Một công cụ đo kiểm giao thức MQTT cho những ứng dụng IoT phục vụ cho việc ra quyết định co dãn tài nguyên trên đám mây

Phạm Mạnh Linh
Trung tâm Công nghệ tích hợp liên ngành
Giám sát hiện trường (FIMO)
Trường Đại học Công Nghệ, ĐHQGHN
Hà Nội, Việt Nam
Email: linhmp@vnu.edu.vn

Trần Mạnh Đông, Nguyễn Trường Thắng
Viện Công nghệ thông tin
Viện Hàn lâm Khoa học và Công nghệ Việt Nam
Hà Nội, Việt Nam
Email: {dongtm, ntthang}@ioit.ac.vn

***Tóm tắt nội dung*—Trong sự phát triển của Internet vạn vật (IoT), giao thức MQTT (Message Queuing Telemetry Transport) có một vai trò rất quan trọng bởi lẽ nó được sử dụng rộng rãi bởi những ứng dụng IoT thường được triển khai theo mô hình "Xuất bản-Đăng ký". Với khả năng cung ứng và giải phóng những tài nguyên ảo và mịn (chức năng co dãn - elasticity), điện toán đám mây có khả năng làm gia tăng sức mạnh cho những máy chủ môi giới MQTT. Khả năng co dãn giúp các máy chủ môi giới MQTT có thể đương đầu với một khối lượng lớn dữ liệu đang từng ngày được tích hợp vào IoT. Tuy nhiên vẫn có sự thiếu vắng của các công cụ đo kiểm có thể đánh giá đầy đủ tất cả các khía cạnh của MQTT để hỗ trợ việc ra những quyết định co dãn tài nguyên hợp lý và chính xác. Bài báo này tập trung vào chủ đề đo kiểm MQTT thông qua việc giới thiệu MQTTBench một công cụ kiểm thử mới được chúng tôi phát triển cùng với những kết quả ban đầu có được từ việc áp dụng công cụ đo kiểm đó trên một ứng dụng điện toán đám mây giả lập.**

***Keywords*-đo kiểm; điện toán đám mây; Internet vạn vật; MQTT;**

I. Giới thiệu

Internet vạn vật (IoT) đang trở thành sự thật với sự sẵn sàng của những thiết bị truyền thông chi phí thấp ví dụ như các thẻ RFID, cảm biến không dây hay các thiết bị di động thông minh. IoT mở ra cho các công ty cơ hội ứng dụng những mô hình kinh tế mới (ví dụ như trả theo nhu cầu) để cải thiện chất lượng dịch vụ cung cấp cho khách hàng của họ mà vẫn đáp ứng được nhu cầu về trách nhiệm hợp đồng và pháp lý. Mô hình giao tiếp được sử dụng phổ biến trong các kiến trúc IoT là "Xuất bản-Đăng ký" [1] (viết ngắn gọn là PubSub) dùng để phân phối các thông điệp đo lường được tập hợp từ các nút cảm biến và được đưa tới những ứng dụng chuyên sâu cụ thể. Giao thức MQTT (Message Queuing Telemetry Transport) [2] là triển khai được sử dụng phổ biến nhất của mô hình PubSub trong ngữ cảnh IoT nói chung và trong ngữ cảnh của công nghệ Máy tới Máy (M2M) nói riêng. Với MQTT thông điệp được gửi và điều hướng bởi các chủ đề (topics) chứ không phải bởi các máy chủ. Một thách thức đối với sự kết hợp giữa MQTT và IoT đó là khả năng cung cấp dịch vụ với năng lực quản lý tài nguyên động (tài nguyên sẽ được cung ứng hoặc giải phóng dựa trên nhu cầu thực tế). Bởi vậy rất cần thiết để định nghĩa khái niệm "chủ đề co dãn" có thể xử lý điều khiển các kết nối một cách tối ưu. Để định nghĩa ra các ngưỡng co dãn mà các máy chủ môi giới MQTT có thể hỗ trợ trong thực tiễn, rất cần thiết để thực hiện các bài kiểm tra hiệu năng MQTT.

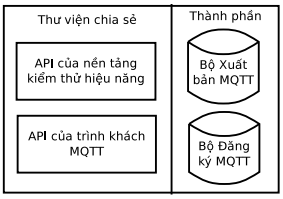
Trong bài báo này, chúng tôi thực hiện những công việc sau: (1) giới thiệu MQTTBench, một công cụ kiểm thử cho phép thực hiện rất nhiều các bài kiểm tra hiệu năng MQTT; (2) cung cấp một cách tiếp cận để phân tích kết quả thống kê bằng cách so sánh các định nghĩa lý thuyết của MQTT để tìm ra các ngưỡng co dãn MQTT. Phần còn lại của bài báo được sắp xếp như sau: Sau khi nêu ra các nghiên cứu có liên quan tới công việc của chúng tôi ở Phần II, kiến trúc và các thành phần của công cụ chúng tôi phát triển được mô tả chi tiết trong Phần III. Phần IV mô tả các thí nghiệm mang tính hợp lệ hóa và trao đổi về những kết quả có được từ các kịch bản khác nhau tương ứng với các triển khai thực tiễn trong ngữ cảnh IoT. Sau cùng, bài báo kết luận và đưa ra những nghiên cứu tiềm năng tương lai trong Phần V.

II. Các nghiên cứu liên quan

Mạng lưới liền mạch và linh hoạt của các đối tượng thường ngày sẽ trở thành miền ứng dụng quan trọng cho truyền thông dựa trên Internet, với tầm nhìn hướng về IoT. Hiệu năng của những hệ thống M2M là một nhân tố quan trọng, đặc biệt với những yêu cầu thời gian thực. Trong một vài hệ M2M việc sử dụng pin và năng lượng hiệu quả là một mối quan tâm lớn ví dụ như trong mạng cảm biến không dây. Tuy nhiên thiết lập một môi trường đánh giá hiệu năng cho hệ thống M2M rất phức tạp và khá thách thức đối với lập trình viên. Bởi vậy nhu cầu về một công cụ hiệu quả và thân thiện cho kiểm thử MQTT là cấp thiết và dưới đây là một vài nghiên cứu có liên quan tới cách tiếp cận của chúng tôi.

Durkop đã tiến hành những thí nghiệm để so sánh 3 giao thức M2M là CoAP, MQTT và OPC UA có liên quan đến cơ chế giao vận của chúng để đánh giá thời gian truyền và phân tích những tiềm năng cho tối ưu hóa [10]. Bằng cách sử dụng những thành phần trung gian (middleware), một nhóm nghiên cứu từ trường Đại học quốc gia Singapore đã thực hiện những thí nghiệm đánh giá hiệu năng của MQTT và CoAP với các đại lượng về độ trễ đầu cuối và tiêu thụ băng thông [11]. Một số ít công ty nghiên cứu như Ekito và Scalagent [12], [13] đã tiến hành việc đo kiểm MQTT. Tuy nhiên, những công ty này đã sử dụng mã nguồn tự phát triển của riêng họ để làm những việc này.

Có một vài công cụ đo kiểm phổ biến như JMeter, CLIF Server, Tsung, hay Gatling. Chúng đều được thiết kế để dễ dàng sử dụng với khả năng bảo trì và hiệu năng cao. Giống như MQTTBench, chúng cung cấp một nền tảng cho phép thực hiện các bài test về tải cho một vài kiểu máy chủ và có khả năng giả lập các bài test hiệu năng. Nhưng đối với các giao thức chuyển nhận thông điệp sử dụng hàng đợi, tất cả những công cụ này chỉ hỗ trợ JMS. Chỉ có một công cụ để thực hiện đo kiểm MQTT trên Internet là SmartBear [14], nhưng nó không phải là mã nguồn mở và chỉ đo kiểm được một vài chức năng cơ bản của giao thức MQTT. Trong giới hạn kiến thức của chúng tôi, hiện nay không có công cụ mã nguồn mở nào đo kiểm MQTT cho các ứng dụng IoT trên môi trường đám mây. Bằng việc phát triển trình cắm MQTTBench, chúng tôi đã lấp đầy khoảng trống tồn tại trong thế giới IoT về một công cụ đo kiểm linh hoạt cho giao thức được sử dụng phổ biến là MQTT. MQTTBench được phát triển mở rộng từ JMeter, bởi vậy tất cả người sử dụng JMeter có thể trải nghiệm việc đo kiểm cả JMS và MQTT với công cụ quen thuộc của họ. Thêm vào đó, những nhóm nghiên cứu khác nhau có một công cụ dùng chung, thân thiện, và linh hoạt để đánh giá đầy đủ các khía cạnh của MQTT. Như một hệ quả, sẽ dễ dàng hơn khi so sánh các kết quả từ các nhóm này với nhau.



Hình 1: Kiến trúc phần mềm tổng thể của MQTTBench

III. MQTTBench

MQTTBench [1] là một nền tảng kiểm thử MQTT phân tán cho những ứng dụng MQTT tuân theo những nguyên lý chính của kiến trúc dựa trên thành phần. Nó được thiết kế để tái sử dụng những thành phần dựng sẵn (COTS: component-off-the-shelf) và kết hợp chúng lại một cách linh hoạt. Bởi vậy, lõi của MQTTBench được giữ gọn nhẹ với những hoạt động của API được đơn giản hóa như là những điểm neo để gắn thêm trình cắm (plug-in) cụ thể tới những thành phần dựng sẵn. Hình 1 chỉ ra kiến trúc phần mềm tổng thể của MQTTBench bao gồm sự kết hợp của một vài mô đun.

A. API của Nền Tảng Kiểm Thử Hiệu Năng

API của nền tảng kiểm thử hiệu năng được triển khai trên Java để tạo và chèn các hồ sơ tải vào những hệ thống khác nhau và để đo lường hiệu năng. Ban đầu nó được thiết kế để kiểm thử MQTT trong những ứng dụng IoT nhưng đã được mở rộng để bao gồm những chức năng kiểm thử khác nữa. Chúng tôi đã phát triển một trình cắm cho mô đun này để gắn vào hệ thống Apache JMeter [3], một phần mềm đo kiểm mã nguồn mở hỗ trợ tính năng di chuyển gọn nhẹ một cách tự nhiên. JMeter cung cấp khả năng thực hiện các bài test với tải lớn và có thể test hiệu năng với nhiều kiểu dịch vụ và giao thức khác nhau (ví dụ như Web - HTTP, HTTPS, SOAP, FTP, Database với JDBC, Message-Oriented Middleware với JMS). Nó cũng hỗ trợ xử lý đa luồng để tăng tốc cho những bài test. Những phần mềm cạnh tranh của JMeter có thể nói đến CLIF

[1]https://github.com/fimocode/mqttbench

Server [8] hay Gatling [9]. Tuy nhiên, chúng đều thiếu một bộ chèn tải cho giao thức MQTT nói riêng và đây chính là điểm đóng góp của chúng tôi muốn giới thiệu trong bài báo này.

Tầm quan trọng của giao thức MQTT trong thế giới IoT cũng ngang như của giao thức HTTP trong thế giới Web. Thực vậy, chúng ta đã có những công cụ rất mạnh mẽ cho việc đo kiểm HTTP, những gì chúng ta thiếu là một công cụ mạnh tương tự để đo kiểm MQTT.

B. API của Trình Khách MQTT cho Cả Bên Xuất Bản và Đăng Ký

Chúng tôi đã phát triển giao diện cho công cụ bên phía khách dựa trên cả giao diện đồ họa (GUI) và dòng lệnh (CLI) được cung cấp bởi JMeter. Bởi vì các ứng dụng là không giống nhau, kịch bản test của mỗi ứng dụng phải được tùy chỉnh để đo lường hiệu năng lưu lượng của thế giới thực [6]. Chúng ta có thể dùng giao diện GUI để phát triển các trường hợp test và sử dụng giao diện CLI để thực hiện các bài test thực sự.

Mô đun MQTT bên máy khách gồm 2 phần: một bộ Xuất bản đại diện cho máy khách gửi dữ liệu tới những chủ đề trong máy chủ môi giới, và một bộ Đăng ký đại diện cho máy khách nhận dữ liệu từ máy chủ môi giới tương ứng với những chủ đề cụ thể mà bộ Đăng ký đó đã thuê bao.

Với bộ Xuất bản MQTT, chúng ta có thể thực hiện rất nhiều những kịch bản test nhờ vào các tùy chọn đa dạng của nó, nhờ vậy mà tất cả khía cạnh của giao thức có thể được đánh giá. Bộ Xuất bản gồm 4 mô đun nhỏ: Thông tin kết nối, Mã hóa, Tùy chọn và Nội dung. Mô đun "Thông tin kết nối" cho phép nhập vào định danh máy khách và một tập hợp của các chủ đề mà chúng ta muốn xuất bản tới. Nó cũng cung cấp khả năng xác thực, xóa phiên (máy chủ không nhớ máy khách của nó sau khi ngắt kết nối) và hai chiến lược xuất bản khác nhau: Ngẫu nhiên (Random) hoặc Luân phiên (Round Robin). Với chiến lược ngẫu nhiên, máy khách xuất bản dữ liệu tới những chủ đề ngẫu nhiên trong vòng tập hợp các chủ đề đã nhập, ngược lại, với chiến lược luân phiên, máy khách xuất bản dữ liệu tới những chủ đề với tần suất bằng nhau và theo thứ tự xoay vòng. Mô đun "Mã hóa" cho phép gửi thông điệp dưới các dạng như không được mã hóa, nhị phân, base64, binhex và văn bản thô với nhiều kiểu nén khác nhau. Mô đun "Tùy chọn" cung cấp khả năng gửi thông điệp duy trì và không duy trì, thêm nhãn thời gian hoặc chuỗi số tới thông điệp. Nó cho phép điều khiển ba cấp độ của chất lượng dịch vụ: Nhiều nhất một lần (0), Ít nhất một lần (1), và Chính xác đúng một lần (2). Mô đun "Nội dung" cung cấp 4 kiểu thông điệp để được xuất bản: một đoạn văn bản, một giá trị được tạo ra nằm trong một khoảng tùy biến, một giá trị cố định, hoặc một mảng kiểu byte ngẫu nhiên. Bốn kiểu này của thông điệp thích ứng với rất nhiều các kịch bản test khác nhau, từ những kiểu test đơn giản như gửi một đoạn văn bản tới những kịch bản test phức tạp hơn như những thông điệp đại diện dữ liệu cảm biến được gửi với dung lượng nhỏ nhưng với số lượng lớn.

Bộ Đăng ký MQTT bao gồm hai mô đun nhỏ: Thông tin kết nối và Tùy chọn cung cấp những thuộc tính giống nhau cho kết nối của bên Xuất bản và Đăng ký. FuseSource API cho trình khách MQTT [2] cũng được sử dụng bởi vì nó cung cấp API theo giấy phép ASL 2.0 (Apache Software License) và nó có thể thực hiện được việc tự động tái kết nối tới máy chủ MQTT và khôi phục lại những phiên kết nối phía máy khách nếu mất điện trên mạng lưới xảy ra.

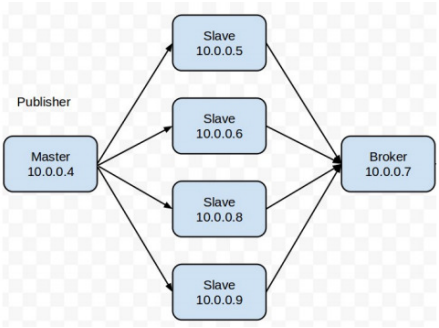
IV. Đánh giá

A. Môi Trường Kiểm Thử

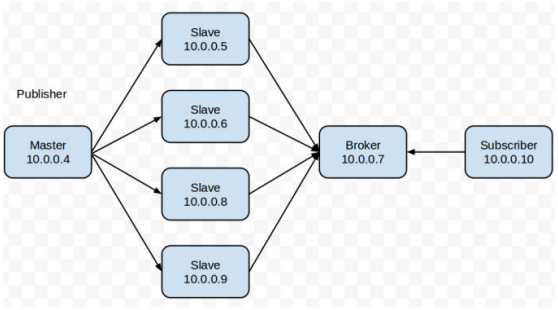
Dữ liệu IoT giả lập được tạo ra bằng cách cấu hình mô đun "Nội dung" bên bộ xuất bản MQTT. Trong các thí nghiệm này, các bộ xuất bản MQTT đóng vai trò của các thiết bị cửa ngõ (gateway) giữa các cảm biến IoT và các máy chủ ứng dụng trong môi trường đám mây. Dữ liệu này sau khi đi qua qua các thiết bị cửa ngõ sẽ được đưa tới các máy chủ môi giới MQTT. Môi trường đám mây được sử dụng trong các thí nghiệm là Windows Azure với giấy phép sử dụng 32 CPU. Dịch vụ máy ảo (VM) của Azure có khả năng cung cấp tài nguyên điện toán co dãn theo nhu cầu. Một máy ảo Azure là một máy chủ trên đám mây mà chúng ta cấu hình và duy trì tương ứng với nhu cầu sử dụng. Trong các bài test của chúng tôi, mỗi VM đều được cài đặt hệ điều hành Ubuntu 16.04 LTS và các VMs này được triển khai chung trên một mạng ảo.

Chúng tôi đã sử dụng Mosquitto [5] với vai trò là một máy chủ môi giới MQTT. Mosquitto là một máy chủ môi giới MQTT miễn phí triển khai giao thức MQTT 3.1. Mosquitto cung cấp cấu hình ban đầu cơ bản vì vậy sau khi cài đặt nó phải được cấu hình tùy biến cho những chức năng tương ứng mà chúng ta muốn đo kiểm. Như đã đề cập ở Phần II,

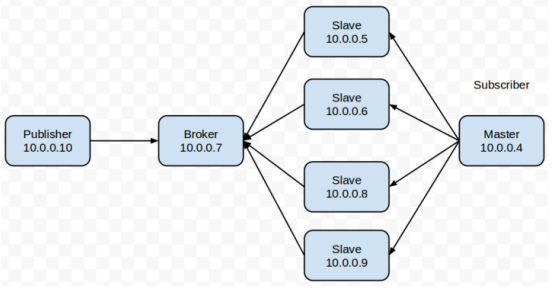
[2]https://github.com/fusesource/mqtt-client



Hình 2: Kịch bản có nhiều bộ Xuất bản - Không có bộ Đăng ký



Hình 3: Kịch bản có nhiều bộ Xuất bản - Chỉ có một bộ Đăng ký



Hình 4: Kịch bản có một bộ Xuất bản - Nhiều bộ Đăng ký

hiện giờ không có công cụ mã nguồn mở nào đo kiểm MQTT cho các ứng dụng IoT trên môi trường đám mây. Vì vậy chúng tôi xây dựng các kịch bản test khác nhau chạy trên MQTTBench để hợp lệ hóa về mặt chức năng của công cụ này.

B. Kịch Bản

Trong những thí nghiệm của chúng tôi, có 75 kế hoạch test đã được tạo ra đại diện ít nhất 56.25 giờ thí nghiệm trên đám mây Azure. Sau đây là 3 kịch bản test đại diện cho các kiểu chuyên biệt của giao thức MQTT.

1) Nhiều bộ Xuất bản - Không có bộ Đăng ký: Kịch bản này triển khai trường hợp có rất nhiều bộ Xuất bản sản xuất dữ liệu mà không có sự tồn tại của một bộ Đăng ký nào để tiêu thụ dữ liệu. Nó phân tích chi phí của việc xuất bản được sử dụng cho kịch bản thứ hai dưới đây (xem Hình 2).

2) Nhiều bộ Xuất bản - Một bộ Đăng ký: Kịch bản này triển khai nhiều bộ Xuất bản với chỉ một bộ Đăng ký. Điều này tương ứng với trường hợp có sự tham gia với số lượng lớn của các cảm biến hoặc nguồn đám đông (crowdsourcing) ví dụ như các điện thoại di động thông minh. Trong trường hợp này các dữ liệu đo được gửi tới chỉ một hệ thống IT trung tâm duy nhất.

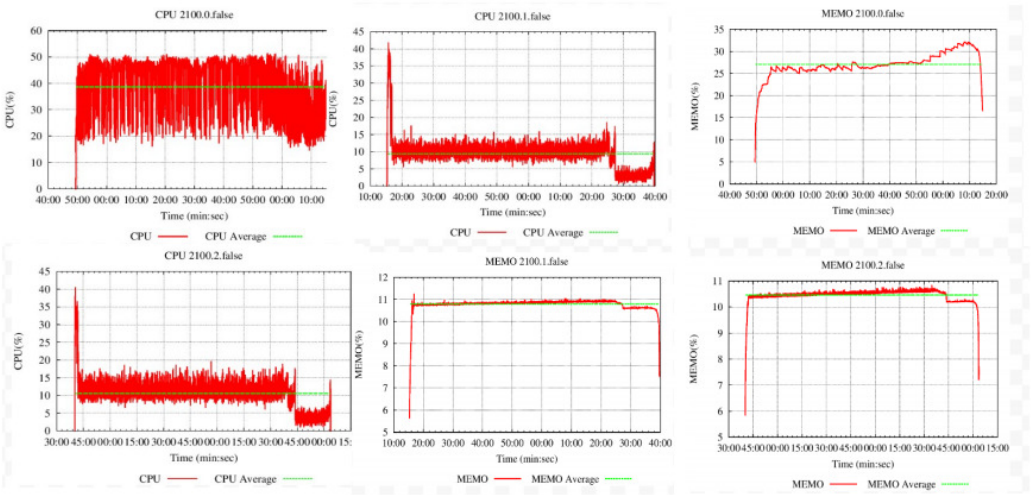
3) Một bộ Xuất bản - Nhiều bộ Đăng ký: Kịch bản này sử dụng một bộ Xuất bản và một số lượng lớn các bộ Đăng ký. Điều này tương ứng với trường hợp đăng ký nhận thông báo thời gian thực được gửi đi bởi một hệ thống IT trung tâm của một số lượng lớn điện thoại di động thông minh (xem Hình 4).

Để đạt được một số lượng lớn máy khách song song, MQTTBench được sử dụng theo cách phân tán, tức là một máy chủ chính (master) điều khiển bốn máy chủ phụ (slave) và tất cả chúng đều là máy chủ MQTTBench. Kết quả test được phục hồi để thống kê bởi những bộ tổng hợp MQTTBench. Các mã kịch bản dạng Bash được cài đặt để đo lường các đại lượng đo trên tất cả các máy tham gia vào kịch bản test. Chúng được cài đặt trong mỗi máy để điều khiển Sysstat [4] từ xa qua SSH.

C. Kết Quả và Thảo Luận

Trong phần này, chúng tôi thảo luận chi tiết về kết quả của hai kịch bản test là "Nhiều bộ Xuất bản - Không có bộ Đăng ký" và "Nhiều bộ Xuất bản - Chỉ có một bộ Đăng ký". Kịch bản còn lại được lược bỏ do giới hạn không gian của bài báo này.

1) Nhiều bộ Xuất bản - Không có bộ Đăng ký: Hình 5 chỉ ra một ví dụ trong trường hợp có 1 máy chủ chính (là thực thể máy ảo Azure mã A3 có cấu hình với 4 lõi, CPU 1.6GHz và 7GB RAM), 4 máy chủ phụ (với cùng cấu hình như máy chủ chính), 1 máy chủ môi giới Mosquitto (A2, 2 lõi, CPU 1.6GHz, 3.5 GB RAM). Số lượng của luồng trong mỗi máy chủ phụ là 2100. Mỗi luồng thực hiện 5 kết nối, bởi vậy tổng số kết nối là 42000. Dòng chữ "2100.0.false" có nghĩa là số lượng luồng của mỗi máy chủ phụ là 2100 với chất lượng dịch vụ là 0 và từ "false" ngầm chỉ rằng thông điệp không phải kiểu duy trì. Nhìn vào đồ thị chúng ta có thể



Hình 5: Tải CPU và bộ nhớ sử dụng của máy chủ môi giới MQTT trong kịch bản Nhiều bộ Xuất bản - Không có bộ Đăng ký

kết luận rằng với 3 cấp độ của chất lượng dịch vụ (QoS):

- Máy chủ môi giới Mosquitto tiêu thụ CPU theo thứ tự: QoS 0 $>$ QoS 1 $\approx$ QoS 2.
- Máy chủ môi giới Mosquitto tiêu thụ bộ nhớ theo thứ tự: QoS 0 $>$ QoS 1 $>$ QoS 2.

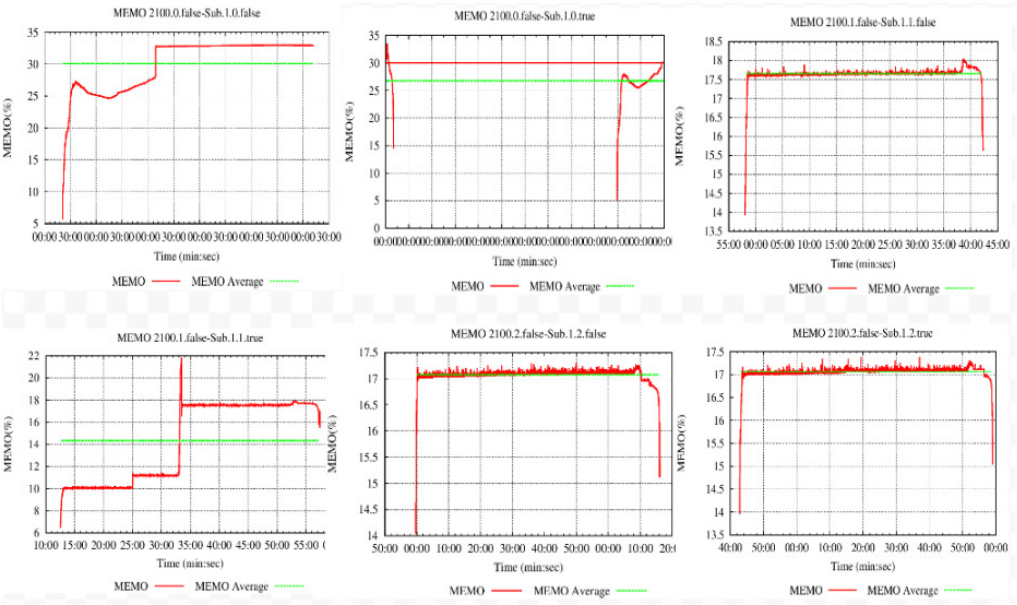
Điều này được giải thích bằng cách định nghĩa về QoS trong MQTT. Với QoS 0, bộ Xuất bản gửi các thông điệp mà không phải đợi sự xác nhận từ phía máy chủ môi giới, thông điệp gửi đi với số lượng lớn nhất đồng nghĩa với việc máy chủ môi giới phải xử lý nhanh nhất và bởi vậy tiêu thụ CPU là lớn nhất. Mặt khác, với QoS 1 và 2, đồ thị ổn định hơn, số lượng thông điệp gửi đi ổn định hơn bởi vậy kết quả đáng tin cậy hơn. Chúng ta cũng tìm ra các điểm bão hòa (tại đỉnh đồ thị) và sử dụng chúng để thiết lập các ngưỡng ra quyết định cho các cỗ máy thực hiện việc co dãn tài nguyên như được mô tả trong [7].

2) Nhiều bộ Xuất bản - Chỉ có một bộ Đăng ký: Trong Hình 6, dòng chữ "2100.0.false-Sub.1.0.false" chỉ ra số lượng của luồng trên mỗi máy chủ phụ là 2100 với chất lượng dịch vụ là 0. Từ "false" ngầm chỉ rằng thông điệp không phải kiểu duy trì. "Sub" là bộ Đăng ký với chất lượng dịch vụ của bộ này là 0 và tham số xóa phiên được chỉ ra là "false".

Nhìn vào đồ thị, chúng ta nhận ra rằng trong kịch bản "Nhiều bộ Xuất bản - Chỉ có một bộ Đăng ký", máy chủ môi giới Mosquitto phải xử lý nhiều hơn để gửi thông điệp tới bộ Đăng ký có thuê bao những thông điệp này. Bởi vậy cả hai đại lượng tải CPU và bộ nhớ được tiêu thụ nhiều hơn khi so sánh với kịch bản "Nhiều bộ Xuất bản - Không có bộ Đăng ký". Trong cả hai kịch bản, QoS 2 đem lại những kết quả tin cậy hơn cả và những đồ thị của cấp độ QoS này ổn định hơn hẳn. Đồ thị tải CPU chỉ ra cùng xu hướng với đồ thị bộ nhớ và bởi vậy bị lược bỏ để tiết kiệm không gian bài báo.

V. Kết luận

Điện toán đám mây đang ngày càng trở nên phổ biến và quan trọng. Trong cuộc cách mạng về IT này, việc tìm hiểu về giao thức MQTT là đặc biệt cần thiết bởi vì nó làm tăng hiệu năng của các ứng dụng IoT giúp kết nối hàng tỷ các thiết bị lại với nhau ở một mức độ chúng ta chưa từng thấy. Đóng góp của chúng tôi là sự phát triển MQTTBech, một công cụ để đo kiểm giao thức MQTT có thể được sử dụng trên những máy chủ môi giới mới nhất. Bằng cách sử dụng MQTTBench, chúng tôi đã giả lập thành công một trường hợp với 42000 kết nối MQTT tới một máy chủ môi giới Mosquitto sử dụng chỉ 7 máy ảo trên nền tảng điện toán đám mây Windows Azure. Kết quả cho thấy MQTTBench hoàn toàn hiệu quả và phù hợp với nhu cầu đo kiểm giao thức MQTT với cả những bài test hiệu năng và bài test về tải lớn để tìm ra điểm bão hòa cho các cỗ máy điều khiển co dãn. Điều này sẽ giúp điều khiển và kiểm soát số lượng và hiệu năng của các máy chủ môi giới MQTT cả về mặt cung ứng và điều chỉnh tối ưu khi cần.



Hình 6: Bộ nhớ của máy chủ môi giới MQTT trong kịch bản Nhiều bộ Xuất bản - Chỉ có một bộ Đăng ký

CẢM TẠ

Nghiên cứu này được thực hiện trong phạm vi các đề tài khoa học mã số CS19.19 của Viện Công nghệ thông tin và GUST.STS.ĐT2019-TT02 của Học viện Khoa học và Công nghệ, Viện hàn lâm KHCN Việt Nam và được hỗ trợ bởi Ban Quản lý dự án FIRST, Bộ Khoa học và Công nghệ thông qua Thỏa thuận tài trợ Số 12/FIRST/2a/IoIT. Nghiên cứu cũng được hỗ trợ bởi dự án mã số VNU QMT.17.03 của Đại học Quốc gia Hà Nội.

TÀI LIỆU

[1] Birman, K. and Joseph, T. "Exploiting virtual synchrony in distributed systems" in Proceedings of the eleventh ACM Symposium on Operating systems principles (SOSP '87), 1987. pp. 123–138.

[2] MQTT, 'http://mqtt.org/," visited on April 2019.

[3] Apache JMeter, 'http://jmeter.apache.org/," visited on April 2019.

[4] Sysstat, 'http://sebastien.godard.pagesperso-orange.fr/," visited on April 2019.

[5] Mosquitto, 'http://mosquitto.org/," visited on April 2019.

[6] E. Cecchet, V. Udayabhanu, T. Wood, and P. Shenoy, "BenchLab: An Open Testbed for Realistic Benchmarking of Web Applications," in USENIX WebApps, 2011, pp. 4–4.

[7] L. M. Pham and T. M. Pham, "Autonomic fine-grained migration and replication of component-based applications across multi-clouds," 2015 2nd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS), HCM City, 2015, pp. 5-10.

[8] CLIF Server, 'http://clif.ow2.org," visited on April 2019.

[9] Gatling, 'http://gatling.io," visited on April 2019.

[10] Durkop, L.; Czybik, B.; Jasperneite, J., "Performance evaluation of M2M protocols over cellular networks in a lab environment," ICIN, pp.70,75, 17-19 Feb. 2015.

[11] Thangavel, D.; Xiaoping Ma; Valera, A.; Hwee-Xian Tan; Tan, C.K.-Y., "Performance evaluation of MQTT and CoAP via a common middleware," IEEE ISSNIP, pp.1,6, 21-24 April 2014.

[12] Giuliani, A., "MQTT benchmarks with RabbitMQ & ActiveMQ," June 2014. [Online]. Available: http://www.ekito.fr/people/mqtt-benchmarks-rabbitmq-activemq/.

[13] Scalagent, "Benchmark of MQTT servers," Jan 2015. [Online]. Available: www.scalagent.com/IMG/pdf/Benchmark_MQTT_servers-v1-1.pdf.

[14] SmartBear, "MQTT Test Steps," [Online]. Available: https://smartbear.com/plugins/mqtt-test-steps-page/, visited on April 2019.