. 25-25.

[111]. Yazdanov, Lenar and Fetzer, Christof (2013), Vscaler: Autonomic virtual machine scaling, *2013 IEEE Sixth International Conference on Cloud Computing*, IEEE, pp. 212-219.

[112]. Zhang, Qi, Cherkasova, Ludmila, and Smirni, Evgenia (2007), A regression-based analytic model for dynamic resource provisioning of multi-tier

applications, *Autonomic Computing, 2007. ICAC'07. Fourth International Conference on*, IEEE, pp. 27-27.

[113]. Zhang, Ying, et al. (2010), Integrating resource consumption and allocation for infrastructure resources on-demand, *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, IEEE, pp. 75-82.

[114]. Zhu, Q. and Agrawal, G. (2012), "Resource Provisioning with Budget Constraints for Adaptive Applications in Cloud Environments", *IEEE Transactions on Services Computing*. 5(4), pp. 497-511.