Computers and Electrical Engineering 76 (2019) 13–23

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Fair service matching agent for federated cloud☆

S. Sudhakar[a,*], N.S. Nithya[b], B.L. Radhakrishnan[a]

[a] Department of Computer Science and Engineering, Karunya Institute of Technology and Sciences, Coimbatore, Tamilnadu 641114, India
[b] Department of Computer Science and Engineering, K.S.R College of Engineering, Namakkal, Tamilnadu 637215, India

ARTICLE INFO

Article history:
Received 8 December 2018
Revised 5 March 2019
Accepted 6 March 2019

Keywords:
Agent based service selection
Federated cloud
Fair service matching
Quality of services
Free riding

ABSTRACT

A federated cloud is a group of cloud service providers operating together for service sharing. In the federation, cloud service providers obtain services from providers when their services are insufficient to meet the demand. The federated cloud uses agent-based, incentive-based or reciprocal resource fairness-based techniques for service selection. In agent-based service selection, the agent selects the best service provider for service sharing. Distributing the service requests among the service providers of the federation is a challenging task. The agent-based service selection techniques do not distribute service requests to all cloud providers in the federated cloud. Additionally, the agent-based service selection techniques rarely present service offers to the newly joined cloud provider. To fairly distribute the service requests, a new agent named "A Fair Service Matching Agent (FSMA)", fairly distributes service requests to all cloud providers in the federated cloud and provides a fair chance to the newly joined cloud provider. The experimental results demonstrate FSMA outperforms the existing traditional Cloud Resource Bartering system.

1. Introduction

The federated cloud is a system for sharing, migrating and aggregating services among divergent, heterogeneous cloud providers through the centralized control mechanism. Peer-to-peer and vertical are two models of federated cloud. In peer-to-peer, communication happens among cloud providers through a centralized agent or a gateway at the same level. In vertical federation, communication happens at different levels of cloud hierarchy. A free rider is a selfish cloud provider in the federation who do not share its service with others [1]. Several free riders exist among the cloud providers in the federation. Permitting free riders to consume services creates service unavailability for legitimate cloud providers. Allowing a cloud provider to consume services, based on its service contribution, solves free rider problem in federated cloud [2,3]. There should be a mechanism to coordinate, control, and monitor service sharing in the federation.

An agent is application software that makes independent decisions, conducts autonomous actions, and communicates with other software agents [4]. The federated cloud serves customers by aggregating services from many cloud providers using the agent. The agent in the federated cloud performs discovery, matching, selection, composition, monitoring, negotiation, and schedule services [5]. Service selection in federated cloud is performed and controlled by intelligent software agents. The agent-based service selection approaches presented in [6–8], and [9] employ intelligent software agents to select best services in federated cloud.

☆ This paper is for regular issues of CAEE. Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. S. Smys.

* Corresponding author.
E-mail address: sudhakar.sree@gmail.com (S. Sudhakar).

https://doi.org/10.1016/j.compeleceng.2019.03.002
0045-7906/

Cloud providers are permitted to reasonably consume and offer services [10]. The agent uses parameters like Quality of Service (QoS), performance cost ratio and service contributions while selecting service provider to contribute services. The QoS based service provider selection always selects the highest-quality service provider repeatedly to share the services [11]. This approach does not distribute service requests proportionately and denies other service providers the service offerings based on QoS. Moreover, when new cloud providers join the federation, they do not have QoS value, and QoS-based selection do not select new cloud providers when the existing cloud providers do have services [12]. The service contribution-based service selection method selects the provider with reasonable service contribution history. This method solves the free-riding problem but do not guarantee the best QoS and performance-cost ratio. In a service contribution-based method, new cloud provider is never selected as they do not have any service contribution history.

None of the existing agent-based service selection approach distributes the service requests equally to all service providers. Additionally, existing approaches do not solve the new cloud provider problem. The proposed Fair Service Matching Agent System (FSMA) selects the high-quality cloud provider that has a fair service contribution record with the federation. The agent in the FSMA distributes the service requests evenly to all registered cloud providers. The FSMA applies an innovative method to solve the new cloud provider problem.

The paper has been organized as follows: Section 2 discusses related works, Section 3 illustrates the FSMA system, Section 4 analyzes the results, and Section 5 presents the findings of the study on FSMA.

2. Related works

Many agent-based service selection approaches are available. The distributed multi agent framework proposed in [6] selects web services based on the services' QoS. The Dynamic Programming Optimization Protocol for Quality of Service (DPOP4QoS) algorithm proposed in this system computes utility value based on QoS of web services, constructs the Depth First Search (DFS) tree and ranks web services based on attributes' QoS. The DPOP4QoS algorithm selects the best web service that has high QoS value from considerable number of services, available in the market in less duration. The disadvantages of the DPOP4QoS algorithm are increase in the dimensions of hypercube (message size) and the amount of memory required to process service request.

The agent-based cloud computing methods implemented in [4,13,14], and [15] are approaches for cloud service discovery, negotiation, and composition. These approaches select a service provider that meets the required QoS for the service. The ontology-based multi agent systems proposed in [7], and [16] are service recommendation systems for cloud computing. Multiple agents used to discover and match services from the cloud based on similarity between services and service requests. These systems recommend service providers to meet the requested QoS.

The service selection across multi cloud environment was proposed in [17,18], and [19] selects services from multiple clouds and combines them as a single service. The service selection uses multiple agents to discover and integrate services from multiple cloud providers. Services are aggregated from the service providers to meet the requested QoS and integrated into a single service. The Service Level Agreement (SLA) based service selection was proposed in [20] and [21] choose a service provider based on their SLAs. The number of times a service provider met and violated SLA is considered while selecting service from service providers.

The service contribution fairness based service selection system that was developed in [3,22,23] and [24] assigns service contribution values based on service providers' past contributions. Services selected from service provider have highest service contribution value. The adaptive learning agent developed in [25] presents several dynamic service selection algorithms for service selection. The intermediate service agent in the system uses the dynamic service selection algorithms to find service providers. The facility location agent in [26] uses greedy algorithm to find web services. The greedy algorithm considers two factors while selecting web services: (a) a number of services, connected to community (b) a number of simultaneous service requests. A QoS based data-intensive service selection system employs a multiphase ant colony algorithm to match service providers. This system uses multiparty negotiation protocol for service negotiation.

The various models, discussed above consider the factors such as QoS, service contribution, and fairness individually while selecting services. The models do not consider a combination of these factors while selecting services. Additionally, none of the models discussed service distributions equitably if more than one service provider meets the service requirement. We propose a Fair Service Matching Agent (FSMA) system that considers a combination of the factors while selecting the service provider and distributes service requests fairly.

3. Fair service matching agent system

3.1. Cloud provider classes

All cloud providers voluntarily register to offer services with the cloud federation. The FSMA agents in federated cloud records the details of cloud providers, such as list of services, number of services available, duration, cost of services and cost adjustment factor. The agent maintains cloud providers in two classes, namely, (a) new class (b) regular class.

The new class contains cloud providers who have newly joined the federation. A new cloud provider does not have any transaction history in order to calculate a QoS while joining the federation. The QoS-based cloud provider selection selects

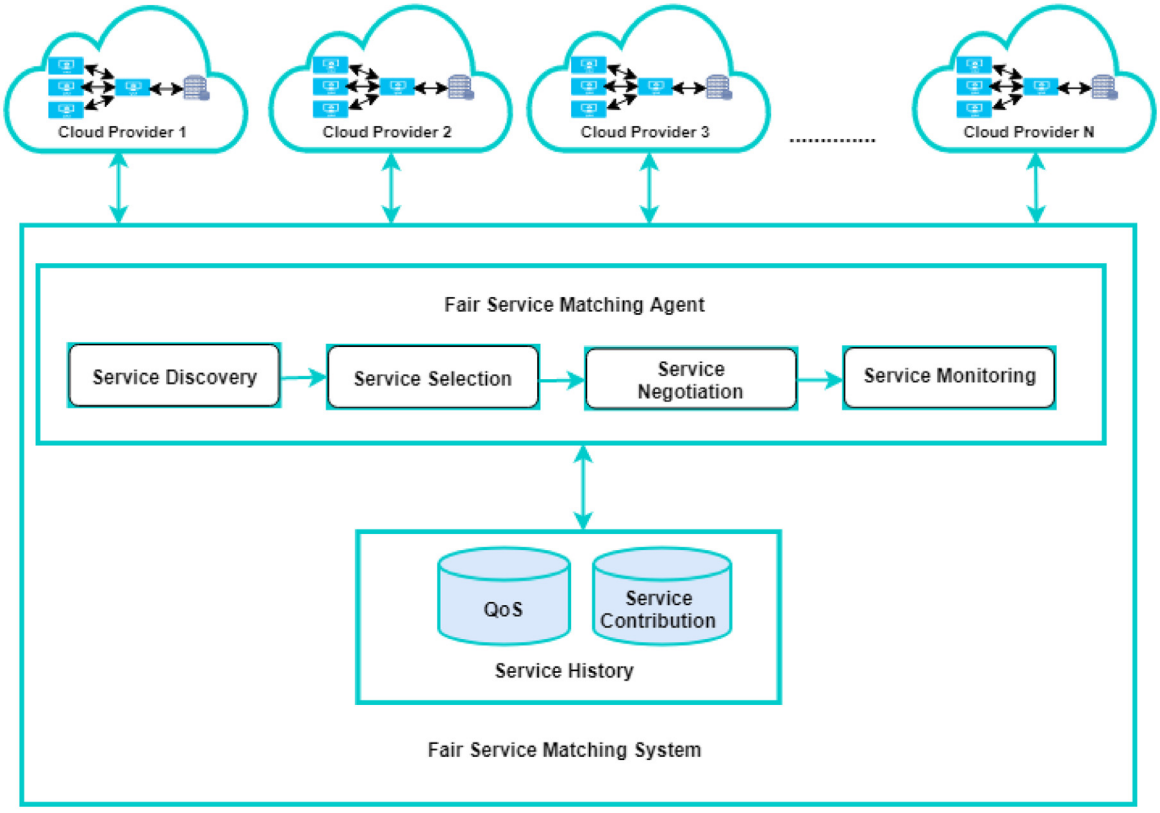

Fig. 1. Fair service matching system architecture.

the new cloud provider to deliver a service when existing cloud providers do not have services [12], which leads to starvation. The starvation prolongs selection of a cloud provider to share its services with others. A free rider is a cloud provider unable to clear previous debt over some length of time due to service unavailability to reciprocate. The free rider needs to obtain services from federation in order to meet current demand but is prohibited from getting services until previous debt is cleared [1]. The FSMA proposes new approach to solve these problems. The FSMA determines service requesting cloud provider type when it receives a service request. If service requester is a free rider, then service request is assigned to cloud provider in the new class. The FSMA solves free riding and starvation problems by allowing free riders to obtain services from new cloud providers, helping new cloud providers to obtain QoS values. Additionally, FSMA enables free riders to meet their current demand. A cloud provider in new class can move to regular class after establishing sufficient service contribution history. The regular class contains cloud providers with past service contribution histories. If the requesting cloud provider is not a free rider, cloud provider is permitted to avail services from regular class.

3.2. FSMA system architecture

The service selection of FSMA system involves complex tasks; service discovery, selection, negotiation, and monitoring. The FSMA system performs service selection task 24/7 and updates every change in services. The architecture of FSMA system is shown in Fig. 1. The service requesting cloud provider makes a service request to FSMA system. The FSMA agent receives a request from cloud provider and processes it.

The FSMA agent initially identifies request making cloud provider's type, as a regular or free rider. The agent chooses new class if requesting cloud provider's type is free rider, else it choose regular class. After identification, FSMA agent determines service offering cloud provider in consultation with cloud provider's service history database. The agent filters few service offering cloud providers from the list based on required type of service, required number of services, QoS, cost of service, and duration. Then, FSMA agent orders service offering cloud provider based on performance-cost value. The cloud provider with least service offering count is selected for negotiation process. Once negotiation process succeeds, service delegation happens, else next cloud provider is chosen for negotiation. After service delegation, FSMA agent monitors services, records QoS and service contribution in service history database. This process repeats whenever a request is made by cloud provider.

3.3. Cloud provider selection

A registered service requesting cloud provider makes service request to agent in order to obtain services from others. The cloud provider's service request includes the following parameters: <service type, number of services required, duration, QoS value, maximum service cost, and service cost adjustment factor>. If the requesting cloud provider is a free rider, the agent selects cloud providers from new class (NC) and assigns them to cloud provider list (CPl).

The Performance Cost Ratio (PCR) and Service Contribution (SC) are used for service selection in the federation. PCR is the ratio between QoS and cost of service. PCR is used to determine cost worthiness of any cloud service. SC is the ratio between amount of resources that are consumed and amount of resources that are provisioned in federation. The SC is used to ensure continuous service exchange.

If requesting cloud provider is not a free rider, agent calculates QoS, PCR and SC values in order to select a suitable cloud provider for service sharing. A discrete inconsistency of service delivery in federation over a period of time is known as service inconsistency problem. Hence, it is necessary to determine whether selected cloud provider has any service inconsistency [27].

To solve service inconsistency issue, weighted sum method is used. A weighted sum is a total in which a relative weight based on its importance is assigned to each attribute. For example, assume 0.11, 0.09, 0.08, and 0.12 are QoS values and 0.4, 0.3, 0.2, and 0.1 are corresponding weights. Then, weighted sum is calculated as $0.11*0.4 + 0.09*0.3 + 0.08*0.2 + 0.12*0.1$, which is equal to 0.099. The FSMA calculates cloud providers' QoS, PCR, and SC values using weighted sum in order to ensure consistent services over past years. QoS value of cloud provider is calculated using Eq. (1):

$$\text{CPl[i].QoS} = \sum_{j=1}^{m} wj * CPl[i].QoSj \tag{1}$$

CPl[i] is the list of selected cloud providers. i = 1, 2... n and j = 1, 2... m, where i and j represent the cloud provider number and the year, respectively. w is the relative weight that is assigned for past years.

QoSj is past years' Quality of Service.

$$\text{CPl[i].PCR} = \sum_{j=1}^{m} wj * CPl[i].PCRj \tag{2}$$

PCRj is past years' Performance Cost Ratio.

$$\text{CPl[i].SC} = \sum_{j=1}^{m} wj * CPl[i].SCj \tag{3}$$

SCj is past years' Service Contribution.

The cloud providers are divided into N regular classes (CPreg-1, CPreg-2 ... CPreg-N) based on their QoS. Dividing cloud providers into N regular class efficiently performs service matching. The agent chooses cloud provider class (CPreg-i) that is closer to requested QoS and assigns it to CPl. It executes Fair Service Matching (FSM) algorithm to find a suitable cloud provider from list (CPl).

3.4. Fair service matching (FSM)

The FSM algorithm takes requesting cloud provider CPreq and CPl as inputs and selects a suitable cloud provider (CP) from class as follows;

If CPreq is a free rider, then agent compares CPreq's service type (CPreq.service_type), service required (CPreq.required), and duration (CPreq.duration) with all cloud providers' service types (NC[i].service_type), services available (NC[i].available), and durations (NC[i].duration) in NC. Then, cloud providers who meet the request are moved to Cloud Provider result class (CPr).

Otherwise, agent compares CPreq's quality of service (CPreq.QoS), service type (CPreq.service_type), service required (CPreq.required), and duration (CPreq.duration) with all cloud providers' quality of service (CPl[i].QoS), service types (CPl[i].service_type), services available (CPl[i].available), and durations (CPl[i].duration) in regular class CPl. Cloud providers who meet the request are moved to CPr.

If CPr is empty, agent informs cloud provider regarding no availability of cloud provider. CPreq has two options: (a) wait and make same service request again after some time, (b) alter service request and make service request again. If CPr contains only one cloud provider, provider is labeled as selected cloud provider (CPs) and it proceeds to price negotiation process.

A cloud provider joins federation to increase its resource utilization and revenues. QoS-based cloud provider selection selects same cloud provider repeatedly to match with subsequent service requests [11]. This selection makes other providers in federation wait for a long time to offer services. This selection causes starvation problem. FSMA algorithm sets a counter for all cloud providers and avoids starvation problem as follows;

If CPr has more than one cloud provider, CPr is sorted in descending order based on CPr[i].pcr. PCR gives preference to best performer. Now, best performing cloud provider is selected from CPr and labeled as CPs. The number of times that a resource is provisioned by cloud provider over a fixed period of time is stored in a counter variable (CPl[i].count). All cloud providers' counts are zero at beginning and are updated whenever cloud provider provides service to others. CPs count value is compared with count of next best cloud provider in CPr. CPs does not change if count of CPs is less than or equal to next best cloud provider in CPr. Otherwise, next best cloud provider in CPr is selected as CPs, and process repeats. Finally, a CPs with a low count is selected, and it proceeds to price negotiation. Selection of service offering cloud provider based on count ensures fairness among cloud providers.

Price negotiation is a process through which both cloud customer and cloud provider resolve and determine price. All service providers mention adjustment value (changes in quoted costs) during service advertisement. Similarly, requesting service provider provides an adjustment value while making service request. Adjustment value is used either to increase CPreq.cost or decrease CPs.cost during price negotiation. If CPs.cost is greater than CPreq.cost, cost difference is calculated (CPs.cost - CPreq.cost). If cost difference is less than or equal to CPs.adjustment or CPreq.adjustment, service matching is treated as successful, and CPs.count is updated. The resource control transfer between CPs and CPreq happens after a successful price negotiation. Otherwise, service matching is treated as unsuccessful, CPs is removed from CPr and service matching continues. If CPs.cost is less than or equal to CPreq.cost, service matching is treated as successful, and CPs.count is updated. The resource control transfer between CPs and CPreq happens after successful service matching (Algorithm 1).

Algorithm 1 Fair service matching (FSM).

```
Input: new class (NC[1...k]), regular class (CPl[1...n]), requesting cloud provider (CPreq)
Output: a cloud provider (CPs)
  i = 1, CPr = ϕ, m = 0, CPs = ϕ;
  if (CPreq.type == 'F')
    while i <= k do
      if (CPreq.service_type == NC[i].service_type && CPreq.required <= NC[i].available
          && CPreq.duration <= NC[i].duration)
        Add NC[i] to CPr[++m];
      endif;
    endwhile;
  else
    while i <= n do
      if (CPreq.QoSreq <= CPl[i].qos && CPreq.service_type == CPl[i].service_type &&
          CPreq.required <= CPl[i].available && CPreq.duration <= CPl[i].duration)
        Add CPl[i] to CPr[++m];
      endif;
    endwhile;
  endif;
  cps_selection:
  if (m == 0)
    return CPs;
  else if (m == 1)
    CPs = CPr[1];
  else
    sort CPr[1...m] in descending order based on pcr;
    CPs = CPr[1];
    i = 2;
    while i <= m do
      if (CPs.count <= CPr[i].count)
      i++;
      else
        CPs = CPr[i];
        i++;
      endif;
    endwhile;
  endif;
  if (CPs.cost> CPreq.cost) then
    if (CPs.adjust >= CPs.cost-CPreq.cost || CPreq.adjust >= CPs.cost-CPreq.cost) then
      CPs.count++;
      resource delegation;
      return CPs;
    else
      remove CPs from the CPr;
      m−;
      CPs = ϕ;
      goto cps_selection;
    endif;
  endif;
```

Table 1
Cloud providers QoS, PCR and SC details.

Cloud provider/year	CY			CYM1			CYM2			CYM3			Weighted sum		
	QoS	PCR	SC	QoS	PCR	SC	QoS	PCR	SC	QoS	PCR	SC	QoS	PCR	SC
CP1	79.95	416.4	126.7	77.65	404.4	120.0	81.91	426.6	125.0	77.33	402.8	88.9	79.4	413.5	120.6
CP2	57.60	303.2	75.0	59.63	313.8	75.0	59.79	314.7	57.1	58.60	308.4	100.0	58.7	309.2	73.9
CP3	57.80	301.0	102.6	56.63	294.9	102.6	59.47	309.7	102.6	59.99	312.4	109.8	58.0	302.1	103.3
CP4	66.92	346.8	100.0	70.34	364.5	105.0	69.95	362.4	95.2	64.37	333.5	89.4	68.3	353.9	99.5
CP5	86.86	452.4	90.0	88.46	460.7	110.0	84.44	439.8	112.5	85.91	447.4	87.5	86.8	451.9	100.3
CP6	82.24	430.6	80.0	84.41	441.9	80.0	80.31	420.5	80.0	81.11	424.7	100.0	82.4	431.4	82.0
CP7	62.42	326.8	108.9	64.77	339.1	108.9	63.76	333.8	102.1	63.45	332.2	102.1	63.5	332.4	106.8
CP8	72.44	375.3	125.0	76.38	395.8	95.2	73.33	379.9	125.0	75.00	388.6	111.1	74.1	383.7	114.7
CP9	62.53	320.7	121.4	63.93	327.8	106.3	65.55	336.2	113.3	64.21	329.3	123.1	63.7	326.8	115.4
CP10	81.92	435.7	80.0	80.55	428.5	120.0	82.11	436.8	80.0	81.99	436.1	100.0	81.6	433.8	94.0

CY – Current year, CYM1 - Current year minus 1, CYM2 - Current year minus 2, and CYM3 - Current year minus 3.

Table 2
Cloud Providers list (CPl).

Cloud provider/ parameters	No. of VMs	Available durations (in days)	QoS value (in percentage)	PCR	Cost per VM (in $/per hour)	Cost adjustment factor (in $)	Count
CP1	10	60	79.40	413.50	0.192	0.019	3
CP5	8	45	86.80	451.90	0.192	0.029	4
CP6	7	45	82.40	431.40	0.191	0.019	2
CP8	4	45	74.10	383.70	0.193	0.023	5
CP10	12	30	81.60	433.80	0.188	0.019	0

Table 3
Cloud provider result list (CPr).

Cloud provider/ parameters	No. of VMs	Available durations (in days)	QoS value (in percentage)	PCR	Cost per VM (in $/per hour)	Cost adjustment factor (in $)	Count
CP5	8	45	86.80	451.90	0.192	0.029	4
CP6	7	45	82.40	431.40	0.191	0.019	2
CP1	10	60	79.40	413.50	0.192	0.019	3

3.5. Working example

Let us assume that CP1, CP2… and CP15 are registered cloud providers that advertise their service offering details with federation. Agent divided cloud providers into three classes (one new class and two regular classes based on their QoS). The regular class and new class contain cloud providers ⟨CP1, CP2 … CP10⟩ ⟨CP11, CP12 … CP15⟩, respectively. The agent receives service request from CPreq as ⟨VM, 3 nos, 45 days, 75%, $0.183, $0.00915⟩ and then, agent found that CPreq was not a free rider.

The federation maintains service provider's transaction history, which includes its QoS, PCR, and SC. The service provider's QoS is calculated based on its transaction history using the Analytical Hierarchical Processing (AHP) method that has been proposed in [28]. The agent calculates QoS, PCR, and SC values for cloud providers in regular class by using Eqs. (1), (2), and (3), respectively. The previous four years QoS, PCR, and SC values are shown in the Table 1. Additionally, Table 1 shows calculated weighted sum of QoS, PCR, and SC. The agent divided regular class cloud providers in Table 1 into regular class1 and regular class2 based on their weighted sum QoS, as stated below.

regular class1 < CP1, CP5, CP6, CP8, CP10 > and regular class2 < CP2, CP3, CP4, CP7, CP9 >

Regular class1 contains cloud providers with a QoS of 70% or above, and regular class2 contains the cloud providers with a QoS less than 70%. Agent chose regular class1, which was near CPreq.QoS value and assign it to CPl. Details of cloud provider in CPl are given in the Table 2.

The agent executed FSM algorithm and selected cloud providers CP1, CP5, and CP6 from CPl since they met service type, number of services required, duration, and QoS given by CPreq. CP8 and CP10 were eliminated from CPl because CP8's QoS was less than CPreq's QoS and CP10′s duration was less than CPreq's duration. Subsequently, agent added the selected CP1, CP5, and CP6 to CPr. Finally, the agent sorted them based on their PCRs since CPr had more than one cloud provider. The details of cloud in CPr are shown in the Table 3.

At first, cloud provider CP5 was selected as CPs, and count of CPs was compared with count of the next best cloud provider (CP6) in CPr. Since count of CPs was greater than CP6, CP6 was selected as CPs. This comparison continued, and CP6 was selected for price negotiation. The maximum service cost of CPreq was less than that of CP6. Then, the agent used price adjustment factor of CP6 and calculated the adjusted service price of CP6. The calculated adjusted price of CP6

was less than CPreq's maximum service cost. Hence, the price negotiation was successful. The resource control of CP6 was transferred to CPreq, and count of cloud provider increased by one.

4. System evaluations

The effectiveness of FSMA was evaluated by matching underutilized services among many cloud providers. Exchanging services in a continuously changing environment has many risks. The uncertainty in service demand, service underutilization, service unavailability, free riders, and new cloud providers are means of risk. These risks hinder service exchange. For example, creating a service transaction record to calculate QoS for new cloud provider is a considerable challenge as new cloud provider does not have a service history transaction record when joining federation. The comparison of FSMA against an existing system is a difficult task, since no such system existed in past. Therefore, FSMA system is compared with the existing federated cloud system's common properties. The properties of existing federated cloud systems are cloud provider selection and allocation. The federated cloud system supports monetary-based service exchange, which is contradictory to FSMA. Monetary based service exchange does not require a ranking, service contribution history, or service negotiation since service costs are fixed. The only way to compare FSMA with federated cloud system is through cloud provider selection and allocation.

4.1. Experimental setup

Many datasets were prepared randomly by considering real market scenarios so as to assess the performance of FSMA. The dataset consists of cloud providers that are involved in process and its requirements as shown in the Tables 4 and 5. The whole dataset has two parts. The first part contains service details, such as service type, number of available services, cost of service, and duration. The second part consists of request details, such as required service, number of required services, QoS, maximum cost, and duration. The cloud providers were divided into four classes. The classes contain 15, 30, 45, and 60 cloud providers, respectively. The purpose of dividing dataset into classes is to evaluate the performance of FSMA under various conditions. Initially, the case studies were executed using FSMA in a cloud simulator and performances were recorded. The same case studies were executed using Cloud Resource Bartering System (CRBS) and results were recorded. The comparison of the FSMA with the CRBS system is discussed below.

4.2. Efficiency of handling a new cloud provider

The existence of a new cloud provider that does not have a service history in federation creates starvation problem, as discussed in the Section 3.1. A service offering cloud provider is selected based on their QoS in existing federated cloud systems. The existing systems assign a low QoS, i.e., zero, for new cloud providers when they join federation and select a new cloud provider until a regular cloud provider does not have services to share. However, new service providers that benefit are significantly less in existing systems. The FSMA effectively solves this problem by using fair service matching algorithm, which is discussed in Section 3.4.

A dataset with a few free riders that are randomly included was used in order to evaluate effectiveness of FSMA system. The evaluation aimed to find the improvement of FSMA over existing systems (CRBS) while solving free rider and new cloud provider problems. The evaluation results showed the new cloud providers received a rare chance (shown in the Fig. 2), and free riders were blocked until they cleared their previous debts in CRBS system. Besides, the number of successful transactions decreased, since free riders were blocked. FSMA increased the number of new cloud providers that benefitted and free riders were allowed to avail services from federation. Therefore, a number of successful transactions increased as expected, which increased transaction history for new cloud providers.

The experiment was conducted for 10, 20, 30, 40, and 50 service requests. CRBS selected 0, 0, 1, 1, and 2 new cloud providers, while FSMA selected 1, 3, 6, 8, and 11 new cloud providers, respectively, for above mentioned service requests.

Table 4
Sample dataset1.

	Experiment1	Experiment2	Experiment3	Experiment4
Number of cloud providers	15	15	30	30
Number of service requests	20	40	30	60
Number of services available	25	25	50	50

Table 5
Sample dataset2.

	Experiment1	Experiment2	Experiment3	Experiment4	Experiment5
Number of service requests	10	20	30	40	50
Number of cloud providers	5	10	15	20	25
Number of services available	15	25	30	35	40

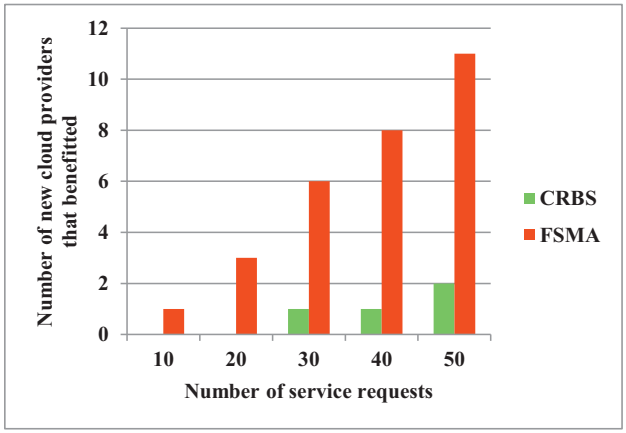

Fig. 2. Comparison of a number of the new cloud providers that benefitted.

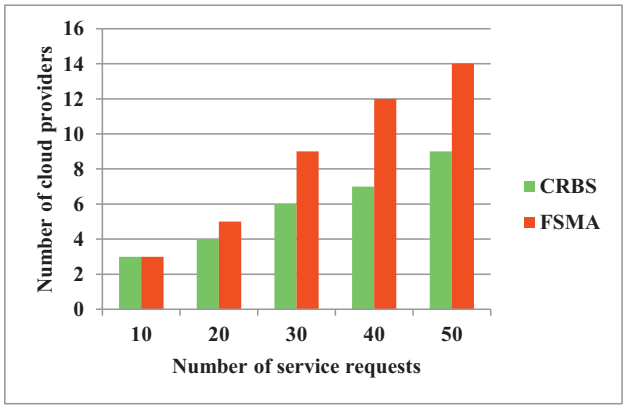

Fig. 3. Comparison of the service request distribution.

On average, FSMA provided 17% of service requests to new cloud providers, while CRBS provided 3% of service requests to new cloud providers.

4.3. Fair service matching

The progress of federation depends on excellent service matching. An excellent service matching fairly distributes service requests to all cloud providers in federation. In order to equitably distribute service request, FSMA assigns service requests considering QoS and the number of service requests that were assigned to them in past. If QoS values match two providers, then allocation is based on number of service requests that were assigned to them in the past. Compared to the FSMA, federated cloud systems assign service requests based on QoS, utility, and service contribution. Since federated cloud systems choose service provider based on QoS ranking, same service provider can be selected for many service requests until a service is available with a service provider. The service provider selection based on the QoS ranking does not distribute service requests evenly to all the service providers in federation.

A dataset was taken in order to carefully evaluate fairness in distributing service requests. The number of service providers that advertised services in system was fixed as 15. A number of service requests that were submitted to both CRBS and FSMA systems were 10, 20, 30, 40, and 50. FSMA distributed service requests fairly to all cloud providers. Besides, the number of cloud providers that were involved in transactions was compared to that of CRBS system (shown in the Fig. 3).

CRBS system failed to distribute service requests fairly to all service providers. In addition to that, number of successful transactions was less and number of negotiations that were necessary for successful transactions was more (shown in the Figs. 4 and 5). In the Fig. 4, number of service requests that were submitted was 50. Among the 50 service requests, CRBS system matched 32 service requests successfully, and FSMA system matched 44 service requests successfully. Compared to CRBS system, FSMA increased the number of successful transactions and required fewer negotiations.

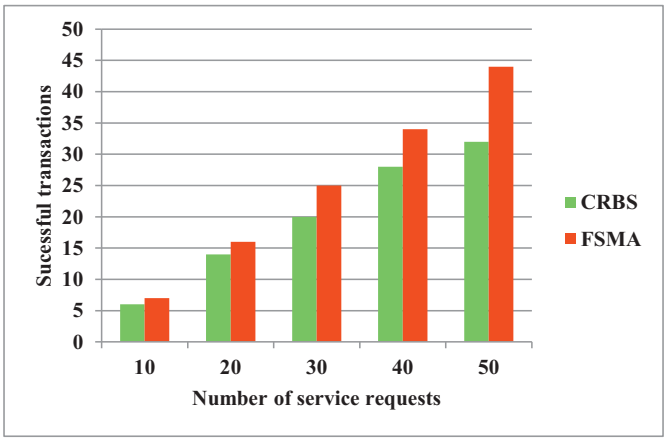

Fig. 4. Comparison of successful transactions.

Fig. 5. Comparison of price negotiations.

4.4. Service consumption vs service contribution

Stabilizing service consumption and contribution is essential for federations. This step ensures service availability and balances services in federations. The nonexistence of a stabilizing method can lead a federation to a resource scarce state [29]. CRBS system does not permit cloud provider to offer or consume services until clearing their previous debt or credit. Cloud providers should consume or offer same indebted or noncredit services. This method prevents cloud provider from possibly offering or consuming services. In FSMA, in order to clear their previous debt, cloud provider does not need to consume or offer the same indebted or noncredit service. Instead, cloud provider can consume or offer same services or different services that are equal to the value of services they already consumed or offered. FSMA fixes a maximum service contribution threshold value and cloud providers are permitted to offer or consume services until reaching maximum value. FSMA method ensures resource availability and stabilizes service consumption and contributions in federation.

4.5. Service matching efficiency

The number of successful transactions determines service matching ability of system. Several experiments have been conducted to evaluate service matching ability of FSMA. Several experiments were designed to determine service matching rate of FSMA in the different working environments. Initially, the system was set up with 20 cloud providers and 10 service requests so as to evaluate the ability of system. Then, number of service requests was continuously increased in order to evaluate the performance of system under varying conditions.

Evaluation results showed that performance of FSMA was same under different conditions. The same experiments were conducted on CRBS system, and results showed that service matching ability of system varied under different workloads. The comparison of FSMA system with CRBS system is shown in Fig. 6.

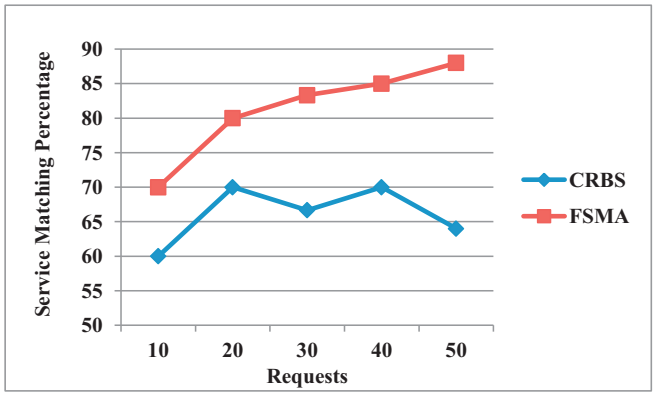

Fig. 6. Comparison of service matching efficiency.

5. Conclusions and future work

The existing agent-based service selection method, such as cloud resource bartering system, has failed to address problems of service request distribution and starvation. Fair service matching agent selects cloud providers by considering quality of service, a fair service request distribution, and a consistent service history. This agent gives a chance to free riders to consume services and clear all debt before making their next service request, which increases number of successful transactions and reduces number of price negotiations for each service request. Additionally, fair service matching agent improves number of cloud providers that benefited from federation and maintains availability of a fair number of services for genuine cloud providers. Maintaining a fair number of services at federation solves service scarcity problem. A further study is made focusing on dynamically balancing service costs based on services that are available in federation. This adjustment may encourage cloud providers to contribute more services and avoid free riding problems.

References

[1] Falcão E, de L, Brasileiro F, Brito A, Vivas JL. Enhancing fairness in P2P cloud federations. Comput Electr Eng 2016;56:884–97. doi:10.1016/J.COMPELECENG.2016.05.007.
[2] Xu X, Yu H. A game theory approach to fair and efficient resource allocation in cloud computing. Math Probl Eng 2014;2014. doi:10.1155/2014/915878.
[3] Liu H, He B. Reciprocal resource fairness: towards cooperative multiple-resource fair sharing in IaaS clouds. In: Int Conf High Perform Comput Networking, Storage Anal SC; 2014. p. 970–81. 2015–Janua:. doi:10.1109/SC.2014.84.
[4] Sim KM. Agent-based cloud computing. IEEE Trans Serv Comput 2012;5:564–77. doi:10.1109/TSC.2011.52.
[5] Sim KM. Agent-based approaches for intelligent intercloud resource allocation. IEEE Trans Cloud Comput 2018:1 PP:. doi:10.1109/TCC.2016.2628375.
[6] Temglit N, Chibani A, Djouani K, Nacer MA. A distributed agent-based approach for optimal QoS selection in web of object choreography. IEEE Syst J 2018;12:1655–66. doi:10.1109/JSYST.2016.2647281.
[7] Kang J, Sim KM. Ontology-enhanced agent-based cloud service discovery. Int J Cloud Comput 2016;5:144. doi:10.1504/IJCC.2016.075125.
[8] Cakmaz Y.E., Alaca O.F., Durmaz C., Akdal B., Tezel B., Challenger M., Kardas G. Engineering a BDI agent-based semantic e-barter system. 2nd Int Conf Comput Sci Eng UBMK 2017 2017:1072–7. doi:10.1109/UBMK.2017.8093474.
[9] Demirkol S, Getir S, Challenger M, Kardas G. Development of an agent based e-barter system. In: 2011 Int. Symp. Innov. Intell. Syst. Appl. IEEE; 2011. p. 193–8. doi:10.1109/INISTA.2011.5946060.
[10] Bonald T, Comte C. Balanced fair resource sharing in computer clusters. Perform Eval 2017;116:70–83. doi:10.1016/j.peva.2017.08.006.
[11] Ezenwoke A, Daramola O, Adigun M. QoS-based ranking and selection of SaaS applications using heterogeneous similarity metrics. J Cloud Comput 2018:7. doi:10.1186/s13677-018-0117-4.
[12] ZarAfshan Goher S, Bloodsworth P, Ur Rasool R, McClatchey R. Cloud provider capacity augmentation through automated resource bartering. Futur Gener Comput Syst 2018;81:203–18. doi:10.1016/j.future.2017.09.080.
[13] Gutierrez-Garcia JO, Sim KM. Agent-based cloud service composition. Appl Intell 2013;38:436–64. doi:10.1007/s10489-012-0380-x.
[14] Nikbazm R, Ahmadi M. Agent-based resource discovery in cloud computing using bloom filters. In: Proc 4th Int Conf Comput Knowl Eng ICCKE 2014; 2014. p. 352–7. doi:10.1109/ICCKE.2014.6993399.
[15] Erdil DC. Autonomic cloud resource sharing for intercloud federations. Futur Gener Comput Syst 2013;29:1700–8. doi:10.1016/j.future.2012.03.025.
[16] Şensoy M, Yolum P. Ontology-based service representation and selection. IEEE Trans Knowl Data Eng 2007;19:1102–15. doi:10.1109/TKDE.2007.1045.
[17] Zhang M, Liu L, Liu S. Genetic algorithm based QoS-aware service composition in multi-cloud. In: Proc - 2015 IEEE Conf Collab Internet Comput CIC 2015; 2016. p. 113–18. doi:10.1109/CIC.2015.23.
[18] Mezni H. A multi-recommenders system for service provisioning in multi-cloud environment. In: Proc - Int Work Database Expert Syst Appl DEXA; 2017. p. 142–6. 2017–Augus:. doi:10.1109/DEXA.2017.45.
[19] Shirvani MH. Web service composition in multi-cloud environment : a bi-objective genetic optimization algorithm. 2018 Innov Intell Syst Appl 2018:1–6.
[20] Wu L, Kumar Garg S, Versteeg S, Buyya R. SLA-based resource provisioning for hosted software as a service applications in cloud computing environments. IEEE Trans Serv Comput 2014;7:465–85. doi:10.1109/TSC.2013.49.
[21] Taha A, Manzoor S, Suri N. SLA-based service selection for multi-cloud environments. In: Proc - 2017 IEEE 1st Int Conf Edge Comput EDGE 2017; 2017. p. 65–72. doi:10.1109/IEEE.EDGE.2017.17.
[22] Sotiriadis S, Bessis N, Kuonen P. Advancing inter-cloud resource discovery based on past service experiences of transient resource clustering. In: Proc - 3rd Int Conf Emerg Intell Data Web Technol EIDWT 2012; 2012. p. 38–45. doi:10.1109/EIDWT.2012.16.
[23] Zhao L, Du M, Chen L. A new multi-resource allocation mechanism: a tradeoff between fairness and efficiency in cloud computing. China Commun 2018;15:57–77. doi:10.1109/CC.2018.8331991.

[24] Tang S, Niu Z, He B, Lee BS, Yu C. Long-term multi-resource fairness for pay-as-you use computing systems. IEEE Trans Parallel Distrib Syst 2018;29:1147–60. doi:10.1109/TPDS.2017.2788880.
[25] Wang X, Cao J, Wang J. A dynamic cloud service selection strategy using adaptive learning agents. Int J High Perform Comput Netw J High Perform Comput Netw 2016;9:70–81. doi:10.1504/IJHPCN.2016.074660.
[26] Gao C, Ma J. A k-median facility location agent for low-cost service selection in digital community network. China Commun 2014;11:174–86. doi:10.1109/CC.2014.7004535.
[27] O'Dywer R, Neville SW. Assessing QoS consistency in cloud-based software-as-a-service deployments. In: 2017 IEEE Pacific Rim Conf Commun Comput Signal Process PACRIM 2017 - Proc; 2017. p. 1–6. 2017–Janua. doi:10.1109/PACRIM.2017.8121889.
[28] Garg SK, Versteeg S, Buyya R. A framework for ranking of cloud computing services. Futur Gener Comput Syst 2013;29:1012–23. doi:10.1016/j.future.2012.06.006.
[29] Kaur M, Kaur K, Kapoor L. An efficient resource provisioning technique in inter-cloud using peer-to-peer approach. ACM 2017;10(2):529–39. doi:10.3844/ajeassp.2017.529.539.

S. Sudhakar received his M.E. in Computer Science and Engineering from Anna University, Chennai, India. He is currently working in the Department of Computer Science and Engineering, Karunya Institute of Technology and Sciences, Coimbatore, India. His research interests include cloud computing and big data analytics.

N.S. Nithya is an Associate Professor in the Department of Computer Science and Engineering, K.S.R College of Engineering, Namakkal, India. She received her Ph.D. in Information and Communication Engineering from Anna University, Chennai, India. She has published more than 10 research articles in national and international journals. Her research interests include cloud computing and big data analytics.

B.L. Radhakrishnan received his M.Tech. in Computer Science and Engineering from Dr. M.G.R. University, Chennai, India. He is currently working in the Department of Computer Science and Engineering, Karunya Institute of Technology and Sciences, Coimbatore, India. His research interests include cloud computing, Blockchain, and healthcare.