ĐỊNH VỊ TÀI NGUYÊN CHO CÁC TÁC VỤ TRÊN TÍNH TOÁN ĐÁM MÂY DỰA TRÊN RÀNG BUỘC DEADLINE VÀ NGÂN SÁCH

RESOURCE ALLOCATION FOR TASKS ON CLOUD COMPUTING BASED ON DEADLINE AND BUDGET CONSTRAINTS

Nguyễn Hoàng Hà[1], Lê Văn Sơn[2], Nguyễn Mậu Hân[1], Nguyễn Thanh Bình[1]
[1]Trường Đại học Khoa học, Đại học Huế; Email: nhha76@gmail.com, nmhan2005@yahoo.com, ntbinh@gmail.com
[2]Trường Đại học Sư phạm, Đại học Đà Nẵng; Email: levansupham2004@yahoo.com

Tóm tắt - Chất lượng dịch vụ (QoS) là một yếu tố không thể thiếu được khi lập lịch cho các tác vụ thời gian thực trên tính toán đám mây. Bài báo này đưa ra một thuật toán để ánh xạ tập các tác vụ với các tham số đầu vào như thời gian đến, deadline, ngân sách và khối lượng công việc vào tập con của tài nguyên có chi phí và tốc độ khác nhau. Chúng tôi xây dựng bài toán như một bài toán ràng buộc tối ưu và đưa ra một thuật toán với độ phức tạp thời gian đa thức để ánh xạ các tác vụ vào các tài nguyên một cách có hiệu quả, với mục tiêu tổng thời gian thực hiện (makespan) của các tác vụ là nhỏ nhất nhưng vẫn thỏa mãn deadline và ngân sách của tác vụ. Sau đó, chúng tôi sử dụng công cụ mô phỏng CloudSim để cài đặt và so sánh thuật toán này với thuật toán Earliest Deadline First (EDF).

Từ khóa - cloud computing, scheduling algorithms, QoS constraint, resource allocation, QoS

Abstract - Quality of services (QoS) is an inevitable issue to be dealt with in real time task scheduling of cloud computing. This paper proposes an algorithm to map a set of tasks with input parameters such as time, deadlines, budgets and workload to subset resources with cost and speed differences. The scheduling algorithm will be complexity polynomial time with optimal constraints in it, which maps effectively the resources with makespan of minimal tasks, but this still satisfies deadlines and budget tasks. Afterward, we use CloudSim tool to install and compare this algorithm with the algorithm Earliest Deadline First(EDF).

Key words - cloud computing, scheduling algorithms, QoS constraint, resource allocation, QoS

1. Giới thiệu

Tính toán đám mây cũng là mô hình tính toán phân tán với quy mô lớn. Nó cung cấp dịch vụ cho người dùng bằng cách thuê tài nguyên (phần cứng, phần mềm, tài nguyên lưu trữ, …) thông qua Internet. Người dùng có thể thuê các tài nguyên khác nhau dựa trên yêu cầu của họ và trả chi phí khi họ sử dụng.

Trong tính toán đám mây, mỗi tài nguyên là một máy ảo với tốc độ và chi phí khác nhau, nó đảm bảo hiệu suất cho người dùng. Mỗi máy ảo được thuê trong nhiều giờ và người dùng phải trả một chi phí cố định trong giờ được thuê, nếu họ không sử dụng hết một giờ thì họ cũng phải trả chi phí cho toàn bộ một giờ. Điều này thúc đẩy nhu cầu tìm kiếm một định vị hiệu quả về chi phí cho tập các tác vụ.

Lập lịch tác vụ tức là việc ánh xạ các tác vụ với các tham số thời gian đến, deadline, ngân sách và khối lượng công việc vào các tài nguyên với tốc độ và chi phí khác nhau. Đây là một bài toán NP-đầy đủ [1]. Để đưa ra một giải pháp tối ưu, chúng ta phải tìm kiếm vét cạn, khi đó độ phức tạp sẽ là hàm mũ, do đó cách này không thể được áp dụng. Để khắc phục nhược điểm này, người ta thường dùng các phương pháp heuristic nhằm đưa ra một giải pháp gần tối ưu như phương pháp tối ưu hóa đàn kiến (ACO) [2], kỹ thuật tối ưu hóa đàn ong mờ [3], phương pháp tham lam EDF[12][13], …

Trong môi trường tính toán đám mây, bộ lập lịch là thành phần rất quan trọng của hệ thống, nó quyết định tính hiệu quả của toàn bộ hệ thống. Thuật toán lập lịch trên môi trường tính toán đám mây có hai hướng chính: Dựa vào hiệu năng về hệ thống và dựa vào ràng buộc QoS (Quality of service).

Lập lịch dựa vào hiệu năng về hệ thống cố gắng để đưa ra một lịch trình có tổng thời gian thực hiện nhỏ nhất cho hệ thống. Các nghiên cứu [2],[7],[8],[9] lập lịch trên các tác vụ độc lập và phụ thuộc dữ liệu, sử dụng các heuristic và cải tiến thuật toán Max-min để đưa ra một lịch trình tối ưu về thời gian, nhưng các nghiên cứu chỉ tập trung vào lập lịch để tổng thời gian thực hiện cho hệ thống là nhỏ nhất không xét đến chi phí, deadline và ngân sách cho các tác vụ.

Trong môi trường tính toán đám mây, người sử dụng thuê các dịch vụ thông qua Internet và trả phí khi sử dụng. Do đó, các thuật toán lập lịch dựa vào ràng buộc QoS thường được sử dụng. Trong trường hợp này các tham số của người dùng như thời gian, phí dịch vụ cho người sử dụng, phí dịch vụ cho nhà cung cấp, độ tin cậy,… được ưu tiên xem xét khi lập lịch. Jzau-Sheng Lin và các đồng nghiệp [4] đưa ra mô hình lập lịch cho các tác vụ trên môi trường tính toán đám mây nhưng mục đích của thuật toán là làm sao đem lại lợi nhuận cao nhất cho nhà cung cấp dịch vụ. Các nghiên cứu [5],[6] lại tập trung vào lập lịch trên các tác vụ để tiết kiệm điện năng trên các trung tâm dữ liệu. Các nghiên cứu gần đây của Ramkumar N [11] về lập lịch trên tác vụ thời gian thực đã sử dụng hàng đợi ưu tiên để ánh xạ tác vụ vào tài nguyên nhưng chỉ tập trung lập lịch để giải quyết công việc một cách nhanh nhất thỏa mãn deadline của tác vụ mà không quan tâm đến chi phí và ngân sách của các tác vụ. S. Liu [10] đưa ra thuật toán lập lịch sử dụng các tham số của người dùng như chi phí, deadline và độ tin cậy. Trong bài báo này chúng tôi đưa ra thuật toán lập lịch cho các tác vụ thời gian thực sử dụng các ràng buộc QoS như chi phí, deadline và ngân sách của các tác vụ với mục tiêu chi phí nhỏ nhất nhưng vẫn thỏa mãn deadline và ngân sách của các tác vụ. Vì vậy, công việc của Jzau-Sheng Lin [4] đem lại lợi ích cho nhà cung cấp dịch vụ còn công việc của chúng tôi đem lại lợi ích cho người dùng.

Bài báo bao gồm 4 phần: Phần 1 giới thiệu, phần 2 lập lịch cho các tác vụ bao gồm mô tả bài toán, xây dựng bài

toán và đưa ra thuật toán, phần 3 mô phỏng và đánh giá, phần 4 kết luận.

2. Lập lịch cho các tác vụ

2.1. Mô tả bài toán

Ứng dụng bao gồm tập các tác vụ T, mỗi tác vụ $t_i \epsilon T$ có thời gian đến là a_i, deadline là d_i (chỉ rõ bằng phút) và ngân sách (budget) là b_i (chỉ rõ bằng $) và khối lượng công việc là w_i. R là tập tài nguyên có sẵn. Mỗi tài nguyên $r_j \in R$ có tốc độ tính toán s_j và tương ứng chi phí c_j để thuê máy ảo. Tốc độ s_j là số chu kỳ của tài nguyên có thể hoàn thành trên phút. Người dùng trả chi phí c_j để thuê tài nguyên r_j trong khoảng D phút liên tục, D là đơn vị nhỏ nhất để thuê. Bài toán có thể mô tả như sau: *Tìm một ánh xạ từ T vào một tập con của R để có tổng thời gian hoàn thành là nhỏ nhất trong khi vẫn thỏa mãn deadline và ngân sách của tất cả các tác vụ.*

2.2. Xây dựng bài toán

Đặt R={$r_1,r_2,..,r_m$} là tập m tài nguyên tính toán, mỗi tài nguyên r_j là bộ <s_j,c_j>, trong đó s_j là tốc độ tính toán tính trên phút, c_j là chi phí để thuê tài nguyên theo giờ. Các tài nguyên này là nguồn tính toán cho n tác vụ, được biểu diễn bởi tập T={$t_1,t_2,\ldots,t_n$}, mỗi tác vụ là một bộ 4 <a_i,d_i,b_i,w_i> trong đó:

- a_i: thời gian đến của tác vụ t_i
- d_i: deadline của t_i, thời gian để hoàn thành tác vụ t_i phải nhỏ hơn hoặc bằng d_i.
- b_i: ngân sách, chi phí thực hiện tác vụ t_i phải nhỏ hơn hoặc bằng b_i
- w_i: khối lượng công việc của t_i tính bằng chu kỳ.

Tại một thời điểm các tác vụ có thể thuê một tài nguyên nhiều lần, ta sử dụng $\alpha(j,k)$ để thể hiện việc chọn tài nguyên:

$$\alpha(j,k) = \begin{cases} 1, & \text{nếu } r_j \text{ được sử dụng } k \text{ lần} \\ 0, & \text{ngược lại} \end{cases} \quad (1)$$

Tổng chi phí của tất cả các tài nguyên được tính như công thức (2)

$$\sum_{j=1}^{m}\sum_{k=1}^{y} c_j \times \alpha(j,k) \quad (2)$$

Trong đó chỉ số trên của y được xác định như sau: Trong tính toán đám mây người dùng có thể lựa chọn bất kỳ tài nguyên nào với số lượng khác nhau, giá trị y có thể là vô cùng, tuy nhiên ta có thể suy ra chỉ số trên của y từ tập các tác vụ T đã có. Tổng khối lượng của T được cho bởi:

$$C = \sum_{i=1}^{n} w_i \quad (3)$$

Nếu chúng ta xét trường hợp tốc độ nhỏ nhất của các tài nguyên là p, khi đó số tài nguyên lớn nhất có thể được dùng là $\frac{C}{p}$, vì vậy chỉ số trên của y được xác định bởi:

$$y \le \frac{1}{p} \times \sum_{i=1}^{n} w_i \quad (4)$$

Như công thức (1) chúng ta không biết khi nào các tài nguyên được sử dụng, do đó, chúng ta xét thêm yếu tố thời gian để các tác vụ chọn các tài nguyên. Ký hiệu $\beta(j,k,x)$ để thể hiện tại phút thứ x thuê k tài nguyên thứ j

$$\beta(j,k,x) = \begin{cases} 1, \text{nếu } r_j \text{ được sử dụng } k \text{ lần tại phút thứ } x \\ 0, \text{ngược lại} \end{cases} \quad (5)$$

Như vậy tại phút thứ x, tổng số chu kỳ được thực hiện là:

$$\sum_{j=1}^{m}\sum_{k=1}^{y} s_j \times \beta(j,k,x) \quad (6)$$

Mục tiêu của chúng ta là tổng thời gian hoàn thành là nhỏ nhất, do đó tổng số chu kỳ được thực hiện của tất cả các tác vụ là lớn nhất như thể hiện ở công thức (7):

$$max \sum_{i=1}^{n}\sum_{x=a_i}^{d_i}\sum_{j=1}^{m}\sum_{k=1}^{y} s_j \times \beta(j,k,x) \quad (7)$$

Để đạt được mục tiêu như công thức (7) thì phải thỏa mãn các ràng buộc:

- Chi phí của tất cả các tác vụ phải thỏa mãn ngân sách của nó tức là:

$$\sum_{j=1}^{m}\sum_{k=1}^{y} c_j \times \alpha(j,k) \le \sum_{i=1}^{n} b_i \quad (8)$$

- Thời gian thực hiện của các tác vụ phải thỏa mãn deadline của nó, chúng ta cần phải chọn đủ số tài nguyên và số lượng của mỗi tài tài nguyên cần thuê ở (2) để mỗi tác vụ có thể hoàn thành sau thời gian đến và trước deadline. Mỗi tác vụ $t_i \epsilon T$ cần phải thỏa mãn ràng buộc:

$$\sum_{x=a_i}^{d_i}\sum_{j=1}^{m}\sum_{k=1}^{y} s_j \times \beta(j,k,x) \ge w_i \quad (9)$$

2.3. Thuật toán

Phần này chúng tôi xây dựng thuật toán dựa trên bài toán đặt ra ở phần 2.1 và 2.2. Thuật toán 1: CTO chúng tôi xây dựng bảng dò tìm để xác định số lượng của mỗi tài nguyên cho mỗi tác vụ. Thuật toán 2: MINC xác định khoảng thời gian gối đầu lên nhau giữa các tác vụ để tận dụng khoảng thời gian còn hiệu lực của mỗi tác vụ và đưa ra lịch trình để ánh xạ các tác vụ vào các tài nguyên.

Thuật toán CTO.

Đầu vào: Tập n các tác vụ T={$t_1,t_2,\ldots,t_n$}, mỗi tác vụ là một bộ 4 <a_i,d_i,b_i,w_i>. Tập R={$r_1,r_2,..,r_m$} là tập m tài nguyên tính toán, mỗi tài nguyên r_j là bộ <s_j,c_j>. Ngưỡng ρ, dùng ngưỡng này để nhóm các tài nguyên có tốc độ gần bằng nhau trên 1 nhóm.

Đầu ra: Bảng dò tìm để xác định số lượng các tài nguyên cho các tác vụ.

Thuật toán:

Bước 1: Sắp xếp các tài nguyên theo thứ tự giảm dần của tốc độ.

Bước 2: Duyệt qua các tài nguyên đã được sắp xếp theo tốc độ, dựa vào ngưỡng ρ để nhóm các tài nguyên có tốc độ gần bằng nhau trên từng nhóm, sau đó sắp sếp các tài nguyên trên từng nhóm theo thứ tự tăng dần của chi phí.

Bước 3: Trên mỗi nhóm ưu tiên ánh xạ tác vụ vào tài nguyên có chi phí thấp, sau đó nếu tác vụ chưa hoàn thành ánh xạ tác vụ vào các tài nguyên còn lại của nhóm. Đưa ra bảng dò tìm để xác định số lượng các tài nguyên cho các tác vụ.

Một số hạn chế của thuật toán CTO:

- Các tác vụ có thể hoàn thành trước deadline của nó nhưng có thể không thõa mãn ngân sách.

- Thời gian thuê tài nguyên là D phút nhưng có thể tác vụ t_i nào đó không sử dụng hết D phút. Do đó, trong phần này chúng ta định nghĩa tập T_i bao gồm các tác vụ gối lên tác vụ t_i và các tác vụ này có thể chia sẻ cùng 1 tài nguyên:

$$T_i = \left\{ t_j | d_j \geq d_i \text{ và } a_j < d_i \right\} \quad (10)$$

Sau khi xác định được tập T_i, ta tiến hành tính khoảng thời gian gối đầu lên nhau. Ta định nghĩa t_{ij} là thời gian của tài nguyên định vị cho t_i vẫn còn hiệu lực để tính toán cho t_j. Giá trị t_{ij} phụ thuộc tốc độ s của tài nguyên cuối cùng được ánh xạ cho t_i, phụ thuộc vào thời gian đến, deadline và khối lượng công việc của t_i và t_j. T_{ij} được tính như sau:

$$t_{ij} = \begin{cases} min\left(D - U_{ij}, d_j - a_j\right) & Nếu\ a_j - a_i \geq \frac{w_i}{s} \\ D - U_{ij} & Nếu\ a_j - a_i < \frac{w_i}{s}\ và\ d_j - a_i \geq D \\ d_j - \left(a_i + U_{ij}\right) & Nếu\ a_j - a_i < \frac{w_i}{s}\ và\ d_j - a_i < D \end{cases} \quad (11)$$

Trong đó: $U_{ij} = \frac{w_i}{s} + max\left(a_j - d_i, 0\right)$, s là tốc độ của tài nguyên cuối cùng được ánh xạ cho t_i

Ví dụ: Giả sử D=60 và t_1(0,30,1.5,500) có thời gian thực hiện trên tài nguyên r_2(20,1.5) của nhóm 1là: 25 phút, còn dư từ phút 26 đến 60.t_2(30,50,2,1200) có $a_j - a_i$ =30> $\frac{w_i}{s}$ =25 nên rơi vào trường hợp đầu tiên của công thức (11). Khi đó U_{ij} =25+max(30-30,0)=25 và $min\left(D - U_{ij}, d_j - a_j\right)$ =min(60-25,50-30)=20, như vậy tài nguyên r_2 còn hiệu lực cho 60-25=35 phút cho t_2 nhưng thời gian đến của t_2 từ phút thứ 30 nên chỉ sử dụng được min(35,20)=20 phút. Hai trường hợp còn lại của công thức (11) xét tương tự.

Thuật toán MINC

Đầu vào:

- ST={}; tập các tác vụ đã lập lịch

- UST=T; tập các tác vụ chưa lập lịch

- TB: Bảng dò tìm để xác định số lượng các tài nguyên cho các tác vụ của thuật toán CTO.

- 1 ngăn xếp để lưu các tác vụ

Đầu ra: Một lịch trình để ánh xạ các tác vụ vào các tài nguyên

Thuật toán:

1. Dựa vào TB để tìm ra tác vụ t_i đầu tiên thỏa mãn deadline và chi phí
2. PUSH(t_i); *// Lưu trữ t_i vào ngăn xếp*
3. ST=ST+{t_i}; UST=UST-{t_i};
4. Do while $UST \neq \varnothing$
5. t_i=POP(); *// Lấy t_i từ ngăn xếp*
6. Tìm $T_i = \left\{ t_j | d_j \geq d_i \text{ và } a_j < d_i \right\}$
7. Tính t_{ij} của các tác vụ trong T_i. t_{ij} được tính trên tài nguyên cuối cùng được ánh xạ bởi tác vụ t_i:

.

8. Tính max(t_{ij}), t_j có thời gian gối đầu lớn nhất làm tiếp vụ tiếp theo
9. Dựa vào max(t_{ij}) để tính lại w_j cho tác vụ t_j.
10. Trên mỗi nhóm của TB ưu tiên ánh xạ tác vụ t_j vào tài nguyên có chi phí thấp, sau đó nếu t_j chưa hoàn thành ánh xạ t_j vào các tài nguyên còn lại của nhóm. Tìm ra được các tài nguyên đầu tiên thỏa mãn deadline và ngân sách cho t_j
11. PUSH(t_j);
12. ST=ST+{t_j}; UST=UST-{t_j};
13. **Loop**
14. Dựa vào tập ST để đưa ra lịch trình để ánh xạ các tác vụ vào các tài nguyên

Cơ sở lý luận.

- Sự hình thành của hai tập ST và UST chắc chắn rằng chỉ có giới hạn vì các tác vụ được lập lịch theo lô và theo một chu kỳ tức là ta sử dụng i tập tác vụ chưa lập lịch. Cứ tập tác vụ này đang được lập lịch thì tập tác vụ kia tiếp tục nhận tác vụ chưa lập lịch và lưu vào hàng đợi, khi tập tác vụ này lập lịch xong thì hệ thống sẽ lập lịch cho tập tác vụ lưu ở hàng đợi và quá trình cứ lặp lại.

- Mục tiêu đặt ra là tổng thời gian hoàn thành là nhỏ nhất như ở công thức (7)và thỏa mãn hai ràng buộc (8) và (9). Với dữ liệu đầu vào của thuật toán là danh sách các tài nguyên có sẵn trên trung tâm dữ liệu của tính toán đám mây, chúng tôi sắp xếp các tài nguyên này theo thứ tự giảm dần của tốc độ, sau đó dựa vào ngưỡng ρ để phân nhóm các tài nguyên và sắp sếp tăng dần theo giá trên từng nhóm này. Như vậy, nếu ánh xạ các tác vụ vào tài nguyên đầu tiên trên mỗi nhóm sẽ có tốc độ thực hiện là lớn nhất và chi phí thì nhỏ nhất, điều này sẽ làm giảm chi phí và thời gian hoàn thành cho cả hệ thống.

- Thời gian thuê tài nguyên là D phút, do đó có thể một tác vụ t_i hoàn thành công việc của mình với thời gian ít hơn D phút nhưng phải trả chi phí trong vòng D phút. Thuật toán trên tận dụng khoảng thời gian còn hiệu lực này để thực hiện cho tác vụ tiếp theo, điều này dẫn đến chi phí thực hiện cho cả hệ thống giảm xuống.

3. Mô phỏng và đánh giá thuật toán

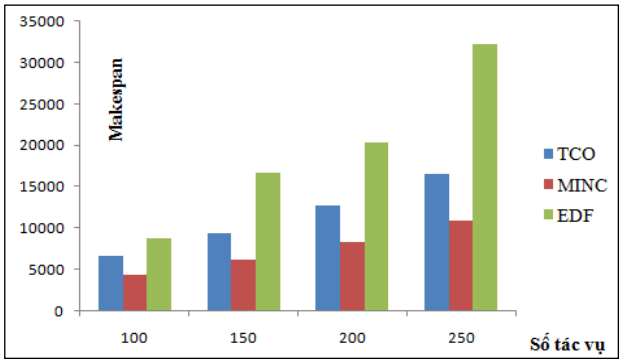
Các thuật toán được cài đặt mô phỏng bằng ngôn ngữ Java (NetBean 7.1.1, JDK 6), gói công cụ CloudSim 2.0 [14] với các thông số sau: sử dụng 1 Datacenter, 2 host vật lý, 50 máy ảo, 4 PE và 512 RAM trên một máy ảo và số tác vụ (Cloudlet) thay đổi từ 100 đến 250

Trong cài đặt, chúng tôi sử dụng các API của CloudSim

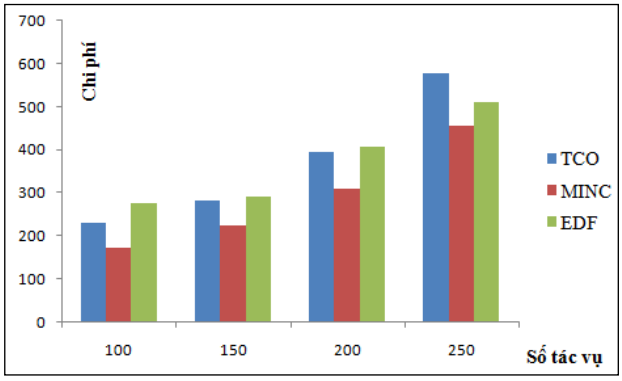
2.0, kế thừa từ lớp DataCenterBroker để tạo ra chính sách lập lịch mới tương ứng với thuật toán CTO và MINC đề xuất. Kế thừa lên lớp Vm của CloudSim 2.0 để tạo ra các máy ảo với các thông số tốc độ và chi phí được xác định như sau: tốc độ được lấy ngẫu nhiên từ 10 đến 50 tương ứng với chi phí là số thực được lấy ngẫu nhiên từ 0.1 đến 1. Kế thừa lên lớp Cloudlet để tạo ra các tác vụ với các thông số: thời gian đến, khối lượng công việc, ngân sách và deadline được xác định như sau: Thời gian đến được lấy ngẫu nhiên từ 1 đến 500, chúng ta phát sinh deadline một cách ngẫu nhiên giữa (d_l,d_u) phút và các giá trị khác nhau d_l và d_u có giới hạn từ 10 đến 1500, deadline phải lớn hơn thời gian đến. Khối lượng công việc được lấy ngẫu nhiên từ 10 đến 5000 chu kỳ, căn cứ vào khối lượng để ước lượng ngân sách cho các tác vụ.

3.1. Phân tích tổng thời gian và tổng chi phí thực hiện

Nếu chúng ta sử dụng thuật toán tìm kiếm vét cạn để ánh xạ các tác vụ vào các tài nguyên thì luôn tìm ra thời gian hoàn thành và chi phí thấp nhất nhưng thời gian đưa ra lịch trình rất lớn vì độ phức tạp của thuật toán là hàm mũ.



Hình 1. *So sánh tổng thời gian thực hiện (makespan) của 3 thuật toán khi thay đổi số tác vụ*

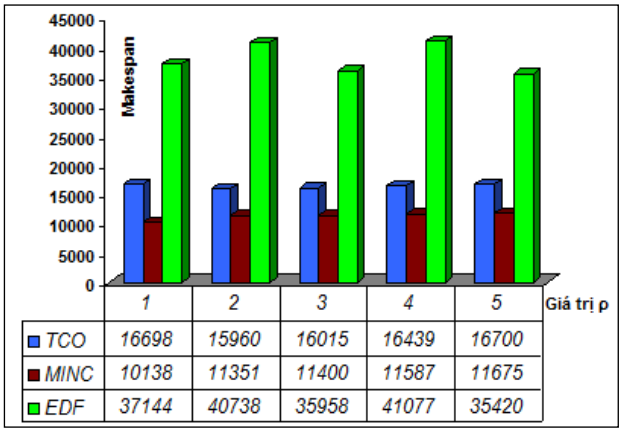


Hình 2. *So sánh tổng chi phí của 3 thuật toán khi thay đổi số tác vụ*

Hình 1 và hình 2 chỉ ra tổng thời gian thực hiện (makespan) và tổng chi phí của 3 thuật toán CTO, MINC và EDF khi $\rho = 5$, 50 máy ảo và các tác vụ thay đổi từ 100 đến 250. Các giá trị trong hình 1 và 2 là kết quả của 5 lần chạy thử và lấy kết quả trung bình. Khi số tác vụ càng cao thì tổng thời gian thực hiện và tổng chi phí của thuật toán MINC luôn nhỏ hơn thuật toán CTO và EDF vì thuật toán MINC xem xét việc chia sẻ thời gian gối đầu của các tác vụ lên các tài nguyên, trong khi đó thuật toán CTO không xem xét thời gian gối đầu, còn thuật toán EDF chỉ xem xét đến tỉ số sử dụng: $U = \sum_{i=1}^{m} \frac{C_i}{T_i} \le 1$ (trong đó C_i là thời gian thực hiện và T_i tương ứng với Deadline)[15][16] để ánh xạ tác vụ vào các tài nguyên. Do đó, thuật toán EDF chỉ đảm bảo các tác vụ hoàn thành trước deadline của nó chứ không quan tâm đến chi phí cho các tác vụ.

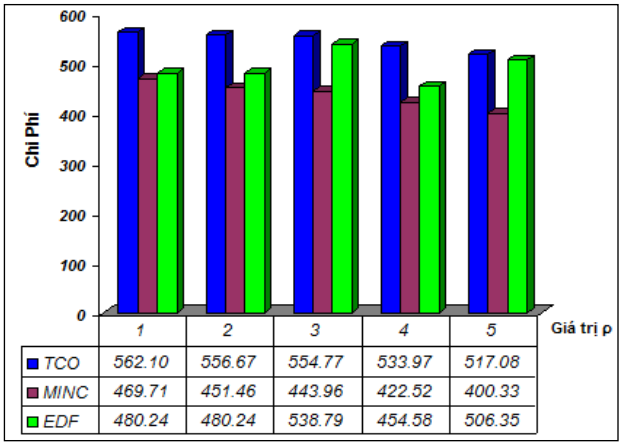
3.2. Thay đổi ngưỡng ρ

Chúng tôi thay đổi ngưỡng ρ từ1 đến 5, sử dụng 250 tác vụ và 50 máy ảo, kết quả được thể hiện ở hình 3 và hình 4. Khi ngưỡng ρ càng nhỏ như ở hình 3 thì các tài nguyên có tốc độ cao sẽ nằm ở nhóm đầu tiên và MINC sẽ tập trung ánh xạ các tác vụ vào tập các tài nguyên này và tiến hành chia sẻ thời gian gối đầu giữa các tác vụ, do đó khi ngưỡng ρ càng nhỏ thì tổng thời gian thực hiện của MINC sẽ nhỏ hơn CTO và EDF. Nhưng ngược lại khi tài nguyên có tốc độ càng cao thì chi phí của nó càng lớn, do đó khi ngưỡng ρ càng nhỏ thì cả hai thuật toán MINC và CTO đều có tổng chi phí cao như ở hình 4.



	1	2	3	4	5	Giá trị ρ
TCO	16698	15960	16015	16439	16700	
MINC	10138	11351	11400	11587	11675	
EDF	37144	40738	35958	41077	35420	

Hình 3. *So sánh thời gian tổng thời gian thực hiện (makespan) của 3 thuật toán khi thay đổi ρ*



	1	2	3	4	5	Giá trị ρ
TCO	562.10	556.67	554.77	533.97	517.08	
MINC	469.71	451.46	443.96	422.52	400.33	
EDF	480.24	480.24	538.79	454.58	506.35	

Hình 4. *So sánh tổng chi phí của 3 thuật toán khi thay đổi ρ*

Phân tích độ phức tạp.

- Thuật toán EDF duyệt qua các tác vụ và các máy ảo, dựa vào tỉ số sử dụng để ánh xạ tác vụ vào máy ảo, do đó độ phức tạp của thuật toán EDF là O(n.m) với n là số tác vụ, m là số máy ảo.

- Thuật toán CTO sử dụng thuật toán Quick Sort để sắp xếp các tác vụ theo chiều giảm dần của tốc độ và trên từng nhóm sắp xếp tăng dần của chi phí, do đó độ phức tạp của

hai lần sắp xếp là $O\left(2.m.log_2^m\right)$. Sau đó duyệt qua các tác vụ để ánh xạ các tác vụ vào các máy ảo nên độ phức tạp của thuật toán CTO là $O\left(2.m.log_2^m + n\right)$.

- Đối với thuật toán MINC, chúng ta duyệt qua các tác vụ chưa lập lịch, cứ mỗi tác vụ chưa lập lịch sẽ duyệt qua các tác vụ để tìm ra tập T và tính max(t_{ij}), từ đó tìm ra tác vụ t_j để ánh xạ vào các máy ảo của nhóm đầu tiên trong tập các máy ảo. Do đó, độ phức tạp của hai vòng lặp sẽ là $O(n^2)$. Kết quả đầu ra của thuật toán CTO là đầu vào của thuật toán MINC, do đó độ phức tạp của thuật toán MINC là $O\left(2.m.log_2^m + n\right) + O\left(n^2\right) = O\left(2.m.log_2^m + n(n+1)\right)$. Độ phức tạp này vẫn đảm bảo độ phức tạp thời gian đa thức cho thuật toán được đề xuất.

4. Kết luận

Bài báo đã tập trung nghiên cứu các tác vụ thời gian thực với các ràng buộc QoS, mỗi tác vụ chúng tôi xem xét đến các yếu tố như thời gian đến, deadline, khối lượng và ngân sách, mỗi máy ảo bao gồm tốc độ và chi phí khác nhau. Chúng tôi xây dựng bài toán ánh xạ tác vụ vào các máy ảo như là bài toán ràng buộc tối ưu và đưa ra thuật toán với thời gian đa thức để giải quyết bài toán này. Thông qua việc phân tích và kết quả thực nghiệm đối sách trên cùng một bộ mẫu và sử dụng cùng một công cụ mô phỏng CloudSim đã cho thấy kết quả của thuật toán MINC có sự cải tiến đáng kể về tổng thời gian thực hiện và chi phí so với giải thuật CTO và EDF hiện có.

TÀI LIỆU THAM KHẢO

[1] P. Brucker, *Scheduling Algorithms*, Fifth Edition, Springer Press, 2007.

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

[3] Jzau-Sheng Lin and Shou-Hung Wu, *Fuzzy Artificial Bee Colony System with Cooling Schedule for the Segmentation of Medical Images by Using of Spatial Information*, Research Journal of Applied Sciences, Engineering and Technology 4, 2973-2980, 2012

[4] Jianguang Deng, Yuelong Zhao, Huaqiang Yuan, *A Service Revenue-oriented Task Scheduling Model of Cloud Computing*, Journal of Information & Computational Science 10:10 (2013) 3153–3161 July 1, 2013.

[5] Mao, Ming and Li, Jie and Humphrey, Marty, *Cloud auto-scaling with deadline and budget constraints, Grid Computing (GRID)*, 11th IEEE/ACM International Conference, 2010.

[6] K. H. Kim et al, *Power-aware provisioning of cloud resources for real-time services*. In International Workshop on Middleware for Grids, Clouds and e-Science, pages 1–6, 2009.

[7] O. M. Elzeki, M. Z. Reshad,M. A. Elsoud, *Improved Max-Min Algorithm in Cloud Computing*, International Journal of Computer Applications (0975 – 8887), Volume 50 – No.12, July 2012.

[8] Suraj Pandey, Scheduling and Management of Data Intensive ApplicationWorkflows in Grid and Cloud Computing Environments, December 2010.

[9] T. Kokilavani, Dr. D.I: George Amalarethinam, *Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing*, International Journal of Computer Applications (0975 – 8887) Volume 20– No.2, April 2011.

[10] S. Liu et al, *On-Line Scheduling of Real-Time Services for Cloud Computing*. In World Congress on Services, pages 459–464, 2010.

[11] Ramkumar N, Nivethitha S, *Efficient Resource Utilization Algorithm (ERUA) for Service Request Scheduling in Cloud*, International Journal of Engineering and Technology (IJET), Vol 5 No 2 Apr-May 2013.

[12] A. Burns, R.I. Davis, P. Wang and F. Zhang, *Partitioned EDF Scheduling for Multiprocessors using a C=D Scheme*. Department of Computer Science, University of York, UK.

[13] Łukasz Kruk, John Lehoczky, Kavita Ramanan And Steven Shreve, *Heavy traffic analysis for EDF queues with reneging*, The Annals of Applied Probability. Vol. 21, No. 2, 484–545, 2011.

[14] Buyya, R., Ranjan, R., Calheiros, R.N, *Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities*, Proceedings of the 7th High Performance Computing and Simulation (HPCS 2009) Conference, Leipzig, Germany, DOI: 10.1109/HPCSIM.2009. 5192685, 2009.

[15] Http://en.wikipedia.org/wiki/Earliest_deadline_ first_ scheduling

[16] Http://www.tcbin.com/search?q=bin-packing-assignment - algorithm-for-edf-scheduling.

(BBT nhận bài: 24/03/2014, phản biện xong: 19/04/2014)