LẬP LỊCH TRONG MÔI TRƯỜNG TÍNH TOÁN ĐÁM MÂY DỰA TRÊN RÀNG BUỘC THỜI HẠN

SCHEDULING TASKS IN CLOUD COMPUTING ENVIRONMENT BASED ON DEADLINE

Nguyễn Anh Tuấn[1], Lê Văn Sơn[2], Nguyễn Hoàng Hà[3], Lê Thành Công[2]

[1]Trường Cao đẳng Công nghệ Thông tin, Đại học Đà Nẵng; Email: tuanna@ud.edu.vn
[2]Trường Đại học Sư phạm, Đại học Đà Nẵng; Email: levansupham2004@yahoo.com
[3]Trường Đại học Khoa học, Đại học Huế; Email: nhha76@gmail.com

Tóm tắt - Lập lịch tác vụ trong môi trường tính toán đám mây có hai hướng chính: hướng đến hiệu năng về hệ thống và hướng đến hiệu năng về kinh tế. Bài báo này chỉ tập trung lập lịch hướng đến hiệu năng về hệ thống. Bởi vì bài toán lập lịch tác vụ việc trên tính toán đám mây là một bài toán NP - đầy đủ [2], do đó cần thiết phải xây dựng các thuật toán heuristic để giải quyết vấn đề này. Bài báo này sử dụng thuật toán ACO [1] để đưa ra một thuật toán heuristic mới về lập lịch trên các tác vụ trong môi trường tính toán đám mây. Mục đích của thuật toán là làm cho tổng thời gian hoàn thành công việc của hệ thống là nhỏ nhất những vẫn thỏa mãn thời hạn cho các tác vụ. Thuật toán này được cài đặt hoàn chỉnh trên CloudSim [9] và kết quả có sự cải tiến đáng kể so với thuật toán MinMin [10] hiện có.

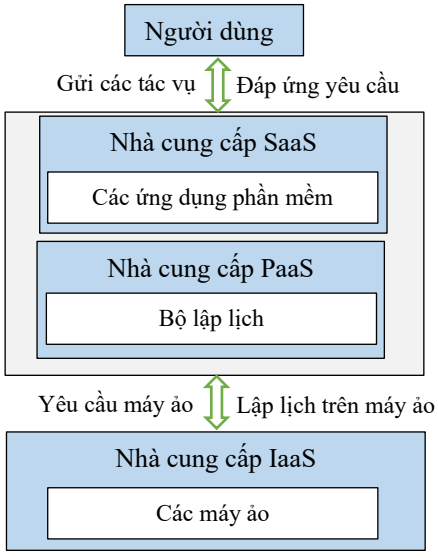
Từ khóa - tính toán đám mây; quản lý tài nguyên; lập lịch tác vụ; DSACO; MinMin.

Abstract - Scheduling taskss on the cloud computing has two main problems to solve: the performance of the system and the economic performance. This paper focuses on scheduling on system performance. The task scheduling problem on the cloud computing is a complete NP [2]. Therefore, it is necessary to develop heuristics to solve this problem. This paper uses Ant Colony Optimization – ACO [1] algorithm to provide a new heuristic to schedule tasks on the cloud computing. The purpose of the algorithm is to minimize the total time to complete the task of the system but it still satisfies the minimum duration for the task. This algorithm is officially installed on CloudSim [9] and it has brought about significantly improved results compared with the current MinMin algorithm [10].

Key words - cloud computing; resource management, scheduling; DSACO; MinMin.

1. Đặt vấn đề

Tính toán đám mây đã mở ra một thời đại mới để chia sẻ cơ sở hạ tầng công nghệ thông tin, các hệ thống được liên kết với nhau thông qua internet để chia sẻ dịch vụ. Tính toán đám mây bao gồm các thành phần người dùng, nhà cung cấp dịch vụ SaaS, PaaS và IaaS như thể hiện ở Hình 1. Người dùng gởi tập các tác vụ yêu cầu tài nguyên đến nhà cung cấp SaaS, nhà cung cấp PaaS sử dụng bộ lập lịch để tìm kiếm tài nguyên thích hợp trên nhà cung cấp IaaS, nếu tìm thấy tài nguyên thỏa mãn yêu cầu sẽ ánh xạ tài nguyên vào tác vụ tương ứng.



Hình 1. *Mô hình tổng quát để lập lịch theo hướng hiệu năng về hệ thống*

Với sự hỗ trợ công nghệ ảo hóa, nền tảng của tính toán đám mây cho phép các doanh nghiệp cho thuê năng lực tính toán của mình dưới hình thức các máy ảo đến người dùng. Những người dùng có thể sử dụng hàng trăm, hàng ngàn máy ảo nên rất khó để gán tác vụ đến các tài nguyên trong môi trường đám mây. Vì vậy, vấn đề cần thiết đặt ra là cần một số thuật toán có khả năng mang lại hiệu quả để lập lịch tối ưu hóa các tác vụ trong môi trường tính toán đám mây. Hiện nay các nhà khoa học đã nghiên cứu nhiều thuật toán để đạt được hiệu quả cho việc lập lịch tác vụ trong môi trường đám mây. Nhiều thuật toán đã được phát triển để có được giải pháp gần tối ưu phù hợp với mục tiêu đặt ra của người dùng được đề xuất gần đây tại vGrADS [2] như MinMin [10], MaxMin [8]. Các thuật toán này cơ bản tính toán thời gian hoàn thành sớm nhất của từng tác vụ ứng với tài nguyên có sẵn, sau đó có được thời gian dự kiến hoàn thành cho mỗi công việc và giá trị tối thiểu cho tất cả các tác vụ tương ứng.

Thuật toán ACO là thuật toán tìm kiếm ngẫu nhiên, giống như các thuật toán bầy đàn (PSO) và thuật toán di truyền (GA) [5]. Nó bắt chước hành vi của đàn kiến trong tự nhiên để tìm kiếm thức ăn và kết nối với nhau bằng mùi thức ăn đặt lại trên đường đi. Nhiều nhà nghiên cứu đã sử dụng ACO để giải các bài toán khó về tổ hợp như bài toán người du lịch, tô màu đồ thị.

Trong bài báo này, chúng tôi đề xuất một thuật toán lập lịch tác vụ trên môi trường đám mây dựa trên thuật toán tối ưu hóa đàn kiến (DSACO) để tìm và ánh xạ các tác vụ vào tài nguyên một cách tối ưu nhằm giảm thiểu tổng thời gian thực hiện của hệ thống. Sau đó, thuật toán lập lịch này được thực nghiệm mô phỏng bằng công cụ CloudSim để so sánh về thời gian tối ưu hệ thống với thuật toán MinMin vừa nêu trên.

2. Mô hình hệ thống

Mô hình tổng quát để lập lịch cho các tác vụ bao gồm các thành phần như Hình 1. Trong mô hình này bao gồm

các mô hình con: mô hình người dùng SaaS, mô hình nhà cung cấp SaaS, mô hình nhà cung cấp PaaS. Chúng tôi trọng tâm trên mô hình của nhà cung cấp dịch vụ PaaS để tìm kiếm tài nguyên hợp lý trên IaaS.

2.1. Mô hình người dùng

Người dùng gửi n tác vụ ($T_1, T_2,..,T_n$) lên hệ thống, mỗi tác vụ T_i là một bộ 5 thành phần<a_i,w_i,d_i,in_i,out_i> trong đó:

- a_i: thời gian đến của tác vụ.

- w_i: khối lượng tính bằng MI của tác vụ.

- d_i: thời hạn lớn nhất để hoàn thành công việc.

- in_i, out_i: kích cỡ file đầu vào.

Mỗi tác vụ có thể ánh xạ bất kỳ nguồn tài nguyên tính toán có trên đám mây và mỗi tài nguyên tính toán có thể kết nối với một hoặc nhiều tài nguyên lưu trữ để tải về các file đầu ra và lưu lại các file đầu ra khi tác vụ tính toán xong.

2.2. Mô hình nhà cung cấp IaaS

Tài nguyên hệ thống bao gồm m các tài nguyên tính toán ($PC_1, PC_2, …, PC_m$), chúng liên kết với nhau với các băng thông khác nhau và tài nguyên tính toán kết nối với một hoặc nhiều tài nguyên lưu trữ ($S_1, S_2,.., S_k$). Các tài nguyên tính toán này có thể thực hiện được bất kỳ công việc nào, nhưng thời gian thực hiện khác nhau.

Mỗi tài nguyên tính toán PC_j là một bộ 3 thành phần <r_j,s_j,sr_j>, trong đó:

r_j: thời gian sẵn sàng của tài nguyên.

s_j: tốc độ tính toán của tài nguyên.

sr_j: là tập các tài nguyên lưu trữ liên kết với tài nguyên tính toán PC_j .

2.3. Mô hình nhà cung cấp PaaS

Gọi thời gian hoàn thành của tác vụ i trên tài nguyên tính toán thứ j là CT_{ij}, CT_{ij} được xác định như sau:

$$CT_{ij} = TP_{ij} + R_j + TF_{ij} \quad (1)$$

Trong đó:

- TP_{ij} là thời gian thực hiện của tác vụ i trên tài nguyên j, phụ thuộc vào tốc độ tính toán s_{ij} của tác vụ i trên tài nguyên j và khối lượng w_i của tác vụ i.

$$TP_{ij} = \frac{w_i}{s_{ij}} \quad (2)$$

- R_j là thời gian sớm nhất mà tài nguyên tính toán j có thể thực thi tác vụ (thời gian sẵn sàng):

$$R_j = R_j + 1 \quad (3)$$

- TF_{ij} là thời gian nhỏ nhất để chuyển file đầu vào in_i và đầu ra out_i từ sr_j đến tài nguyên tính toán tài nguyên i.

$$TF_{ij} = \min\left(\frac{(in_i+out_i)}{bw_{jk}}\right),\ k = 1..m' \quad (4)$$

Trong đó: bw_{jk} là băng thông kết nối từ sr_j đến tài nguyên tính toán j (k=1..m', m' là số tài nguyên lưu trữ kết nối với tài nguyên tính toán j).

Mục tiêu của bài báo là tìm kiếm các tài nguyên tính toán trên IaaS để cho makespan của hệ thống là nhỏ nhất, tức là phải đi tìm:

$$\min\left(\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{p}\left(\begin{array}{c}\frac{w_i}{s_{ij}} + (R_j + 1) \\ + \min\left(\frac{(in_i + out_i)}{bw_{jk}}\right)\end{array}\right)\right),$$

$$i = 1..n, j = 1..m, k = 1..p \quad (5)$$

Để đạt được mục tiêu trên thì phải thỏa màn ràng buộc:

$$CT_i \le d_i \ i = 1..n' \quad (6)$$

3. Xây dựng bài toán

Thuật toán đàn kiến là một trong những giải pháp giải quyết các bài toán tối ưu hóa tổ hợp. Nó mô phỏng hành vi của đàn kiến trong tự nhiên nhằm tìm kiếm đường đi ngắn nhất giữa nguồn thức ăn và tổ kiến dựa trên mật độ mùi mà các con kiến để lại trên đường đi. Khi một con kiến tìm kiếm thức ăn nó sẽ để lại một số lượng mùi trên đường đi. Nếu một con kiến cố gắng để di chuyển từ nơi này sang nơi khác, nó sẽ gặp một dấu vết trước đó từ những con kiến đã đi trước đó, kiến có thể phát hiện các dấu vết mùi và dựa vào xác suất mùi cao nó sẽ quyết định chọn đường đi. Sau khi đi qua một chu kỳ, con kiến này để lại mùi trên đường và mùi này sẽ mất dần theo thời gian, do đó khi nhiều con kiến tìm đường đi, mùi trên con đường ngắn hơn sẽ được tăng lên một cách nhanh chóng. Số lượng mùi trên mỗi con đường sẽ ảnh hưởng đến khả năng con kiến khác để lựa chọn đường đi. Cuối cùng dựa vào xác suất mùi cao, những con kiến sẽ chọn con đường ngắn nhất. Thuật toán ACO đươc mô tả như sau:

Bước 1. Khởi tạo các điều kiện ban đầu.

Bước 2. Các con kiến xây dựng phương án để tìm kiếm nguồn thức ăn.

Bước 3. Các con kiến cập nhật lại mùi.

Bước 4. Nếu điều kiện kết thúc chưa thỏa mãn thì sang bước 2, ngược lại dựa trên các con kiến để tìm ra phương án tốt nhất. Để áp dụng thuật toán đàn kiến người ta phải xác định được các thông tin hàm cực tiểu F, thông tin heuristic η_i, cập nhật lại mùi và xác xuất P.

3.1. Hàm cực tiểu F và thông tin heuristic η_i

Hàm cực tiểu F và thông tin heuristic η_i được sử dụng để tìm ra nhà cung cấp IaaS tốt nhất, phụ thuộc vào thời gian thực hiện (CTi_{ij}) của tác vụ i trên tài nguyên tính toán j:

$$F = Max(CT_{ij}), i = 1..n, j = 1..m \quad (7)$$

$$\eta_{\ i} = \frac{1}{CT_{ij}}, i = 1..n, j = 1..m \quad (8)$$

Sử dụng η_i để tìm ra tài nguyên có độ ưu tiên cao nhất vì thời gian thực hiện CT_{ij} càng nhỏ thì thông tin η_i của tác vụ i càng cao. Hàm cực tiểu F dùng để tính xác xuất cho tác vụ i chọn tài nguyên j.

3.2. Cập nhật lại mùi

Mỗi con kiến bắt đầu từ nguồn tài nguyên và tác vụ một cách ngẫu nhiên. Tất cả các con kiến được duy trì trong một danh sách, bất kỳ khi nào chúng chọn tác vụ vào tài nguyên kết tiếp chúng được lưu vào danh sách. Tại mỗi bước lặp các con kiến tính hàm cực tiểu và cập nhật lại mùi như sau:

$\tau_{ij} = \rho\tau_{ij} + \Delta\tau_{ij}$

Trong đó:

- τ_{ij} là mật độ mùi của tác vụ i trên tài nguyên j.

- ρlà giá trị bay hơi được xác định trong khoảng (0,1).

- $\Delta\tau_{ij}=\frac{1-\rho}{F_k}$ với F_k là hàm cực tiểu của con kiến k.

- Δ_{ij}được thêm vào mật độ mùi.

3.3. Tính xác suất

Xác xuất để ánh xạ tác vụ i vào tài nguyên j được xác định như sau [7]:

$$\rho_{ij} = \frac{\tau_{ij}.\eta_{ij}}{\sum_{j=1}^{m}\tau_{ij}.\eta_{ij}} \quad (9)$$

Sau khi áp dụng các công thức (1),(2),(3),(4),(7), (8) và (9) chúng tôi xác định công thức để ánh xạ tác vụ i vào tài nguyên tính toán j như sau:

$$p_{ij} = \frac{\left(\rho\tau_{ij} + \frac{1-\rho}{F_k}\right).\frac{1}{TP_{ij}+R_j+TF_{ij}}}{\sum_{j=1}^{m}(\rho\tau_{ij} + \frac{1-\rho}{F_k}).\frac{1}{TP_{ij}+R_j+TF_{ij}}} \quad (10)$$

Trong đó: τ_{ij} là mật độ mùi để lại khi con kiến di chuyển. p_{ij} là xác suất để tác vụ i ánh xạ vào tài nguyên tính toán j.

4. Thuật toán DSACO

Ý tưởng của thuật toán này được lấy từ ý tưởng của thuật toán SACO [7], chúng tôi thêm vào các ràng buộc như thời gian đến, thời hạn cho các tác vụ, sau đó xây dựng thuật toán như mô hình ở mục 2 và mục 3.

4.1. Các ký hiệu và các giá trị khởi tạo của thuật toán

- Dựa vào các tài nguyên có sẵn trên đám mây, ước lượng thời gian thực hiện lưu vào TP_{ij}, dựa vào băng thông kết nối giữa các tài nguyên tính toán và tài nguyên lưu trữ ta ước lượng thời gian chuyển tải các file đầu vào từ tài nguyên lưu trữ đến tài nguyên tính toán lưu ở PS_{ij}.

- T, CR, SR: tập các tác vụ, tập các tài nguyên tính toán và tập các tài nguyên lưu trữ.

- CL: Tập các tác vụ chưa lập lịch.

- S: tập các tác vụ đã lập lịch.

- T_i, CR_j: tác vụ thứ i, tài nguyên thứ j.

- R_j: Thời gian sớm nhất mà tài nguyên CR_j có thể cung cấp cho tác vụ.

- Giá trị bốc hơi của mùi ban đầu của ρ là 0,05.

- Giá trị mùi ban đầu là 0,01.

4.2. Đầu vào của thuật toán

- Số tác vụ cần lập lịch: N, số tài nguyên tính toán: M.

- Ma trận TP_{nxm}, PS_{mxn}.

- ρ =0,05, τ_{ij} = 0,01.

4.3. Đầu ra của thuật toán

Bảng ánh xạ của tác vụ vào các nguồn tài nguyên.

4.4. Giải thuật xử lý bài toán

B1.THỰC HIỆN LẶP VỚI MỖI CON KIẾN

B2.S = {}; CL = T; R_j= 0;

B3.Gán ngẫu nhiên tác vụ $T_i$vào tài nguyên CR_j.

B4.Ánh xạ tác vụ T_i vào nguồn tài nguyên CR_j và cập nhật lại R_j.

B5.S = S + {T_i}; CL = CL – {T_i}; $R_j = TP_{ij} + R_j + TF_{ij}$

B6.THỰC HIỆN LẶP MỖI TÁC VỤ T_j TRONG TẬP TÁC VỤ CHƯA LẬP LỊCH

B7.Tính thông tin heuristic cho tác vụ T_i trên CR_j:

$$\eta_{ij} = \frac{1}{TP_{ij} + R_j + TF_{ij}}$$

B8.Tìm ra giá trị của mùi hiện tại: τ_{ij}

B9.Cập nhật lại mùi: $\tau_{ij} = \rho\tau_{ij} + \frac{1-\rho}{F_k}$

B10. Tính xác suất:

$$p_{ij} = \frac{\left(\rho\tau_{ij} + \frac{1-\rho}{F_k}\right).\frac{1}{TP_{ij}+R_j+TF_{ij}}}{\sum_{j=1}^{m}(\rho\tau_{ij} + \frac{1-\rho}{F_k}).\frac{1}{TP_{ij}+R_j+TF_{ij}}}$$

B11. Tìm ra P_{ij} cao nhất và có $CT_i<d_i$, sau đó, ánh xạ tác vụ T_i vào nguồn tài nguyên.

B12. S=S+{T_i}; CL = CL – {T_i}; $R_j = TP_{ij} + R_j + TF_{ij}$

B13. QUAY TRỞ LẠI BƯỚC LẶP (6).

B14. Phân tích các con kiến trong để tìm ra giải pháp có thể tối ưu nhất.

B15. QUAY TRỞ LẠI BƯỚC LẶP (1).

Mỗi con kiến ta ánh xạ ngẫu nhiên một tác vụ vào tài nguyên tính toán j nào đó và cập nhật lại các trạng thái (bước 3 đến bước 5). Sau đó duyệt qua các tác vụ chưa lập lịch (CL), cứ mỗi tác vụ chưa lập lịch ta dựa vào các thông tin η_{ij} và τ_{ij} để tính xác xuất p_{ij} (bước 6 đến bước 10) và tìm ra xác xuất cao nhất và có thời gian thực hiện nhỏ hơn thời hạn của nó (bước 11). Sau đó tiến hành ánh xạ tác vụ i vào nguồn tài nguyên tính toán j và cập nhật lại các trạng thái lập lịch và chưa lập lịch. Sau khi kết thúc vòng lặp (bước 14) ta tiến hành phân tích các giải pháp có được của các con kiến để tìm ra một bảng ánh xạ tối ưu.

5. Kết quả thực nghiệm

5.1. Các thông số trong cài đặt thực nghiệm

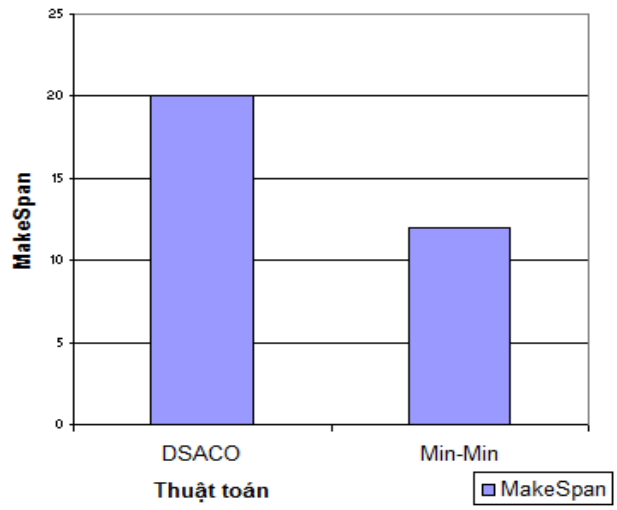
Các thuật toán được cài đặt mô phỏng bằng ngôn ngữ Java (NetBean 7.1.1, JDK 6), gói công cụ CloudSim 2.0 [9] với các thông số sau: Sử dụng 1 Datacenter, số máy ảo thay đổi từ 50 đến 100, 4 PE và 512 RAM trên một máy ảo và số yêu cầu thay đổi từ 100 đến 500. Các thông số khác được lấy ngầm định như trong CloudSim. Thời gian thực hiện của mỗi tác vụ trên mỗi máy ảo (ma trận TP) và thời gian chuyển tải file đầu vào của mỗi tác vụ từ tài nguyên lưu trữ đến tài nguyên tính toán (ma trận PS) được lấy ngẫu nhiên từ 1 đến 10.

5.2. Phân tích makespan và số tác vụ vi phạm thời hạn khi số tác vụ nhỏ hơn số máy ảo

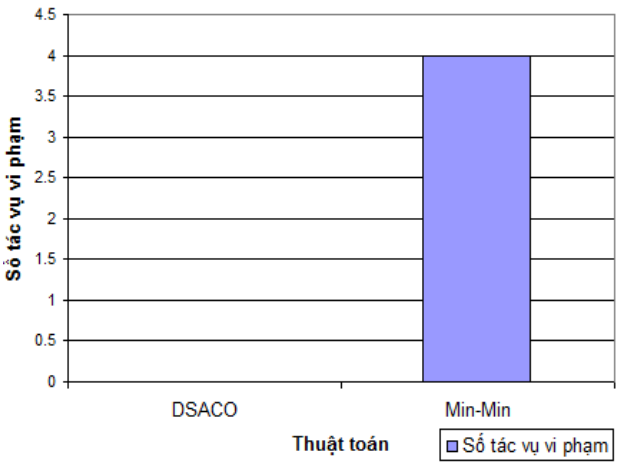
Đối với thuật toán MinMin [10], ý tưởng chính của thuật toán cũng giống như thuật toán MaxMin [8]. Cả hai thuật toán này dựa trên thời gian hoàn thành nhỏ nhất (MCT) của tác vụ trên các tài nguyên. Thuật toán sẽ tính toán thời gian hoàn thành của tất cả các tác vụ trên tất cả các tài nguyên hiện có. Tác vụ nào có giá trị MCT nhỏ nhất sẽ được phép chọn tài nguyên trước. Sau đó cập nhật thời

gian sẵn sàng cho tài nguyên được chọn, tính toán lại thông số với tất cả các tác vụ chưa được điều phối. Quá trình lặp lại cho đến khi tất cả các tác vụ đều đã chọn được tài nguyên thực thi.

Thuật toán MinMin không xét đến tác vụ có vi phạm thời hạn hay không. Sau khi thuật toán này thực hiện xong sẽ có nhiều tác vụ không thỏa mãn ràng buộc thời hạn như thể hiện ở Hình 3. Trong khi đó, thuật toán DSACO kiểm tra trước máy ảo có thỏa mãn ràng buộc của tác vụ không, nếu thỏa mãn mới tiến hành ánh xạ tác vụ vào máy ảo. Do đó, khi số máy ảo nhiều hơn số tác vụ thì hầu như thuật toán DSACO không có tác vụ vi phạm vi phạm thời hạn. Như thể hiện ở Hình 2, nếu số tác vụ nhỏ và số tài nguyên lớn thì thuật toán MinMin sẽ tối ưu hơn thuật toán DSACO vì MinMin sẽ duyệt qua tất cả các tài nguyên để tìm tài nguyên tốt nhất, trong khi đó thuật toán DSACO chỉ xét đến xác xuất để tìm ra phương án chấp nhận được.



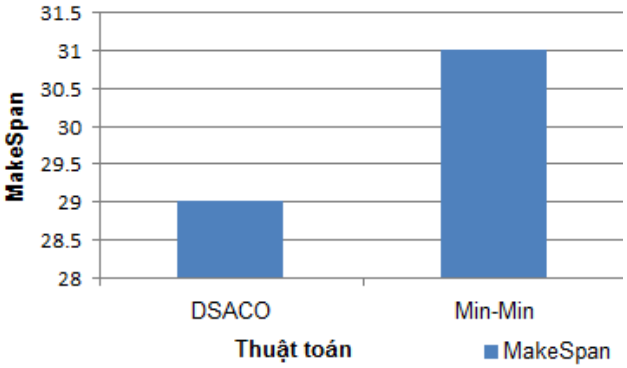
Hình 2. *So sánh MakeSpan của hai thuật toán khi số tác vụ là 15 và số máy ảo 50*



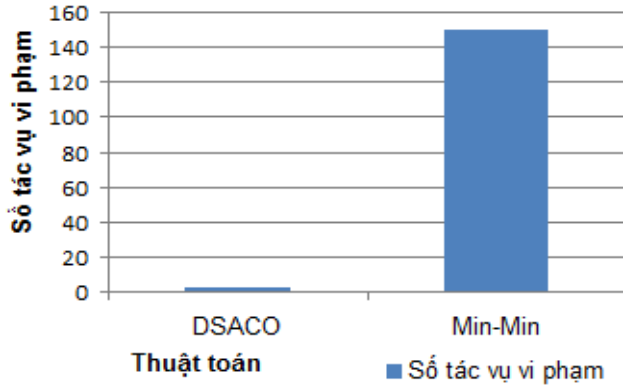
Hình 3. *So sánh số tác vụ vi phạm của hai thuật toán khi số tác vụ là 15 và số máy ảo 50*

5.3. Phân tích makespan và số tác vụ vi phạm thời hạn khi số tác vụ lớn hơn số máy ảo

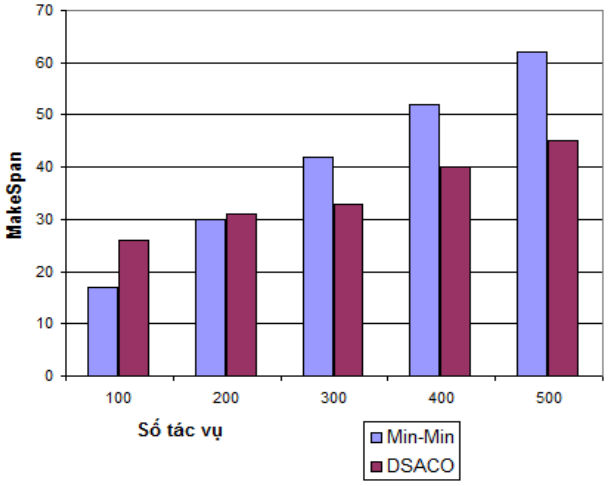
Nếu số tác vụ lớn hơn số tài nguyên thì thuật toán DSACO sẽ tối ưu hơn vì mỗi con kiến sẽ tìm ra một phương án tối ưu cục bộ, sau khi duyệt qua các con kiến đã lập lịch ta chọn ra phương án tối ưu nhất, trong khi đó thuật toán MinMin chỉ tìm ra phương án tối ưu cục bộ. Do đó, khi số tác vụ lớn hơn nhiều so với số máy ảo thì thuật toán DSACO sẽ tối ưu hơn và cân bằng tải cho toàn bộ hệ thống, đảm bảo không có vị trí nào xử lý quá nhiều dữ liệu và các tác vụ có quyền bình đẳng chia sẻ tài nguyên trên hệ thống làm cho makespane của hệ thống giảm xuống như thể hiện ở Hình 4 và Hình 6. Khi số máy ảo ít hơn nhiều so với tác vụ thì thuật toán MinMin sẽ có nhiều tác vụ không thỏa mãn ràng buộc thời hạn, trong khi đó có rất ít số tác vụ vi phạm thời hạn của thuật toán DSACO có như thể hiện ở Hình 5 và Hình 7.



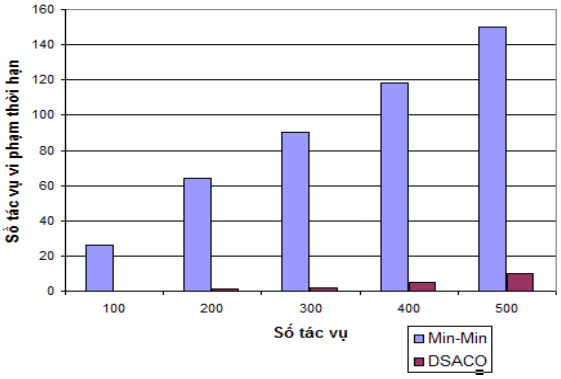
Hình 4. *So sánh MakeSpan của hai thuật toán khi số tác vụ là 200 và số máy ảo 50*



Hình 5. *So sánh số tác vụ vi phạm thời hạn của hai thuật toán khi số tác vụ là 200 và số máy ảo 50*



Hình 6. *So sánh MakeSpan của hai thuật toán khi thay đổi số tác vụ*



Hình 7. *So sánh số tác vụ vi phạm thời hạn của hai thuật toán khi thay đổi số tác vụ*

6. Kết luận

Bài báo tập trung nghiên cứu bài toán lập lịch cho các tác vụ theo hướng hiệu năng về hệ thống và dựa vào thuật toán ACO để thiết lập một mô hình hệ thống và đã đề xuất một thuật toán lập lịch mới (DSACO) cho các tác vụ trên môi trường tính toán đám mây dựa vào rang buộc thời hạn. Thông qua việc phân tích và kết quả thực nghiệm đối sánh trên cùng một bộ mẫu và sử dụng cùng một công cụ mô phỏng CloudSim đã cho thấy kết quả có sự cải tiến đáng kể về tổng thời gian thực hiện (makespan) của hệ thống so với thuật toán MinMin hiện đang sử dụng. Hướng nghiên cứu tiếp theocủa bài báo là tiếp tục so sánh đánh giá thực nghiệm với các giải thuật heuristic khác như MET (Min Execute Time) và MaxMin trên cùng các mẫu thử, môi trường giả lập đám mây. Qua đó tiếp tục cải tiến thuật toán DSACO để rút ngắn tổng thời gian thực hiện của hệ thống.

TÀI LIỆU THAM KHẢO

[1] A. Colorni, M. Dorigo and V. Maniezzo (1991), "Distributed optimization by ant colonies", *Proceedings of ECAL'91, European Conference on Artificial Life*, Elsevier Publishing, Amsterdam.

[2] Agarwal, A., Kumar (2011), "Economical Duplication Based Task Scheduling for Heterogeneous and Homogeneous Computing System", Mousumi Paul et al, *Int. J. Comp. Tech. Appl.*, Vol 2 (5), pp. 1552-1556

[3] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman (2000), "Heuristics for Scheduling Parameter Sweep Applications in Grid Environments", *Ninth Heterogeneous Computing Workshop*, IEEE Computer Society Press, pp. 349-363.

[4] JiayinLi, MeikangQiu, ZhongMing, GangQuan, XiaoQin, ZonghuaGu (2012), "Online optimization for scheduling preemptable tasks on IaaS cloud systems", *J. ParallelDistrib Comput.*

[5] Jing Liu, Xing-Guo Luo, Xing-Ming Zhang and Bai-Nan Li (2013), "Job Scheduling Model for Cloud Computing Based on Multi-Objective Genetic Algorithm", *IJCSI International Journal of Computer Science Issues*, Vol 10 (3).

[6] Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong and Dan Wang (2011), "*Cloud Task scheduling based on Load Balancing Ant Colony Optimization*", Sixth Annual ChinaGrid Conference.

[7] Nguyễn Hoàng Hà, Nguyễn Mậu Hân, Lê Văn Sơn (2013), "Một thuật toán lập lịch trong môi trường tính toán đám mây dựa trên thuật toán ACO", *Kỷ yếu Hội nghị Fair*, trang 331-340.

[8] O. M. ElzekiM. Z. ReshadM. A. Elsoud (2012), "Improved Max-Min Algorithm in Cloud Computing", *International Journal of Computer Applications*, Vol 50 (12).

[9] Ranjan Kumar, G. Sahoo (2014), "Cloud Computing Simulation Using CloudSim", *International Journal of Engineering Trends and Technology* (IJETT), Vol 8 (2).

[10] Soheil Anousha, Mahmoud Ahmadi (2013), "An Improved Min-Min Task Scheduling Algorithm in Grid Computing", *Grid and Pervasive Computing*, Vol 7861.

(BBT nhận bài: 15/09/2014, phản biện xong: 09/12/2014)