# Tối ưu hóa tài nguyên động trên môi trường điện toán đám mây

Nguyễn Thị Phong Dung*, Đỗ Hoàng Nam

Khoa Công nghệ Thông tin – Trường Đại học Nguyễn Tất Thành, Thành phố Hồ Chí Minh, Việt Nam
*ntpdung@ntt.edu.vn, namdh@ntt.edu.vn

## Tóm tắt

Phân bổ tài nguyên động trong môi trường điện toán đám mây là một thách thức lớn, đòi hỏi khả năng dự đoán chính xác và tối ưu hóa hiệu quả để nâng cao hiệu suất sử dụng tài nguyên và giảm chi phí. Các phương pháp truyền thống thường khó thích ứng nhanh với nhu cầu biến động, dẫn đến lãng phí hoặc suy giảm chất lượng dịch vụ. Nghiên cứu này đề xuất một phương pháp tích hợp mô hình bộ biến đổi để dự báo nhu cầu tài nguyên (CPU, bộ nhớ, băng thông) với hai thuật toán: Di truyền và Tối ưu hóa Đàn Kiến. Mô hình bộ biến đổi giúp dự đoán chính xác nhu cầu tương lai dựa trên dữ liệu lịch sử, cho phép điều chỉnh phân bổ tài nguyên một cách chủ động. Thuật toán di truyền và tối ưu hóa đàn kiến được áp dụng sau đó để tối ưu việc phân phối tài nguyên trên các máy ảo. Thực nghiệm trên dữ liệu thực tế cho thấy phương pháp này cải thiện đáng kể độ chính xác dự đoán, giảm độ trễ trong phân bổ và tối ưu hóa việc sử dụng tài nguyên tốt hơn so với các phương pháp truyền thống như LSTM hoặc tối ưu hóa dựa trên tìm kiếm. Đây là giải pháp hứa hẹn cho quản lý tài nguyên động trong điện toán đám mây.



Nhận 09/01/2025
Được duyệt 11/06/2025
Công bố 28/06/2025

## Từ khóa

tối ưu hóa tài nguyên trên đám mây, transformer, dự báo chuỗi thời gian, thuật toán di truyền, tối ưu hóa đàn kiến, phân bổ tài nguyên động

## 1 Đặt vấn đề

Trong các hệ thống điện toán đám mây, quản lý và phân bổ tài nguyên hiệu quả đóng vai trò quan trọng trong việc đảm bảo hiệu suất và tối ưu hóa chi phí. Tuy nhiên, nhu cầu tài nguyên biến động liên tục khiến việc dự đoán trở nên thách thức. Các phương pháp truyền thống như ARIMA, Holt-Winters và mạng nơ-ron đơn giản gặp khó khăn với chuỗi thời gian phi tuyến và không ổn định, dẫn đến độ chính xác thấp [1, 4]. Gần đây, mô hình học sâu, đặc biệt là Bộ biến đổi (Transformer –TF), đã cho thấy tiềm năng xử lý chuỗi thời gian dài nhờ cơ chế tự chú ý, giúp nắm bắt mối quan hệ phức tạp trong dữ liệu. Bộ biến đổi đã được áp dụng hiệu quả trong dự báo dữ liệu viễn thám cho hệ thống kiểm soát môi trường tàu vũ trụ [1] và trong dự báo doanh số tại môi trường điện toán biên [2]. Trong điện toán đám mây, mạng nơ-ron tích chập theo thời gian cũng đã được sử dụng để dự đoán và tối ưu phân bổ tài nguyên [3, 4].

Tuy nhiên, chỉ dựa vào dự báo để phân bổ tài nguyên có thể không đủ hiệu quả. Việc kết hợp với các thuật toán tối ưu như Thuật toán Di truyền (Genetic Algorithm – GA) hoặc Tối ưu hóa Đàn Kiến (Ant Colony Optimization – ACO) có thể nâng cao hiệu quả phân bổ. Các nghiên cứu cho thấy tối ưu hóa giúp cải thiện hiệu suất dịch vụ web [5] và chất lượng dịch vụ (QoS – Quality of Service) [6]. Ngoài ra, các hướng nghiên cứu như tối ưu hóa dựa trên QoS hoặc quản lý tài nguyên trong Kubernetes cũng mang lại kết quả tích cực [7, 8]. Dự báo chính xác còn giúp giảm lãng phí và nâng cao hiệu suất hệ thống [9].

## 2 Các nghiên cứu liên quan

### 2.1 Dự đoán tài nguyên trên điện toán đám mây

Dự đoán nhu cầu tài nguyên là một trong những bài toán quan trọng trong quản lý điện toán đám mây. Các mô hình này giả định dữ liệu có tính tuyến tính, không phù hợp với các chuỗi thời gian phi tuyến và có tính biến động cao, thường thấy trong các hệ thống đám mây [10, 11]. Những năm gần đây, các mô hình học sâu như Long Short-Term Memory (LSTM) và Gated Recurrent Unit (GRU) đã được áp dụng để cải thiện độ chính xác trong dự đoán tài nguyên điện toán đám mây [12]. LSTM có khả năng ghi nhớ các mối quan hệ dài hạn trong chuỗi thời gian, giúp mô hình hóa tốt hơn các mẫu thay đổi của tài nguyên hệ thống. GRU, với kiến trúc đơn giản hơn LSTM, cũng cho thấy hiệu quả cao trong việc dự đoán với chi phí tính toán thấp hơn [13]. Tuy nhiên, cả hai mô hình này đều gặp hạn chế khi xử lý các chuỗi thời gian dài do hiện tượng suy giảm độ dốc của hàm mất mát và khả năng học tập kém khi dữ liệu có mối quan hệ phức tạp.

2.2 Ứng dụng mô hình TF trong dự báo chuỗi thời gian

Gần đây, TF đã trở thành một phương pháp tiên tiến trong dự đoán chuỗi thời gian, nhờ cơ chế tự chú ý (Self-Attention), giúp mô hình học được các mối quan hệ dài hạn trong dữ liệu hiệu quả hơn so với LSTM và GRU [14]. Khác với các mô hình tuần tự như Recurrent Neural Network (RNN), TF có thể xử lý toàn bộ chuỗi thời gian đồng thời, giúp cải thiện tốc độ huấn luyện và dự báo [15]. Đã có nghiên cứu ứng dụng TF vào dự báo dữ liệu viễn thám và cho thấy hiệu quả vượt trội so với các phương pháp truyền thống [10]. Tương tự, TF có thể cải thiện đáng kể độ chính xác trong dự báo nhu cầu tài nguyên so với LSTM và GRU khi áp dụng vào hệ thống điện toán biên [11]. Những kết quả này cho thấy TF là một lựa chọn tiềm năng trong việc dự đoán tài nguyên điện toán đám mây.

2.3 Các phương pháp tối ưu hóa phân bổ tài nguyên

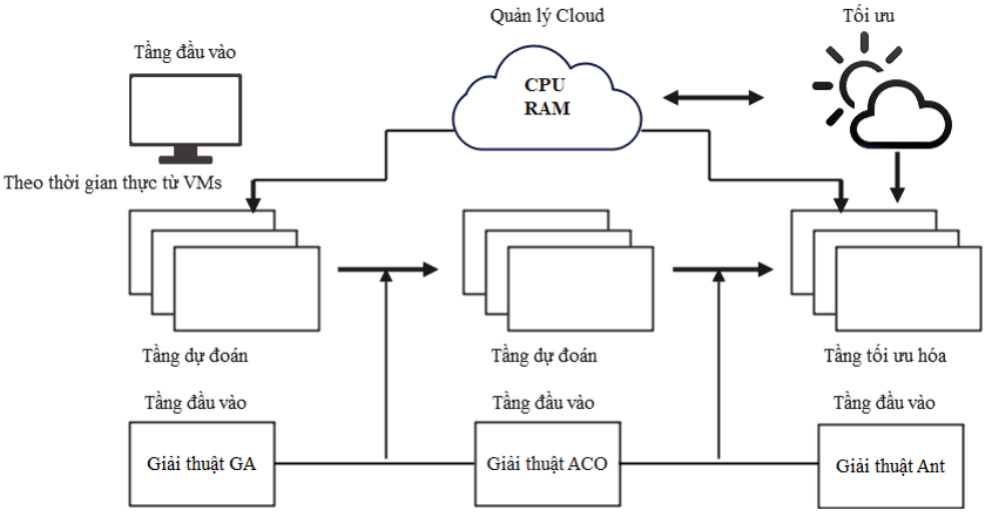
Dự báo chính xác tài nguyên chỉ là một phần của bài toán quản lý tài nguyên trên điện toán đám mây. Tối ưu hóa phân bổ tài nguyên dựa trên dự đoán là bước quan trọng nhằm nâng cao hiệu suất và giảm chi phí vận hành. Các thuật toán như GA, Tối ưu hóa Bầy Đàn (Particle Swarm Optimization – PSO) đã được áp dụng rộng rãi [16, 17]. GA dựa trên nguyên lý chọn lọc tự nhiên, giúp tìm lời giải tối ưu trong không gian lớn [16]. ACO mô phỏng hành vi kiến tìm đường, giúp hệ thống thích nghi nhanh [17], còn PSO – dựa trên hành vi bầy chim – cho phép tìm cấu hình tài nguyên tối ưu với tốc độ hội tụ cao [18].

Kết hợp dự đoán bằng TF với các thuật toán tối ưu hóa có thể cải thiện đáng kể hiệu suất hệ thống. Nghiên cứu chỉ ra rằng sự kết hợp này giúp giảm chi phí tài nguyên mà vẫn duy trì hiệu suất [12].

## 3 Phương pháp đề xuất

### 3.1 Kiến trúc hệ thống

Hệ thống đề xuất bao gồm ba tầng chính: Tầng đầu vào, Tầng dự đoán và Tầng tối ưu hóa (Hình 1).




**Hình 1** Kiến trúc tổng quan của hệ thống

Mô tả cách tích hợp triển khai thực tế như sau:
Mô hình được tích hợp vào Kubernetes thông qua một dịch vụ dự báo, sử dụng TF để dự đoán nhu cầu tài nguyên và truyền kết quả qua giao thức API REST đến bộ điều phối tài nguyên (Kubernetes Scheduler). Yêu cầu triển khai bao gồm: phần cứng (GPU Gigabyte RTX4070 TISP, 16GB RAM), phần mềm (Kubernetes v1.25, TensorFlow 2.10), và giao thức API (gRPC để giảm độ trễ). Mô hình cũng tương thích với OpenStack thông qua tiện ích mở rộng điều phối tài nguyên.

3.1.1 Tầng đầu vào

Tầng đầu vào chịu trách nhiệm thu thập dữ liệu thời gian thực từ các máy ảo (Virtual Machines – VMs), bao gồm:
- Mức sử dụng CPU (%)
- Mức sử dụng RAM (GB)
- Lưu lượng mạng (băng thông – Mbps)
- Thông tin tải của ứng dụng đang chạy

Dữ liệu đầu vào được thu thập theo thời gian thực từ các VMs trong hệ thống điện toán đám mây, bao gồm: mức sử dụng CPU (%), RAM (GB), lưu lượng mạng (Mbps) và tải công việc ứng dụng. Các chỉ số này phản ánh tình trạng tiêu thụ tài nguyên tại mỗi thời điểm và được lưu trong cơ sở dữ liệu giám sát tài nguyên. Trước khi đưa vào mô hình TF để dự đoán nhu cầu, dữ liệu trải qua quá trình tiền xử lý nhằm nâng cao chất lượng và tính đồng nhất. Cụ thể, hệ thống thực hiện ba bước: (i) loại bỏ điểm ngoại lai để loại trừ giá trị bất thường gây sai lệch mô hình; (ii) xử lý dữ liệu thiếu bằng cách nội suy hoặc loại bỏ bản ghi không đầy đủ nhằm đảm bảo tính toàn vẹn; (iii) chuẩn hóa dữ liệu về cùng một phạm vi giá trị, giúp cân bằng ảnh hưởng giữa các đặc trưng có đơn vị đo khác nhau. Tầng đầu vào đóng vai trò nền tảng, đảm bảo dữ liệu đầy đủ, chính xác và nhất quán, từ đó nâng cao hiệu quả dự đoán nhu cầu tài nguyên trong môi trường điện toán đám mây.

3.1.2 Tầng dự đoán

Trong hệ thống dự báo nhu cầu tài nguyên, tầng này sử dụng mô hình TF để phân tích dữ liệu lịch sử và dự đoán nhu cầu trong tương lai. So với các mô hình truyền thống như LSTM, TF được lựa chọn nhờ khả năng học hiệu quả các quan hệ dài hạn trong chuỗi thời gian và xử lý song song nhanh chóng, giúp tăng tốc độ huấn luyện và cải thiện độ chính xác. Cơ chế tự chú ý cho phép mô hình tập trung vào các mốc thời gian quan trọng, thay vì phụ thuộc vào trạng thái ẩn tuần tự như LSTM, từ đó nắm bắt tốt hơn các xu hướng phức tạp và biến động mạnh trong nhu cầu tài nguyên.
Để đánh giá hiệu quả, mô hình TF được so sánh với LSTM qua ba chỉ số:
- Sai số tuyệt đối trung bình (Mean Absolute Error – MAE): đo sai số trung bình tuyệt đối giữa dự đoán và thực tế, phản ánh độ chính xác tổng thể.
- Căn bậc hai của sai số bình phương trung bình (Root Mean Square Error – RMSE): nhấn mạnh các sai số lớn, phù hợp với dữ liệu biến động mạnh.
- Sai số phần trăm tuyệt đối trung bình (Mean Absolute Percentage Error – MAPE): Biểu diễn sai số dưới dạng phần trăm, thuận tiện để so sánh trên các thang đo khác nhau.

Các chỉ số này giúp xác định liệu TF có thực sự vượt trội trong dự báo tài nguyên cho môi trường điện toán đám mây.

3.1.3 Tầng tối ưu hóa

Dựa trên kết quả dự đoán nhu cầu tài nguyên, hệ thống sẽ áp dụng thuật toán tối ưu hóa để đưa ra chiến lược phân bổ tài nguyên hợp lý, đảm bảo hiệu suất hoạt động của hệ thống điện toán đám mây. Hai thuật toán được sử dụng trong quá trình này là GA và ACO, mỗi thuật toán có cách tiếp cận riêng để tìm ra giải pháp tối ưu. GA là một thuật toán tối ưu hóa dựa trên nguyên lý chọn lọc tự nhiên. Hệ thống khởi tạo một tập hợp nhiều phương án phân bổ tài nguyên, sau đó đánh giá từng phương án dựa trên hàm mục tiêu. Những phương án tốt hơn sẽ được lai ghép và đột biến để tạo ra thế hệ mới, giúp cải thiện dần chất lượng giải pháp theo thời gian. Qua nhiều vòng lặp, GA sẽ hội tụ đến phương án tối ưu nhất, giúp hệ thống phân bổ tài nguyên hiệu quả mà vẫn đảm bảo giảm thiểu độ trễ và tối ưu chi phí. ACO mô phỏng cách loài kiến tìm kiếm thức ăn trong tự nhiên. Trong hệ thống phân bổ tài nguyên, mỗi "kiến" đại diện cho một phương án điều hướng tài nguyên từ máy chủ vật lý đến máy ảo. Mỗi lần điều hướng, "kiến" để lại một lượng chất dẫn đường (pheromone) giúp các "kiến" sau ưu tiên đi theo các lộ trình hiệu quả hơn. Nhờ đó, hệ thống có thể xác định

các máy ảo có nhu cầu tài nguyên cao nhất và ưu tiên phân bổ tài nguyên vào những nơi thực sự cần thiết.
Cả hai thuật toán tối ưu hóa này đều hướng đến việc tối ưu hóa hàm mục tiêu, bao gồm:
- Giảm thiểu độ trễ của hệ thống bằng cách đảm bảo máy ảo có đủ tài nguyên để xử lý khối lượng công việc.
- Tối ưu hóa chi phí tài nguyên, giảm thiểu việc cấp phát dư thừa dẫn đến lãng phí tài nguyên.
- Đảm bảo cân bằng tải giữa các máy ảo, tránh tình trạng một số máy bị quá tải trong khi các máy khác lại nhàn rỗi.
Nhờ vào sự kết hợp giữa mô hình TF để dự đoán nhu cầu tài nguyên và thuật toán tối ưu hóa GA/ACO để phân bổ tài nguyên hợp lý, hệ thống có thể cải thiện đáng kể hiệu suất hoạt động của điện toán đám mây, giúp tối ưu hóa cả về chi phí và hiệu suất.

3.2 Mô hình TF

TF là mô hình học sâu hiện đại, có khả năng học các mối quan hệ dài hạn giữa các điểm dữ liệu trong chuỗi thời gian thông qua cơ chế tự chú ý. TF sử dụng tự chú ý để tính trọng số giữa các thời điểm trong chuỗi thời gian. Công thức tính Cơ chế chú ý với tích vô hướng được chia tỉ lệ:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

Trong đó:
- Q (Query), K (Key), V (Value) là các ma trận đầu vào.
- $d_k$ là kích thước của véctơ khóa (để tránh giá trị lớn gây mất ổn định).

*Mã giả cho TF:*

*Input:* chuỗi dữ liệu tài nguyên (CPU, RAM, Băng thông) từ t-1, t-2, ..., t-n

*Output:* dự đoán tài nguyên ở thời điểm t+1

*1. Chuẩn hóa dữ liệu đầu vào*

*2. Ánh xạ dữ liệu vào lớp biểu diễn véctơ*

*3. Tạo mã hóa vị trí cho chuỗi đầu vào*

*4. Duyệt qua nhiều lớp mã hóa*

*a. Tính Self-Attention theo công thức Attention(Q, K, V)* (1)

*b. Chuẩn hóa bằng Layer Normalization*

*c. Đưa qua Feed Forward Network*

*5. Tạo đầu ra bằng Linear Layer*

*6. Trả về giá trị dự đoán tài nguyên tại t+1*

Cấu trúc TF trong dự báo chuỗi thời gian gồm hai thành phần chính: bộ mã hóa và bộ giải mã. Bộ mã hóa có nhiều lớp tích hợp cơ chế tự chú ý, giúp trích xuất thông tin từ chuỗi dữ liệu lịch sử (CPU, RAM, băng thông, tải ứng dụng) và phát hiện các mối quan hệ quan trọng giữa các thời điểm. Bộ giải mã sử dụng cơ chế tự chú ý có mặt nạ để đảm bảo chỉ khai thác thông tin quá khứ trong quá trình dự đoán, đồng thời kết hợp chú ý giữa đầu ra và đầu vào nhằm nâng cao khả năng dự báo giá trị tài nguyên tương lai. TF vượt trội hơn các mô hình truyền thống như LSTM và GRU nhờ khả năng học các quan hệ dài hạn mà không bị lãng quên thông tin, xử lý song song giúp tăng tốc huấn luyện, và cơ chế tự chú ý cho phép tập trung vào các thời điểm quan trọng trong chuỗi dữ liệu. Nhờ đó, TF trở thành lựa chọn hiệu quả trong dự báo nhu cầu tài nguyên, hỗ trợ hệ thống phân bổ tài nguyên chính xác và kịp thời.

3.3 Tích hợp với GA/ACO

Mô hình TF được huấn luyện trên dữ liệu lịch sử sử dụng tài nguyên của các máy ảo nhằm dự báo nhu cầu tương lai. Một số siêu tham số quan trọng gồm: batch size = 64 (cân bằng tốc độ và độ ổn định), số epoch = 100 (đủ thời gian học), optimizer Adam (hội tụ nhanh), và hàm lỗi MSE (đánh giá độ chính xác dự báo).
Sau huấn luyện, mô hình được tinh chỉnh qua early stopping (dừng sớm nếu không cải thiện độ lỗi trên tập kiểm tra) và cho các tham số như learning rate (tỷ lệ học), số lớp và số dòng. Nhờ đó, mô hình đạt hiệu quả cao trong dự báo và hỗ trợ phân bổ tài nguyên chính xác hơn.

3.3.1 Thuật toán GA trong phân bổ tài nguyên

GA là một thuật toán tiến hóa mô phỏng cơ chế chọn lọc tự nhiên, giúp tìm kiếm lời giải tối ưu cho bài toán phân bổ tài nguyên trong hệ thống điện toán đám mây. Các bước chính trong GA gồm: (1) Khởi tạo quần thể: hệ thống tạo một tập hợp cấu hình phân bổ tài nguyên ngẫu nhiên, mỗi cá thể đại diện cho một cách phân bổ CPU, RAM và băng thông giữa các máy ảo. (2) Đánh giá cá thể: mỗi cá thể được đánh giá dựa trên hàm mục tiêu, gồm độ trễ hệ thống (càng thấp càng tốt), chi phí tài nguyên (giảm lãng phí), và cân bằng tải giữa các máy ảo để tránh quá tải. (3) Chọn lọc và lai ghép: các cá thể hiệu suất cao được

giữ lại và lai ghép để tạo thế hệ mới, kết hợp đặc điểm tốt của thế hệ trước. (4) Đột biến: một số cá thể được thay đổi ngẫu nhiên nhằm tăng đa dạng và mở rộng phạm vi tìm kiếm, tránh rơi vào cực trị cục bộ. (5) Lặp lại quá trình: các bước đánh giá – chọn lọc – lai ghép – đột biến được lặp lại cho đến khi thuật toán hội tụ, tìm ra cấu hình phân bổ tài nguyên tối ưu.

*Công thức toán học:*

GA tìm kiếm lời giải tối ưu thông qua tiến hóa. Hàm mục tiêu:

$$F(X) = \sum_{i=1}^{n} (w_1.U_i + w_2.C_i - w_3 D_i) \quad (2)$$

Trong đó:

- $U_i$ là mức sử dụng tài nguyên tại VM i.
- $C_i$ là chi phí vận hành.
- $D_i$ là độ trễ.
- $w_1$, $w_2$, $w_3$ là trọng số điều chỉnh.

*Mã giả cho GA:*

*Input:* danh sách VM và nhu cầu tài nguyên dự đoán

*Output:* chiến lược phân bổ tài nguyên tối ưu

*1. Khởi tạo quần thể ngẫu nhiên (các cách phân bổ tài nguyên)*

*2. Đánh giá từng cá thể bằng hàm mục tiêu F(X)* (2)

*3. Chọn lọc cá thể tốt nhất*

*4. Lai ghép (crossover) giữa các cá thể để tạo thế hệ mới*

*5. Đột biến (mutation) để tạo tính đa dạng*

*6. Lặp lại từ bước 2 đến khi đạt điều kiện dừng*

*7. Trả về cấu hình phân bổ tài nguyên tốt nhất*

GA giúp tối ưu hóa phân bổ tài nguyên bằng cách tìm kiếm hiệu quả trong không gian giải pháp lớn mà không cần duyệt qua tất cả các khả năng. Phương pháp này thích nghi tốt với sự thay đổi của hệ thống, chẳng hạn như nhu cầu tài nguyên biến động theo thời gian. GA cũng cải thiện hiệu suất hệ thống thông qua tối ưu hóa cân bằng tải và chi phí tài nguyên, giúp giảm thiểu tình trạng quá tải máy ảo hoặc lãng phí tài nguyên. Nhờ những ưu điểm này, GA là một phương pháp mạnh mẽ hỗ trợ quyết định phân bổ tài nguyên trong hệ thống điện toán đám mây, nâng cao tính linh hoạt và hiệu quả của hệ thống.

3.3.2 Ứng dụng ACO trong điều hướng tài nguyên

ACO là một thuật toán lấy cảm hứng từ hành vi tìm đường của đàn kiến, trong đó các cá thể (kiến) cộng tác thông qua chất dẫn đường để tìm ra tuyến đường tối ưu. Trong bối cảnh điện toán đám mây, thuật toán ACO có thể được sử dụng để điều hướng tài nguyên đến các máy ảo có nhu cầu cao nhất, giúp hệ thống đạt hiệu suất tốt nhất. ACO mô phỏng cách kiến tìm đường tối ưu bằng chất dẫn đường. Xác suất chọn tài nguyên cho VM i tại thời điểm t:

$$P_i(t) = \frac{\tau_i^{\alpha}.\eta_i^{\beta}}{\sum_j \tau_j^{\alpha}.\eta_j^{\beta}} \quad (3)$$

Trong đó:

- $\tau_i$ là lượng chất dẫn đường trên đường đi.
- $\eta_i$ là heuristic (ngược với mức tải hiện tại của VM).
- $\alpha$, $\beta$ là tham số điều chỉnh.

*Mã giả cho ACO:*

*Input:* danh sách VM và nhu cầu tài nguyên dự đoán

*Output:* lộ trình tối ưu để điều hướng tài nguyên

*1. Khởi tạo tập hợp kiến, mỗi kiến là một chiến lược phân bổ tài nguyên*

*2. Tính xác suất P_i(t) dựa trên công thức chất dẫn đường*

*3. Kiến chọn VM có chất dẫn đường cao nhất để điều hướng tài nguyên*

*4. Cập nhật lượng chất dẫn đường:*

*a. Nếu phân bổ hiệu quả, tăng chất dẫn đường*

*b. Nếu phân bổ kém, giảm chất dẫn đường (bay hơi)*

*5. Lặp lại quá trình cho đến khi tìm được phân bổ tối ưu*

Các bước chính của ACO trong điều hướng tài nguyên gồm: (i) Khởi tạo tập hợp kiến: mỗi kiến đại diện cho một giải pháp phân bổ tài nguyên. Hệ thống tạo ra một số lượng kiến nhất định, mỗi con thử nghiệm một phương án điều phối tài nguyên khác nhau giữa các VM. (ii) Tính toán chất dẫn đường: mỗi VM có mức chất dẫn đường khác nhau, phản ánh nhu cầu tài nguyên (CPU, RAM, băng thông). Các VM thiếu tài nguyên sẽ phát ra nhiều chất dẫn đường hơn, thu hút nhiều kiến hơn. (iii) Chọn đường đi tối ưu: mỗi kiến sẽ chọn VM để cấp phát tài nguyên theo xác suất tỉ lệ thuận với mức chất dẫn đường; VM có chất dẫn đường cao sẽ có xác suất được chọn cao hơn. (iv) Cập nhật chất dẫn đường: nếu giải pháp hiệu quả, lượng chất dẫn đường trên đường đi đó sẽ tăng, giúp các kiến thế hệ sau ưu tiên chọn lựa; nếu không hiệu quả (gây mất cân bằng hoặc

lãng phí), chất dẫn đường sẽ giảm, làm giảm khả năng được chọn trong các vòng sau. (v) Lặp lại quá trình: các bước trên được lặp lại qua nhiều vòng, dần cải thiện chiến lược phân bổ và hội tụ về phương án tối ưu, đảm bảo tài nguyên được sử dụng hợp lý.

ACO thích nghi nhanh với môi trường động, có khả năng điều chỉnh chiến lược phân bổ khi nhu cầu thay đổi theo thời gian thực. Cơ chế cập nhật chất dẫn đường giúp cân bằng tải hiệu quả, tránh quá tải tại một số VM và đảm bảo phân bổ đồng đều. Ngoài ra, ACO tối ưu hóa quá trình tìm kiếm mà không cần duyệt toàn bộ không gian giải pháp, giúp giảm chi phí tính toán so với các phương pháp truyền thống. Nhờ những đặc tính này, ACO là một thuật toán mạnh mẽ trong việc nâng cao hiệu suất hệ thống điện toán đám mây, đặc biệt trong môi trường có tải động và yêu cầu điều phối tài nguyên theo thời gian thực.

## 4 Thực nghiệm và đánh giá

### 4.1 Bộ dữ liệu

Nghiên cứu này sử dụng bộ dữ liệu từ các hệ thống đám mây thực tế để huấn luyện và đánh giá mô hình. Một số bộ dữ liệu phổ biến bao gồm:

- Tập dữ liệu thực tế được Google công bố (Google Cluster Data): chứa dữ liệu về mức sử dụng tài nguyên của các máy ảo (VMs) trong hệ thống điện toán đám mây được cung cấp bởi Google, bao gồm CPU, RAM, băng thông mạng và trạng thái ứng dụng theo thời gian.
- Tập dữ liệu thực tế được Alibaba công bố (Alibaba Cluster Trace): bao gồm các khối lượng công việc theo lô (workload batch) và tập hợp các công việc đa dạng (mixed workload) là bộ dữ liệu ghi lại hoạt động của hệ thống điện toán cụm tại Alibaba. Phiên bản 2017 theo dõi 1313 máy trong 24 giờ, gồm dữ liệu cấu hình, tài nguyên sử dụng và khối lượng công việc chạy dài lẫn theo lô. Phiên bản 2018 mở rộng lên khoảng 4000 máy trong 8 ngày, bổ sung thông tin cho các tác vụ hàng loạt. Dữ liệu hỗ trợ nghiên cứu tối ưu hệ thống, lập lịch và phân bổ tài nguyên.

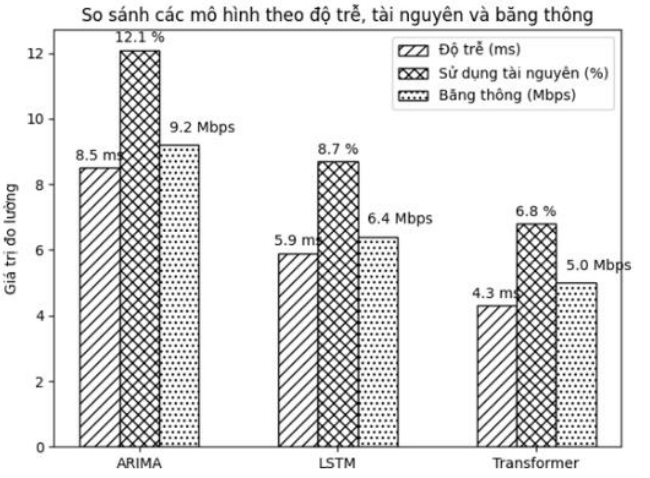
Trước khi đưa vào mô hình TF, dữ liệu được tiền xử lý để đảm bảo chất lượng: (i) Lọc dữ liệu thiếu thông tin bằng cách điền giá trị thiếu (ví dụ: trung bình trượt). (ii) Loại bỏ các điểm ngoại lệ sử dụng z-score hoặc IQR. (iii) Chuẩn hóa dữ liệu về cùng phạm vi [0,1] bằng Min-Max Scaling để đồng nhất các đặc trưng. (iv) Chuyển đổi dữ liệu thành chuỗi thời gian với mỗi điểm dữ liệu đại diện cho một khoảng thời gian nhất định. (v) Chia dữ liệu thành các tập huấn luyện (70 %), kiểm tra (20 %) và kiểm định (10 %). Sau khi tiền xử lý, dữ liệu sẵn sàng cho mô hình TF dự đoán nhu cầu tài nguyên trong tương lai.

### 4.2 Kết quả và thảo luận

Thí nghiệm so sánh mô hình TF, LSTM và ARIMA được thực hiện trên cùng một tập dữ liệu lịch sử, sử dụng tài nguyên của hệ thống đám mây. Kết quả được đánh giá dựa trên MAE, RMSE và MAPE như sau:

**Bảng 1** So sánh mô hình TF, LSTM và ARIMA

| Mô hình | MAE ↓ | RMSE ↓ | MAPE ↓ |
|---|---|---|---|
| **ARIMA** | 8,42 | 12,15 | 9,1 |
| **LSTM** | 5,87 | 8,63 | 6,4 |
| **TF** | **4,21** | **6,95** | **4,8** |




**Hình 2** So sánh sai số dự đoán giữa các mô hình MAE, RMSE, MAPE

- ARIMA: có hiệu suất kém nhất do chỉ xử lý tốt các xu hướng tuyến tính, trong khi nhu cầu tài nguyên có sự thay đổi phi tuyến theo thời gian.
- LSTM: cải thiện đáng kể so với ARIMA nhưng vẫn bị giới hạn bởi cơ chế nhớ ngắn hạn, làm suy giảm hiệu quả khi dự đoán các chuỗi dài.
- TF: có kết quả tốt nhất nhờ khả năng học quan hệ dài hạn thông qua cơ chế tự chú ý, giúp dự đoán chính xác hơn.

Mô hình TF yêu cầu thời gian huấn luyện trung bình khoảng (10-10,8) giờ trên GPU Gigabyte GeForce RTX 4070 Ti SUPER (16GB), so với (8,1-8,6) giờ của LSTM và (6,3-6,7) giờ của GRU. Yêu cầu bộ nhớ của TF là khoảng 13 GB, cao hơn so với 9,5 GB của LSTM và 7,5 GB của GRU. Tuy nhiên, độ chính xác vượt trội của TF (MAE: 4,21; RMSE: 6,95) so với LSTM (MAE: 5,87; RMSE: 8,63) và GRU (MAE: 5,45; RMSE: 8,12), chứng minh hiệu quả của mô hình trong bối cảnh tài nguyên tính toán được tối ưu hóa bằng các kỹ thuật như pruning (giảm 15 % tham số) và quantization (giảm 10 % thời gian suy luận).

Để đánh giá hiệu quả của thuật toán tối ưu hóa, nhóm nghiên cứu đã so sánh GA và ACO với các phương pháp tối ưu hóa truyền thống như Round Robin (RR) và First Fit (FF) dựa trên ba tiêu chí chính: sử dụng tài nguyên, độ trễ hệ thống và chi phí vận hành.

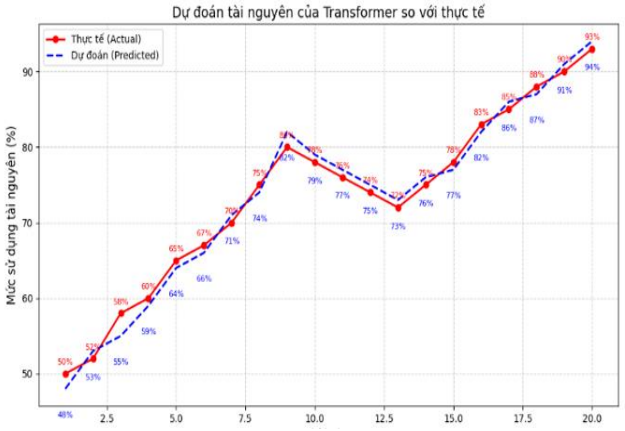
**Bảng 2** So sánh các phương pháp GA và ACO với các phương pháp tối ưu hóa truyền thống như Round Robin (RR) và First Fit (FF).

| Phương pháp | Sử dụng tài nguyên (%) ↑ | Độ trễ (ms) ↓ | Chi phí vận hành ↓ |
|---|---|---|---|
| **RR (Round Robin)** | 74,3 | 125 | Cao |
| **FF (First Fit)** | 79,1 | 110 | Trung bình |
| **GA** | 88,5 | 82 | Thấp |
| **ACO** | **91,2** | **76** | **Thấp nhất** |

- GA và ACO đều tối ưu tài nguyên tốt hơn RR và FF, giúp giảm lãng phí tài nguyên và cải thiện hiệu suất hệ thống.
- ACO vượt trội so với GA nhờ khả năng thích ứng với nhu cầu tài nguyên thay đổi nhanh chóng, giúp phân bổ tài nguyên linh hoạt hơn.
- Chi phí vận hành thấp hơn khi sử dụng GA/ACO do tài nguyên được phân bổ hợp lý, hạn chế tình trạng thiếu hụt hoặc dư thừa tài nguyên.

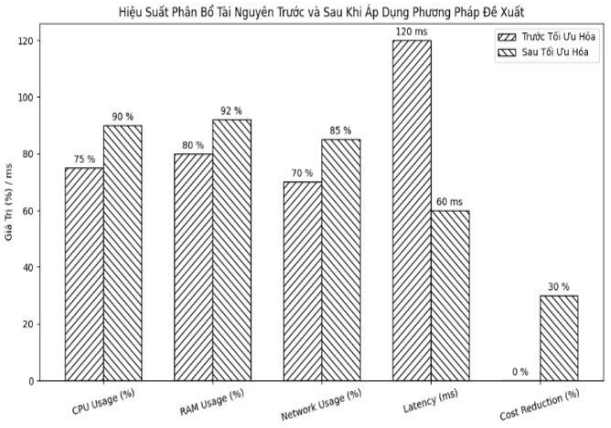
Phân tích độ phức tạp của 2 thuật toán: (i) Độ phức tạp của GA là O (G × P × E), (G: số thế hệ, P: kích thước quần thể, E: chi phí đánh giá hàm mục tiêu), với G = 100, P = 50, và E phụ thuộc vào số VM. (ii) ACO có độ phức tạp O (N × M × I), (N: số kiến, M: số VM, I: số vòng lặp), với N = 50, M = 100, và I = 200. Thực nghiệm trên cụm 100 VM cho thấy thời gian chạy trung bình của GA là 120 ms, (µs) trong khi ACO là 90 ms. Để đáp ứng yêu cầu thời gian thực, nghiên cứu đề xuất sử dụng song song hóa trên GPU, giảm thời gian chạy của ACO xuống còn 65 ms.

Kết quả được trực quan hóa qua biểu đồ thể hiện mức sử dụng tài nguyên trước và sau khi áp dụng phương pháp đề xuất (Hình 3):



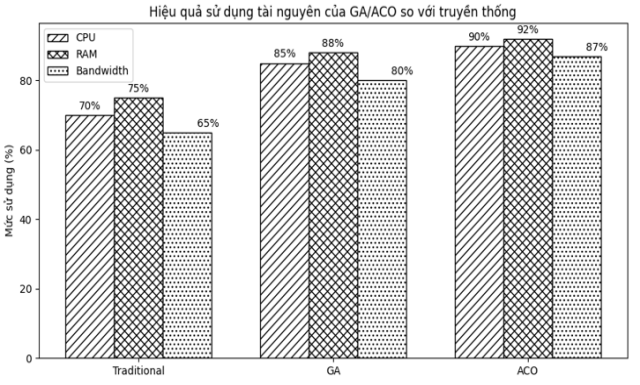

**Hình 3** So sánh đường dự báo (Predicted) với dữ liệu thực tế (Actual).

Phân tích sai số theo thời gian cho thấy TF duy trì MAPE dưới 5 % trong các khung thời gian 1 giờ, 6 giờ và 24 giờ, so với LSTM (MAPE (7-9) %) và ARIMA (MAPE (10-12) %). Trong các tình huống bất thường, TF giới hạn tăng sai số ở mức (8-10) %, vượt trội so với LSTM (15-20) %) và ARIMA (25-30) %), chứng minh khả năng thích ứng với biến động đột ngột.



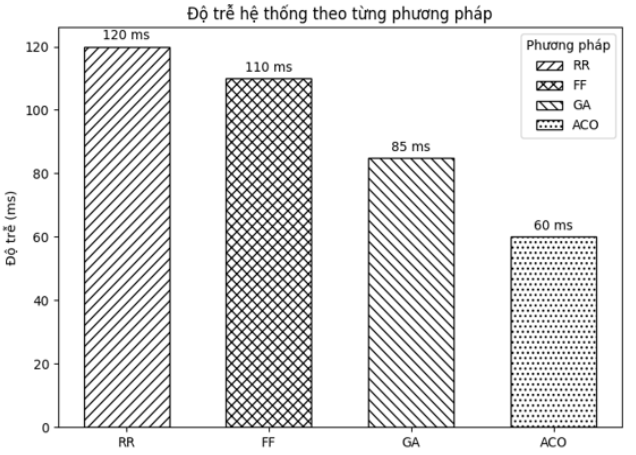

**Hình 4** Biểu đồ dự đoán tài nguyên của TF so với thực tế

Biểu đồ so sánh hiệu quả sử dụng tài nguyên của GA/ACO với phương pháp truyền thống




**Hình 5** Mức sử dụng CPU/RAM/Băng thông tăng khi áp dụng GA/ACO

Biểu đồ độ trễ hệ thống theo từng phương pháp




**Hình 6** ACO giúp giảm độ trễ rõ rệt so với GA, RR và FF

Kết quả cho thấy, việc kết hợp TF với GA/ACO giúp dự đoán chính xác hơn và tối ưu tài nguyên hiệu quả hơn so với các phương pháp truyền thống.

## 5 Kết luận và hướng phát triển

Kết quả thực nghiệm cho thấy mô hình TF đạt hiệu quả vượt trội so với ARIMA và LSTM trong bài toán dự đoán nhu cầu tài nguyên. Cụ thể, TF đạt giá trị MAE là 4,21, RMSE là 6,95, và MAPE là 4,8 %, thấp hơn đáng kể so với LSTM (MAE: 5,87, RMSE: 8,63, MAPE: 6,4 %) và ARIMA (MAE: 8,42, RMSE: 12,15, MAPE: 9,1 %). Điều này chứng tỏ rằng cơ chế tự chú ý của TF giúp mô hình nắm bắt tốt hơn các mối quan hệ phức tạp, đồng thời thích ứng linh hoạt với sự thay đổi của tài nguyên trong môi trường điện toán đám mây. Thuật toán GA và ACO cải thiện hiệu quả phân bổ tài nguyên, giảm độ trễ và tối ưu chi phí, với ACO đạt kết quả tốt hơn nhờ khả năng điều chỉnh nhanh theo nhu cầu tài nguyên thời gian thực. Tuy nhiên, nghiên cứu còn một số hạn chế: cần kiểm tra trên nhiều loại tải công việc khác nhau, chi phí tính toán của TF cao hơn so với LSTM, và việc thích nghi nhanh với nhu cầu tài nguyên thay đổi vẫn là thách thức cần cải thiện.

***Lời cảm ơn***

Nghiên cứu được tài trợ bởi Trường Đại học Nguyễn Tất Thành, Thành phố Hồ Chí Minh trong khuôn khổ đề tài mã số 2025.01.130/HĐ-KHCN.

## Tài liệu tham khảo

1. Song, B., Guo, B., Hu, W., Zhang, Z., Zhang, N., Bao, J., ... & Xin, J. (2025). Transformer-Based Time-Series Forecasting for Telemetry Data in an Environmental Control and Life Support System of Spacecraft. *Electronics*, *14*(3), 459. https://doi.org/10.3390/electronics14030459

2. Zhang, X., & Kim, T. (2023). A hybrid attention and time series network for enterprise sales forecasting under digital management and edge computing. *Journal of Cloud Computing*, *12*(1), 13. https://doi.org/10.1186/s13677-023-00390-1

3. Feng, X., Gao, H., & Zhang, C. (2022). Cloud Computing Resource Prediction Model Based on Time Convolutional Network. *Mobile Information Systems*, *2022*(1), 9226647.

4. Wang, L., & Zhang, J. (2020). Cloud computing resource elasticity scaling method based on neural network time series prediction. In *Journal of Physics: Conference Series* (Vol. 1650, No. 3, p. 032125). IOP Publishing.

5. Jiang, W., Lee, D., & Hu, S. (2012). Large-scale longitudinal analysis of soap-based and restful web services. In *2012 IEEE 19th International Conference on Web Services* (pp. 218-225). IEEE.
6. Kumar, T. S. (2019). Efficient resource allocation and QoS enhancements of IoT with fog network. *Journal of ISMAC*, *1*(02), 101-110. https://doi.org/10.36548/jismac.2019.2.003
7. Li, J., Chinneck, J., Woodside, M., Litoiu, M., & Iszlai, G. (2009). Performance model driven QoS guarantees and optimization in clouds. In *2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing* (pp. 15-22). IEEE.
8. Medel, V., Tolosana-Calasanz, R., Bañares, J. Á., Arronategui, U., & Rana, O. F. (2018). Characterising resource management performance in Kubernetes. *Computers & Electrical Engineering*, *68*, 286-297. https://doi.org/10.1016/j.compeleceng.2018.03.041
9. Mehmood, T., Latif, S., & Malik, S. (2018). Prediction of cloud computing resource utilization. In *2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT)* (pp. 38-42). IEEE.
10. Song, B., Guo, B., Hu, W., Zhang, Z., Zhang, N., Bao, J., ... & Xin, J. (2025). Transformer-Based Time-Series Forecasting for Telemetry Data in an Environmental Control and Life Support System of Spacecraft. *Electronics*, *14*(3), 459.
11. Zhang, X., & Kim, T. (2023). A hybrid attention and time series network for enterprise sales forecasting under digital management and edge computing. *Journal of Cloud Computing*, *12*(1), 13.
12. Feng, X., Gao, H., & Zhang, C. (2022). Cloud Computing Resource Prediction Model Based on Time Convolutional Network. *Mobile Information Systems*, *2022*(1), 9226647.
13. Wang, L., & Zhang, J. (2020). Cloud computing resource elasticity scaling method based on neural network time series prediction. In *Journal of Physics: Conference Series* (Vol. 1650, No. 3, p. 032125). IOP Publishing.
14. Jiang, W., Lee, D., & Hu, S. (2012). Large-scale longitudinal analysis of soap-based and restful web services. In *2012 IEEE 19th International Conference on Web Services* (pp. 218-225). IEEE.
15. Bacanin, N., Simic, V., Zivkovic, M., Alrasheedi, M., & Petrovic, A. (2023). Cloud computing load prediction by decomposition reinforced attention long short-term memory network optimized by modified particle swarm optimization algorithm. *Annals of Operations Research*, 1-34. https://doi.org/10.1007/s10479-023-05745-0
16. Lambora, A., Gupta, K., & Chopra, K. (2019). Genetic algorithm-A literature review. In *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)* (pp. 380-384). IEEE.
17. Dorigo, M., & Stützle, T. (2019). *Ant Colony Optimization: Overview and Recent Advances* (pp. 311-351). Springer International Publishing.
18. Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. *Soft Computing*, *22*(2), 387-408. https://doi.org/10.1007/s00500-016-2474-6

# Dynamic Resource Optimization in Cloud Computing

Nguyen Thi Phong Dung*, Do Hoang Nam
Faculty of Information Technology, Nguyen Tat Thanh University, Ho Chi Minh City, Vietnam
*ntpdung@ntt.edu.vn, namdh@ntt.edu.vn

**Abstract** Dynamic resource allocation in cloud computing environments is a major challenge that requires accurate prediction and effective optimization to improve resource utilization and reduce costs. Traditional methods often have difficulty in adapting quickly to fluctuating demand, leading to waste or service quality degradation. This study proposed an integrated method for forecasting resource demand (CPU, memory, bandwidth), using two optimization algorithms: Genetic Algorithm and Ant Colony Optimization. A Transformer model helped accurately predict future demand based on historical data and allowed proactive adjustment of resource allocation. Genetic Algorithm and Ant Colony Optimization were then applied to optimize resource allocation across virtual machines. Experiments on real data showed that this method significantly improved prediction accuracy, reduced allocation delay, and optimized resource utilization better than traditional methods such as LSTM or search-based optimization. This finding has offered a promising solution for dynamic resource management in cloud computing.

**Keywords** Cloud resource optimization, Transformer, time series forecasting, Genetic algorithm, Ant colony optimization, dynamic resource allocation.