Thuật toán PSO cải tiến trong cung cấp tài nguyên cho dịch vụ ảo hóa dựa trên nền tảng máy chủ chia sẻ không đồng nhất

The Improved PSO Algorithm in Resource Allocation for Virtual Service based on Heterogeneous Shared Hosting Platforms

Phạm Nguyễn Minh Nhựt, Lê Văn Sơn, Hoàng Bảo Hùng

Abstract: *Providing resource for virtual services in cloud computing which requires saving the resource and minimizing the amount of energy consumption is critical. In this study, we propose the resource model and linear programming formulation for multi-dimensional resource allocation problem. Based on the Particle Swarm Optimization algorithm, RA-PSO algorithm was designed to solve and evaluate through CloudSim simulation tool compared with FirstFit Decreasing (FFD) algorithm. The parameters include the number of physical machines being used and the amount of energy consumption. The experimental results show that the proposed RA-PSO algorithm yields a better performance than FFD algorithm.*

Keywords: *Resource Allocation, Cloud Computing, Virtual machine, Particle Swarm Optimization.*

I. GIỚI THIỆU

Với sự phát triển về công nghệ và khả năng ứng dụng của điện toán đám mây, nhu cầu sử dụng các máy vật lý cho dịch vụ ảo hóa tại các trung tâm dữ liệu ngày càng tăng. Điều này dẫn đến việc gia tăng năng lượng tiêu thụ trong các trung tâm dữ liệu, có thể trở thành mối đe dọa đối với môi trường sống. Vì thế, tối ưu trong cung cấp tài nguyên cho dịch vụ ảo hóa tại các trung tâm dữ liệu, đáp ứng nhu cầu về chất lượng dịch vụ và giảm thiểu tối đa năng lượng tiêu thụ là vấn đề quan trọng.

Cung cấp tài nguyên với các ràng buộc tối thiểu năng lượng tiêu thụ cho dịch vụ ảo hóa đang được nhiều nhà nghiên cứu quan tâm. Trong đó, Eugen Feller và các cộng sự [6] đã đưa ra mô hình cung cấp tài nguyên cho dịch vụ ảo hóa. Họ sử dụng thuật toán tối ưu "đàn kiến" để ước lượng. Kết quả mô phỏng đã chứng minh rằng, năng lượng tiêu thụ của hệ thống giảm khi số máy vật lý giảm. Tuy nhiên, họ xem xét nền tảng máy vật lý đồng nhất và thực nghiệm trên dữ liệu mô phỏng, còn chúng tôi xem xét trong môi trường không đồng nhất, tức là cấu hình tài nguyên của máy vật lý không giống nhau và thực nghiệm trên dữ liệu thực tế được đưa ra trong [1, 2]. Mark Stillwell và các cộng sự [11] đã trình bày bài toán cung cấp tài nguyên dưới dạng bài toán quy hoạch tuyến tính và sử dụng thuật toán *FFD* để ước lượng. Ngược lại, Thomas Setser [12] cho rằng thuật toán này có xu hướng dẫn đến lãng phí tài nguyên. Vì vậy, chúng tôi dựa trên thuật toán PSO, đề xuất thuật toán *RA-PSO* để giải. Trong các tài liệu [4, 5, 7, 10], các tác giả đã giới thiệu phương pháp cung cấp tài nguyên với mục tiêu tối thiểu năng lượng tiêu thụ của hệ thống, nhưng họ chỉ tập trung xem xét đến việc tiêu thụ năng lượng trên CPU của máy vật lý. Trong khi đó, các tác giả trong tài liệu [8, 9] cho rằng, việc tiêu thụ năng lượng không chỉ trên CPU mà còn trên các tài nguyên khác, ví dụ: đĩa cứng, băng thông, RAM, …

Vì vậy, trong nội dung bài báo này, chúng tôi xem xét bài toán cung cấp tài nguyên (tài nguyên vật lý) đa chiều cho dịch vụ ảo hóa từ nền tảng máy chủ chia sẻ không đồng nhất, với mục tiêu tối thiểu năng lượng tiêu thụ trên tất cả các tài nguyên thông qua việc tối thiểu số máy vật lý được dùng. Những kết quả chính của bài báo được tóm tắt như sau:

(a) Xây dựng mô hình nhu cầu tài nguyên và cung cấp tài nguyên, mô hình tiêu thụ năng lượng của nền tảng máy chủ chia sẻ không đồng nhất khi cung cấp tài nguyên cho dịch vụ ảo hóa, với ràng buộc mỗi dịch vụ ảo hóa là một máy ảo đơn lẻ;

(b) Phát biểu bài toán cung cấp tài nguyên từ nền tảng máy chủ chia sẻ không đồng nhất dưới dạng bài toán quy hoạch tuyến tính;

(c) Xây dựng thuật toán *RA-PSO* được cải tiến từ thuật toán PSO và sử dụng công cụ CloudSim [3] để giải. So sánh năng lượng tiêu thụ, số máy vật lý được dùng và thời gian thực thi thuật toán giữa *RA-PSO* và *FFD*, thông qua dữ liệu thực tế.

Phần còn lại của bài báo được tổ chức như sau: Mục 2 trình bày mô hình của bài toán dưới dạng bài toán quy hoạch tuyến tính và mô hình tiêu thụ năng lượng. Mục 3 trình bày thuật toán *RA-PSO* và mục 4 trình bày phương pháp thực nghiệm, kết quả đánh giá. Phần kết luận và đề xuất hướng phát triển được giới thiệu ở mục 5.

II. MÔ HÌNH CUNG CẤP TÀI NGUYÊN CHO DỊCH VỤ ẢO HÓA

II.1. Mô hình tài nguyên và nhu cầu tài nguyên

Xét một nền tảng máy chủ chia sẻ không đồng nhất gồm các máy vật lý có cấu hình tài nguyên khác nhau, được kết nối bằng thiết bị mạng tốc độ cao để chia sẻ tài nguyên [11]. Khi hệ thống nhận được yêu cầu cung cấp tài nguyên thì bộ cung cấp tài nguyên có nhiệm vụ ra quyết định từ chối hoặc đáp ứng yêu cầu, phân chia tỷ lệ tài nguyên từ các máy vật lý cho dịch vụ ảo hóa.

Đối với mỗi loại tài nguyên tại một máy vật lý có thể có một hoặc nhiều phần tử tài nguyên riêng biệt (ví dụ, một hoặc nhiều CPU, một hoặc nhiều RAM, …). Vì thế, tài nguyên của mỗi máy vật lý được biểu diễn bởi một cặp thứ tự các vector. Trong đó, vector *tài nguyên thành phần* biểu diễn tài nguyên của một phần tử đơn lẻ và vector *tài nguyên tổng hợp* biểu diễn tổng tài nguyên được tính trên tất cả các phần tử.

Tương tự, đối với nhu cầu tài nguyên của dịch vụ ảo hóa cũng được biểu diễn bởi cặp thứ tự các vector, bao gồm vector *nhu cầu tài nguyên thành phần* và vector *nhu cầu tài nguyên tổng hợp*. Hơn nữa, nhu cầu tài nguyên gồm có hai loại: *nhu cầu tất yếu* và *nhu cầu tùy biến* [11]. Nhu cầu tất yếu biểu thị phần cụ thể của tài nguyên yêu cầu. Dịch vụ không hưởng lợi từ phần lớn hơn và không thể hoạt động với phần nhỏ hơn từ tài nguyên được cung cấp. Nhu cầu tùy biến biểu thị phần bổ sung của tài nguyên mà dịch vụ có thể sử dụng. Dịch vụ không hưởng lợi từ phần lớn hơn nhưng có thể hoạt động với phần nhỏ hơn với chi phí giảm.

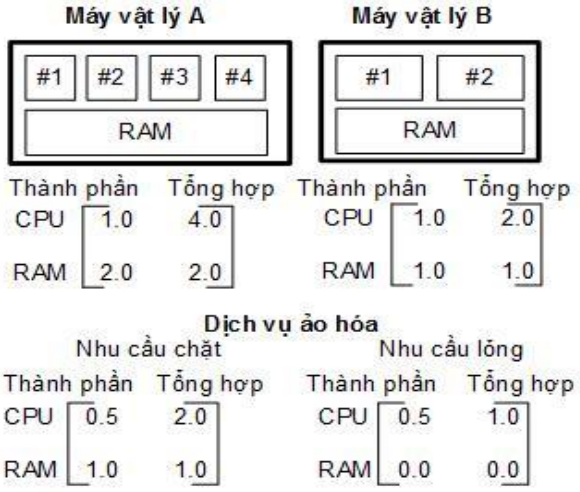
Như vậy, nhu cầu tất yếu tài nguyên k của dịch vụ i được biểu diễn bởi một cặp vector thứ tự (r_{ik}^e, r_{ik}^a), biểu thị nhu cầu tài nguyên cần thiết để chạy dịch vụ ở mức tối thiểu chấp nhận được. Nếu nhu cầu tài nguyên tất yếu không được đáp ứng thì việc cung cấp tài nguyên thất bại. Nhu cầu tùy biến tài nguyên k của dịch vụ i được biểu diễn bởi một cặp vector thứ tự (f_{ik}^e, f_{ik}^a), biểu thị tài nguyên bổ sung cần thiết để chạy dịch vụ ở mức tối đa. Do vậy, nhu cầu tùy biến có thể được biểu diễn bởi tích số giữa nhu cầu tùy biến với hệ số yêu cầu bổ sung và được gọi là *nhu cầu tùy biến ràng buộc*.

Việc sử dụng tài nguyên đối với nhu cầu tùy biến thường là quan hệ tuyến tính. Chẳng hạn, nếu dịch vụ được cung cấp chỉ một nửa tài nguyên CPU so với nhu cầu, thì khả năng nó chỉ sử dụng một nửa băng thông I/O so với nhu cầu. Điều này phù hợp với thực tế vì khi tài nguyên CPU cần cung cấp giảm, dẫn đến tiêu hao tài nguyên khác cũng bị giảm (trong trường hợp này là băng thông I/O). Như vậy, để đơn giản hệ số bổ sung của tất cả nhu cầu tùy biến có thể biểu diễn cùng một giá trị và giá trị của nó nằm trong khoảng 0 và 1. Một dịch vụ với hệ số bổ sung bằng 0 sẽ thực thi ở trạng thái không có nhu cầu tài nguyên tùy biến, trong khi một dịch vụ với hệ số bổ sung bằng 1 sẽ được thực thi ở mức tài nguyên được cung cấp tối đa. Do đó, nhu cầu tài nguyên k của dịch vụ ảo hóa i tại máy vật lý j được xác định bởi $(r_{ik}^e + Q_{ij} \times f_{ik}^e, r_{ik}^a + Q_{ij} \times f_{ik}^a)$. Trong đó, Q_{ij} là hệ số bổ sung nhu cầu tài nguyên tùy biến của dịch vụ i tại máy vật lý j.

Việc cung cấp tài nguyên bị lỗi nếu nhu cầu tài nguyên tổng hợp của dịch vụ ảo hóa vượt quá dung

lượng tài nguyên máy vật lý. Tương tự như [2], chúng tôi cũng giả định rằng nhu cầu tài nguyên thành phần của dịch vụ ảo hóa không vượt quá dung lượng tài nguyên thành phần của máy vật lý.

Hình 1, minh họa về mô hình tài nguyên và nhu cầu tài nguyên, gồm có 2 máy vật lý và một dịch vụ ảo hóa. Trong đó, máy vật lý A gồm có 4 lõi CPU (tức là, 4 tài nguyên CPU thành phần) và 1 bộ nhớ RAM (tức là, 1 tài nguyên RAM thành phần). Như vậy, cặp vector tài nguyên CPU là $(1.0,\ 4.0)$ và cặp vector tài nguyên RAM là $(2.0, 2.0)$. Máy vật lý B có 2 lõi và 1 RAM nên cặp vector tài nguyên CPU là $(1.0, 2.0)$ và cặp vector tài nguyên RAM là $(1.0, 1.0)$. Đối với dịch vụ ảo hóa, cặp vector nhu cầu tài nguyên CPU là $(0.5 + Q_{ij} \times 0.5, 2.0 + Q_{ij} \times 1.0)$, cặp vector nhu cầu tài nguyên RAM là $(1.0 + Q_{ij} \times 0.0,\ 1.0+Q_{ij} \times 0.0)$. Nếu Q_{ij} = 1 thì chỉ máy vật lý A đáp ứng nhu cầu cung cấp tài nguyên cho dịch vụ ảo hóa. Ngược lại, $Q_{ij} = 0.5$, thì máy vật lý B cũng đáp ứng việc cung cấp tài nguyên nhưng với chi phí tài nguyên giảm.



Hình 1. Ví dụ tài nguyên của 2 máy vật lý và nhu cầu tài nguyên của 1 dịch vụ ảo hóa.

II.2. Hàm mục tiêu và các ràng buộc

Giả định mỗi dịch vụ ảo hóa là một máy ảo đơn lẻ có nhu cầu tài nguyên không đổi. Chúng tôi xây dựng bài toán cung cấp tài nguyên đa chiều cho dịch vụ ảo hóa dựa trên nền tảng máy chủ chia sẻ không đồng nhất (HMDRAP) trên cơ sở bài toán quy hoạch tuyến tính với các biến hữu tỷ và nguyên, như sau:

Xét S_i dịch vụ, với $i = 1, \dots, n$; $S_i > 0$, số máy vật lý không đồng nhất là Y_j với $j = 1, \dots, m$; $Y_j > 0$. Mỗi máy vật lý cung cấp D_k loại tài nguyên, với $k = 1, \dots, d$. Sử dụng r_{ik}^e, r_{ik}^a để biểu thị nhu cầu tài nguyên tất yếu thành phần và nhu cầu tài nguyên tất yếu tổng hợp tương ứng, f_{ik}^e, f_{ik}^a là nhu cầu tài nguyên tùy biến thành phần và nhu cầu tài nguyên tùy biến tổng hợp. Tương tự, C_{jk}^e, C_{jk}^a biểu thị tài nguyên thành phần và tài nguyên tổng hợp của máy vật lý j đối với loại tài nguyên k và Q_{ij} là hệ số bổ sung tài nguyên của dịch vụ i trên máy vật lý j. Biến nhị phân x_{ij} có giá trị 1 nếu dịch vụ i được cung cấp tài nguyên từ máy vật lý j và bằng 0 nếu ngược lại. Cuối cùng, số máy vật lý để cung cấp tài nguyên cho S_i dịch vụ ảo hóa là biến nhị phân y_j. Với những ký hiệu như trên, bài toán cung cấp tài nguyên đa chiều cho dịch vụ ảo hóa được biểu diễn với các ràng buộc và hàm mục tiêu như sau:

$$x_{ij} \in \{0,1\}, Q_{ij} \in [0,1], \quad \forall i,j \tag{1}$$

$$\sum_j x_{ij} = 1, \quad \forall i \tag{2}$$

$$y_j \geq x_{ij}, \quad \forall i,j \tag{3}$$

$$\left(r_{ik}^e + Q_{ij} f_{ik}^e\right) x_{ij} \leq C_{jk}^e, \quad \forall i,j,k \tag{4}$$

$$\sum_i (r_{ik}^a + Q_{ij} f_{ik}^a) x_{ij} \leq C_{jk}^a, \quad \forall j,k \tag{5}$$

$$\text{Và mục tiêu là} \quad min \sum_j y_j \tag{6}$$

Ràng buộc (1) xác định phạm vi của các biến. Ràng buộc (2) thể hiện trạng thái một dịch vụ i chỉ được cung cấp tài nguyên từ một máy vật lý j. Ràng buộc (3) mô tả trạng thái mà máy vật lý j đã được sử dụng hay không. Ràng buộc (4) biểu thị trạng thái mà tổng nhu cầu tài nguyên thành phần loại k của dịch vụ i tại các máy vật lý j không vượt quá năng lực tài nguyên thành phần của máy vật lý j. Ràng buộc (5) thể hiện trạng thái mà tổng nhu cầu tài nguyên tổng hợp loại k của các dịch vụ ảo hóa tại các máy vật lý j không được vượt quá năng lực tài nguyên tổng hợp của nó. Cuối cùng, ràng buộc (6) với mục tiêu là tối thiểu số lượng máy vật lý y_j được dùng để cung cấp cho S_i dịch vụ ảo hóa.

II.3. Mô hình tiêu thụ năng lượng

Để ước lượng năng lượng tiêu thụ của máy vật lý, chúng tôi thừa kế [4] đề xuất công thức tính nguồn điện năng tiêu thụ tại máy vật lý j là hàm tuyến tính $P(u_j)$ như công thức sau:

$$P(u_j) = (P_{max} - P_{idle}) \times u_j + P_{idle} \ , \ \forall j \qquad (7)$$

Trong đó, P_{max} và P_{idle} là công suất của máy vật lý j tương ứng ở trạng thái sử dụng tiện ích tài nguyên tối đa và trạng thái không hoạt động. Để mở rộng cho tất cả các loại tài nguyên, thì u_j là tổng tiện ích sử dụng của tất cả tài nguyên trên máy vật lý j, $u_j \in [0, 1]$ và được tính theo công thức (8).

$$u_j = \sum_k \frac{U_{jk}}{C_{jk}^a} \quad , \quad \forall j \qquad (8)$$

Trong đó, U_{jk} là tài nguyên thứ k của máy vật lý j đã cung cấp cho dịch vụ ảo hóa và C_{jk}^a là dung lượng tài nguyên tổng hợp loại k trên máy vật lý j. Những máy vật lý không được sử dụng sẽ được tắt và năng lượng tiêu thụ bằng 0. Do vậy, năng lượng tiêu thụ của Y_j máy vật lý khi cung cấp tài nguyên trong khoảng thời gian t được thiết lập như sau:

$$E(t) = \begin{cases} t \times \sum_j P(u_j), & nếu \ \ u_j \neq 0 \\ 0 & trường \ hợp \ khác \end{cases} \qquad (9)$$

Cung cấp tài nguyên như trình bày ở trên là bài toán tối ưu hóa tổ hợp. Để giải các bài toán thuộc lớp này, đến nay người ta thường dùng các tiếp cận: tìm kiếm heuristic để tìm lời giải đủ tốt; hoặc tìm kiếm cục bộ để tìm lời giải tối ưu địa phương; hay tìm lời giải gần đúng nhờ các thuật toán mô phỏng tự nhiên: mô phỏng luyện kim, di truyền, … Trong nội dung sau đây, chúng tôi đề xuất một thuật toán cải tiến dựa trên thuật toán PSO [13] để giải.

III. THUẬT TOÁN PSO CẢI TIẾN

Thuật toán PSO là kỹ thuật tối ưu ngẫu nhiên dựa trên kinh nghiệm bầy đàn, được phát triển bởi Eberhart và Kennedy vào năm 1995 [13]. Hiện nay, PSO được áp dụng thành công để giải các bài toán tối ưu trong nhiều lĩnh vực khác nhau. Trong PSO, mỗi giải pháp được xây dựng bởi một "con chim" và được gọi là *particle*. Mỗi *particle* có hai tham số: vị trí và vận tốc. Trong đó, vị trí của *particle* kết hợp với giá trị hàm thích nghi của nó để đánh giá và lựa chọn giải pháp tối ưu. Đầu tiên, PSO được khởi tạo với một quần thể các *particle* ngẫu nhiên và sau đó tìm kiếm tối ưu bằng cách cập nhật giải pháp qua nhiều thế hệ. Trong mỗi lần lặp, mỗi *particle* được cập nhật bởi hai giá trị "tốt nhất" (dựa trên hàm thích nghi). Một là, giải pháp tốt nhất so với các giải pháp mà mỗi *particle* đã đạt được trong mỗi vòng lặp và được gọi là giá trị tốt nhất cục bộ, *pbest*. Hai là, giá trị "tốt nhất" nhận được từ trước cho đến nay của một *particle* trong quần thể và gọi là giá trị tốt nhất toàn cục, *gbest*. Sau khi tìm thấy hai giá trị tốt nhất, các *particle* được cập nhật vận tốc và vị trí của nó theo công thức sau:

$$V_p^{t+1} = \omega V_p^t + c_1 r_1 (X_{pBest}^t - X_p^t) + c_2 r_2 (X_{gBest}^t - X_p^t) \qquad (10)$$

$$X_p^{t+1} = X_p^t + V_p^{t+1} \qquad (11)$$

Trong đó, V_p^t, V_p^{t+1} là vận tốc của *particle* thứ p ở thời điểm trước và sau khi vật tốc được cập nhật. X_p^t, X_p^{t+1} là vị trí của *particle* thứ p trước và sau khi vị trí được cập nhật. X_{pBest}^t và X_{gBest}^t là vị trí của *particle* tương ứng với giải pháp tốt nhất cục bộ và toàn cục. r_1, r_2 là các số ngẫu nhiên $\in [0, 1]$. c_1, c_2 là các hệ số gia tốc và ω là hệ số quán tính.

Áp dụng để giải bài toán HMDRAP, thuật toán PSO [13] truyền thống được cải tiến như sau: thứ nhất, PSO truyền thống thích hợp để giải bài toán tối ưu hóa liên tục và không phù hợp để giải bài toán tối ưu rời rạc. Do vậy, các tham số và phép toán phải được định nghĩa lại. Thứ hai, cấu trúc của các *particle* phải được xây dựng lại cho phù hợp với bài toán HMDRAP và cách thức cập nhật vị trí và vận tốc của các *particle* cũng phải được thay đổi.

III.1. Vị trí của các particle

Vị trí của các *particle* là một vector m bit $X_p^t = (x_{p1}^t, x_{p2}^t, \dots x_{pm}^t)$, biểu diễn một giải pháp cung cấp tài nguyên tại thời điểm t. Trong đó, m là số máy vật lý được dùng để cung cấp tài nguyên.

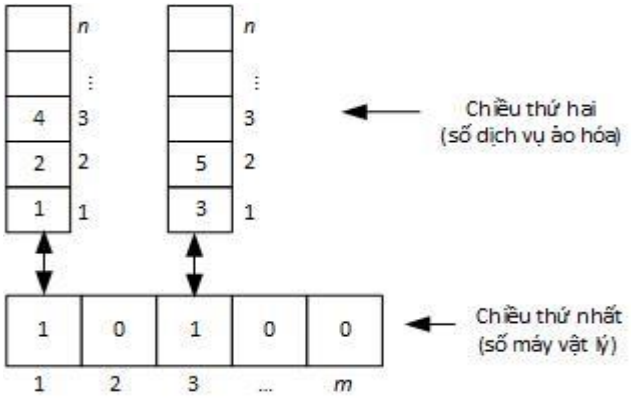
III.2. Vận tốc của các particle

Vận tốc của các *particle* là một vector m bit $V_p^t = (v_{p1}^t, v_{p2}^t, \dots v_{pm}^t)$, nó quyết định sự điều chỉnh

của việc cung cấp tài nguyên. V_p^t hướng các toán tử cập nhật để đạt được giải pháp tối ưu. Giá trị của các phần tử trong vector V_p^t là các *bit* nhị phân. Nếu *bit* bằng 0 thì việc cung cấp tài nguyên được ước tính lại. Ngược lại, bằng 1 thì không thay đổi.

III.3. Cấu trúc của các particle

Mỗi *particle* có cấu trúc 2 chiều: chiều thứ nhất có độ dài m phần tử (m là số máy vật lý). Chỉ số của mỗi phần tử trong chiều này biểu diễn chỉ số của máy vật lý và giá trị của nó là 0 hoặc 1. Giá trị 0 biểu diễn trạng thái mà không có máy vật lý nào được sử dụng và giá trị 1 biểu diễn cho trạng thái có máy vật lý được dùng. Chiều thứ hai có n phần tử (n là số dịch vụ ảo hóa) biểu diễn trạng thái mà dịch vụ ảo hóa được cung cấp tài nguyên từ các máy vật lý.



Hình 2. Cấu trúc của một particle

Hình 2 mô phỏng cấu trúc của một *particle* biểu diễn việc cung cấp tài nguyên từ m máy vật lý cho n dịch vụ ảo hóa. Trong đó, máy vật lý 1 được sử dụng để cung cấp tài nguyên cho các dịch vụ ảo hóa 1, 2 và 4. Máy vật lý 3 được sử dụng để cung cấp tài nguyên cho các dịch vụ ảo hóa 3 và 5. Các máy vật lý còn lại không được sử dụng. Như vậy, số lượng máy vật lý được dùng để cung cấp cho 5 dịch vụ ảo hóa là 2.

III.4. Các toán tử cập nhật vật tốc particle

- Toán tử cộng: được ký hiệu $\oplus$ để thay thế phép toán cộng trong biểu thức cập nhật vật tốc các *particle* của PSO truyền thống. Kết quả của biểu thức cộng phụ thuộc vào giá trị các *bit* phần tử của V_p^t kèm với xác suất P. Tức là, biểu thức $P_1V_1^t \oplus P_2V_2^t \oplus ... \oplus P_nV_n^t$ chỉ ra rằng vận tốc của một *particle* sau khi cập nhật sẽ có giá trị là V_1^t với một xác suất P_1, hoặc V_2^t với một xác suất P_2,…, hoặc V_n^t với một xác suất P_n. Ví dụ, 0.3(0, 0, 1) $\oplus$ 0.7(0, 1, 1) = (0, #, 1) có nghĩa là *bit* # bằng 0 với xác suất 0.3, hoặc bằng 1 với xác suất 0.7. Trong biểu thức cập nhật vận tốc cho các *particle* có 3 toán hạng, nên trong thuật toán PSO cải tiến, chúng tôi sử dụng 3 tác nhân xác suất là P_1, P_2, P_3 và giá trị của nó là một số ngẫu nhiên $\in [0, 1]$.

- Toán tử trừ: được ký hiệu $\ominus$ để thay thế phép toán trừ trong biểu thức cập nhập vận tốc các *particle* của PSO truyền thống. Nó được sử dụng để tính toán sự khác nhau giữa hai giải pháp cung cấp tài nguyên. Giá trị của biểu thức trừ phụ thuộc vào giá trị của các *bit* phần tử trong vector X_p^t. Tức là, giá trị của biểu thức $X_1^t \ominus X_2^t$ được tính theo quy luật như sau: nếu hai phần tử *bit* cùng vị trí tương ứng trong X_1^t và X_2^t có giá trị bằng nhau, thì phần tử kết quả có giá trị là 1. Ngược lại, có giá trị là 0. Ví dụ, (1, 1, 0) $\ominus$ (1, 0, 1)= (1, 0, 0), vì *bit* 1 của hai vector vế bên trái của biểu thức có giá trị giống nhau và giá trị của *bit* 2 và *bit* 3 của hai vector khác nhau.

III.5. Toán tử cập nhật vị trí particle

Toán tử cập nhật vị trí được ký hiệu $\otimes$ để thay thế phép toán cộng trong biểu thức cập nhật vị trí các *particle* của PSO truyền thống. Giá trị *bit* phần tử của X_p^t có được thay đổi hay không phụ thuộc vào giá trị *bit* ở vị trí tương ứng của V_p^{t+1}. Cụ thể, giá trị của biểu thức $X_p^t \otimes V_p^{t+1}$ được tính theo quy luật sau: nếu giá trị *bit* của vector V_p^{t+1} bằng 1 thì giá trị *bit* tương ứng của vector X_p^t sẽ không thay đổi. Ngược lại, giá trị *bit* của vector X_p^t sẽ được thay đổi. Ví dụ, X_p^t = (1, 0, 1) và V_p^{t+1} = (1, 1, 0) thì biểu thức $X_p^t \otimes V_p^{t+1}$ cho biết rằng, vị trí *bit* thứ 3 của X_p^t sẽ được thay đổi. Điều này tương ứng trạng thái mà máy vật lý thứ 3 phải được cập nhật lại quá trình cung cấp tài nguyên cho dịch vụ ảo hóa.

Với các định nghĩa như trên và thừa kế từ các công thức (10) và (11). Công thức cập nhật vận tốc và vị trí

của các *particle* được trình bày như sau:

$$V_p^{t+1} = P_1 V_p^t \oplus P_2(X_{pBest}^t \ominus X_p^t) \oplus P_3(X_{gBest}^t \ominus X_p^t) \quad (12)$$

$$X_p^{t+1} = X_p^t \circledast V_p^{t+1} \quad (13)$$

III.6. Hàm thích nghi

Đối với thuật toán RA-PSO, qua mỗi lần lặp vị trí và vận tốc của các *particle* được cập nhật dựa trên giá trị tốt nhất cục bộ và tốt nhất toàn cục. Các giá trị này có được thông qua ước lượng *hàm thích nghi*. Mục tiêu của bài toán là tối thiểu số máy vật lý được dùng, tức là tối đa tiện ích tài nguyên được cung cấp. Nên, hàm thích nghi của mỗi *particle* tại thời điểm t được xác định dựa vào tiện ích sử dụng tài nguyên của các dịch vụ ảo hóa được cấp tài nguyên từ một máy vật lý j, như sau:

Gọi F_{ijk}^t là tổng tài nguyên thứ k mà dịch vụ ảo hóa i được cung cấp tài nguyên từ máy vật lý j. Dựa vào công thức (5), F_{ijk}^t được tính theo công thức (10).

$$F_{ijk}^t = (r_{ik}^a + Q_{ij} f_{ik}^a),\ \forall i,j,k \quad (14)$$

Do vậy, hàm thích nghi được tính như công thức (11).

$$f_j = \sum_i \left(\sum_k \frac{F_{ijk}^t}{C_{jk}^a} \right),\ \forall j \quad (15)$$

Trong đó, C_{jk}^a là tài nguyên tổng hợp của máy vật lý j đối với loại tài nguyên k (đã nêu ra trong mục II.2). Những *particle* với giá trị hàm thích nghi cực đại sẽ được chọn khi cập nhật vị trí.

Mã giả của thuật toán *RA-PSO* để giải bài toán HMDRAP được trình bày như *Thuật toán 1*.

Thuật toán 1: *RA-PSO*

Đầu vào:

- Số dịch vụ ảo hóa *S*, số loại tài nguyên *D*, nhu cầu tất yếu tài nguyên thành phần r_{ik}^e, nhu cầu tất yếu tài nguyên tổng hợp r_{ik}^a, nhu cầu tùy biến tài nguyên thành phần f_{ik}^e, nhu cầu tất yếu tài nguyên tổng hợp f_{ik}^a, và hệ số bổ sung Q_{ij}.
- Số máy vật lý *Y*, tài nguyên thành phần C_{jk}^e, tài nguyên tổng hợp C_{jk}^a.

Đầu ra: Danh sách các dịch vụ ảo hóa đã được cung cấp tài nguyên tại các máy vật lý (tốt nhất), $gBest$.

```
1: int N ← numPopulation, M ← numLoop;
2: new gBest[Y]; /* new là toán tử khai báo mảng */
3: new pBest[N][Y];
4: new parLoca[N][Y];
5: boolean new parVelo[N][Y];
6: while nLoop ≤ M do
7:     for p := 1 to N do
8:         parLoca[p] ← initSolution(p,Y,S);
9:         for q := 1 to Y do
10:            if (parLoca[p][q].size == 0)
11:                parVelo[p][q] ← 0;
12:            else
13:                parVelo[p][q] ← 1;
14:            end for q := 1 to Y
15:        end for p := 1 to N
16:    for p:= 1 to N do
17:        for q := 1 to parLoca[p].length do
18:            pBest[p][q] ← parLoca[p][q];
19:        end for q := 1 to parLoca[p].length
20:    end for p := 1 to N
21:    gBest = globalBestParticle(parLoca,Y);
22:    for p := 1 to N do
23:        parVelo[p] = speedUpdate();
24:        parLoca[p] = positionUpdate();
25:        pBest[p] = PBestUpdate(pBest[p]);
26:        gBest = GBestUpdate(gBest);
27:    end for p := 1 to N
28:    nLoop++;
29: end while
30: return gBest;
```

Đầu tiên, thiết lập số lượng *particle* là N, số lần lặp M. Khai báo các mảng để lưu giá trị tốt nhất cục bộ và toàn cục; lưu thông tin vị trí và vận tốc các *particle* (từ lệnh 1 đến lệnh 5). Tiếp đến, thực hiện M lần lặp để tìm kiếm giải pháp cung cấp tài nguyên tối ưu (từ lệnh 6 đến lệnh 29). Trong đó, sử dụng *Thuật toán 2* để tạo ra N giải pháp ban đầu cho việc cung cấp tài nguyên bằng phương pháp ngẫu nhiên, kết quả là vị trí của các *particle* được lưu vào mảng $parLoca$ (lệnh 8). Dựa trên thông tin đó, thiết lập giá trị vận tốc các *particle* và lưu trong mảng $parVelo$ (từ lệnh 9 đến lệnh 15). Tiếp đến, tính độ thích nghi của tất cả các *particle* và có được vị trí tốt nhất cục bộ (từ lệnh 16 đến lệnh 20) và vị trí tốt nhất toàn cục (lệnh 21). Ước lượng vị trí tốt nhất toàn cục bằng *Thuật toán 3*. Tiếp đến, cập nhật vị trí, vận tốc của các *particle* theo công thức (12) và (13), tính vị trí tốt nhất cục bộ và toàn cục dựa trên

dân số mới được cập nhật và hàm thích nghi (từ lệnh 22 đến lệnh 27). Cuối cùng, tăng chỉ số vòng lặp $nLoop$ (lệnh 28). Thuật toán kết thúc sau khi thực hiện $nLoop$ lần lặp và trả về danh sách các dịch vụ ảo hóa đã được cung cấp tài nguyên từ các máy vật lý (tốt nhất) được dùng (lệnh 30).

Thuật toán 2: *InitSolution*

Đầu vào:

- Số dịch vụ ảo hóa *S*, cùng với nhu cầu tài nguyên giống như Thuật toán 1 và hệ số bổ sung Q_{ij}.
- Số máy vật lý *Y*, cùng với tài nguyên giống như *Thuật toán 1*.

Đầu ra: Danh sách các dịch vụ ảo hóa được cung cấp tài nguyên tại các máy vật lý, $Allocation$.

1: **new** $Allocation[Y]$ /* *Khai báo mảng Allocation* */
2: **for** $a := 1$ **to** $Allocation.length$ **do**
3: $Allocation[a] := new\ Array()$
4: **end for** $a := 1$ **to** $Allocation.length$
5: **for** $i := 1$ **to** S **do**
6: **int** $j \leftarrow$ **random**(Y);
7: **for** $k := 1$ **to** D **do**
8: $Sum^e_{ik} \leftarrow r^e_{ik} + Q_{ij} \times f^e_{ik}$;
9: $Sum^a_{ik} \leftarrow r^a_{ik} + Q_{ij} \times f^a_{ik}$;
10: **if**$(C^e_{jk} \geq Sum^e_{ik}$ **and** $C^a_{jk} \geq Sum^a_{ik})$
11: $C^a_{jk} \leftarrow C^a_{jk} - Sum^a_{ik}$;
12: **end if**
13: **end for** $k := 1$ **to** D
10: $Allocation[j] \leftarrow i$;
11: **end for** $i := 1$ **to** S
12: **return** $Allocation$;

Thuật toán 3: *globalBestParticle*

Đầu vào:

- Mảng hai chiều $parLoca$
- Số máy vật lý *Y*, tài nguyên thành phần C^e_{jk}, tài nguyên tổng hợp C^a_{jk}.

Đầu ra: Mảng $gBparticle$; mỗi phần tử của mảng là một mảng $parLoca$ có giá trị tốt nhất toàn cục.

1: **double** $minF \leftarrow ConstMax, temp \leftarrow 0$;
2: **new** $gBparticle[Y]$
3: **for** $g := 1$ **to** $gBparticle.length$ **do**
4: $gBparticle[g] = new\ Array()$;
5: **end for** $g := 1$ **to** $gBparticle.length$
6: **for** $p := 1$ **to** $parLoca.length$ **do**
7: $temp \leftarrow fSolution(parLoca[p], Y)$; /* *Tính giá trị của hàm thích nghi theo công thức* (11) */
8: **if** $(temp < minF)$
9: $minF \leftarrow$ temp;
10: **for** $g := 1$ **to** $gBparticle.length$ **do**
11: $gBparticle[p] \leftarrow parLoca[p][q]$;
12: **end for** $j := 1$ **to** $gBparticle.length$
13: **end if**
14: **end for** $i := 1$ **to** $parLoca.length$
15: **return** $gBparticle$;

Gọi S là số dịch vụ ảo hóa, Y là số máy vật lý, N là số lượng *paticle*. Cố định chiều tài nguyên *D*, độ phức tạp của thuật toán là $O(N^{SY})$.

IV. PHƯƠNG PHÁP THỰC NGHIỆM VÀ KẾT QUẢ ĐÁNH GIÁ

IV.1. Phương pháp thực nghiệm

Đánh giá thuật toán *RA-PSO với FFD* [11], chúng tôi sử dụng công cụ mô phỏng đám mây CloudSim [3]. Trong đó, kế thừa các lớp *Vm* và *Host* để mở rộng các đặt tính nhu cầu tài nguyên của máy ảo và tài nguyên của máy vật lý. Đồng thời, kế thừa lớp *VMAlloctonPolicy* để thực chính sách cung cấp tài nguyên cho máy ảo dựa trên thuật toán *RA-PSO*.

Bảng 1. Đặc tính tài nguyên và tiêu thụ công suất của các loại máy vật lý

Loại máy vật lý	CPU (MHz)	RAM (GB)	BW (GB/s)	Disk (GB)	P_{idle} (kW)	P_{max} (kW)
HP proliant G4	2core x 1860	4	1	20	86	117
HP proliant G5	2core x 2660	4	1	40	93.7	135
IBM Server x3250	4core x 2933	8	1	600	46.1	113
IBM Server x3550	6core x 3067	16	1	800	58.4	222

Các dữ liệu thực nghiệm được lấy từ các mẫu dữ liệu thực tế đã được trình bày trong [1, 2]. Trong đó, các máy vật lý có đặc tính tài nguyên và tiêu thụ công suất như trong Bảng 1, các máy ảo có đặc tính tài nguyên giống với các loại máy ảo của đám mây Amazon EC2 và được sửa đổi để phù hợp với bài toán, như trong Bảng 2.

Bảng 2. Đặc tính nhu cầu tài nguyên của các loại máy ảo

Loại máy ảo	CPU(MHz)				RAM(GB)				BW(GB/s)				DISK(GB)			
	f^e_{cpu}	r^e_{cpu}	Số phần tử	Tổng cộng	f^e_{ram}	r^e_{ram}	Số phần tử	Tổng cộng	f^e_{bw}	r^e_{bw}	Số phần tử	Tổng cộng	f^e_{disk}	r^e_{disk}	Số phần tử	Tổng cộng
VM1	1000	1500	1	2500	0.4	0.45	1	0.85	0.2	0.25	1	0.45	0.5	2.0	2	5
VM2	1000	1000	1	2000	1.0	2.75	1	3.75	0.1	0.25	1	0.35	2.0	3.0	2	10
VM3	500	500	1	1000	0.7	1.0	1	1.7	0.1	0.15	1	0.25	2.5	5.0	2	15
VM4	250	250	1	500	0.113	0.5	1	0.613	0.05	0.1	1	0.15	5.0	5.0	2	20

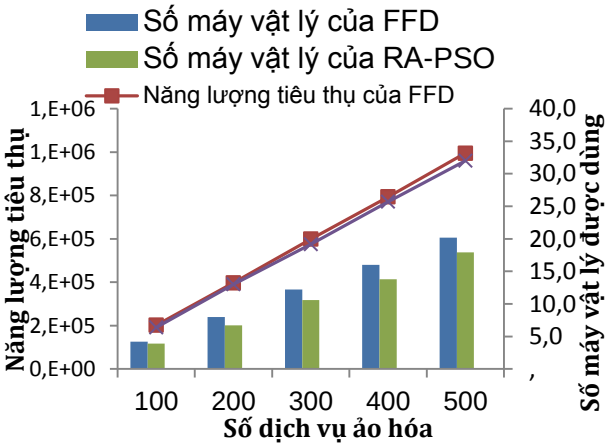
Số lượng *particle* và giá trị *numLoop* là các tham số thực nghiệm, việc lựa chọn giá trị của nó phụ thuộc vào kết quả hàm mục tiêu. Thực nghiệm thấy rằng: số lượng *particle* là 20, số lần lặp *numLoop* là 10 cho kết quả hàm mục tiêu tốt nhất. Với mỗi thuật toán sử dụng ba thước đo: số máy vật lý được dùng, năng lượng tiêu thụ trong khoảng thời gian t = 24 giờ và thời gian thực thi. Số lượng máy ảo trong các kịch bản thực nghiệm lần lượt là 100; 200; 300; 400; 500. Đơn vị tính năng lượng thiêu thụ là *kWh* và thời gian thực thi là giây (s). Thời gian thực hiện được đo trên máy tính đơn có bộ vi xử lý Intel(R) Core(TM) i5-3235M 2.60 GHz, RAM 4Gb.

IV.2. Kết quả nhận xét

Bảng 3. Kết quả thực nghiệm

Số máy ảo	Tên Thuật toán	Số máy vật lý được dùng	Thời gian thực hiên (s)	Năng lượng tiêu thụ (kWh)	Lợi ích năng lượng (%)
100	*FFD*	42	0,031	201.284	
	RA-PSO	39	0,037	190.726	5,536
200	*FFD*	80	0,078	396.706	
	RA-PSO	67	0,084	390.292	1,643
300	*FFD*	122	0,116	597.989	
	RA-PSO	106	0,121	574.440	4,099
400	*FFD*	160	0,144	793.411	
	RA-PSO	138	0,150	770.908	2,919
500	*FFD*	202	0,200	994.694	
	RA-PSO	179	0,216	960.156	3,597

Kết quả thực nghiệm được trình bày như trong Bảng 3 và Hình 3. Nhận thấy rằng: năng lượng tiêu thụ tỷ lệ thuận với số lượng máy vật lý; số lượng máy vật lý được dùng và năng lượng tiêu thụ của thuật toán *RA-PSO* tốt hơn thuật toán *FFD*. Khi bài toán lớn (số lượng máy ảo lớn) thì chênh lệch số máy vật lý được dùng giữa thuật toán *RA-PSO* và *FFD* rõ nét hơn. Thời gian thực thi của *RA-PSO* kém hơn *FFD*, nguyên nhân do *RA-PSO* có sử dụng tham số vòng lặp để ước lượng giải pháp tốt nhất cục bộ và tốt nhất toàn cục. Hơn nữa, thuật toán *RA-PSO* thực hiện tìm kiếm giải pháp tối ưu dựa trên kinh nghiệm bầy đàn, nên số lượng các *particle* cũng góp phần làm tăng thời gian tính toán.



Hình 3. Số máy vật lý được dùng và năng lượng tiêu thụ.

V. KẾT LUẬN

Nội dung bài báo trình bày vấn đề cung cấp tài nguyên vật lý đa chiều cho dịch vụ ảo hóa với ràng buộc tối ưu; mỗi dịch vụ là một máy ảo đơn lẻ; nhu cầu tài nguyên không đổi trong quá trình hệ thống thực thi. Dựa trên phương pháp tối ưu hóa bầy đàn PSO, chúng tôi đưa ra các thuật toán *RA-PSO* và đã cài đặt, đánh giá và so sánh với thuật toán *FFD* thông qua các thước đo: số máy vật lý được dùng, năng lượng tiêu thụ và thời gian thực hiện thuật toán. Các thuật toán được thực thi bằng công cụ CloudSim với các dữ liệu thực tế. Qua kết quả thực nghiệm, chúng ta thấy rằng: các thước đo giá trị hàm mục tiêu và năng lượng tiêu thụ trên thuật toán *RA-PSO* tốt hơn thuật toán *FFD*. Hướng nghiên cứu mở rộng là bài toán cung cấp tài nguyên động cho dịch vụ ảo hóa.

TÀI LIỆU THAM KHẢO

[1] ARIANYAN, E., H. TAHERI, and S. SHARIFIAN, *"Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers",*Computers & Electrical Engineering, vol. 47, 2015, 222-240.

[2] BELOGLAZOV, A. and R. BUYYA, "*Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers",* Concurr. Comput. Pract. Exper., vol.24, No.13, 2012,1397-1420.

[3] CALHEIROS, R.N., et al., *"CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms",* Softw. Pract. Exper., vol. 41, No.1, 2011, 23-50.

[4] CAO, Z. and S. DONG, "*Dynamic VM Consolidation for Energy-Aware and SLA Violation Reduction in Cloud Computing*" in Proceedings of Parallel and Distributed Computing, Applications and Technologies (PDCAT), Beijing, 2012, 363-369.

[5] FARAHNAKIAN, F., et al. "*Energy-Aware Dynamic VM Consolidation in Cloud Data Centers Using Ant Colony System",* in Proceedings of the 7th International Conference on Cloud Computing, Anchorage, 2014, 104-111.

[6] FELLER, E., L. RILLING, and C. MORIN, "*Energy-Aware Ant Colony Based Workload Placement in Clouds",* in Proceedings of the 12th International Conference on Grid Computing, Lyon, 2011, 26-33.

[7] JANSEN, R. and P.R. BRENNER. "*Energy efficient virtual machine allocation in the cloud",* in Green Computing Conference and Workshops, Orlando, 2011,1-8.

[8] LIANG, L., et al. "*A resource scheduling algorithm of cloud computing based on energy efficient optimization method*",Green Computing Conference, San Jose, 2012,1-6.

[9] QUAN, D.M., et al., "*Energy Efficient Resource Allocation Strategy for Cloud Data Centres*", in Proceedings of 26th International Symposium on Computer and Information Sciences, London, 2012, 133-141.

[10] VIGLIOTTI, A. and BATISTA, D.M., "*Energy-Efficient Virtual Machines Placement",* in Proceedings of Computer Networks and Distributed Systems, Florianopolis, 2014, 1-8.

[11] STILLWELL, M., VIVIEN, F., CASANOVA, H. "*Virtual Machine Resource Allocation for Service Hosting on Heterogeneous Distributed Platforms",* in Proceedings of 26th International, 2012, 86 - 797.

[12] THOMAS S. and ALEXANDER S., *"Decision support for virtual machine reassignments in enterprise data centers",* in Proceedings of Network Operations and Management Symposium Workshops, Osaka, 2010, 88-94.

[13] KENNEDY J. and EBERHART R., *"Particle swarm optimization",* in Proceedings of Neural Networks, vol. 4, 1995,1942 – 1948.

Nhận bài ngày: 15/04/2016

SƠ LƯỢC VỀ TÁC GIẢ

PHẠM NGUYỄN MINH NHỰT

Sinh năm 1972 tại Đà Nẵng.

Tốt nghiệp ĐH Huế năm 1994 chuyên ngành Vật lý; nhận bằng thạc sỹ Khoa học Máy tính năm 2005 tại ĐH Huế; đang là nghiên cứu sinh tại ĐH Đà Nẵng.

Hiện công tác tại trường Cao đẳng CNTT Việt-Hàn.

Lĩnh vực nghiên cứu: Xử lý ảnh, nhận dạng, hệ phân tán, điện toán đám mây.

Điện thoại : 090 3 501 421,

Email : minhnhutvh@gmail.com

LÊ VĂN SƠN

Sinh năm 1948 tại Quảng Nam.

Tốt nghiệp ĐH năm 1978, bảo vệ TS năm 1997 tại trường ĐH Tổng hợp Donesk, Ucraina, được công nhận PGS năm 2004. Hiện công tác tại Khoa Tin học, trường ĐH Sư phạm, ĐH Đà Nẵng.

Lĩnh vực nghiên cứu: Hệ điều hành, mạng máy tính, hệ phân tán, điện toán đám mây.

Điện thoại: 090 5 827 499

E-mail : levansupham2004@yahoo.com

HOÀNG BẢO HÙNG

Sinh năm 1971 tại Huế.

Tốt nghiệp ĐH năm 1993, bảo vệ Tiến sĩ tại Viện CNTT, Viện Hàn lâm KHCN Việt nam năm 2007. Hiện công tác tại Trường Cao đẳng CNTT Hữu nghị Việt-Hàn. Lĩnh vực nghiên cứu: Hệ CSDL hướng đối tượng, hệ phân tán, điện toán đám mây, GIS.

Điện thoại: 0914019123, E-mail : hbhung@gmail.com