Multiply-Sectioned Bayesian Network for Multi-Agent Learning Based Meta Resources Scheduling in CloudSim

Khaled M. Khalil
Faculty of Computer and Information Science Ain shams University Cairo, Egypt
kmkmohamed@gmail.com

M. Abdel-Aziz
Faculty of Computer and Information Science Ain shams University Cairo, Egypt
mhaziz@cis.asu.edu.eg

Taymour T. Nazmy
Faculty of Computer and Information Science Ain shams University Cairo, Egypt
timor.mohammad@cis.asu.edu.eg

Abdel-Badeeh M. Salem
Faculty of Computer and Information Science Ain shams University Cairo, Egypt
absalem@cis.asu.edu.eg

***Abstract*—The interest in Cloud Computing is growing significantly. This is for the application requirements to quickly access more resources to expand as needed. Collecting real time data about Cloud resources and finding the best fit resources to user requests are time and space consuming operation. Resources can be added/removed at any time and Cloud Computing System should utilize available resources and schedule user requests within specific time for execution. Considering such dynamic and complex domain, it is important to setup global policy among resources. The global policy can be applied through resources meta-scheduling. Multiply-Sectioned Bayesian Network (MSBN) provides a natural framework for such metaphor. We propose using adaptive Multi-Agent System (MAS) based on Multiply-Sectioned Bayesian Network (MSBN) for the function of meta-scheduling in Cloud Computing Systems. Random variables are representing the uncertainties in workload and resource's attributes. Each set of identical hosts in the Cloud System are grouped as subdomains, where set of agents have partial knowledge about the global system and local observations of hosts in the subdomain. Then, agents work to estimate the state of the local hosts and act accordingly to provide probability of assigning available resources to the user request. In addition, agents need to communicate their beliefs to other agents to improve the overall meta-scheduling performance. Agent with the strongest belief will get the request assigned for local resources scheduling and execution. If the agent failed to satisfy the request needs then the agent with the next strongest belief is assigned and so on. We present our proposed model by extending CloudSim simulation. Validation of the proposed model and improvement in scheduling of Virtual Machines are presented.**

Keywords— Multi-Agent System; Multiply-Sectioned Bayesian Network; Meta-Scheduling; Agent-Based Learning; CloudSim

I. INTRODUCTION

Many parallel and distributed applications require simultaneous access to large number of resources for a specific period of time. These applications are like social networking, web hosting, content delivery and real time data processing applications [1]. In such type of applications, requested resources may expand based on the execution state of the application requests. Cloud Computing Systems satisfy these requirements by creating Virtual Machines (VMs) and allocating resources to them based on users QoS (Quality of Service) necessities [2]. Cloud Computing Systems define VMs and resource scheduling policies to handle allocation and management of resources among various applications [3].

Effective resource assignment strategy is required to satisfy users and maximize the profit for cloud service providers. Thus, resources assignment strategy plays a very crucial role in the Cloud Computing Systems. Cloud state is uncertain and this is coming from multiple sources like adding/removing hosts, workload pattern, and fragmentation of available resources. In practical systems, it is impossible to achieve perfect match between available resources and workload [4]. Meta-Scheduling enables reliable and efficient sharing of resources managed by different local resource management systems [5]. Meta-Scheduling can be defined as a key decision unit that involves tradeoff between cost and performance to redirect requests to the local level schedulers to get best fit resources in terms of QoS like: resources availability, execution time, latency, and cost.

Multi-Agent Systems (MAS) present a natural metaphor to implement meta-scheduling function. Agents can work together to improve the whole system performance. Each agent can cooperate to find mutual accepted resources schedule. Each agent needs to produce a response representing its local believe about the request assignment. Then a bid against each other can be held to schedule in an auction mechanism [4]. To improve the scheduling ability of a multi-agent system, learning capability has to be adopted by agents. Agents need to learn from each assignment and get their beliefs adjusted. On the other hand, evaluation of different meta-scheduling and scheduling approaches in real Cloud Computing Systems is expensive. Simulation-based tools offers significant benefits in these types of problems. One of the most common simulation-

978-1-5386-0821-0/17/$31.00 ©2017 IEEE

based tools called CloudSim. CloudSim is an extendible simulation toolkit that enables modeling and simulation of Cloud Computing environments [6, 7]. It is a customizable tool and it allows extension to core policies. CloudSim simulates datacenters, hosts, VMs, resources provisioning policy, and VM allocation policy.

We introduce a novel concept of meta-resource allocation for Cloud Computing based on the Multiply-Sectioned Bayesian Network (MSBN) for Multi-Agent System. Our proposed system learns different QoS attributes such as cost, latency, and availability of resource to effectively suggest an assignment of the resources. The rest of this paper is organized in four sections. Section 2 includes the background and literature review. While section 3 presents the proposed system model. Section 4 presents the experiments and discussion of results. Finally, section 5 concludes the study and provides future work.

II. BACKGROUND

Yousafzai et al. [8] presented basic and essential concepts of resource allocation in Cloud Computing. In addition, they listed the challenges of resource allocations as follows: more cost-effective allocation schemas, seamless VM migration, reducing the operational cost, and reducing the SLA violations. On the other hand, Vinothina et al. [9] discussed the advantages and limitations of resources scheduling in Cloud Systems. Then, they provided a classification of Resource Allocation Strategies for utilizing and scheduling resources within the limits of the Cloud environment. After that they discussed the impact of each classification on the Cloud Systems.

Endo et al. [10] focused on the uncertainties that increase difficulty in scheduling and matching resources with requests such as users rely on infinite resources, resources heterogeneity, and representing Cloud applications in terms of what is known as resource offering and treatment. Majumdar [11] discussed two additional uncertainties for the scheduling of resources; errors associated with estimation of request execution time and lack of knowledge of local resource management policies.

Alnajdi et al. [12] mentioned that there are four main resource scheduling strategies used. They are service level agreement-based, utility-based, market-based, and priority-based strategies. Service Level Agreement (SLA)-Based Strategy is the most common strategy used. This strategy determines a prior-agreement of terms of QoS like price and latency between the user and the service provider. Market-Based Strategy is focused on the cost of the service provided to the user. This is happening by auction-based mechanism in which all involved components provide a bid for the cost of executing the request and a decision is made to maximize the revenue and reduce the cost. Utilization-based Strategy is used to maximize utilization of resources. This strategy is targeting reducing the unallocated parts of the resources in the whole system. Finally, priority-based strategy is focusing on execution of highest priority requests first to get available resources assigned to the most critical requests.

Meta-Scheduling gives additional functionalities to resource management. The term meta-scheduling is defined as establishing a wide policy control among resources [5]. This includes management of resources controlled by different domains. Xu et al. [13] and Sotiriadis et al. [14] provided a survey of different meta-scheduling algorithms and discussed the additional challenges of using meta-schedulers in Cloud Computing. They mentioned that there are three main architectures of meta-schedulers: Centralized, Hierarchical and Decentralized. Centralized meta-schedulers maintain one instance of the meta-scheduler which tracks information of all resources. Centralized Meta-Schedulers implement algorithms like genetic [15] and greedy [16] to map VMs to hosts. Centralized meta-schedulers are simple and provide good performance but they have drawbacks of bottleneck and single point of failure. Hierarchical meta-schedulers are like the centralized meta-schedulers in which requests are sent to one instance of a meta-scheduler then the request is directed to related schedulers in the hierarchy. Hierarchical meta-schedulers are like [17, 18] and they share the same advantages and drawbacks of the centralized meta-schedulers. Decentralized meta-schedulers like [19, 20] are based on set of meta-schedulers that each meta-scheduler has information about a sub-domain of the resources and they communicate at regular intervals so that to collect load data. Sotiriadis et al. [19] provided a framework called ICMS for decentralized meta-scheduling for inter-cloud. While, Huang et al. [20] provided a community-aware scheduling algorithm. It is based on scheduling tasks by distributing requests among set of hosts without asking for detailed real-time processing information nor control authorities. Sotiriadis et al. [14] provided listing of more decentralized meta-schedulers and algorithms involved inside them. At the end, decentralized meta-schedulers are more complex than the centralized and hierarchical ones but they are more flexible and scalable. As a remark, centralized and hierarchical meta-schedulers need a complete knowledge of the actual resources which is unrealistic. This includes the number of hosts, number of requests submitted, and the workload of each host. On the other hand, in the decentralized meta-schedulers, the information is incomplete and the requests received from the meta-scheduler are assigned to the local schedulers in the same or a different domain.

More flexible and automated meta-schedulers are requested to handle the dynamic and uncertain environment of resources in Cloud Computing Systems. Meta-schedulers need to adapt based on the current state of the environment and take the advantage of historical data of previous assigned requests and workload. Subramani et al. [21] designed a meta-scheduler that adapts-to-changes. They submit the same request to multiple domains simultaneously instead of the domain of the best fit or light loaded domain. They assume that this will reduce the rejected rate of the request but it decreases the system performance. Zhaofeng and Aiwan [22] proposed an ant-colony algorithm with three constraint conditions based meta-scheduler. The three conditions are expected execution time, network delay, and network bandwidth. Kessaci et al. [23] proposed a multi-objective genetic algorithm that focused on scheduling requests based on terms of energy consumption, CO_2 emissions, and the generated profit. They focused on meta-scheduling High-Performance Computing applications on

a Geographically Distributed Cloud Federation. Niu et al. [16] worked on adaptive meta-scheduler for efficiency fairness in data-intensive applications. They proposed FLEX which is a set of multiple existing schedulers and it chooses the most proper scheduler according to the current workload and user SLA. Prado et al. [24] compared two additional techniques for adaptive meta-scheduling, fuzzy rule-based systems (FRBS) – based strategies and Gaussian Scheduling founded on Gaussian Mixture Models (GMM). Both group of strategies model the state of resources and select the best site based on the current environment and request conditions. FRBSs provide fuzzy IF-THEN rules to inference the suitability index of sites to get requests assigned to. While, GMMs are to be trained to model the probability destiny distribution indicating the probability of selecting the resources site based on the request conditions.

The Bayesian networks constitute a reasoning method based on probability theory. Bayesian network consists of a set of nodes and a set of arcs which together constitute a directed acyclic graph (DAG). Multiply-Sectioned Bayesian Network is an extension of the Bayesian Network for flexible and cooperative multi-agent based inferencing [25]. A large and complex domain is divided into sub-domains and represented as sub-networks. Each sub-network contains the knowledge of the sub-domain and an agent or set of agents are responsible for acting based on such knowledge [26]. Belief propagation among the sub-networks is required [27] so when the subnet beliefs are updated, all adjacent subnets beliefs need to be propagated. One of the approaches used for belief propagation is the Junction Tree (JT) [28].

III. The Proposed System Model

We are proposing a distributed meta-scheduler based on multi-agent systems and using MSBN.

Let H_i is host i in the datacenter at given time T, then:
$H_i : (id_i, m_i, b_i, s_i, pen_i, pes_i, c_i)$ where
id_i is the host id,
m_i is the memory available in the host,
b_i is the bandwidth available in the host,
s_i is the storage available in the host,
pen_i is the number of processing elements (PEs) in the host,
pes_i is the MIPs (millions instructions per second) speed of each of the PEs in the host,
c_i is the cost of using a single resource unit in the host,
and $i \in I \{1....h\}$.

Let V_j represents the VM j at given time T, then:
$V_j : (uid_j, m_j, b_j, s_j, pen_j, pes_j, l_j)$ where
uid_j is the user id who submitted the VM,
m_j is the memory needed for the VM,
b_j is the bandwidth needed for the VM,
s_j is the storage needed for the VM,
pen_j is the number of PEs needed for the VM,
pes_j is the MIPs of each of the PEs for the VM,
l_j is maximum latency allowed for VM assignment,
and $j \in J \{1....v\}$.

Let Q_{ij} represents the QoS which the system needs to maintain for V_j assignment at given time T, then:
$Q_{ij} : (b_{ij}, pe_{ij}, s_{ij}, m_{ij}, c_{ij}, l_{ij}, u_{ij})$ where
b_{ij} is the bandwidth matchmaking: H_i, V_j where $b_{ij}=max\ (b_j|b_i)$,
pe_{ij} is the PE matchmaking: H_i, V_j where $pe_{ij}=max\ (pe_j|pe_i)$,
s_{ij} is the storage matchmaking: H_i, V_j where $s_{ij}=max(s_j|s_i)$,
m_{ij} is the memory matchmaking: H_i, V_j where $m_{ij}=max(m_j|m_i)$,
c_{ij} is the cost of assignment of V_j to H_i,
l_{ij} is the latency matchmaking: H_i, V_j where $l_{ij}=max(l_j|l_i)$,
u_{ij} is the utilization of the site of hosts including H_i,
and $i \in I \{1....h\}$, $j \in J \{1....v\}$.

So our proposed system goal is to *min (c), max (u)* where:

$$c = \sum_{i=1}^{h} \frac{\sum_{j=1}^{v} c_{ij}\ a_{ij} \max(b_j|b_i) \max(pe_j|pe_i) \max(s_j|s_i)}{\max(m_j|m_i) \max(l_j|l_i)}$$

$$\text{and } u = \sum_{i=1}^{h} u_i \qquad (1)$$

while the capacity constraints of the H_i as follow:

$$m_i \geq m_j,\ b_i \geq b_j,\ s_i \geq s_j,\ pen_i \geq pen_j,\ pes_i \geq pes_j \text{ where } i \in I \{1....h\}, j \in J \{1....v\} \qquad (2)$$

and assignment constraint of a_{ij} where V_i is assigned to H_j

To implement the MSBN, we assume that we have groups of almost identical hosts in terms of resources units and cost characteristics. Each group of hosts represents a sub-network. Each sub-network has 12 random variables (input size, output size, processing size, available bandwidth, available MIPs, available storage, available memory, available resources, cost, latency, QoS, and Resources Utilization). Fig. 1 shows the sub-networks in the proposed system. While, Fig. 2 shows variables in a sub-network representing a group of hosts and sharing two interfacing nodes with the root sub-network; the Resources Utilization and QoS nodes. Each sub-network is managed by an agent which receives requests for assignment and provides probability for each VM assignment. Fig. 3 shows the interface nodes of each sub-network linked to the root decision utility.

To determine the joint probability distribution $P(x_1, x_2 \ldots x_n)$ in a Bayesian network, it is sufficient to know the conditional probabilities $P(x_i \mid parent\ (x_i))$, where $parent\ (x_i)$ is the parent set of variable x_i, i.e. the set of variables by which x_i is affected:

$$P(x_1, x_2 \ldots x_n) = \prod_{i=1}^{n} P(x_i \mid parent\ (x_i)) \qquad (3)$$

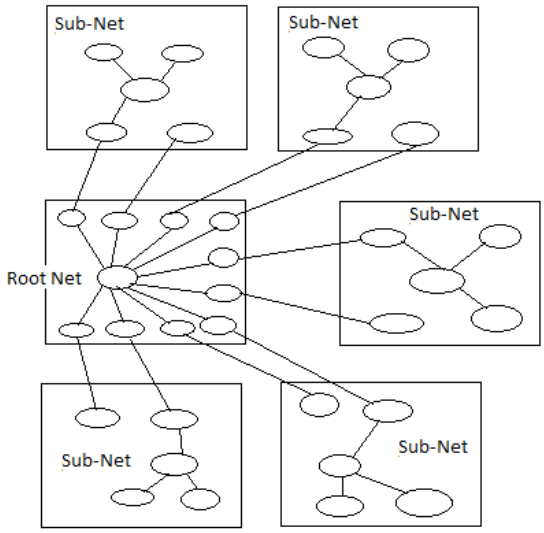

Fig. 1. Sub-networks in the proposed system

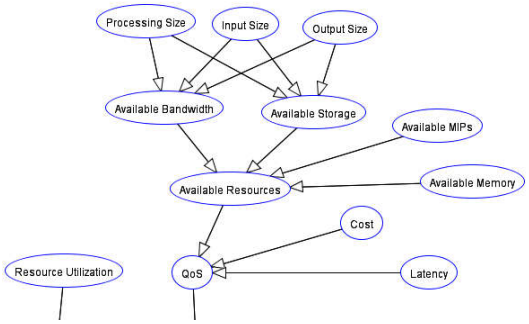

Fig. 2. Sub-network nodes in the proposed model

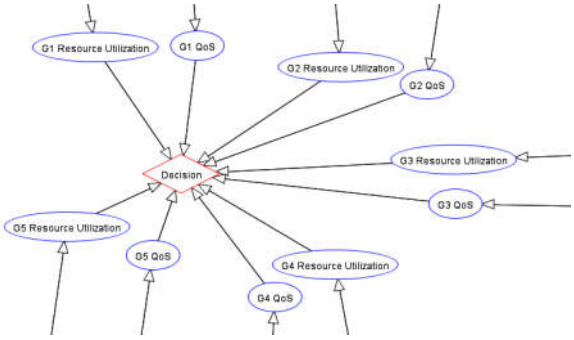

Fig. 3. Root-sub network and interfacing nodes

We mapped each variable to set of constant values. Index of each constant represents the discrete value as shown in Table I. This approach is giving much simplicity in working with the hosts and VMs attributes. In addition, we have sat the initial distribution with equal probability for variable events. Algorithm 1 shows the map function of VM attributes to probability from the probability distribution of the domain.

TABLE I. MAPPING OF THE PROPOSED SYSTEM RANDOM VARIABLES

Random Variable	Mapping	Mapped Values
Input Size	Small, Medium, Large	0, 1, 2
Output Size		
Processing Size		
Cost	Low, Mid, High	0, 1, 2
Latency		
QoS	None, Low, Mid, High	0, 1, 2, 3
Resource Utilization	High, Mid, Low	0, 1, 2
Available Bandwidth	Available, Not Available	0, 1
Available MIPs		
Available Storage		
Available Memory		
Available Resources		

Algorithm 1. Pseudo-code for mapping V_j attribute to a probability

```
01: probability ← 0
02: index ← get mapped value from Vj attribute or -1 if not found
03: if index ≠ -1 then
04:     probability ← probabilityMap [index]
05: end if
```

When a request is received to get a VM assigned to a host, a probability of each group is calculated based on (3). The group agent with the highest probability (belief) will take the assignment of the VM. If the group agent failed to get the VM assigned, then the group agent with the second highest probability is assigned to handle the VM request and so on. Algorithm 2 shows the logic of selecting the sub-network with the highest probability and then getting the probability distribution updated based on the selection of the success sub-network. Variables Input Size, Output Size, Processing Size, Available MIPs, Available Memory, Cost, and Latency probability are challenged against the VM attributes. While Resource Utilization is picked and mapped to a probability from the sub-domain agent.

Adjustment of probabilities are happening as follow:

For group g_k, $$adj_g = r * f * u_g \quad (4)$$

For other groups $g_{l\,(l\neq k)}$, $$adj_l = -1 * r * f * u_l \quad (5)$$

where r is the learning rate to adjust the probabilities (selected as *0.05*), f is used to avoid over fitting (selected as *0.1*), and u_g is the current utilization of the group g. If g is the group with success assignment of V_j (g_k), then the adjustment will be to increase the probability of believing in such VM attributes in future. If g is from the other set of groups ($g_{l\,(l\neq k)}$) then the adjustment will be to reduce the probability of variables to not to accept the assignment of such VM like attributes in future.

Algorithm 2. Pseudo-code for meta-scheduling V_j

```
01: for each sub-network do
02:     bid QoS and Resource Utilization probabilities of V_j assignment (3)
03: end for
04: propagate probabilities to the root sub-net
05: state ← fail
06: for number of groups do
07:     g ← group with highest probability at utility node in root sub-net
08:     get the V_j assigned to Group g
09:     state ← group assignment result
10:     if state = success then
11:         exit for
12:     end if
13: end for
```

IV. Experiments and Discussion

In this section, some experiments are executed on CloudSim to evaluate our proposed model. CloudSim provides a data center based virtual technology, modeling and simulation capabilities including resource monitoring, and host to VM mapping capabilities. We created an extension to CloudSim VM allocation policy in Java with our proposed meta-scheduling model to get the VM assigned to a site with best fit resources and load. We've created five groups with set of different hosts attributes as shown below in Table II, Host Groups: *{G1, G2, G3, G4, G5}*. Then, we got the resource attributes categorized as shown in Table III. After that, cost of each attribute is shown in Table IV.

TABLE II. Resources Specifications of Each Group of Hosts

Host Group	MIPs	RAM	Bandwidth	Storage
G1	1x1.8GHz	1x2048MB	10Gbit/s	1TB
G2	2x2.0GHz	2x2048MB	10Gbit/s	2TB
G3	4x2.3GHz	3x2048MB	10Gbit/s	3TB
G4	8x2.6GHz	4x2048MB	10Gbit/s	4TB
G5	8x2.9GHz	8x2048MB	10Gbit/s	5TB

TABLE III. Resources Mapping to Categories

	Small	Medium	Large
Input Size	0.5GB	1.5GB	3GB
Output Size	64MB	0.75GB	2GB
Processing Size	1GB	1.5GB	2GB

	Low	Mid	High
Cost	$1.0	$2.5	$5.0
Latency	0.01	0.03	0.05

	None	Low	Mid	High
QoS (probability)	0.0	0.2	0.4	0.5

	High	Mid	Low
Resource Utilization (# of VMs assigned)	50	150	250

TABLE IV. Cost of Resources at Each Group of Hosts

Group	Processing (per second)	RAM (per GB)	Storage (per GB)	Bandwidth (per GBit)
G1	$0.3	$0.5	$0.2	$0.01
G2	$0.6	$1.0	$0.4	$0.02
G3	$0.9	$1.5	$0.6	$0.03
G4	$1.2	$2.0	$0.8	$0.04
G5	$1.5	$2.5	$1.0	$0.05

In our experiments we chose to setup 100 hosts per each group. Then, we randomly generated set of 1000, 2000, 3000, 4000, and 5000 VMs to be assigned to hosts in each of the groups and check at the end of the simulation the metrics of the proposed model. The attributes of the random VMs are in these ranges: MIPs: {1GHz, 2GHz, 3GHz}, MIPs Number: {1, 2, 3, 4}, RAM: {1GB, 2GB …. 16GB}, Bandwidth: {1GBit/s}, Storage: {1GB, 2GB …. 12GB}, and Latency: {0.01, 0.02, 0.03, 0.04}.

To validate our model, we ran the original CloudSim scheduling and our proposed model. Fig. 4, 5 and 6 show the results of both scheduling approaches in terms of allocated MIPs, assigned memory and number of successfully assigned VMs. Based on the numbers shown, we have validated our proposed technique to the one used in CloudSim as they are getting almost the same numbers.

Fig. 4. Allocated MIPs, CloudSim Scheduling vs. Proposed Meta-Scheduling

Fig. 5. Assigned RAM, CloudSim Scheduling vs. Proposed Meta-Scheduling

Fig. 6. Assigned VMs, CloudSim Scheduling vs. Proposed Meta-Scheduling

In our proposed model, we have modeled extra VM attributes not found in CloudSim like latency, resource utilization, and QoS. These additional attributes have been considered as input to our next contribution of improving the process of VM assignment and time needed for that. The default scheduling technique and our proposed meta-schedulers are executed to show the improvement in number

of trails required to assign VMs to available hosts. The numbers are shown in Fig. 7. We started training the MSBN with 5000 VMs, then we got the testing phase of the VMs requested for assignment. The source of improvement is that the VM request is directed to the group of hosts that has the highest probability of the VM to be assigned and ignore other groups. Groups with higher load of VMs assigned will provide lower probability even the group has available resources. The figures shows improvement of getting the VM assigned to a host instead of scanning all hosts to find a suitable one with available resources. Trails needed to assign a VM to a group of hosts is marked as first trial to fifth trail. If VM is failed to be assigned to any of the groups then it is marked as failed. The improvement we got is around 40% improvement in search time for a suitable host to get the VM assigned to.

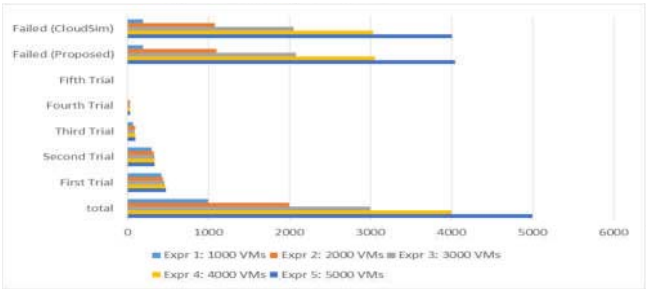

Fig. 7. Trials and time needed to get VM assigned to host

V. Conclusion and Future Work

Due to the huge demand for the cloud applications, it is becoming a critical issue to efficiently manage resources according to users' requests while satisfying the QoS agreement. Furthermore, cloud resources heterogeneity and the uncertain workload pattern complicates the resource management process further more. We proposed a Multiply-Sectioned Bayesian Network for Multi-Agent Learning Based Meta-Scheduling for resources. We selected CloudSim as the simulation tool and extended the scheduling engine with the proposed meta-scheduling model. Our proposed system model is considered as SLA-based, utility based and market based resource scheduler. In addition, our proposed system model is adaptable to the load of VMs. We ran set of experiments with different number of hosts and VMs. Our results show that, while the VMs demand from users have increased competition, in general the system provided an increased ability to satisfy QoS demands over all users. The results presented improved time in finding the group of hosts that the best set of resources resides. In future work, we are looking for supporting VM migration policies and cloudlets meta-scheduling. Furthermore, we need to use a prediction method to predict the workload and get the assignment process changed accordingly.

References

[1] R. Buyya, R. Ranjan, and R.N. Calheiros, "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim toolkit: Challenges and opportunities," in the 7th High Performance Computing and Simulation Conference, IEEE, p. 1-11, 2009.

[2] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," in Grid Computing Environments Workshop, GCE'08, IEEE, pp. 1-10, 2008.

[3] R. Sudeepa and H.S. Guruprasad, "Resource Allocation in Cloud Computing," in International Journal of Modern Communication Technologies & Research (IJMCTR), vol. 2, issue 4, pp. 19-21, 2014.

[4] C. Zhang, V. Lesser, and P. Shenoy, "A Multi-Agent Learning Approach to Online Distributed Resource Allocation," in the 21st International Joint Conference on Artificial Intelligence, vol. 1, pp. 361-366, 2009.

[5] S. Sotiriadis, N. Bessis, F. Xhafa, and N. Antonopoulos, "From meta-computing to interoperable infrastructures: A review of meta-schedulers for HPC, grid and cloud," in the 26th International Conference on Advanced Information Networking and Applications, pp. 874-883, 2012.

[6] P. Sareen and T.D. Singh, "Simulation of Cloud Computing Environment using CloudSim," in International Journal of Emerging Technologies in Engineering Research (IJETER), vol. 4, issue 12, article no. 9, 2016.

[7] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," in Software: Practice and experience, vol. 41, issue 1, pp.23-50, 2011.

[8] A. Yousafzai, A. Gani, R. Md. Noor, M. Sookhak, H. Talebian, M. Shiraz, and M.K. Khan, "Cloud resource allocation schemes: review, taxonomy, and opportunities," in Knowledge and Information Systems, vol. 50, issue 2, pp. 347-381, 2017.

[9] V. Vinothina, R. Shridaran, and G. Padmavathi, "A survey on resource allocation strategies in cloud computing". in International Journal of Advanced Computer Science and Applications, vol. 3, issue 6, pp. 97-104, 2012.

[10] P.T. Endo, A.V. de Almeida Palhares, N.N. Pereira, G.E. Goncalves, D. Sadok, J. Kelner, B. Melander, and J.E. Mangs, "Resource allocation for distributed cloud: concepts and research challenges," in IEEE network, vol. 25, issue 4, pp. 42-46, 2011.

[11] S. Majumdar, "Resource management on cloud: handling uncertainties in parameters and policies," in CSI Communications, vol. 35, issue 2, pp. 16-19, 2011.

[12] S. Alnajdi, M. Dogan, and E. Al-Qahtani, "A Survey on Resource Allocation in Cloud Computing," in International Journal on Cloud Computing: Services and Architecture (IJCCSA), vol. 6, issue 5, article 1 , 2016.

[13] M. Xu, W. Tian, R. Buyya, "A survey on load balancing algorithms for virtual machines placement in cloud computing," in Concurrency Computat: Practice and Experience, vol. 29, issue 12, e4123, 2017.

[14] S. Sotiriadis, N. Bessis, and N. Antonopoulos, "Towards inter-cloud schedulers: A survey of meta-scheduling approaches," in International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), IEEE, pp. 59-66, 2011.

[15] K. Chandrasekaran and U. Divakarla, "Load Balancing of Virtual Machine Resources in Cloud Using Genetic Algorithm," in National Institute of Technology KarnataNa, SurathNal, pp. 156–168, 2013.

[16] Z. Niu, S. Tang, and B. He, "An Adaptive Efficiency-Fairness Meta-scheduler for Data-Intensive Computing," in IEEE Transactions on Services Computing, vol. pp, issue 99, pp. 1-14, 2016.

[17] A. Iosup, T. Tannenbaum, M. Farrellee, D. Epema, and M. Livny, 'Inter-operating grids through delegated matchmaking', in Scientific Programming, vol. 16, vol. 2, pp.233-253, 2008.

[18] M.D. De Assuncao, R. Buyya, and S. Venugopal, "InterGrid: A case for internetworking islands of Grids," in Concurrency and Computation: Practice and Experience, vol. 20, issue 8, pp. 997-1024, 2008.

[19] S. Sotiriadis, N. Bessis, P. Kuonen, and N. Antonopoulos, "The inter-cloud meta-scheduling (ICMS) framework," in 27th International Conference on Advanced Information Networking and Applications (AINA), IEEE, pp. 64-73, 2013.

[20] Y. Huang, N. Bessis, P. Norrington, P. Kuonen, and B. Hirsbrunner, "Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm," in Future Generation Computer Systems, vol. 29, issue 1, pp. 402-415, 2013.

[21] V. Subramani, R. Kettimuthu, S. Srinivasan, and P. Sadayappan, "Distributed job scheduling on computational grids using multiple simultaneous requests," in 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), pp. 23-26, 2002.

[22] Y. Zhaofeng and F. Aiwan, "Ant colony Algorithm based on Three Constraint Conditions for Cloud Resource Scheduling," in International Journal of Grid and Distributed Computing, vol. 9, issue 7, pp.189-200, 2016.

[23] Y. Kessaci, M. Nouredine, E.G. and Talbi "A Pareto-based Metaheuristic for Scheduling HPC Applications on a Geographically Distributed Cloud Federation," in Journal of Cluster Computing, Springer, vol. 16, issue 3, pp. 451-468, 2012.

[24] R.P. Prado, J. Braun, J. Krettek, F. Hoffmann, S. García-Galán, J.E. Muñoz Expósito, and T. Bertram, "Gaussian mixture models vs. fuzzy rule-based systems for adaptive meta-scheduling in grid/cloud computing," in Management Intelligent Systems, vol. 171, pp. 295-304, 2012.

[25] Y. Xiang and H. Geng, "Distributed equipment monitoring and diagnosis with multiply sectioned bayesian networks," in AAAI spring symposium on AI in equipment service maintenance and support, pp. 18-25, 1999.

[26] Y. Xiang and V. Lesser, "On the role of multiply sectioned Bayesian networks to cooperative multiagent systems," in IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol. 33, issue 4, pp. 489-501, 2003.

[27] S. Das and R. Ascano, "Distributed Belief Propagation in Multi-Agent Environment," in International Conference on Practical Applications of Agents and Multi-Agent Systems, pp. 53-65, 2015.

[28] Y. Xiang, "Belief updating in multiply sectioned Bayesian networks without repeated local propagations," in International Journal of Approximate Reasoning, vol. 23, issue 1, pp. 1-21, 2000.