ISSN 2278-3091
Volume 9 No.2, March -April 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse108922020.pdf
https://doi.org/10.30534/ijatcse/2020/108922020

Residue Based Adaptive Resource Provisioning through Multi-Criteria Decision and Horizontal Scaling of VM's in Agent-Based Model for Federated Cloud

Pradeep Kumar Vadla[1], Kolla Bhanu Prakash[2]
[1]Research Scholar, Department of CSE, Koneru Lakshmaiah Education Foundation, Vadeswaram, A.P, India
[2]Professor, Department of CSE, Koneru Lakshmaiah Education Foundation, Vadeswaram, A.P, India

ABSTRACT

Elasticity and scalability are prominent issues in cloud computing which are resolved effectively using federated clouds. The agent-based model is simulated in our work in which all the elements of cloud computing are categorized into specific agents like cloud consumer agent, cloud provider agent and cloud broker agent. The collaborated cloud providers who are contributing resources are treated as collated cloud provider agents. The residue-based resource provisioning is carried at cloud broker by performing multi-criteria decision for finding dominant collated provider agent in providing resources within the limit of service level agreement and horizontal scaling of the virtual machine is done based on greedy cloud ranker algorithm to rank the cloud which contributes the virtual VM which satisfy the consumer agent request within specific turnaround time without violating service level agreement. The features of the interoperable cloud are simulated using python classes and realization of horizontal scaling is tested for computing percentage of request satisfaction with full or partial and transaction rate completion of allocating virtual machine to cloud consumer agent.

Key words: Agent-based Model, Cloud Elasticity, Horizontal Scaling, Resource Provisioning Multi-Criteria decision, Greedy cloud ranker.

1. INTRODUCTION

Cloud Computing plays a vital role in resource provisioning for cloud consumer agents to meet their demands [1] at any instance. Elasticity and Scalability are the two prominent features of cloud computing that occur during the allocation of resources. The basic difference between them is elasticity attempts to balance the resources available at any given point in time with the actual amount of resources required and scalability manages the evolving needs of an application within infrastructure confines by dynamically adding or withdrawing resources to meet the demands of applications if necessary. Scalability is addressed in most situations by scaling up (vertical scaling) and/or scaling-out (horizontal scaling). Furthermore, when it comes to sizing, scalability can be more granular and focused in nature than elasticity.

An agent-based model of cloud federation will solve the elasticity and scalability issues of managing resources in the form of VM's (Virtual Machines). The agent-based model considers providers, users and brokers as CPA (Cloud Provider Agent), CCA (Cloud Consumer Agent), and CBA (Cloud Broker agent). The CBA does the critical task of managing of aggregation of CPA services and provides as CCPA (Cloud Collated Provider Agent). The coalition formation of CCPA depends on the CCA request of types of VMs. The coalition formation of CCPA uses FLA (Federated Level Agreement) and SLA (Service Level Agreement) with specific KPI (Key Performance Indicators) and compute SLA values for satisfying request. The CBA manage the task of the resource provisioning using residue-based VM Provisioning technique. The multi-decision criteria are used to select the VM's of CCPA based on their computed SLA parameter values then horizontal scaling of resources among CCPA providers is done by selecting the appropriate virtual VM using residue-based VM provisioning algorithm. The transaction success rate and turnaround time of CCA request are computed by the creation of virtual VM using residue-based VM provisioning algorithm by CBA. The computations of residue-based VM provisioning is compared to Equi-based VM provisioning algorithm in terms of transaction success rate, turnaround time and successful percentage of fully/partial requests. The popularity of residue-based VM provisioning algorithm lies with the Greedy ranker algorithm for ranking the CCPA agents at a particular instance for easy provisioning of VM.

The remaining of the paper is organized as follows: Section 2 discusses the related work for listing the latest provisioning techniques applied in the federated cloud for inter-cloud operations. Section 3 gives an overview of the agent-based model of the federated cloud for specifying the importance of

the roles of agents involved in satisfying consumer's request. Section 4 gives a complete picture of the proposed residue-based VM provisioning technique. Section 5 highlights the approach of multi-criteria decision and ranking algorithm importance for residue-based VM provisioning technique. Section 6 makes the results and discussion and Section 7 provides the conclusion and the future work**.**

2. RELATED WORK

The cloud services are dynamically made available to its consumers in the form of VMs without violating any SLA and overloading any individual cloud provider servers as they have few resources to server the consumers [2]. The other related work of [3,4] discusses the allocation of VMs within a cloud in a load-balanced manner, thereby attempting to allow efficient use of its resources spread through various servers and datacenters. RESERVOIR [5]-[7] is an Inter-cloud system, not requiring resource provisioning brokers, where all participating cloud resources are divided to accommodate different components of software applications as not dependent of each other for processing.-InterCloud [8],[9] uses a central broker for resource provisioning for all cloud consumers by collaborating resources from all cloud providers and distribute resources at any instance dynamically. In comparison, the same inter-cloud [10]–[12] maintains global databases for information on the availability of VMs across all clouds. The information is replicated at many providers thus broker will negotiate with cloud consumers and distribute resources dynamically.

Federated Cloud Management (FCM)[13]-[15] also creates an inter-cloud environment for creating collaboration among cloud providers with a software feature managing resource provisioning and other critical administrative tasks like maintenance of availability of servers, SLA creation, computing resource pricing etc.. Besides, a broker acts as a mediator between the cloud and user software elements, and thus assists in the allocation of resources. Adaptive resource management [16],[34] approach to helping make decisions dependent on the workflow's execution time. These models reschedule resources to boost performance, based on the usage history. A dynamic adaptive resource provisioning approach [17] uses autonomic computation, hybridization and reinforcement learning. It suggested solution tackles the unforeseen situations, such as job congestion, over availability and under-provisioning. In [18] they proposed a self-organizing method focused on multiple agent schemes. To achieve the features expected by locally communicating agents in the cloud sector, they propose triple-layered self-adaptive multiple agent structures to enable concurrent negotiating operations in cloud commerce. Their method of running a market interface uses an algorithm as a bargaining procedure.

In [19] they aimed at creating a fully automated program in which the consumer agent merely wants to meet his specifications then loads his request of VM's to broker agent based on the communication of messages between the master broker-agent and cloud provider agents. The broker agent who is entrusted with the control of the capital must obey a set of guidelines. In [20] they analyzed multiple strategies for combining a dynamic information management system and effective integration frameworks for the on-demand request generation for independent hybrid cloud maintenance. In this paper, autonomously controlled federated cloud infrastructure architecture was built which focus on migration behaviour with the potential effect on cloud federations. The Knowledge Management program recommends corrective measures to reduce energy usage to avoid breaches of service level agreements (SLA) for efficient usage of resources. Cloud network maintenance is conducted in an automated fashion to manage SLA violation while the allocation of resources.

In [21] an evolutionary approach is proposed to solve the resource provisioning issue in federated cloud providers with the agent replication process. The multi-agent systems are enabled to use their adaptive intelligent behaviour by using multiple leaning schemes. In [22] they discussed an agent-based grid computing resource allocation (ARAM) model. There are three types of entities used: User-agent entity, they are dynamic and responsible for performing the tasks at relevant grid nodes. Broker agents (BAs) entity implement a negotiation model and are responsible for managing services. Resource management agents (RMAs) are static, responsible for reporting resource status to local cluster servers. By using mobile operators, ARAM targeted at versatile, knowledgeable and adaptable resources as opposed to conventional resource management strategies. IN [23] a distributed resource management approach is used to solve resource negotiation and allocation in the inter-cloud environment. In these pricing strategies are used by cloud providers while collaborating their resources to other cloud providers. The cloud provider whenever they request for resources with low price for other's they check which providers can outsource and provide resources if his request is rejected he reduces his leasing rates and make resources available to other providers.

Federated-Cloud-Model in [24],[37] uses a mechanism of merging two or three cloud providers to form one collaborated cloud environment where they share resources at an agreed price. The providers who are having adequate resources will share the majority of resources with reasonable price and meet the demand of cloud consumers. In this work, the authors demonstrated through simulations by framing a framework among different cloud vendors for implementing the negotiation mechanism of the agreed price of sharing resources. In [25], a multi-criteria decision-making approach is used by cloud brokers in routing the cloud user request to

provide resource automatically on specific conditions for the agreed price. The choice of cloud providers is compared by cloud broker among others by framing certain criteria features on service level agreement and make available resources to cloud customers. In [26] they presented a commitment to maximizing the exchange of cloud services**.** They suggested an approach based on the Marko chain model [33] which would allow dynamic information sharing of resources in a distributed environment. In [27] they proposed an autonomous multi-objective framework for SLA management that involves an efficient approach by evaluating patterns in the workload. That decides agreement according to the state of the cloud network.

In [28] the authors proposed a middleware in cloud federation which uses a specific criterion that manages heterogeneous cloud networks wherein any cloud provider can join or leave on collaboration services and discussed specific issues which result in managing the inter-cloud environment [36] for providing an efficient solution in maintaining trust among cloud providers in satisfying cloud customers. In [29] autonomous management of cloud services is achieved based on the context of cloud consumers request satisfaction. The self-organization approach of cloud brokers is tested for different configuration settings and management of cloud provider in delivering resources at the appropriate time without violating SLA agreement among cloud providers and consumers. IN [30] a multi-agent architecture was proposed for grid distribution environment. In which hierarchy of operations are performed for allocating resources with the virtual association by cloud broker and deal with the self-organization [35] of interconnected software components are used by cloud providers to join or leave for sharing resources**.**

3. BACKGROUND

3.1 Agent-Based Model in Federated Cloud

Cloud computing, uses a market paradigm in which enterprises make big profits by minimizing investments in infrastructure**.** Cloud consumers typically pay for the usage (i.e. computing power, data storage, and inter-network connectivity, applications, and other client operational services) that is measured by the number of instance-hours consumed**.** Apart from providing a wide variety of infrastructure, the cloud consortium provides offerings from different aggregated vendors giving clients the ability to select the right cloud service vendors**.**

The dynamic nature of open federated cloud can be realized using our agent-based model approach. The agent-based model provides the flexibility of adding cloud provider agents dynamically and also remove them if they are not contributing to collation formation. The agent-based model considers four type of agents Cloud Consumer Agents (CCA_i where iϵ[1,n]),Cloud Broker Agent(CBA_i where iϵ[1,n]),Cloud Collated Provider Agent($CCPA_i$ where iϵ[1,n]), and Cloud Provider Agent (CPA_i where iϵ[1,n]).In the agent-based model as CCA send the request of type (Svm, Mvm, Lvm, XLvm) where Svm-Small Virtual Machines, Mvm-Medium Virtual Machines, Lvm-Large Virtual Machines and XL-Extra Large Virtual Machines. Table 1 gives the VM configurations which are referred from AWS Cloud. Along with the request, a specified SLA value related to availability, response time and process time is also sent to CBA.CBA will further forward the request to CPA and Seed CPA will start computing its KPI parameters with other CPA and form CCPA using FLA-SLA aware Collation formation approach [31] by verifying its computed SLA with CCA SLA and try to initiate collation formation in terms of contributing resources to the CCA request. Figure 1 shows the architecture model for Residue Based Adaptive Resource Provisioning in Agent-Based Federated Cloud. The Cloud Broker Agent does the critical tasks of managing resource provisioning using Residue-based adaptive VM provisioning algorithm which uses the multi-decision criteria approach and greedy ranker to rank the CCPA agents based on their provision of resources.

Table 1: Example of VM Configurations

Parameters	Small VM	Medium VM	Large VM	Extra Large VM
Number of Cores(1.6 GHz CPU)	1	2	4	8
Memory (GB)	1.7	3.75	7.5	15
Storage (TB)	22	48	98	199
Price	0.12	0.24	0.48	0.96

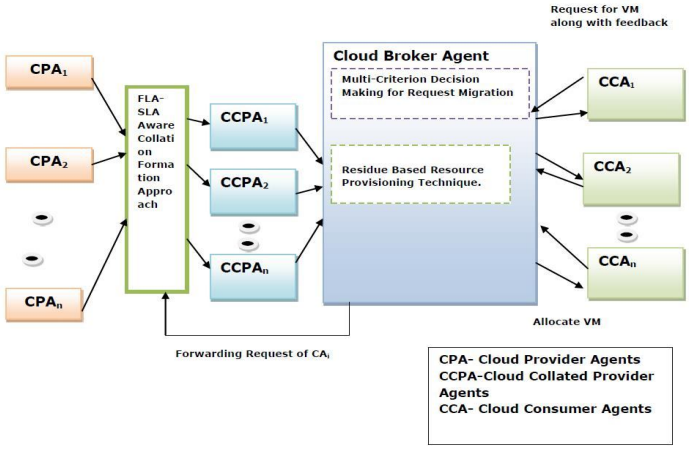

Figure 1: Architecture Model for Residue Based Adaptive Resource Provisioning in Agent-Based Federated Cloud

3.2 Virtual Machine (VM)

In agent-based model federated cloud the resources are acquired by ordered pair of four types of VM's i.e. (Svm, Mvm, Lvm, XLvm) and the Table 1 gives their configurations and each CCPA agent while forming collation they share there set of VM's with the other cloud providers for satisfying the CCA request. Each CCPA while sharing resources some part of VM's will become essential (e_i, $1 \le i \le n_e$) and another part of VM's become non-essential (ne_i, $1 \le i \le n_{ne}$).Each request of VM's made by CCA are categorized to set essential and non-essential request of VM's there denoted by set $XR_e=\{xre_i, \forall 1 \le i \le n_e\}$ and $XR_{ne}=\{xrne_i, \forall\ 1 \le i \le n_{ne}\}$ Where X is (Svm, Mvm, Lvm, XLvm).

The virtual machine satisfying the request of VM's is denoted by

$$XVM= \left\{ \bigcup_{i=1}^{n_e} xr_{e_i} \right\} \cap \left\{ \bigcup_{i=1}^{n_{ne}} xr_{ne_i} \right\} \quad , \quad xr_{e_i} \ge e_i^{\min} \quad , \quad xr_{ne_i} \ge 0 \qquad (1)$$

Where X is (Svm, Mvm, Lvm, XLvm), x is an instance of that type of VM required as essential and non-essential resources. The conditions for creating XVM resources for satisfying the CCA request are all essential XVM's should not be falling short of minimal essential $e_i^{\min}$ XVM's and non-essential XVM's may be specified to zero of not available or some specified value without any limitation.

3.3 Basic-Placer VM Provisioning

Basic-Placer VM Provisioning is a simple VM Provisioning algorithm which runs the request of essential XVM's for required no of resources on CCPA agents list and checks for the list of collaborated resources are matching with a request. If the resources are satisfying to exact request of essential XVM's then availability resources are confirmed to CCA agent. It reduces turnaround time as it searches the exact match of request and makes it available to service the CCA agent. It also ensures not only the exact match of resources the requested SLA is within the limit of computed SLA of CCPA agents and the set of XVM is returned as output. The drawback of this approach is it results in more of failures because the resources may not get satisfied with requested essential XVM's even the CCPA agents may have resources but their collaboration may result in not satisfying the request.

3.4 Equi-Placer VM Provisioning

Equi-Placer VM Provisioning is a VM Provisioning algorithm which incorporates horizontal scaling of XVM's at CCPA agents and creates virtual XVM's for satisfying the CCA request dynamically. The horizontal scaling of resources is attained by dividing the request which is not satisfied by the Basic Placer VM Provisioning. The unsatisfied requests of essential XVM are divided into sub-requests and then Basic Placer VM Provisioning is run parallel for all sub-requests. Each sub-request get satisfied by the generation of specific XVM meeting all the essential requirements of SLA as mentioned by CCA. To generated the XVM for satisfying for all sub-requests is taken care of by the algorithm attach_extra_collect and fulfil all requests with not satisfied essential XVM. The Equi-Placer VM Provisioning results in some sub-requests not satisfied i.e. it does partial satisfaction of CCA requests and provide achieving good transaction success rate.

3.5 Motivation of Residue-Based VM provisioning over Equi-Placer VM Provisioning

Many critical resources may be available in abundance in certain CCPA, to generate a new XVM. In the Equi-Placer VM Provisioning, services from these clouds are not included in any of the sub-requests, though adequate to build XVMs. It is because the critical services the CCA needs are split equally for each sub-request meeting the minimum amount of essential XVM's**.** The Residue-Based VM Provisioning proposed in this paper aims at achieving a high transaction rate than Equi-Placer VM Provisioning. Many limitations for requests satisfaction are specified like "horizontal scaling through clouds" and "absolute fulfilment" and considered them as the service level agreement violation parameters during VM provisioning at CCPA. In every CCA request, the scheme of performance of the Equi-Placer VM Provisioning works similar to Basic Placer VM Provisioning satisfying the CCA request in the agent-based model of federated cloud. Satisfying the CCA request ensures that, even if one of the above requirements is not acceptable, both will be regarded as unreasonable and the allocation of services will decline accordingly. This problem is also discussed in the Residue-Placer VM Provisioning is suggested. Here, resource provisioning efficiency is not hampered, although at the same time checking that the SLA is not violated. This leads to a higher success rate for purchases. Within this paper, a new cloud ranking algorithm along with Multi-Decision Criteria is built to further improve the resource provisioning performance.

4. RESIDUE-BASED VM PROVISIONING

The Residue-Based VM provisioning is the combination of the two methodology placers, Basic-Placer VM provisioning and Equi-Placer VM Provisioning. Similarity with Basic-Placer VM provisioning, it does an extensive search of a match of the request of XVM's among the CCPA and from Equi-Placer VM Provisioning, it does horizontal scaling of XVM's among the CCPA. Thus Residue-Based VM Provisioning has advantages of achieving low turnaround time and increased transaction success rate. Unlike horizontal scaling done by Equi-Placer VM Provisioning here, it is managed dynamically based on the division of request by considering the SLA of availability of the XVM's for

satisfying the CCA request. The Residue based VM provisioning has taken much of its features from Residue Placer Algorithm [32] which is used for computing utilities for forming a single VM.

4.1 Parameter definition

The availability resources in CCPA are represented as A_{e_i} (C) , $1 \leq i \leq n_e$ and A_{ne_i} (C), $1 \leq i \leq n_{ne}$ for essential and non-essential XVM's where X is(Svm, Mvm, Lvm, XLvm) available by each CCPA .Each CCPA will have set of essential VM's which are satisfied and unsatisfied denoted by Resat={$Xresat_i$, $1 \leq i \leq n_e$ },Rensat={$Xrnesat_i$, $1 \leq i \leq n_e$ } similarly non-essential VM's which are satisfied and unsatisfied, Rnesat={$Xrnesat_i$, $1 \leq i \leq n_{ne}$ },Rnensat={$Xrnensat_i$, $1 \leq i \leq n_{ne}$ } where X is (Svm, Mvm, Lvm, XLvm) Following inequalities are considered

$$Xrensat = Xre - Xresat \leq Xre,\ Xrensat \in Rensat,\ Xresat \in Resat,\ Xre \in Re \quad (2)$$

$$Xrnensat = Xrne - Xrnesat \leq Xrne,\ Xrnensat \in Rnensat,\ Xrnesat \in Rnesat,\ Xrne \in Rne \quad (3)$$

Re and Rne are considered for essential and non-essential resources ,Rsla for partial satisfaction and Rsla for horizontal scaling is taken as boolean values by checking the availability of resources along with computed SLA value by CCPA agents.

4.2 Main Concept

Provided a request of XVM made by CCA the aim of the cloud broker is to obtain the VM's from CCPA and make available to CCA for a particular instance. The request satisfaction of XVM for any CCA agent leads to three possible cases in Residue-based VM provisioning.

Case 1: Where complete request of XVM of CCA was found at any CCPA at that instance by meeting all SLA requirements. Thus no residue is generated or needed for CCA request satisfaction.

Case 2: Minimum amount of VM's are available for satisfying the request of CCA. Thus cloud broker will need to generate the XVM from other CCPA and parallel check for meeting SLA requirements. The XVM which is generated should be ensured not to be queried by other CCA request until it is getting serviced for that instance.

Case 3: No virtual XVM is needed for satisfying the CCA that means the CCPA has inadequate resources and thus no extra XVM resources are needed to be generated by CCPA, leaving as residue the entire request.

4.3 Algorithms description

Figure 2 shows the main algorithm of Residue-Based VM Provisioning where a set of a request of essential and non-essential XVM's, the AvailList provides the list of collated cloud providers based on availability SLA parameter, CCResx gives the list of XVM's provided during the collaboration of resources by CCPA, CP-list of CPA available for collation formation. Minessential XVM's are assumed for satisfying the essential XVM's is considered. It returns the virtual XVM which will be generated for the request of CCA. VmC is a dictionary which holds the key as CPA agent ID or CCPA agent ID and values list of resources which they contribute to sharing XVM's. Line 5 computes the SLA value for collaborated CCPA cloud agents along with Rsla for partial satisfaction and Rsla for horizontal scaling. At line 7-13 the CCPA SLA is cross-checked with the computed SLA value and proceeded for acquiring resources using Available Resources algorithm as mentioned in Figure 3 to gather information of VM, a flag value for confirming the SLA, resources of the left amount of XVM's which are essential not satisfied and non-essential non-satisfied for particular CCPA and include it in VmC. At line 14 will check for availability from other CCPA agents for other partial request satisfaction of XVM's using AcquireAvailableFromCCPA as shown in Figure 5. From Line 15 to Line 22 will try to generate a new combination of CCPA for satisfying the CCA request of XVM's and finally at line 25 return for the unsuccessful generation of XVM's which is not satisfied or at line 27 the algorithm will try to return the XVM's by attach_extra_collect algorithm as shown in Figure 6. Figure 5 provides the cloud ranker algorithm where the CCPA are ranked by two techniques one by shuffle ranker to generate a new set of CCPA which are creating virtual XVM for satisfying the CCA request or Greedy Ranker for ranking the hierarchy of a set of clouds in VmC whose virtual VM will meet the CCA request without violating its SLA.

Multi-criteria decision and ranking of cloud mechanisms are major components during the residue-based VM provisioning. The Multi-Criteria decision is done through dominance relationship where the CCPA computed SLA values are taken in consideration and various factors like location of collaborated CPA, Minimum no of requests serviced by that CCPA within time duration specified by the user i.e. the turnaround time of request serviced is less than equal to time duration.

Following are the dominance relationship rules:

F1: Minimal no of CCA requests serviced by CCPA

F2: CCPA turnaround time of CCA request less than the specified time duration of their request.

F3: Location of CPA who are contributing XVM's for CCPA

Dominance Relation [2]:X dominates Y if the two following conditions meet

Condition1:$\forall J \in \{1, 2....k\}$ $f^{X}_{g^k}(Ti) \leq f^{Y}_{g^k}(Ti)$ – " X is not worse than Y for all Criteria " (4)

Condition2:$\exists J \in \{1, 2 ...k\}$ $f^{X}_{g^k}(Ti) < f^{Y}_{g^k}(Ti)$-"X is strictly better than Y for at least one Criterion" (5)

Ranking of clouds is done based on two mechanisms first is shuffle cloud ranker and second is greedy cloud ranker. Shuffle cloud ranker generates the possible sets of CCPA clouds contributing for virtual VM and Greedy cloud ranker uses Cloud ranker algorithm mentioned in Figure 4 where the dictionary values of VmC are compared dynamically for every CCA subrequest and does two-level sorting for generating rank for the CCPA agents who are successful or partially servicing the request.

Algorithm 1: Residue-BasedVMProvisioning(Re,Rne,AvailList,CCResx,Rsla,CPres,CP):

Input:Re-Request of essential of XVM's ,Rne-Request of nonessential of XVM's,AvailList –List of CCPA agents,CCResx-List of collaborated XVM's of each CCPA,Rsla-computed SLA value of CCPA,CPres-Available VMs with CPA and CP-No of cloud providers

Output:Returns the VirtualVM

Assumptions: MinEssentialVMs are computed from set of essential XVM.

```
1.  VmC={}
2.  for i in range(len(CP)):
3.      VmC[CP[i].nm]=CP[i].ARes
4.  for i in range(4):
5.  ACRsla[i]=abs(Rsla[(AvailList[0][i])-1]-(CheckRslaPartial()+CheckRslahorizontal( )))
6.  for i in range(4):
7.  if (Rsla[AvailList[0][i]-1]<=ACRsla[i]):
8.      vm,more,Rensat,Rnensat=AvailableResources(CCResx[i],Re,Rne,Rsla[AvailList[0][i]-
    1],CPres,MinEssentialVMs)
9.      if vm[i]!=0:
10.       if (more==0) or (CheckRslahorizontal()==0) :
11.         return vm
12.     else:
13.           VmC[Rsla[AvailList[0][i]-1]]=vm
14. T,more,Rensat,Rnensat=AcquireAvailableFromCloudSet(CP,Re,Rne,Rsla,CPres,MinEssentialVMs)
15. VmC[6]=T
16. Cnew=np.zeros((4,4),dtype=int)
17. if more==1:
18.   for i in range(4):
19.     for j in range(4):
20.         Cnew[i][j]=CPres[i][j]-CCResx[i][j]
21.       T,more,Rensat,Rnensat=AcquireAvailableFromCloudSet(CP,Re,Rne,Rsla,CPres,MinEssentialVMs)
22. VmC[7]=T
23. res=not VmC
24. if(str(res)==False):
25.     return 0
26. else:
27.     return attach_extra_collect(VmC,Rensat,Rnensat)
```

Figure 2: Residue-Based VM Provisioning Algorithm

Table 3: Computation Results for Different no of Requests with the probability of Acquire Resources

No of CCA Requests (Probability factors of acquire)	Satisfied Percentage of request and full or partial (Equi-Placer VM Provisioning)	Satisfied Percentage of request and full or partial (Residue-Based VM Provisioning)	Transaction rate (Equi-Placer VM Provisioning)	Transaction rate (Residue-Based VM Provisioning)	AvgTurnAround Time (Equi-Placer VM Provisioning)	AvgTurnAroundTime Residue-Based VM Provisioning)
2(0.2)	23.5	50.0	100	50	0.13	0.14
3(0.22)	37.5	25.5	37.5	25	0.23	0.20
4(0.33)	50.0	66.66	50	66.66	0.35	0.36
5(0.5)	50.0	81.75	50	76.25	0.53	0.54
6(0.75)	40.0	55.55	60	95	0.75	0.80

5. RESULTS AND DISCUSSION

The agent-based model is simulated in Python where all CCA, CPA are created as classes and implementation residue-based VM provisioning in CBA class. The collation formation process and multi-decision criteria are used for computing SLA of CCPA agent and select specific SLA which matches with SLA of CCA agent request.

5.1 Evaluation Metrics

The Equi-placer VM Provisioning and Residue-based VM Provisioning satisfy the XVM's CCA agent request by horizontal scaling of CPA by collation formation for forming specific CCPA agents in which multi-decision criteria is used for cross-checking specific SLA value with the SLA

Parameters specified by CCA agent. The similar metrics as mentioned for a residue-placer algorithm, the three important metrics which are used for comparison between residue-based VM provisioning and Equi-Placer VM Provisioning are (i) Request fully or partially Satisfied, (ii) Transaction rate and (iii) Turnaround Time.

1.Satisfied Percentage of request and full or partial ($Sat^{fp}_{Percent}$): The amount of requests completed, in full or in part, by the CCPA is the consecutive number of requests generated by its CCA. The probability of partly satisfying a requirement and is determined as follows:

$$Sat^{fp}_{Percent} = \frac{noof^{FP}_{Req}}{noof_{Req}} * 100 \qquad (6)$$

Where $noof_{req}$ the total no of request is raised by CCA agents and $noof^{FP}_{req}$ is the number of requests that are either full or partial satisfied by the CCPA agents.

2. Transaction rate: It is computed in percentage as follows:

$$Trans_{rate} = \frac{\sum_{i=1}^{noof_{req}} Reqsat_{score}(i)}{noof_{req}} * 100 \qquad (7)$$

Where $Reqsat_{score}$ is the request satisfied score made by CCPA agent on satisfying request made by CCA and it takes values from 0 to 1 based on the amount of request satisfaction

3. AvgTurnAroundTime: It is for all $noof^{FP}_{req}$ requests that are fully or partially satisfied is:

$$Tot_{Avgturnaroundtime} = \frac{\sum_{i=1}^{noof^{FP}_{req}} T_{turnaroundavg}(i)}{noof^{FP}_{req}},$$

$$Reqsat_{score}(i) \neq 0, \forall\ 1 \leq i \leq noof^{FP}_{req} \text{-(8)}$$

Where, $T_{turnaroundavg}$ (i), is the turnaround time of the ith request that is full or part satisfied by the CCPA i.e. time taken for a CCA request to be full or partial satisfied by CCPA.

5.2 Implementation Details

The residue-based VM provisioning is implemented through Python software on Intel Pentium i3 Processor, 4GB RAM. The agent-based is simulated using python classes and following data of Table 2 is considered for CCA request of XVM's. After running the algorithm the results for different no of requests obtained in Table 3 is shown here probability factor of acquiring resources by CCA with the contribution of CCPA agents are listed. The results of satisfied per cent of full or partial for the different probability of request is plotted in Figure 7 the interpretation says after probability factor of 0.5 the satisfying percentage of the request of Residue-Based VM Provisioning is increasing than Equi-Placer VM Provisioning algorithm. Figure 8 represents the transaction rate vs. probability factor of acquires resources. It also interprets that

from probability factor 0.03 the transaction success rate is increasing for Residue-based VM provisioning over Equi-Placer VM Provisioning. Figure 9 shows the Average turnaround time over the probability factor of requests it also reveals the same interpretation after probability factor 0.05 the turnaround time is increasing.

The Cloud Ranker algorithm uses two mechanisms for generating the ranks for CCPA agents who are contributing for creation virtual VM for satisfying request of CCA either full or partial. Figure 10 gives the bar graph plot for shuffle ranker and Greedy cloud ranker for satisfied per cent of requests of full or partial for different probability request. The graph shows the success rate of satisfaction is more for greedy cloud ranker than a shuffle. Figure 11 shows the bar graph plot for shuffle ranker over greedy ranker for transaction success rate for different probability factors of acquiring resources. This graph shows that initial probability factor shuffle ranker has good performance over greedy ranker but as the probability factor increases as more no of requests come from CCA the greedy ranker has good performance then shuffle ranker. Figure 13 results the bar graph between Average turnaround time over the probability factor of acquiring requests. The results show that greedy ranker does more better performance than shuffle ranker by increased turnaround time

Algorithm 2: AvailableResources(CCRes,Re,Rne,Rsla,CPres,minvm):
Input: Re-Request of essential of XVM's ,Rne-Request of nonessential of XVM's, ,Rsla-computed SLA value of CCPA, CCResx-List of collaborated XVM's of each CCPA,Rsla-computed SLA value of CCPA,CPres-Available VMs with CP
Output: vm-created in CCPA,more-flag to denote residue left by CCPA,Rensat-quantites of XVMs essential that are not satisified,Rnensat- quantites of XVMs of non-essential VMs that are not satisified
Assumptions: MinEssentialVMs are computed from set of essential XVM.

```
AvaiesRes=np.array(CCRes)
CPavres=np.array(CPres)
AvainesRes=np.zeros((4,4),dtype=int)
for i in range(4):
  for j in range(4):
      AvainesRes[i][j]=abs(AvaiesRes[i][j]-CPavres[i][j])

for i in range(4):
   for j in range(4):
      more=CheckRslaPartial()
        if (AvaiesRes[i][j]>=Re[j]) and (AvainesRes[i][j]>=Rne[j]):
              vm[i]=Re[i]+Rne[i]
              return vm,0,0,0
       elif more==0:
            return vm,1,Re,Rne
      elif AvaiesRes[i][j]>=minvm[j]:
          if Re[j]-minvm[j]>=minvm[j]:
              if (AvaiesRes[i][j]>=(Re[j]-minvm[j])):
                  Resat[j]=abs(Re[j]-minvm[j])
              else:
                  Resat[j]=AvaiesRes[i][j]
                  Rensat[j]=abs(Re[j]-Resat[j])
                  Rnesat=AcquireMaximum(AvainesRes[i][j],Rne)
                  more=1
        else:
            Resat[j]=AcquireMaximum(AvainesRes[i][j],Re,Rne)
            Rensat[j]=abs(Re[j]-Resat[j])
             more=0
              vm[j]=Resat[j]+Rensat[j]
            Rnensat[j]=abs(Rne[j]-Rnesat[j])
             return vm,more,Rensat,Rnensat
      else:
            return vm,1,Re,Rne
```

Figure 3: Available Resources Algorithm

Algorithm 3: Ranker(CP,Re,Rne,Rsla,MinEssentialVMs,CPres):
Input: Re-Request of essential of XVM's ,Rne-Request of nonessential of XVM's, ,Rsla-computed SLA value of CCPA, CCResx-List of collaborated XVM's of each CCPA,Rsla-computed SLA value of CCPA,CPres-Available VMs with CP
Output:Crank list of CCPA with Ranks

```
for i in range(len(CP)):
    for j in range(4):
        if (CP[i].FixedSLa[0]<=Rsla[j]) and (CPres[i][j]<MinEssentialVMs[j]):
            CPnovm[i]=CP[i]
        if(CP[i].FixedSLa[0]<=Rsla[j]) and (CP[i]!=CPnovm[i]):
            Crank[i]=CP[i]
Crank=Crank.append(CPnovm)
return Crank
```

Figure 4: Cloud Ranker Algorithm

Algorithm 4: AcquireAvailableFromCCPA(CP,Re,Rne,Rsla,CPres,MinEssentialVMs):
Input:Re-Request of essential of XVM's ,Rne-Request of nonessential of XVM's,AvailList –List of CCPA agents,CCResx-List of collaborated XVM's of each CCPA,Rsla-computed SLA value of CCPA,CPres-Available VMs with CPA and CP-No of cloud providers
Output: VmC- is the list of CPA with their computed collaborated resources,more-flag to denote residue left by CCPA,Rensat-quantites of XVMs essential that are not satisified,Rnensat- quantites of XVMs of non-essential VMs that are not satisified
Assumptions: MinEssentialVMs are computed from set of essential XVM

```
VmC={}
Rensat=Re
Rnensat=Rne
ACRsla=[0]*4
more=0
for i in range(4):
    ACRsla[i]=abs(Rsla[i]-(CheckRslaPartial()+CheckRslahorizontal( )))
Crank=Ranker(CP,Rensat,Rnensat,ACRsla,MinEssentialVMs,CPres)
if Crank!=None:
    for i in len(Crank):
        vm,more,Rensat,Rnensat=AvailableResources(Crank[i],Re,Rne,Rsla,CPres,MinEssentialVMs)
        if vm[i]!=0:
            VmC[Crank[i]]=vm
        if more==0 :
            break
        if CheckRslahorizontal()==0 :
            more=0
            break
return VmC,more,Rensat,Rnensat
```

Figure 5: Acquire Available from CCPA Algorithm

Algorithm 5: attach_extra_collect(VmC,Rensat,Rnensat):

Input: VmC- is the list of CPA with their computed collaborated resources, CCPA,Rensat-quantites of XVMs essential that are not satisified,Rnensat- quantites of XVMs of non-essential VMs that are not satisified

Output;VirtualVM

```
VirtualVm=[0]*4
d=list(VmC.values())
for i in range(4):
        for j in range(4):
            if(d[i][j]>=Rensat[j] or d[i][j]>=Rnensat[j]):
                VirtualVm[j]=Rensat[j]+Rnensat[j]
            else:
                VirtualVm[j]=d[i][j]
Rensat[j]=abs(Rensat[j]-d[i][j])
Rnensat[j]=abs(Rnensat[j]-d[i][j])
return(VirtualVm)
```

Figure 6: Attach Extra Collect Algorithm

Table 2: CCA Agent XVM's Request

S.No	CCA-ID	XVM 's Request
1	CCA-01	[12,10,18,19]
2	CCA-02	[10,7,8,9]
3	CCA-03	[23,45,67,89]
4	CCA-04	[3,5,6,7]
5	CCA-05	[12,13,14,15]
6	CCA-06	[14,24,13,4]

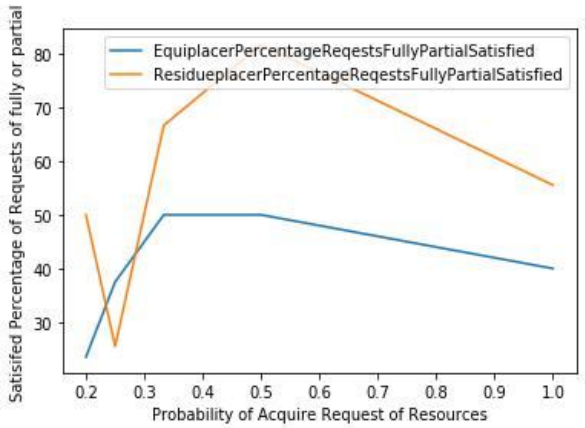

Figure 7: Probability of Requests with Satisfied Percentage

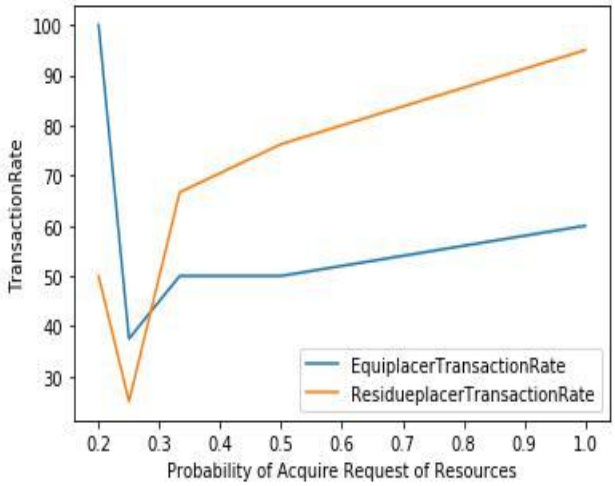

Figure 8: Probability of Requests with Transaction Rate

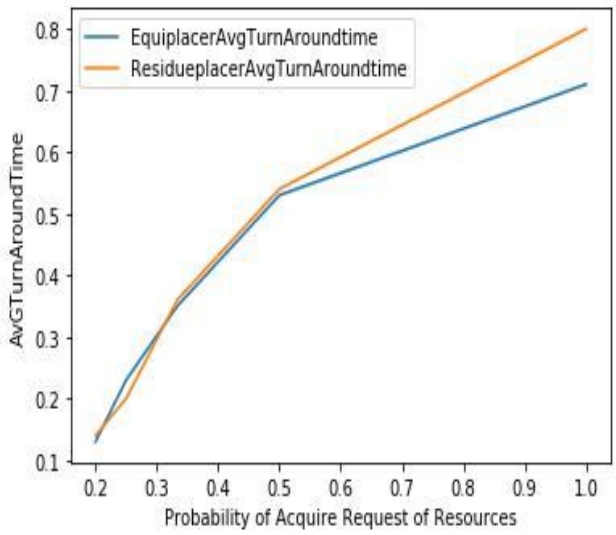

Figure 9: Probability of Requests with AverageTurnAroundTime

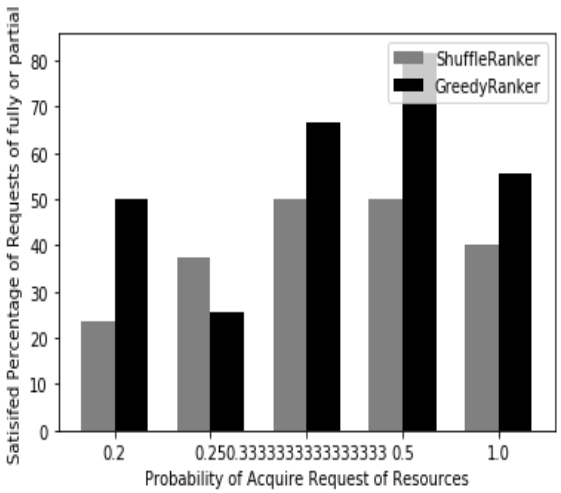

Figure 10: Probability of request with satisfied percentage using shuffle and Greedy ranker

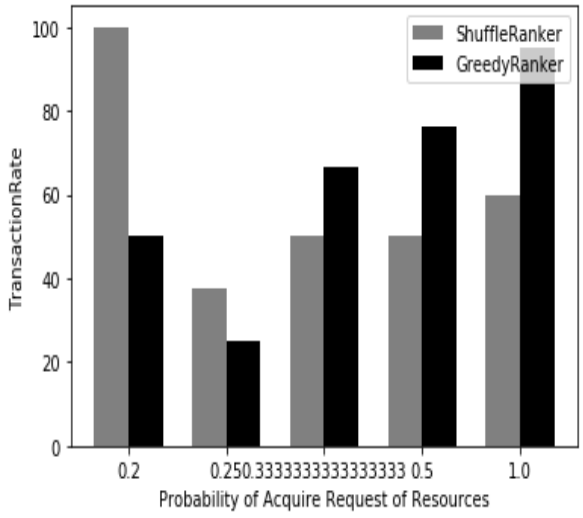

Figure 11: Probability of request with transaction rate using shuffle and Greedy ranker

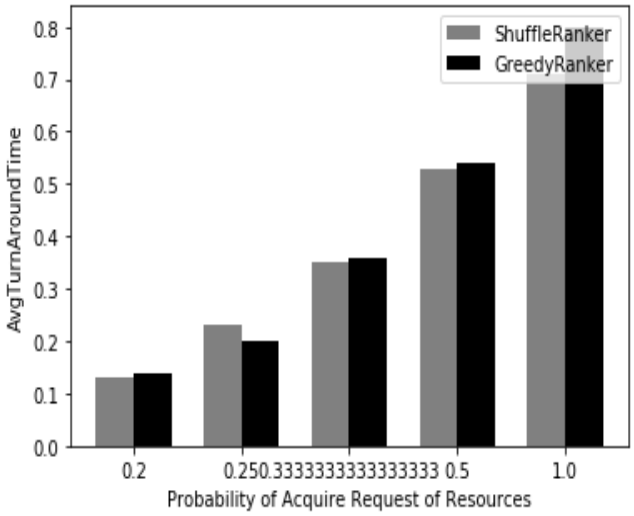

Figure 12: Probability of request with AvgTurnAroundTime using shuffle and Greedy ranker

6. CONCLUSION AND FUTURE WORK

Residue-based VM Provisioning does show the good performance over Equi-placer VM provisioning algorithm but the algorithm performance has tremendous effect with the multi-decision criteria for choosing specific SLA values of CCPA for collation formation and identifying dominant CCPA which will lead successful satisfaction of CCA request. The other algorithm is cloud ranking algorithm which plays a very important role in generating virtual VM which either full or partial satisfy CCA agent request at fast transaction rate.

The replacement of Multi-decision criteria of dominance relationship with a more specific method for computing dominant of CCPA agents by using ELECTRE (Elimination Choice Translation Reality) method for listing outranking relationship which merges ranking of dominance relationship by identifying concordance and discordance relationships among CCPA agents depending on their contribution of resources of XVM's with more SLA parameters like response time, process time and reliability.

REFERENCES

1. D. Chiu, **Elasticity in the cloud**, *ACM Crossroads* Vol.16, no. 3, 2010, pp.3–4. https://doi.org/10.1145/1734160.1734162
2. Xiao Z, Song W, Chen Q. **Dynamic resource allocation using virtual machines for the cloud computing environment**. IEEE Trans Parallel Distrib Syst, Vol.24, no.6, pp.1107–17, 2013. https://doi.org/10.1109/TPDS.2012.283
3. Kumar MRV, Raghunathan S. **Heterogeneity and thermal aware adaptive heuristics for energy-efficient consolidation of virtual machines**

in infrastructure clouds. Journal Computing Systems Sciences, Vol.82, no.2 pp.191–212, 2016. https://doi.org/10.1016/j.jcss.2015.07.005

4. Khani H, LatifiA, Yazdani N, Mohammadi S. **Distributed consolidation of virtual machines for power efficiency in heterogeneous cloud data centres.** Computing Electrical Engineering, Vol.47, pp.173–85, 2015. https://doi.org/10.1016/j.compeleceng.2015.08.001
5. B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P.Massonet, H.Muñoz, G. Tofetti, **RESERVOIR – When one cloud is not enough**, IEEE Computer, Vol.44, no.3 pp.44–51, 2011. https://doi.org/10.1109/MC.2011.64
6. Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I.M., Montero, R., Wolfsthal, Y., Elmroth, E., Caceres, J. and Ben-Yehuda, M., **The reservoir model and architecture for open federated cloud computing**. IBM Journal of Research and Development, Vol.*53*, no.4, pp.4-1,2009 https://doi.org/10.1147/JRD.2009.5429058
7. K. Nagin, D. Hadas, Z. Dubitzky, A. Glikson, I. Loy, B. Rochwerger, L. Schour, **Inter-cloud mobility of virtual machines**, *Annual International Conference on Systems and Storage*, *ACM*, 2011, p.3. https://doi.org/10.1145/1987816.1987820
8. R. Buyya, R. Ranjan, R.N. Calheiros, **InterCloud: the utility-oriented federation of cloud computing environments for scaling of application services, in Algorithms and Architectures for Parallel Processing**, Springer, pp.13–31,2010. https://doi.org/10.1007/978-3-642-13119-6_2
9. R.N. Calheiros, A.N. Toosi, C. Vecchiola, R. Buyya, **A coordinator for scaling elastic applications across multiple clouds**, Future Generation. Computing Systems. Vol.28, no.8, pp.1350–1362, 2012.
10. D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, M. Morrow, **Blueprint for the intercloud-protocols and formats for cloud computing interoperability**, *International Conference on Internet and Web Applications and Services*, *IEEE*, 2009, pp.328–336. https://doi.org/10.1109/ICIW.2009.55
11. D. Bernstein, D. Vij, **Intercloud directory and exchange protocol detail using xmpp and rdf**, *World Congress on Services (SERVICES-1), IEEE*, 2010, pp.431–438.
12. D. Bernstein, D. Vij, **Intercloud exchanges and roots topology and trust blueprint**, *International Conference on Internet Computing*, 2011, pp.135–141.
13. A.C. Marosi, G. Kecskemeti, A. Kertesz, P. Kacsuk, FCM: an architecture for integrating IaaS cloud systems, in 2nd International Conference on Cloud Computing, GRIDs, and Virtualization, IARIA, 2011, pp.7–12.
14. G. Kecskemeti, M. Maurer, I. Brandic, A. Kertesz, Z. Nemeth, S. Dustdar, **Facilitating self-adaptable inter-cloud management,** *Euromicro International Conference on Parallel, Distributed and Network-Based Processing, IEEE*, 2012, pp.575–582. https://doi.org/10.1109/PDP.2012.41
15. A. Kertesz, G. Kecskemeti, M. Oriol, P. Kotcauer, S. Acs, M. Rodríguez, O. Mercè, A.C. Marosi, J. Marco, X. Franch, **Enhancing federated cloud management with an integrated service monitoring approach**, Journal of Grid Computing, Vol.11, no.4, pp.699–720,2013. https://doi.org/10.1007/s10723-013-9269-0
16. Singh H, Randhawa R, **ARMM: Adaptive Resource Management Model for Workflow Execution in Clouds.** In: Negi A, Bhatnagar R, Parida L (eds) Distributed Computing and Internet Technology ICDCIT 2018. Lecture Notes in Computer Science, Vol 10722. Springer, Cham, pp. 315–329
17. Ghobaei-Arani M, Jabbehdari S, Ali Pourmina M, **An autonomic resource provisioning approach for service-based cloud applications: a hybrid approach**. Future Generation Computing System, Vol.78,no.1 pp.191–210,2018 https://doi.org/10.1016/j.future.2017.02.022
18. Xu J, Cao J, **A broker-based self-organizing mechanism for cloud-market**. *International Federation for Information Processing (IFIP).* Springer, Berlin, pp. 281–293
19. Chaabouni T, Khemakhem , **Resource management based on Agent technology**. *In: Proceeding of IEEE on cloud computing*, NOORIC, 2013, pp. 372–375
20. Kecskemeti G, Maurer M, Brandic I, Kertesz A, Nemeth ZS, Dustdar S **Facilitating self-adaptable inter-cloud management**. *Euromicro international IEEE conference on parallel distributed and network-based processing*, 2012, pp 575–582.
21. Comi A, Fotia L, Messina F, Pappalardo G, Rosaci D, Sarné GML, **An evolutionary approach for Cloud learning agents in multi-cloud distributed contexts.** *International conference on enabling technologies: infrastructure for collaborative enterprises* (WETICE'15), IEEE, Cyprus, 2015, pp. 99–104 https://doi.org/10.1109/WETICE.2015.27
22. Manvi SS, Birje MN, Prasad B, **An Agent-based Resource Allocation Model for computational grids**. Multi-Agent Grid System International Journal, Vol. no.1pp.17–27,2005

23. Lee YH, Huang KC, Shieh MR, Lai KC (2017) **Distributed resource allocation in federated clouds**. Journal Supercomputing, Vol.73,no.7 pp.3196–3211,2017
24. Ishikawa T, Fukuta N, **Federated cloud-based resource allocation by automated negotiations using strategy changes**. In: Fujita K, Ito T, Zhang M, Robu V (eds) *Next frontier in agent-based complex automated negotiation.Studies in computational intelligence*, Springer, Tokyo, Vol. 596. pp 111–125,2015 https://doi.org/10.1007/978-4-431-55525-4_7
25. Zulkar Nine Md SQ, Abul Kalam Azad Md, Abdullah S, Ahmed N, **Dynamic load sharing to maximize resource utilization within cloud federation.** *CloudCom-Asia, Springer International Publishing* Switzerland,2015, pp. 125–137.
26. Carlini E, Coppola M, Dazzi P, Mordacchini M, Passarella, **A Self-optimising decentralised service placement in heterogeneous cloud federation**, *IEEE international conference on self-adaptive and self-organizing systems* (SASO),2016, pp. 110–119
27. Son S, Kang DJ, Huh SP, Kim WY, Choi W, **Adaptive trade-off strategy for bargaining-based multi-objective SLA establishment under varying cloud workload**. Journal of Supercomputing, Vol. 72,no.4 pp.1597–1622,2016
28. Andronico G, Fargetta M, Monforte S, Paone M, Villari M **A model for accomplishing and managing dynamic cloud federations**, *International conference on utility and cloud computing, IEEE/ACM*,2014, pp. 744–749 https://doi.org/10.1109/UCC.2014.121
29. Hou F, Mao X, W. Wu W, Liu L, Panneerselvam J, **A cloud-oriented services self-management approach based on multi-agent system technique.** *International conference on utility and cloud computing, IEEE/ACM*,2014 pp 261–268
30. De Meo, P., Messina, F., Rosaci, D. and Sarné, G.M., **An agent-oriented, trust-aware approach to improve the QoS in dynamic grid federations**, Concurrency and Computation: Practice and Experience, Vol.27, no.17, pp.5411-5435, 2015. https://doi.org/10.1002/cpe.3604
31. Vadla PK, Kolla BP, Perumal T, **FLA-SLA Aware Cloud Collation Formation Using Fuzzy Preference Relationship Multi-Decision Approach for Federated Cloud**. Pertanika Journal of Science & Technology. Vol.28, no.1, pp.117-140,Jan 1 ,2020.
32. Kirthica, S. and Sridhar, R., **A residue-based approach for resource provisioning by horizontal scaling across heterogeneous clouds.** International Journal of Approximate Reasoning, Vol.101, pp.88-106.2018
33. Kolla B.P., Dorairangaswamy M.A., Rajaraman A., **A neuron model for documents containing multilingual Indian texts**. International Conference on Computer and Communication Technology, ICCCT-2010, pp. 451-454.2010 https://doi.org/10.1109/ICCCT.2010.5640489
34. Abdulla G, Jelidi Md, **Experimental Evaluation of a Hybrid Intrusion Detection System for Cloud Computing**.International Journal of Advanced Trends in Computer Science and Engineering,Vol.8,no.6,pp.3065-3073,2019. **https://doi.org/10.30534/ijatcse/2019/65862019**
35. Prakash K.B., Dorai Rangaswamy M.A., Raman A.R.**Text studies towards multi-lingual content mining for web communication**. Proceedings of the 2nd *International Conference on Trendz in Information Sciences and Computing*, TISC-2010, 2010 pp. 28-31.
36. Trinath Basu, M., Sastry, J.K.R., **Enhancing Data Security under Multi-Tenancy within Open Stack**, International Journal of Advanced Trends in Computer Science and Engineering,vol.9,no.1,pp.533-544,2020. https://doi.org/10.30534/ijatcse/2020/73912020
37. Faten A. Saif , MN Derahman, Ali A. Alwan , Rohaya Latip. **Performance Evaluation of Task Scheduling using Hybrid Meta-heuristic in Heterogeneous Cloud Environment**, International Journal of Advanced Trends in Computer Science and Engineering,Vol.8,no.6,pp.3249-3257,2019, https://doi.org/10.30534/ijatcse/2019/93862019