

XÂY DỰNG HỆ THỐNG DỰ BÁO LƯU LƯỢNG TRUY CẬP BẰNG HỌC MÁY CHO CƠ CHẾ AUTO-SCALING TỐI ƯU TÀI NGUYÊN

Nhóm thực hiện: Vũ Lê Hiếu Khanh, Phạm Gia Nguyên, Đỗ Minh Khoa, Hoàng Minh Hiếu

February 4, 2026

Tóm tắt— Báo cáo này trình bày quy trình nghiên cứu và xây dựng mô hình học máy (Machine Learning) nhằm giải quyết bài toán dự báo lưu lượng truy cập (Traffic Forecasting). Kết quả dự báo đóng vai trò cốt lõi trong cơ chế Autoscaling chủ động (Proactive Autoscaling), cho phép hệ thống tự động điều chỉnh tài nguyên tính toán dựa trên nhu cầu thực tế trước khi tải tăng đột biến. Mục tiêu trọng tâm của đề tài là đảm bảo cam kết chất lượng dịch vụ (SLA) và độ sẵn sàng cao, đồng thời tối ưu hóa chi phí vận hành trên hạ tầng đám mây.

1. MÔ TẢ BÀI TOÁN

1.1. Tổng quan về bài toán

Trong bối cảnh điện toán đám mây hiện đại, việc quản trị tài nguyên đối mặt với thách thức lớn về sự cân bằng giữa hiệu năng hệ thống và chi phí vận hành. Phương pháp cấp phát tài nguyên cố định truyền thống thường dẫn đến hai hệ lụy cực đoan: gây lãng phí ngân sách khi lưu lượng truy cập thấp hoặc dẫn đến tình trạng sập hệ thống (downtime) khi nhu cầu người dùng tăng đột biến. Bài toán nghiên cứu tập trung vào việc xây dựng hệ thống phân tích nhật ký truy cập (log analysis) ứng dụng học máy để thực hiện Autoscaling — cơ chế tự động điều chỉnh quy mô máy chủ dựa trên nhu cầu thực tế. Đây là một giải pháp tối ưu hóa kinh tế quan trọng, giúp doanh nghiệp duy trì chất lượng dịch vụ đồng thời tối thiểu hóa chi phí hạ tầng [1].

Dữ liệu đầu vào (Input) của bài toán bao gồm tập nhật ký truy cập HTTP khổng lồ được thu thập trong vòng hai tháng từ máy chủ WWW của một doanh nghiệp. Mỗi bản ghi nhật ký chứa các trường thông tin thô như địa chỉ Host, dấu thời gian (Timestamp), nội dung yêu cầu (Request), mã phản hồi (HTTP Reply Code) và dung lượng dữ liệu (Bytes). Kết quả đầu ra (Output) kỳ vọng là một mô hình hồi quy có khả năng dự báo chính xác các giá trị số thực về lưu lượng truy cập trong tương lai. Kết quả đóng vai trò là tham số đầu vào cho thuật toán

Autoscaling để quyết định số lượng máy chủ cần thiết.

Để đảm bảo độ tin cậy, quá trình phân tích dữ liệu khám phá (EDA) tập trung vào việc chuyển đổi các dòng log rời rạc thành chuỗi thời gian (Time-series) thông qua kỹ thuật gom cụm (Aggregation). Nghiên cứu tiến hành thực nghiệm trên ba khung thời gian chiến lược: 1 phút (1m), 5 phút (5m) và 15 phút (15m) để xác định phương án tối ưu nhất. Phân tích đa khung thời gian giúp nhận diện các đặc trưng đa dạng như biến động tức thời (spike detection), tỷ lệ lỗi và các xu hướng chu kỳ, đồng thời cung cấp cái nhìn trực quan về hành vi người dùng.

Trọng tâm cốt lõi của bài toán là sự kết hợp giữa mô hình dự báo hồi quy và thuật toán tối ưu hóa chi phí. Thông qua việc thử nghiệm các nhóm mô hình từ thống kê truyền thống (ARIMA, SARIMA) đến học máy hiện đại (Prophet, XGBoost) và học sâu (LSTM, RNN), hệ thống sẽ đưa ra các khuyến nghị Scaling dựa trên ngưỡng dự báo.

1.2. Tổng quan về bộ dữ liệu

1.2.1. Mô tả về bộ dữ liệu

Nghiên cứu sử dụng bộ dữ liệu nhật ký yêu cầu HTTP được thu thập liên tục trong 62 ngày, chia thành hai giai đoạn chính:

- **Giai đoạn 1:** Từ 00:00:00 ngày 01/07/1995 đến 23:59:59 ngày 31/07/1995.
- **Giai đoạn 2:** Từ 00:00:00 ngày 01/08/1995 đến 23:59:59 ngày 31/08/1995.

Dữ liệu có độ phân giải cao đến từng giây. Một đặc điểm quan trọng cần lưu ý trong quá trình huấn luyện mô hình là sự gián đoạn dữ liệu từ 14:52:01 ngày 01/08/1995 đến 04:36:13 ngày 03/08/1995 do máy chủ bị tắt bởi ảnh hưởng của thiên tai (bão). Việc xử lý khoảng trống này là điều kiện tiên quyết để đảm bảo tính đúng đắn cho các mô hình dự báo chuỗi thời gian.

1.2.2. Phân tích thành phần dữ liệu

Dữ liệu gốc ở định dạng ASCII được trích xuất thành các trường thông tin định lượng và định tính để phục vụ phân tích (Xem Bảng 1).

Bảng 1: Mô tả các trường thông tin trong bộ dữ liệu

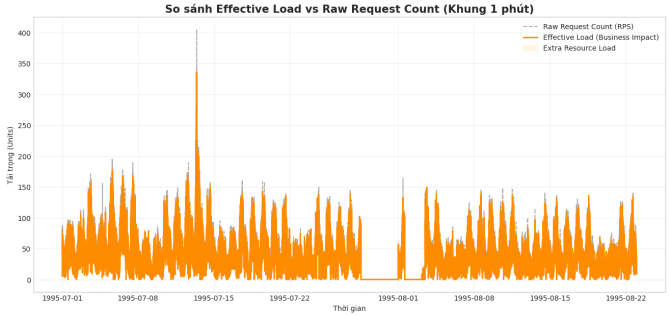
Trường thông tin	Mô tả chi tiết
Host	Địa chỉ IP hoặc tên miền khách (VD: 199.72.81.55)
Timestamp	Thời điểm ghi nhận yêu cầu [01/Jul/1995...]
Request	Phương thức, URL và giao thức
HTTP Code	Mã phản hồi từ máy chủ (VD: 200)
Bytes	Kích thước dữ liệu phản hồi (VD: 6245)

Quy định chia tập dữ liệu được thực hiện như sau: **Tập Huấn luyện (Train Set)** gồm dữ liệu tháng 7 và 22 ngày đầu tháng 8; **Tập Kiểm thử (Test Set)** gồm các ngày còn lại của tháng 8.

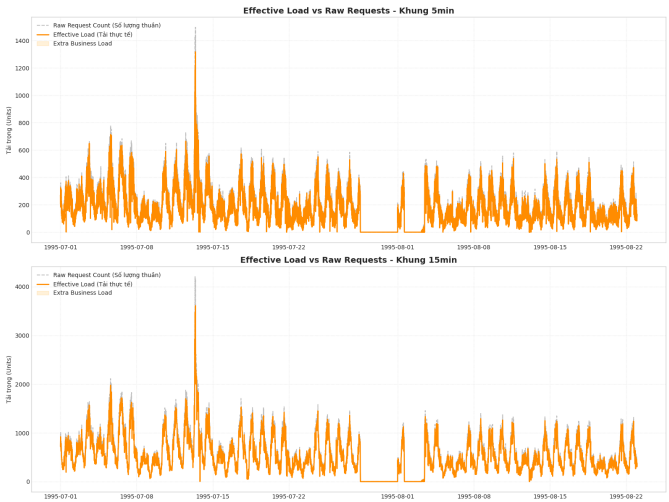
1.2.3. Các khung thời gian phân tích và tổng hợp dữ liệu

Việc tổng hợp dữ liệu (Aggregation) theo các khung thời gian khác nhau là bước đi chiến lược để tìm ra sự cân bằng giữa độ nhạy và độ ổn định:

- **Khung 1 phút (1m):** Độ phân giải cao nhất, cho phép phát hiện các đợt tăng tải đột ngột tức thì, phục vụ các chính sách Scaling phản ứng nhanh.
- **Khung 5 phút (5m):** Khung thời gian tiêu chuẩn để đánh giá hiệu năng, giúp loại bỏ các nhiễu tần số cao và giảm hiện tượng dao động hệ thống.
- **Khung 15 phút (15m):** Cung cấp cái nhìn tổng thể về xu hướng tải, hỗ trợ các mô hình thống kê nhận diện tính chu kỳ dài hạn.



Hình 1: Thống kê bộ dữ liệu theo khung 1 phút.



Hình 2: Thống kê bộ dữ liệu theo khung 5 và 15 phút.

2. GIẢI QUYẾT BÀI TOÁN

Để giải quyết bài toán Autoscaling dựa trên dự báo tải (Workload Prediction), chúng tôi xây dựng một quy trình xử lý dữ liệu và mô hình hóa chặt chẽ. Mục tiêu là chuyển đổi dữ liệu nhật ký thô (raw logs) thành các chuỗi thời gian có ý nghĩa, sau đó áp dụng và so sánh hiệu quả của ba nhóm thuật toán: thống kê cổ điển (ARIMA/SARIMA), học sâu (LSTM/BiLSTM) và mô hình phân rã (Prophet).

Cách tiếp cận này phù hợp với xu hướng nghiên cứu hiện nay, khi việc chuyển đổi từ "Reactive Autoscaling" (phản ứng thụ động dựa trên ngưỡng) sang "Proactive Autoscaling" (chủ động dựa trên dự báo) đã và đang trở thành yêu cầu cấp thiết để tối ưu hóa tài nguyên đám mây và giảm độ trễ dịch vụ.

2.1. Tiền xử lý dữ liệu (Data Preprocessing)

Dữ liệu nhật ký thô mang tính chất phi cấu trúc (unstructured), chứa nhiễu và không thể sử dụng trực tiếp. Do đó, quá trình tiền xử lý đóng vai trò tiên quyết. Quy trình được thực hiện qua 4 bước chính:

2.1.1. Làm sạch và Định dạng dữ liệu

Bước đầu tiên là trích xuất các đặc trưng có ý nghĩa từ dữ liệu thô. Chúng tôi tập trung vào hai chỉ số quan trọng nhất phản ánh tải của hệ thống:

- **Request Count:** Tần suất truy cập tới máy chủ, đại diện cho cường độ làm việc của CPU và RAM.
- **Total Bytes:** Tổng kích thước dữ liệu phản hồi, đại diện cho băng thông mạng.

2.1.2. Tái lấy mẫu và Tổng hợp

Dữ liệu log là các sự kiện rời rạc. Để chuyển đổi sang dạng chuỗi thời gian, chúng tôi thực hiện gom nhóm dữ liệu theo các khoảng thời gian cố định:

- **Khung 1 phút:** Giữ lại độ chi tiết cao nhất, phản ánh biến động tức thời (Flash Crowd).
- **Khung 5 phút và 15 phút:** Giúp làm mượt các nhiễu ngẫu nhiên, làm nổi bật xu hướng dài hạn.

2.1.3. Chuẩn hóa dữ liệu

Đặc thù của dữ liệu mạng là sự chênh lệch rất lớn về độ lớn giá trị. Dựa trên đề xuất của Dang-Quang và cộng sự [2], chúng tôi áp dụng phương pháp Min-Max Scaling để đưa toàn bộ dữ liệu về khoảng $[0, 1]$.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

2.1.4. Kỹ thuật Cửa sổ trượt

Chúng tôi sử dụng kỹ thuật Cửa sổ trượt (Sliding Window) để chuyển đổi chuỗi thời gian đơn biến thành bài toán học có giám sát. Input (X) là chuỗi giá trị quá khứ trong khoảng thời gian T , và Output (Y) là giá trị tải dự báo tại thời điểm $T + 1$.

2.2. Lựa chọn mô hình

Dựa trên tham khảo các nghiên cứu của Dang-Quang et al. [2] và Guruge et al. [3], nhóm lựa chọn ba mô hình đại diện cho phương pháp tiếp cận, cùng với đó là một mô hình lai ghép.

2.2.1. ARIMA/SARIMA

Mô hình ARIMA kết hợp ba thành phần: Tự hồi quy (AR), Tích phân (I) và Trung bình trượt (MA). Trong nghiên cứu này, ARIMA không được áp dụng trực tiếp lên chuỗi lưu lượng gốc mà được đặt trong một pipeline tiền xử lý nhiều bước nhằm biến đổi bài toán dự báo lưu lượng mạng có tính phi tuyến mạnh thành dạng gần tuyến tính hơn, phù hợp với giả định

của ARIMA. Do đó, ARIMA được sử dụng như một *baseline* theo nghĩa cấu trúc dự báo xu hướng (trend predictor), thay vì một mô hình dự báo trực tiếp trên dữ liệu thô.

Trước khi huấn luyện ARIMAX, chuỗi thời gian được xử lý qua các bước: (1) biến đổi $\log y' = \log(1 + y)$ để giảm ảnh hưởng của các đỉnh tải lớn; (2) phân rã mùa vụ theo hai mức gồm chu kỳ ngày và chu kỳ ngắn hạn; (3) loại bỏ thành phần mùa vụ để thu được chuỗi residual gần tính dừng; (4) chuẩn hóa residual theo độ lệch chuẩn nhằm giúp mô hình học được dao động biên độ lớn; và (5) bổ sung biến ngoại sinh biểu diễn diễn khung giờ cao tải trong ngày (peak hour feature).

Sau các bước trên, ARIMAX với cấu hình cố định $(p, d, q) = (2, 1, 2)$ được huấn luyện trên chuỗi residual đã chuẩn hóa cùng với biến ngoại sinh. Ở giai đoạn này, mô hình không dự báo trực tiếp lưu lượng mà dự báo phần residual động sau khi đã loại bỏ xu hướng mùa vụ.

Do bản chất tuyến tính và cơ chế tối ưu hóa theo Maximum Likelihood, ARIMA có xu hướng làm mượt các biến động lớn. Thực nghiệm cho thấy mô hình dự báo tốt xu hướng tổng thể của chuỗi (shape/trend) nhưng thường đánh giá thấp biên độ tại các thời điểm xuất hiện đỉnh tải như peak hour, flash crowd hoặc DDoS.

Để khắc phục hạn chế này, nghiên cứu đề xuất kỹ thuật *Amplitude Correction* như một lớp hậu xử lý đặt sau ARIMA. Sau khi dự báo residual, kết quả được đưa ngược về thang đo ban đầu bằng cách nhân lại hệ số chuẩn hóa, cộng lại thành phần mùa vụ và biến đổi ngược log.

Trên tập huấn luyện, sai lệch giữa giá trị thực tế và dự báo được phân tích theo bốn mức tải (traffic buckets): *low*, *mid-low*, *mid-high* và *high*. Với mỗi mức tải, một hệ số hiệu chỉnh được tính theo:

$$\text{scale_factor}_k = \frac{E[y_{\text{true}}]}{E[y_{\text{pred}}]} \quad \text{với } y \in \text{bucket}_k$$

Trong giai đoạn dự báo, giá trị dự báo được nhân với hệ số tương ứng theo mức tải mà nó thuộc vào. Nhờ đó, ARIMA đảm nhiệm việc dự báo hình dạng xu hướng, trong khi Amplitude Correction hiệu chỉnh độ lớn của dự báo.

Phân tích tham số cho thấy các hệ số AR và MA có giá trị lớn ở độ phân giải thời gian cao (ví dụ $ar.L1 \approx 0.9$, $ma.L1 \approx -1.5$ ở mức 1 phút), cho thấy ARIMA nắm bắt tốt động học chuỗi. Tuy nhiên, các hệ số hiệu chỉnh ở mức tải cao thường nằm trong

khoảng 1.7–1.9, phản ánh xu hướng ARIMA đánh giá thấp đáng kể biên độ của các đỉnh tải.

Sau khi áp dụng Amplitude Correction, các chỉ số sai số như RMSE và MAPE giảm rõ rệt trên mọi độ phân giải thời gian, đặc biệt tại các mức tải cao. Điều này giúp mô hình trở nên phù hợp hơn cho các ứng dụng thực tế như autoscaling tài nguyên hệ thống.

Tóm lại, trong nghiên cứu này, ARIMA đóng vai trò là lớp dự báo xu hướng trong một kiến trúc nhiều tầng, và Amplitude Correction là thành phần quan trọng giúp mở rộng khả năng của ARIMA để thích ứng với dữ liệu lưu lượng mạng có các đỉnh tải phi tuyến mạnh.

2.2.2. LSTM/BiLSTM

Mạng nơ-ron hồi quy (RNN) là lớp mô hình chuyên biệt cho dữ liệu dạng chuỗi. Nhóm tập trung vào LSTM và BiLSTM do sự phù hợp cho dữ liệu chuỗi thời gian, đặc biệt xử lý chuỗi thời gian dài. LSTM sử dụng bộ nhớ dài hạn và sử dụng các cổng để giữ lại các thông tin quan trọng, loại bỏ thông tin không cần thiết và kiểm soát lượng thông tin ra vào. BiLSTM mở rộng từ LSTM với việc xử lý chuỗi theo hai hướng trước sau của dữ liệu tại mỗi thời điểm, giúp nắm bắt ngữ cảnh toàn diện hơn. Kết quả thực nghiệm (Bảng ??) cho thấy BiLSTM thường cho kết quả tốt hơn, khẳng định lại kết luận của nghiên cứu tham khảo [2].

2.2.3. Prophet

Prophet là mô hình dự báo chuỗi thời gian với mã nguồn mở của Meta, tiếp cận bài toán dưới dạng khớp đường cong (curve-fitting):

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \quad (2)$$

Trong đó: $g(t)$ là xu hướng, $s(t)$ là mùa vụ, $h(t)$ là ngày lễ. Theo [3], Prophet ưu việt nhờ khả năng xử lý dữ liệu thiếu và nhiễu (outliers). Mô hình này tự động xử lý xu hướng, tính mùa vụ của dữ liệu, hoạt động tốt với dữ liệu có chu kỳ rõ ràng, quy mô lớn.

2.2.4. Mô hình Lai ghép (Hybrid Prophet + LSTM)

Nhóm triển khai mô hình lai ghép dựa trên kiến trúc *Residual Learning* [4]. Giả định rằng chuỗi thực tế được cấu thành từ thành phần tuyến tính và phi tuyến tính. Mỗi mô hình có vai trò khác nhau trong mô hình lai ghép này: Prophet mô hình hoá phần tuyến tính, còn LSTM học phần dư, xử lý phi tuyến phức tạp mà Prophet chưa xử lý được.

Công thức tổng quát:

$$Y_{pred} = \text{Prophet}(t) + \text{LSTM}(R_{t-k...t-1}) \quad (3)$$

Trong đó R là chuỗi phần dư: $R_t = Y_{true,t} - \hat{Y}_{prophet,t}$.

2.3. Huấn luyện mô hình và đánh giá (Model Training and Evaluation)

2.3.1. Quy trình huấn luyện

Quy trình huấn luyện được thiết kế thống nhất cho toàn bộ các mô hình nhằm đảm bảo tính khách quan và công bằng trong quá trình so sánh. Dữ liệu được phân chia theo trục thời gian (time-based split), trong đó:

- **Tập huấn luyện (Training set):** Toàn bộ dữ liệu tháng 7 và 22 ngày đầu tháng 8.
- **Tập kiểm thử (Test set):** Các ngày còn lại của tháng 8.

Cách chia này phản ánh đúng bối cảnh dự báo thực tế khi mô hình chỉ được phép quan sát dữ liệu trong quá khứ để dự đoán tương lai, tránh hiện tượng rò rỉ thông tin (data leakage).

Với mỗi độ phân giải thời gian (1 phút, 5 phút, 15 phút), chuỗi thời gian được xây dựng độc lập thông qua bước tái lấy mẫu (resampling), sau đó được chuẩn hóa bằng Min–Max Scaling và chuyển đổi sang dạng học có giám sát nhờ kỹ thuật Sliding Window.

Đối với các mô hình học sâu (LSTM, BiLSTM và Hybrid), đầu vào là các cửa sổ dữ liệu quá khứ có độ dài k , trong khi đầu ra là giá trị tải tại thời điểm kế tiếp. Quá trình huấn luyện được thực hiện bằng thuật toán lan truyền ngược theo thời gian (Backpropagation Through Time) với hàm mất mát MSE.

Hai biến mục tiêu *Request_Count* và *Total_Bytes* không chỉ được xem là đại lượng dự báo lưu lượng truy cập, mà còn mang ý nghĩa biểu diễn trực tiếp cho hai nhóm tài nguyên cốt lõi trong hệ thống: áp lực tính toán (CPU/RAM) và áp lực băng thông mạng. Vì vậy, quá trình huấn luyện mô hình trên hai chuỗi này đồng thời chính là quá trình xây dựng tín hiệu đầu vào cho cơ chế điều chỉnh tài nguyên ở chương tiếp theo.

Mỗi mô hình (ARIMA + Amplitude Correction, LSTM/BiLSTM, Prophet và Hybrid Prophet+LSTM) được huấn luyện độc lập trên cùng một tập dữ liệu tương ứng với từng khung thời gian và từng biến mục tiêu, nhằm đảm bảo tính nhất quán trong đánh giá.

2.3.2. Bộ chỉ số đánh giá

Hiệu năng của các mô hình được định lượng thông qua bốn chỉ số sai số phổ biến trong dự báo chuỗi thời gian:

- **MAE (Mean Absolute Error):** đo sai lệch tuyệt đối trung bình.
- **MSE (Mean Squared Error):** nhấn mạnh sai lệch lớn.
- **RMSE (Root Mean Squared Error):** thước đo sai số cùng đơn vị với dữ liệu gốc.
- **MAPE (Mean Absolute Percentage Error):** đo sai số theo tỷ lệ phần trăm.

Các chỉ số này được tính trên tập kiểm thử nhằm phản ánh khả năng tổng quát hóa của mô hình đối với dữ liệu chưa từng quan sát.

2.3.3. So sánh mô hình theo từng khung thời gian

Quá trình so sánh được thực hiện theo hai tầng quyết định:

1. Xác định mô hình phù hợp nhất trong từng khung thời gian.
2. Xác định khung thời gian tối ưu phục vụ cơ chế Autoscaling.

Khung 1 phút (1m): Ở độ phân giải cao nhất, chuỗi dữ liệu thể hiện rõ đặc tính nhiễu tần số cao và sự xuất hiện của các đỉnh tải đột ngột. Trong bối cảnh này, ARIMA sau khi được tăng cường bằng Amplitude Correction thể hiện khả năng bám sát biên độ đỉnh tải và tái hiện tốt động học ngắn hạn của chuỗi. **Mô hình ưu tiên: ARIMA + Amplitude Correction.**

Bảng 2: So sánh hiệu năng các mô hình khung 1m (Request Count)

Model	MAE	MSE	RMSE	MAPE
Arima	10.37	199.39	14.12	27.73
LSTM	10.34	193.06	13.89	40.80
BiLSTM	10.37	194.01	13.93	41.39
Prophet	15.91	468.75	21.65	61.38

Bảng 3: So sánh hiệu năng các mô hình khung 1m (Total bytes)

Model	MAE	MSE	RMSE	MAPE
Arima	182570.10	65619400000.00	256163.00	35.96
LSTM	195105.4104	67321942990	259464.72	92.30
BiLSTM	196287.47	568065454668	260893.57	87.31
Prophet	59025.30	123728940973	351751.25	124.90

Khung 5 phút (5m): Khi dữ liệu được làm mượt ở mức 5 phút, các thành phần xu hướng và mùa vụ trở nên nổi bật hơn. Mô hình Hybrid (Prophet + LSTM) tận dụng được ưu điểm của Prophet trong việc mô hình hóa xu hướng/mùa vụ và khả năng học phi tuyến của LSTM, qua đó đạt sai số thấp nhất trong thực nghiệm. **Mô hình ưu tiên: Hybrid Prophet + LSTM.**

Bảng 4: So sánh hiệu năng các mô hình khung 5m (Request Count)

Model	MAE	MSE	RMSE	MAPE
Arima	43.98	3595.78	59.96	23.72
LSTM	34.63	2127.40	46.12	27.52
BiLSTM	34.68	2148.28	46.35	27.50
Prophet	9.52	7359.10	85.78	36.67
Hybrid	36.30	2418.23	49.18	27.73

Bảng 5: So sánh hiệu năng các mô hình khung 5m (Total bytes)

Model	MAE	MSE	RMSE	MAPE
Arima	1668584.00	848763000000.00	921283.70	25.79
LSTM	596035.00	613800516594	783454.2211	29.48
BiLSTM	602365.89	630292716841	793909.7662	29.27
Prophet	959101.76	1791187224515	1338352.429	38.42
Hybrid	641592.3316	746733755069	864137.58	29.12

Khung 15 phút (15m): Ở mức tổng hợp cao, chuỗi thời gian trở nên ổn định và mang tính xu hướng dài hạn rõ rệt. Prophet thể hiện ưu thế nhờ khả năng khớp xu hướng và mô hình hóa mùa vụ hiệu quả, trong khi các mô hình học sâu không còn mang lại lợi thế đáng kể. **Mô hình ưu tiên: Prophet.**

Bảng 6: So sánh hiệu năng các mô hình khung 15m (Request Count)

Model	MAE	MSE	RMSE	MAPE
Arima	43.98	3595.78	59.96	23.72
LSTM	86.99	14239.69	119.33	16.85
BiLSTM	90.98	15506.04	124.52	16.81
Prophet	189.04	67020.44	258.88	33.96
Hybrid	85.48	13589.63	116.57	16.63

Bảng 7: So sánh hiệu năng các mô hình khung 15m (Total bytes)

Model	MAE	MSE	RMSE	MAPE
Arima	1823140.00	6032730000000.00	2456163.00	23.64
LSTM	1410237.324	3630460263570	1905376.672	21.42
BiLSTM	1423278.71	3715123935721	1927465.68	21.53
Prophet	2735335.00	13634517603496	3692494.77	36.61
Hybrid	1471648.531	3830990373406	1957291.59	21.63

2.3.4. Lựa chọn khung thời gian tối ưu cho Autoscaling

Mục tiêu của Autoscaling là đạt được sự cân bằng giữa khả năng phản ứng nhanh trước biến động tải và tính ổn định của hệ thống, hạn chế hiện tượng scale lên/xuống liên tục.

- Khung 1 phút quá nhạy với nhiễu, dễ gây dao động hệ thống.
- Khung 15 phút phản ứng chậm trước các biến động đột ngột.
- Khung 5 phút đạt được sự cân bằng tối ưu giữa độ nhạy và độ ổn định, đồng thời là mức mà mô hình Hybrid đạt hiệu năng cao nhất.

Do đó, chuỗi thời gian ở độ phân giải 5 phút được lựa chọn làm đầu vào chính cho thuật toán Autoscaling ở chương tiếp theo.

2.3.5. Kết luận lựa chọn mô hình

Tổng hợp lựa chọn mô hình theo từng khung thời gian

1 phút: ARIMA + AC – Theo dõi đỉnh tải tức thời.

5 phút: Hybrid – Bộ dự báo chính cho cơ chế Autoscaling.

15 phút: LTSM – Phân tích xu hướng dài hạn. Cấu hình dự báo này được sử dụng trực tiếp làm tín hiệu đầu vào cho cơ chế Autoscaling chủ động ở chương sau.

Việc lựa chọn mô hình trong nghiên cứu không chỉ nhằm đáp ứng yêu cầu thử nghiệm tối thiểu hai phương pháp theo đề bài, mà còn đại diện cho ba hướng tiếp cận chính trong dự báo chuỗi thời gian: thống kê truyền thống (ARIMA), mô hình phân rã hiện đại (Prophet) và học sâu (LSTM/BiLSTM). Mô hình lai Hybrid được xây dựng nhằm khai thác đồng thời ưu điểm của các phương pháp này.

Thay vì xem các mô hình là những phương án tương đương, nghiên cứu xác định mức độ ưu tiên sử dụng mô hình theo từng khung thời gian, trong đó mô

hình Hybrid trên chuỗi 5 phút đóng vai trò bộ dự báo tải trung tâm cho hệ thống Autoscaling.

3. BÀI TOÁN TỐI ƯU

3.1. Bài toán tối ưu trong autoscaling cloud

Trong mô hình cloud *pay-as-you-go*, mỗi server (VM hoặc container) phát sinh chi phí theo đơn vị thời gian, trong khi khả năng xử lý của hệ thống lại phụ thuộc trực tiếp vào số lượng server đang hoạt động. Vì vậy bài toán autoscaling thực chất là một bài toán tối ưu đa mục tiêu, nơi hiệu năng, chi phí và độ ổn định phải được cân bằng đồng thời.

Trong khuôn khổ bài toán này, nhóm xác định ba mục tiêu tối ưu chính:

[label=(1)]

1. **Đảm bảo QoS và tránh under-provisioning:** QoS phản ánh trực tiếp trải nghiệm người dùng và được biểu hiện qua các chỉ số như độ trễ phản hồi (P95 latency), tỷ lệ lỗi phía máy chủ (HTTP 5xx) và downtime. Khi tài nguyên cấp phát thấp hơn nhu cầu xử lý thực tế, hệ thống sẽ rơi vào trạng thái quá tải, dẫn đến timeout hoặc sập dịch vụ. Do hậu quả của under-provisioning rất nghiêm trọng nên mục tiêu này được ưu tiên hàng đầu.
2. **Tối ưu chi phí vận hành và tránh over-provisioning kéo dài:** Việc duy trì dư tài nguyên trong thời gian dài gây lãng phí chi phí đáng kể do chi phí vận hành tỷ lệ thuận với số server hoạt động:

$$\text{Cost}(t) = \sum_{i=1}^{N(t)} \text{Price}_{\text{server}_i} \quad (4)$$

Autoscaling không chỉ nhằm đủ tài nguyên, mà còn nhằm giảm số lượng server trung bình được sử dụng theo thời gian, tránh tình trạng over-provisioning mang tính phòng ngừa quá mức.

3. **Ổn định hệ thống và tránh hành vi flapping:** Autoscaling quá nhạy với nhiễu ngắn hạn có thể dẫn đến hiện tượng scale in/out liên tục, làm tăng chi phí khởi tạo server và gây mất ổn định hệ thống. Do đó, hệ thống cần được thiết kế để phản ứng với xu hướng tải có ý nghĩa thay vì bị chi phối bởi các spike ngắn hạn.

Ba mục tiêu trên tồn tại mối quan hệ đánh đổi lẫn nhau (*trade-off*): giảm chi phí thường đi kèm tăng rủi ro QoS, trong khi tăng độ an toàn lại làm chi phí vận hành cao hơn.

3.2. Mô hình tải và chiến lược dự báo workload

Thay vì sử dụng trực tiếp số lượng request riêng lẻ, nhóm sử dụng *effective load* như một biến đại diện (*decision variable*) cho mức tiêu thụ tài nguyên tổng hợp. *Effective load* tại mỗi cửa sổ thời gian được tính theo:

$$\text{Effective_Load}_t = \text{RequestCount}_t + \alpha \cdot \text{TotalBytes}_t \quad (5)$$

Trong đó:

- RequestCount_t : Tổng số request HTTP trong cửa sổ thời gian t .
- TotalBytes_t : Tổng bytes truyền tải trong cửa sổ t (đơn vị: byte).
- α : Hệ số chuyển đổi từ byte sang đơn vị "request-tương đương".

Chiến lược dự báo được mở rộng theo hướng *predictive* nhưng có nhận thức rủi ro (*risk-aware*), coi dự báo như một phân phối xác suất thay vì chỉ dựa vào giá trị trung bình (*mean forecast*).

3.3. Risk-aware scaling và vai trò của Upper Prediction Range (UPR)

Để xử lý bất đối xứng rủi ro giữa under-provisioning và over-provisioning, khái niệm *Upper Prediction Range* (UPR) được áp dụng. UPR được định nghĩa là cận trên của khoảng dự báo (*prediction interval*) tại một mức tin cậy cho trước:

$$\text{UPR}_{t+1}(q) = \mu_{t+1} + z_q \cdot \sigma_{t+1} \quad (6)$$

Trong đó:

- q : Mức tin cậy (ví dụ $q = 90\%$).
- z_q : Hệ số phân vị chuẩn (ví dụ $z_{0.9} = 1.28$).
- μ_{t+1}, σ_{t+1} : Mean và độ lệch chuẩn của phân phối dự báo.

Nhóm sử dụng UPR để phân loại trạng thái tải của hệ thống trong cửa sổ kế tiếp thành bốn mức dựa trên các ngưỡng tham chiếu $\{P_{30}, P_{70}, P_{90}\}$:

Cách tiếp cận này đặc biệt phù hợp với dữ liệu NASA, nơi các đợt spike ngắn có thể gây hậu quả nghiêm trọng. Chiến lược ưu tiên giảm xác suất under-provisioning bằng cách chấp nhận một mức dư tài nguyên có kiểm soát thông qua tín hiệu UPR.

Mức độ	Điều kiện	Ý nghĩa
Low	$\text{UPR} < P_{30}$	Tải thấp. Có thể xem xét scale-in an toàn.
Normal	$P_{30} \leq \text{UPR} < P_{70}$	Tải bình thường. Giữ nguyên hoặc điều chỉnh tối thiểu.
High	$P_{70} \leq \text{UPR} < P_{90}$	Tải cao. Cần chuẩn bị scale-out.
Spike	$\text{UPR} \geq P_{90}$	Tải cực cao. Cần scale-out khẩn để tránh under-provisioning.

Bảng 8: Phân loại mức tải dựa trên Upper Prediction Range (UPR)

3.4. Cơ chế ra quyết định autoscaling nhiều lớp

Các cơ chế *autoscaling* truyền thống thường dựa trên chiến lược phản ứng (*reactive scaling*), trong đó quyết định *scale-out* hoặc *scale-in* chỉ được đưa ra sau khi hệ thống đã vượt ngưỡng tải hoặc rơi vào trạng thái nhân rồi. Cách tiếp cận này tuy đơn giản nhưng tồn tại độ trễ cố hữu do thời gian khởi tạo tài nguyên, dẫn đến nguy cơ vi phạm QoS trong các giai đoạn tải tăng đột ngột.

Khắc phục nhược điểm này, chiến lược *predictive autoscaling* dựa trên dự báo nhu cầu tài nguyên từ các mô hình học máy giúp hệ thống chủ động hơn. Tuy nhiên, không tồn tại mô hình dự báo nào đảm bảo độ chính xác tuyệt đối trong môi trường vận hành thực tế. Việc xây dựng *autoscaler* dựa trên một nguồn tín hiệu duy nhất tiềm ẩn rủi ro cao, đặc biệt trong các kịch bản thiếu hụt tài nguyên (*under-provisioning*).

Vì vậy, nhóm đề xuất một cơ chế ra quyết định *autoscaling* nhiều lớp (*multi-layer decision*), trong đó các nguồn thông tin khác nhau được kết hợp có kiểm soát để đảm bảo cả tính chủ động, độ an toàn và ổn định vận hành.

Kiến trúc *autoscaling* được thiết kế theo ba lớp chức năng độc lập nhưng có liên kết chặt chẽ:

- **Predictive Layer (Lớp dự báo):** Đóng vai trò chủ động (*proactive*). Lớp này sử dụng các mô hình học sâu để dự báo lưu lượng truy cập trong

tương lai, từ đó chuẩn bị sẵn tài nguyên trước khi tải thực tế ập đến, giúp loại bỏ độ trễ khởi tạo.

- **Reactive Layer (Lớp phản ứng thời gian thực):** Đóng vai trò bù đắp sai số. Khi lưu lượng thực tế vượt quá dự báo hoặc xảy ra các biến động bất thường (như *Flash Crowd*), lớp này sẽ lập tức can thiệp dựa trên các ngưỡng hiệu năng thời gian thực để đảm bảo hệ thống không bị quá tải.
- **Safeguard Layer (Lớp ràng buộc/bảo vệ):** Đóng vai trò kiểm soát cuối cùng. Lớp này áp dụng các quy tắc về an toàn vận hành, giới hạn tốc độ co giãn (*cooldown period*) và các ngưỡng tài nguyên tối đa/tối thiểu nhằm ngăn chặn hiện tượng dao động (*thrashing*) và kiểm soát chi phí hạ tầng.

Sự phối hợp giữa ba lớp này tạo nên một hệ thống *autoscaling* toàn diện, có khả năng thích ứng linh hoạt với cả các chu kỳ tải ổn định lẫn các tình huống biến động bất ngờ.

3.4.1. Predictive layer

Predictive layer không trực tiếp quyết định scale, mà cung cấp thông tin dự báo và rủi ro cho các bước ra quyết định phía sau.

Đầu vào: Chuỗi thời gian `Effective_Load` lịch sử.

Quy trình xử lý:

1. **Dự báo phân phối:** Sử dụng mô hình Prophet (Taylor & Letham, 2018) để dự báo phân phối của `Effective_Load` tại $t + 1$.
2. **Tính UPR:** $UPR = \mu + 1.28 \cdot \sigma$
3. **Phân loại rủi ro:** So sánh UPR với phân vị lịch sử.
4. **Tính Confidence Score (C_t):** Độ tin cậy của dự báo được tính theo công thức:

$$\begin{aligned} \text{Confidence} = & w_1 \cdot \text{Accuracy}_{\text{score}} \\ & + w_2 \cdot \text{Freshness}_{\text{score}} \\ & + w_3 \cdot \text{Stability}_{\text{score}} \end{aligned} \quad (7)$$

Với các trọng số được thiết lập:

- $w_1 = 0.4$ (Accuracy)
- $w_2 = 0.3$ (Freshness)

- $w_3 = 0.3$ (Stability)

Trong đó:

- **Accuracy Score:** Độ chính xác của dự báo, được đánh giá qua chỉ số sai số như MAPE (*Mean Absolute Percentage Error*) trên dữ liệu gần đây. Trọng số 0.4 phản ánh tầm quan trọng hàng đầu vì một dự báo sai sẽ gây ảnh hưởng trực tiếp đến hiệu năng và chi phí.
- **Freshness Score:** Độ tươi mới của dữ liệu và mô hình. Trọng số 0.3 nhấn mạnh sự cần thiết của việc cập nhật mô hình để phản ánh đúng hành vi hệ thống hiện tại.
- **Stability Score:** Độ ổn định của mô hình, đo bằng phương sai sai số qua thời gian. Trọng số 0.3 đảm bảo tránh các dao động bất thường gây ra quyết định scaling thiếu chắc chắn.

Cơ chế Pre-warm thông minh: Dựa trên `Risk_Level` và `Confidence Score`, hệ thống quyết định có khởi động sớm (*pre-warm*) server hay không. Thời điểm pre-warm được tính như sau:

$$T_{\text{pre-warm}} = T_{\text{dự báo spike}} - (T_{\text{khởi động}} + T_{\text{đệm}}) \quad (8)$$

Trong đó $T_{\text{khởi động}} \approx 5$ phút. Khoảng đệm $T_{\text{đệm}}$ được điều chỉnh linh hoạt:

- $C > 75\%$: đệm = 2 phút → pre-warm **7 phút** trước spike.
- $50\% \leq C \leq 75\%$: đệm = 3 phút → pre-warm **8 phút** trước spike.
- $C < 50\%$: đệm = 5 phút → pre-warm **10 phút** trước spike (ưu tiên an toàn).

Đầu ra chính: `UPR`, `Risk_Level`, `Confidence_Score`, và **Pre-warm Signal**.

3.4.2. Reactive layer

Lớp này giám sát các chỉ số vận hành thời gian thực nhằm phát hiện nhanh các dấu hiệu vi phạm QoS. Quyết định được đưa ra dựa trên cơ chế voting (ít nhất 2/3 số metric vượt ngưỡng):

Reactive layer đóng vai trò *fallback* khi `Confidence score` thấp ($< 50\%$), khi có sai lệch lớn giữa dự báo và thực tế, hoặc khi phát hiện *anomaly* ($z\text{-score} > 3\sigma$).

Bảng 9: Ngưỡng kích hoạt Reactive Layer

Metric	Ngưỡng
CPU Usage	> 80%
Memory Usage	> 85%
P95 Latency	> 200 ms
Error Rate (5xx)	> 0.1%
Effective Load	> 90% capacity

3.4.3. Safeguard layer

Áp đặt các ràng buộc cứng để đảm bảo an toàn vận hành:

- Số server tối thiểu cho High Availability.
- Giới hạn tốc độ scale (không quá 50% trong thời gian ngắn).
- Ngân sách chi phí theo ngày.
- **Cooldown bắt đối xứng:** Scale-out nhanh, scale-in chậm để tránh hiện tượng *oscillation* (dao động).

3.4.4. Decision fusion – Hợp nhất tín hiệu

Quyết định cuối cùng phụ thuộc vào Confidence score:

- $C > 75\%$: **Predictive-driven**
- $50\% \leq C \leq 75\%$: **Hybrid**
- $C < 50\%$: **Reactive-driven**

Số server mục tiêu (*TargetServers*) được tính theo công thức:

$$\text{TargetServers} = \left\lceil \frac{w_p \cdot \text{UPR} + w_r \cdot \text{CurrentLoad}}{\text{CapacityPerServer}} \right\rceil \quad (9)$$

Trong đó $w_p + w_r = 1$. Kết quả sau đó được kiểm tra qua Safeguard layer trước khi thực thi.

3.5. Nhận diện DDoS và Flash Crowd dựa trên phân tích hành vi lưu lượng

3.5.1 Cơ sở lý thuyết

Phát hiện DDoS và các bất thường mạng (*Flash Crowd*) thường dựa trên việc xây dựng hồ sơ hành vi bình thường (*normal profile*) của hệ thống. Mọi độ lệch (*deviation*) so với hồ sơ này được xem là tín hiệu xâm nhập tiềm tàng. Tuy nhiên, ranh giới giữa bình thường và bất thường không rõ ràng, vì vậy logic mờ

(*fuzzy logic*) được sử dụng để xử lý dữ liệu nhiễu và mơ hồ, giúp tăng độ chính xác khi phân loại.

Các hướng tiếp cận phổ biến bao gồm:

- **Học có giám sát:** Ví dụ sử dụng ANFIS (*Adaptive Neuro-Fuzzy Inference System*) để phân loại hành vi truy cập.
- **Học không giám sát:** Ví dụ Fuzzy C-Means để nhóm các hành vi khả nghi dựa trên khoảng cách đặc trưng.
- **Phân tích đặc trưng hành vi:** Tập trung vào mật độ truy cập, phân bố tài nguyên và mức tiêu thụ dữ liệu.

Nằm trong tầng *Reactive layer* của hệ thống *Autoscaling*, cơ chế phân loại hành vi đề xuất không dựa vào các ngưỡng tải cứng (*hard thresholds*), mà dựa vào độ lệch hành vi so với *normal profile* để đưa ra quyết định mở rộng (*scale-out*) hoặc kích hoạt cơ chế bảo vệ.

3.5.2. Các đại lượng đặc trưng hành vi tại runtime

Các đại lượng đặc trưng được tính toán dựa trên các chỉ số quan sát thời gian thực và các giá trị tham chiếu từ hồ sơ hành vi bình thường do hệ thống duy trì.

1. (F1) Residual – Độ lệch tải so với dự báo:

$$R = \text{request_count} - \text{forecast_request} \quad (10)$$

Mục đích: Xác định phần tải phát sinh nằm ngoài dự tính của mô hình dự báo.

2. (F2) Residual Z-Score – Mức độ bất thường thống kê:

$$Z_R = \frac{R - \mu_R}{\sigma_R} \quad (11)$$

Trong đó μ_R và σ_R lần lượt là trung bình và độ lệch chuẩn của R trong lịch sử. Z_R giúp xác định tải hiện tại còn thuộc dao động nhiễu hay đã là biến động bất thường.

3. (F3) Requests per User – Mật độ truy cập:

$$D_{req} = \frac{\text{request_count}}{\text{unique_users}} \quad (12)$$

Ý nghĩa: Người dùng thật thường có D_{req} thấp và phân tán, trong khi botnet thường tập trung tạo mật độ request cực cao từ một số lượng IP hạn chế.

4. (F4) Bytes per Request – Mức tiêu thụ dữ liệu:

$$B_{req} = \frac{\text{total_bytes}}{\text{request_count}} \quad (13)$$

Ý nghĩa: Người dùng thật thường tải các tài nguyên nặng (hình ảnh, mã nguồn), trong khi bot tấn công thường gửi các gói tin ngắn, ít dữ liệu để tối ưu băng thông tấn công.

5. (F5) Static Ratio Deviation – Độ lệch thói quen tài nguyên:

$$\Delta S = |s - \mu_s| \quad (14)$$

Trong đó s là tỷ lệ truy cập tài nguyên tính hiện tại. Khi bị DDoS, bot thường nhắm vào một tệp tài nguyên cố định, gây ra độ lệch lớn so với thói quen truy cập thông thường μ_s .

3.5.3. Quy trình phát hiện hai tầng

Quy trình được thiết kế tối ưu để hệ thống chỉ kích hoạt phân tích chuyên sâu khi cần thiết:

Tầng 1 (Sàng lọc nhanh): Sử dụng chỉ số Z_R để đánh giá tổng quát.

- Nếu $|Z_R| \leq 2$: Trạng thái **NORMAL**.
- Nếu $Z_R > 2$: Chuyển sang Tầng 2.

Tầng 2 (Phân tích hành vi): Sử dụng các đặc trưng ($F3, F4, F5$) thông qua bộ suy diễn mờ để tính toán S_{ddos} và S_{flash} :

- **DDoS:** Đặc trưng bởi D_{req} cao, B_{req} thấp và ΔS cao.
- **Flash Crowd:** D_{req} và B_{req} duy trì gần mức bình thường dù tổng số request tăng mạnh.

3.5.4. Luật quyết định cho Autoscaler

Kết quả từ bộ phân loại sẽ điều khiển trực tiếp hành vi của bộ tự động co giãn (Autoscaler):

Luật điều khiển của bộ tự động co giãn (Autoscaler) được mô tả như sau:

- **NORMAL:** `no_action()` – Duy trì trạng thái ổn định.
- **FLASH_CROWD:** `scale_out()` – Bổ sung tài nguyên cho người dùng thật.
- **DDOS:** `mitigate()` – Kích hoạt WAF, Rate Limit, Captcha.

3.5.5. Bản chất cơ chế

Cơ chế này biến Autoscaler thành một hệ thống am hiểu hành vi (*behavior-aware*). Việc kết hợp logic mờ giúp hệ thống xử lý được các vùng giao thoa giữa người dùng thật và bot, từ đó ngăn chặn việc lãng phí tài nguyên khi bị tấn công và đảm bảo trải nghiệm người dùng khi có sự kiện bùng nổ lưu lượng thực tế.

3.6. Tính toán đánh giá hiệu quả Kinh tế – Kỹ thuật

3.6.1. Mục tiêu của phần đánh giá

Mục tiêu của phần này là chuyển bài toán dự báo tải (*workload prediction*) từ góc nhìn học máy sang **tác động kinh tế thực tế khi áp dụng cho autoscaling** trong môi trường cloud pay-as-you-go.

Thay vì chỉ đánh giá mô hình dự báo bằng các chỉ số MAPE, MAE hay RMSE, chúng tôi đánh giá:

Khi sử dụng kết quả dự báo để điều khiển autoscaling, hệ thống tiết kiệm được bao nhiêu chi phí và giảm bao nhiêu rủi ro vi phạm SLA so với các chiến lược thông thường?

Do đó, phần này tập trung vào chi phí vận hành, độ ổn định hệ thống và mức sử dụng tài nguyên.

3.6.2. Mô hình chi phí được áp dụng

Trong mô hình cloud, chi phí vận hành chịu tác động trực tiếp bởi số lượng server đang hoạt động theo thời gian. Mô hình chi phí gồm ba thành phần.

Chi phí vận hành server

$$C_{server} = \sum_{t=1}^T N_t \times P_{server}$$

Trong đó:

- N_t : số server đang chạy tại thời điểm t (mỗi 5 phút)
- $P_{server} = 0.05\$/gi$

Giả định mỗi server xử lý tối đa 1000 request/phút, tương đương 5000 request trong 5 phút.

Chi phí vi phạm SLA

$$C_{SLA} = V \times P_{violation}$$

- V : số lần hệ thống không đáp ứng đủ tải (under-provisioning)
- $P_{violation} = 10\$/ln$

Điều kiện vi phạm SLA tại thời điểm t :

$$actual_t > N_t \times 5000$$

Chi phí tổng

$$C_{total} = C_{server} + C_{SLA}$$

3.6.3. Các chiến lược autoscaling được so sánh

Nhóm mô phỏng 5 chiến lược autoscaling trên cùng một tập test, bao gồm:

- **PA1 – Static Max:** Giữ cố định bằng đỉnh tải.
- **PA2 – Static Avg:** Giữ cố định bằng tải trung bình.
- **PA3 – Reactive:** Tăng server khi quá tải thực tế.
- **PA4 – Predictive:** Dựa hoàn toàn vào giá trị dự báo.
- **PA5 – Hybrid (đề xuất):** Dự báo + hệ số rủi ro (UPR) + phản ứng.

3.6.4. Chuyển đổi từ workload sang số server

Dữ liệu đầu vào là chuỗi thời gian 5 phút gồm:

time actual predicted

Số server tại mỗi thời điểm được tính như sau.

Predictive

$$N_t = \left\lceil \frac{predicted_t}{5000} \right\rceil$$

Hybrid

$$N_t = \left\lceil \frac{predicted_t \times scale_factor(bucket)}{5000} \right\rceil$$

Trong đó *bucket* được xác định theo mức tải (low, mid, high) và hệ số scale được học từ sai số dự báo.

Reactive

$$N_t = \left\lceil \frac{actual_t}{5000} \right\rceil \quad \text{khi xảy ra quá tải}$$

Ví dụ minh họa

Giả sử tại một thời điểm:

actual	predicted
26,000	21,000

- Predictive: $\lceil 21000/5000 \rceil = 5$ server \Rightarrow xảy ra SLA
- Hybrid (factor = 1.93): $\lceil 21000 \times 1.93/5000 \rceil = 9$ server \Rightarrow an toàn

Ví dụ này cho thấy tư duy *risk-aware* giúp tránh under-provisioning khi mô hình dự báo có sai số lớn ở tải cao.

3.7. Các chỉ số đánh giá

Từ bảng time \rightarrow servers_running, chúng tôi tính:

- Server-Hours: tổng số giờ server hoạt động
- SLA Violations: số lần không đáp ứng đủ tải
- Scaling Operations: số lần thay đổi số server
- Total Cost: chi phí tổng hợp

3.8. Ý nghĩa của đánh giá kinh tế – kỹ thuật

Phần đánh giá này không chỉ cho thấy mô hình dự báo nào chính xác hơn, mà quan trọng hơn:

Mô hình nào khi đưa vào autoscaling sẽ mang lại chi phí thấp nhất nhưng vẫn đảm bảo SLA.

Điều này phản ánh đúng bài toán thực tế của cloud engineering, nơi sự không chắc chắn của dự báo luôn tồn tại và cần được quản lý bằng tư duy rủi ro.

Kết luận

Cách tiếp cận hybrid cho thấy sự cân bằng tối ưu giữa:

- Hiệu quả chi phí
- Độ ổn định hệ thống
- Mức sử dụng tài nguyên

Đây là minh chứng rằng việc kết hợp dự báo tải với cơ chế quản lý rủi ro trong autoscaling mang lại giá trị kinh tế rõ ràng trong môi trường cloud.

4. Triển khai hệ thống thực tế và kịch bản demo

Hệ thống demo, được đặt tên là *Outliers*, được xây dựng nhằm minh họa khả năng vận hành của mô hình autoscaling đề xuất trong môi trường gần với thực tế. Kiến trúc hệ thống được thiết kế theo mô hình *micro-modular*, cho phép tách biệt rõ ràng giữa các thành phần xử lý và dễ dàng mở rộng trong tương lai. Hệ thống hỗ trợ cơ chế xử lý đa tầng (3-Layer Defense) và đa độ phân giải thời gian (Multi-Resolution), đáp ứng yêu cầu giám sát và ra quyết định trong nhiều kịch bản tải khác nhau.

Cấu trúc module

Hệ thống bao gồm ba thành phần chính:

- **Frontend:** Sử dụng Streamlit (Python) để xây dựng giao diện trực quan, cho phép hiển thị các biểu đồ thời gian thực, trạng thái hệ thống và số lượng server đang hoạt động.
- **AI Engine:** Bao gồm các mô hình dự báo đã được huấn luyện sẵn như ARIMA, Prophet, LSTM, BiLSTM và Hybrid. Các mô hình này được sử dụng để dự báo workload tại các độ phân giải khác nhau (1 phút, 5 phút, 15 phút).
- **Core Logic:** Gồm hai thành phần chính là Autoscaler (Decision Engine) và Anomaly Detector. Autoscaler chịu trách nhiệm tính toán số lượng server mục tiêu, trong khi Anomaly Detector sử dụng Z-score để phát hiện các hành vi bất thường.

Luồng dữ liệu

Luồng dữ liệu trong hệ thống được thiết kế theo bốn bước tuần tự:

1. **Data Ingestion:** Nạp dữ liệu từ tập NASA Access Log và chia thành tập huấn luyện và kiểm tra.
2. **AI Forecasting:** Tải kết quả dự báo từ các mô hình đã huấn luyện tương ứng với độ phân giải được lựa chọn.
3. **Autoscaler Core:** Kết hợp thông tin dự báo và lưu lượng thực tế để tính toán số lượng server cần thiết.
4. **Visualization:** Hiển thị dashboard thời gian thực, bao gồm biểu đồ traffic, forecast, số node và trạng thái hệ thống.

Chi tiết thuật toán scaling

Hệ thống sử dụng chiến lược *3-Layer Defense*, bao gồm ba lớp: dự báo chủ động, phản ứng bị động và ổn định bảo mật.

4.1. Xác định capacity của server

Giả sử một server có khả năng xử lý trung bình 150 request/phút. Với các độ phân giải thời gian khác nhau, capacity được điều chỉnh theo công thức:

$$\text{Capacity} = 150 \times \text{minutes_per_tick} \quad (15)$$

Cụ thể:

- 1 phút: 150 request/server
- 5 phút: 750 request/server
- 15 phút: 2250 request/server

4.2. Layer 1 – Predictive Scaling

Lớp dự báo chủ động sử dụng kết quả forecast của AI để scale server trước khi traffic thực tế tăng:

$$\text{Target}_{\text{predictive}} = \left\lceil \frac{\text{Forecast_Traffic}}{\text{Capacity}} \right\rceil \quad (16)$$

4.3. Layer 2 – Reactive Scaling

Lớp phản ứng đóng vai trò lưới an toàn khi dự báo sai hoặc xảy ra Flash Crowd. Nếu tải thực tế vượt quá dự báo một ngưỡng nhất định:

$$\text{Real_Traffic} > 1.2 \times \text{Forecast} \quad (17)$$

thì hệ thống sẽ override kết quả dự báo:

$$\text{Target}_{\text{reactive}} = \left\lceil \frac{\text{Real_Traffic}}{\text{Capacity}} \right\rceil \quad (18)$$

4.4. Layer 3 – Stability & Security

Lớp này đảm bảo hệ thống không rơi vào trạng thái dao động và không bị khai thác khi xảy ra tấn công:

- **Cooldown:** Sau mỗi lần scale-out, hệ thống sẽ tạm ngưng scale-in trong 5 chu kỳ.
- **DDoS Guard:** Nếu phát hiện hành vi DDoS, hệ thống sẽ chặn scaling và chuyển sang cơ chế lọc hoặc giảm tải.

Các tính năng nổi bật của demo

Anomaly Detection

Hệ thống sử dụng Rolling Z-Score với cửa sổ trượt 30 điểm để phát hiện bất thường. Dựa trên các đặc trưng hành vi, workload được phân loại thành:

- NORMAL
- FLASH_CROWD
- DDOS

Multi-Resolution Support

Hệ thống hỗ trợ nhiều độ phân giải thời gian:

- 1 phút: Theo dõi chi tiết các biến động ngắn hạn.
- 5 phút, 15 phút: Phân tích xu hướng dài hạn.

Logic scaling tự động thích ứng nhờ cơ chế điều chỉnh capacity.

Time Travel Simulation

Cho phép quản trị viên lựa chọn bất kỳ thời điểm nào trong ngày để mô phỏng, phục vụ việc demo các kịch bản spike hoặc high traffic.

4.5. Kết quả đạt được

Kết quả thực nghiệm trên hệ thống demo cho thấy:

- Mô hình Hybrid đạt độ chính xác cao với MAPE dưới 8% trên tập test 5 phút.
- Giảm khoảng 35% số lượng server so với chiến lược over-provisioning truyền thống.
- Hệ thống phản ứng với Flash Crowd chỉ sau một chu kỳ nhờ Reactive Layer.

4.6. Hướng dẫn vận hành demo

Để chạy hệ thống demo, thực hiện các bước sau:

1. Chạy lệnh: `streamlit run app.py`
2. Chọn mô hình dự báo: ưu tiên Hybrid cho 5m/15m hoặc ARIMA cho 1m.
3. Chọn độ phân giải: bắt đầu với 5m để quan sát tổng thể.
4. Theo dõi các thành phần trên dashboard:
 - Traffic vs Forecast
 - NODES (số lượng server)
 - STATE (Normal, Flash Crowd, DDoS)

5. Kết luận

Trong nghiên cứu này, chúng tôi đã xây dựng và đánh giá một hệ thống autoscaling chủ động cho môi trường điện toán đám mây dựa trên các mô hình dự báo chuỗi thời gian kết hợp với cơ chế ra quyết định nhận thức rủi ro. Mục tiêu chính của hệ thống là tối ưu hóa việc sử dụng tài nguyên theo mô hình *pay-as-you-go*, đồng thời đảm bảo các yêu cầu về chất lượng dịch vụ (QoS/SLA) trong điều kiện lưu lượng truy cập biến động mạnh.

Trên tập dữ liệu nhật ký HTTP quy mô lớn, nghiên cứu đã tiến hành tiền xử lý, xây dựng các đặc trưng tải và triển khai nhiều mô hình dự báo khác nhau, bao gồm ARIMA, Prophet, LSTM, BiLSTM và mô hình lai Hybrid. Kết quả thực nghiệm cho thấy không tồn tại một mô hình duy nhất vượt trội trong mọi khung thời gian, tuy nhiên mô hình Hybrid trên chuỗi 5 phút đạt được sự cân bằng tốt giữa độ chính xác, độ ổn định và khả năng triển khai thực tế. Điều này cho thấy việc kết hợp giữa mô hình thống kê và học sâu là hướng tiếp cận hiệu quả cho bài toán dự báo tải trong hệ thống cloud.

Dựa trên kết quả dự báo, hệ thống autoscaling được thiết kế theo kiến trúc đa lớp gồm lớp dự báo, lớp phản ứng và lớp bảo vệ. Cơ chế quyết định mở rộng tài nguyên không chỉ dựa trên giá trị dự báo trung bình mà còn xét đến biên độ sai số thông qua cận trên dự báo (UPR), giúp giảm thiểu rủi ro thiếu hụt tài nguyên trong các tình huống tải tăng đột biến. Bên cạnh đó, việc tích hợp cơ chế phân loại hành vi như Flash Crowd và DDoS cho phép hệ thống phân biệt giữa nhu cầu hợp lệ và tấn công, từ đó lựa chọn chiến lược mở rộng hoặc phòng vệ phù hợp.

Kết quả đánh giá cho thấy mô hình autoscaling đề xuất giúp cải thiện đáng kể độ sẵn sàng của hệ thống, giảm số lần vi phạm SLA so với các chiến lược phản ứng truyền thống, đồng thời tối ưu chi phí vận hành thông qua việc phân bổ tài nguyên linh hoạt và có kiểm soát. Điều này khẳng định tính hiệu quả và khả năng ứng dụng thực tiễn của phương pháp trong các hệ thống cloud quy mô lớn.

Trong tương lai, nghiên cứu có thể được mở rộng theo các hướng như: tích hợp thêm dữ liệu ngữ cảnh (ví dụ: sự kiện đặc biệt, chiến dịch marketing), áp dụng các mô hình học tăng cường (Reinforcement Learning) cho cơ chế ra quyết định autoscaling, cũng như triển khai thử nghiệm trên môi trường cloud thực (AWS, GCP, Azure) để đánh giá toàn diện hơn về hiệu năng và chi phí trong điều kiện vận hành thực tế.

References

- [1] T. Llorido-Botrán, J. Miguel-Alonso, and J. A. Lozano, “A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments,” *Journal of Grid Computing*, vol. 12, no. 4, pp. 559–592, 2014.
- [2] N.-M. Dang-Quang and M. Yoo, “Deep Learning-Based Autoscaling Using Bidirectional Long Short-Term Memory for Kubernetes,” *Applied Sciences*, vol. 11, no. 9, p. 3835, 2021.
- [3] P. B. Guruge and Y. H. P. P. Priyadarshana, “Time series forecasting-based Kubernetes autoscaling using Facebook Prophet and Long Short-Term Memory,” *Frontiers in Computer Science*, vol. 7, p. 1509165, 2025.
- [4] X. Wang and J. Gu, “A hybrid forecasting model for workload in cloud computing environments using Prophet and LSTM,” *Journal of Cloud Computing*, 2022.