Research Article

Open Access

# A multi-agent based load balancing system in IaaS cloud environment

**Abstract**

Infrastructure as a service (IaaS) is a type of cloud computing in which a third-party provider hosts virtualized computing resources over the Internet for executing tasks in the cloud computing. Whenever some VMs are overloaded and some VMs are under loaded, this situation may cause to SLA violation and leads to the reduction of customer satisfaction level and further affects the cloud provider leading to penalty. However, in this study, we propose a Multiple Agent-based Load Balancing Algorithm (MA) in which shift the load in the IaaS to achieve well dynamic load balancing across virtual machines for maximizing the utilization. The proposed algorithm with regard to changing environment and characteristics of the VMs, perform both of sender-initiated and receiver-initiated approach to balances the load of an IaaS in such a way that the amount of waiting time of the tasks in the queue is minimal and at the same time the SLA is guaranteed. We have compared the proposed algorithm with existing load balancing and scheduling algorithms via simulation. The simulation results show that the proposed algorithm is more effective and there is a good improvement in the load-balance, response time and makespan.

**Keywords:** cloud computing, load balancing, scheduling, mobile agents

Volume 1 Issue 1 - 2016

Sina Keshvadi,[1] Behnam Faghih[2]

[1]Department of IT, Payame Noor University (PNU), Iran
[2]Department of Technical and Vocational University, Iran

**Correspondence:** Sina Keshvadi, Department of IT, Payame Noor University (PNU), Tehran, Iran,
Email Keshvadi@pnu.ac.ir

**Received:** August 27, 2016 | **Published:** October 05, 2016

**Abbreviations:** IAAS, infrastructure as service; VMM, virtual machine monitor; PAAS, platform as a service; SAAS, software as a service; VMS, virtual machines

## Introduction

As defined by NIST, Cloud Computing is "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g, networks, servers, storage applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". Cloud Computing is a broad term that describes a broad range of services. In these systems, resources are provided as service on-demand. The services provided by cloud computing are infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) that are made available as pay-as-you-go model to subscribers, and it guarantees to them that it sticks to the Service Level Agreement (SLA). IaaS usage allows the enterprise IT or web operations group to install and build their application environments on infrastructure without the pain of purchasing, installing and ongoing maintenance of the infrastructure.[1]

Moreover, day by day subscribers' requires are rising for computing resources and their needs have dynamic heterogeneity and platform irrelevance in IaaS. This make datacenters expensive to maintain which is accompany with wasted energy and floor space, low resource utilization, and significant management overhead.[2] The virtualization technology come into sight, provides cloud datacenters have become more flexible, more secure, and provide better support for on-demand allocation and it address the issue of heterogeneity and platform irrelevance in a better way, and at the same time the SLA is guaranteed. With IaaS, the physical infrastructure of IT systems is typically moved offsite and a service provider grants access to a virtualized environment of computing resources to its customers. In these systems, cloud datacenters should have ability to migrate an application from one set of resources to another in a non-disruptive manner. Such agility becomes a key in modern cloud computing infrastructures that aim to efficiently share and manage extremely large data centers.[2]

The processing units in cloud virtualization environments are called as virtual machines (VMs).[3] In order to improve resource utility, resources must be properly allocated and load balancing must be guaranteed. Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload or sitting idle of any single resource. Therefore, how to schedule VM resources to realize load balancing in cloud computing and to improve resource utility becomes an important research point. In this case, it is the responsibility for the scheduler to balance the loads across the machines in IaaS. In the absence of load balancing provision, efficiency of some overloaded virtual machines can sharply degrade at times, leading to violation of SLA.[4] The violation of the SLA is a major aspect of IaaS and the violation leads to the reduction of customer satisfaction level and further affects the cloud provider leading to penalty.

According to the whole information of each VM in an IaaS, the performance will be managed and enhanced. There are several methods can monitor and collect the relevant information of node that includes broadcasting, the centralized polling and agent.[5] The agent mechanism is used to collect the related node information to achieve efficient utilization resource and enhance work efficiency. It has inherent navigational autonomy and it could collect related information of each VM, such as CPU utilization, remaining CPU capability, remaining memory, transmission rate, etc. However, in this study, multiple agents are used to gather the related information in IaaS environment and reduce the resources wasting and cost. We proposed a load balancing algorithm in cloud IaaS environment

Submit Manuscript | http://medcraveonline.com

© 2016 Keshvadi et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and build upon your work non-commercially.

by using agents that also the distinctive characteristics of virtual machines are considered.

The rest of this paper is organized as follows:

1. A review of the related work is discussed in Section.
2. The proposed algorithm is detailed in Section.
3. Section.
4. Describes the simulation setup and the results. Finally, Section.
5. Concludes this study.

## Related work

There are basically two kinds of load balancing techniques: static and dynamic.[6] Static algorithms are mostly suitable for homogeneous and stable environments and can produce very good results in these environments. However, they are usually not flexible and cannot match the dynamic changes to the attributes during the execution time. In the static approach, the task allocation does not change during execution while in the dynamic it does. A good comparison of these techniques is mentioned in.[7] Dynamic load balancing algorithms are advantageous over static algorithms. But to gain this advantage, we need to consider the additional cost associated with collection and maintenance of the load information.

A dynamic load balancing algorithm has three main components.[6] The information, transfer, and location strategies. Information strategy is responsible for collecting information about nodes in the system. Transfer strategy selects a job for transfer from a local node to a remote node. Location strategy selects a destination node for a transferred task. Two main issues concerning load balancing activity that depend on the transfer strategy employed are: when is the right time to start it and what jobs are subjected to it. Two approaches are commonly used to start the load balancing activity: the time a new job arrives or is created at a node and the time a finished job departs from a node. Algorithms which make load balancing decisions at the arrival or creation of a new job are referred to as sender-initiated, while algorithms which make load balancing decisions at the departure of a finished job are referred to as receiver-initiated.[6,8] Discusses the different qualitative metrics or parameters like performance, scalability, associated overhead etc.[2] Introduces online load balance scheduling for Cloud data centers. They propose an online resource scheduling algorithm (OLRSA), which considers real-time and multidimensional resource scheduling. Then Lowest Integrated-load First Algorithm (LIF) is also introduced.[4] Proposes an Autonomous Agent-Based Load Balancing Algorithm (A2LB) which provides dynamic load balancing for cloud environment by using agents. They introduced a fitness value metric in which the fitness values give the status of a virtual machine.

As cloud task scheduling is an NP-hard optimization problem.[9] Recently numerous nature inspired models have received a lot of research attentions. A good task scheduler should adapt its scheduling strategy to the changing environment and the types of tasks Hu et al.[10] Proposed genetic algorithm based scheduling mechanism for load balancing among virtual machines Xu et al.[11] Introduced a model for load balancing in the public cloud by using game theory.[12] Proposed a load balancing technique for cloud computing environments based on the behaviour of honey bee foraging strategy. The tasks removed from these VMs are treated as honey bees, which are the information updaters globally. Their work also considers the priorities of the tasks. Load Balancing can also be classified in centralized and distributed load balancing or in application-level and system-level load balancing. In the next part, we propose our mechanism to load balance in cloud IaaS datacenters.

## Proposed algorithm

Whenever a VM becomes overloaded, the service provider has to distribute the load in such a manner that the available resources will be utilized in a proper manner and load at all the virtual machines will remain balanced. Identifying when it is best to migrate an application in an overloaded virtual machine to another place has a direct impact on resource utilization. This issue can be best achieved by an efficiently monitoring the utilization of computing resources. Agent is an autonomous unit, which is capable of performing specified tasks on their own. In follow, we introduce our agents.

### Agents

Our proposed mechanism comprises of three agents: Virtual Machine Monitor (VMM) Agent, Datacenter Monitor (DCM) Agent and Negotiator Ant (NA) Agent. VMM and DCM agents are generally agents whereas NA agent is an ant, which is a special category of mobile agents. Since load balancing in IaaS cloud computing systems would require searching for under loaded servers and resources, ant agents suit the purpose and fulfil it appropriately without putting additional burden on network.

### VM monitor agent (VMM agent)

To enhance the monitoring services, each virtual machine in IaaS is supported with a Virtual machine Monitor agent that we call it VMM Agent. VMM collects the CPU, memory and bandwidth utilization of individual virtual machine hosted with different types of tasks to monitor the load. This agent is supported with a table named as VM Local_ Table 1. VMLocal_Table store the state of a vm. Where load satus is categorized in under loaded, normal and overloaded statuses. Each VMM agent uses its VMLocal_Table to monitor its load periodically and determines the load status.

Load of a VM can be calculated as follow[13]:

$$Load\,(\%)=\frac{CPU_{\mu}+MEM_{\mu}+NET_{\mu}}{3}\times 100 \quad (1)$$

Where $CPU_{\mu}$, $MEM_{\mu}$, $NET_{\mu}$ are average utilization of CPU, memory, network bandwidth during each observed period, respectively. VMM agents use eq. 1 to determine the load of associated virtual machine.

### Datacenter monitor agent (DcM agent)

This agent monitors all VMM agents in a datacenter. Actually, it performs information policy in a datacenter by monitoring the VMM's information. This agent is supported with table termed as DcMonitor_table. This table maintains all information about status and characteristics of all virtual machines in a datacenter. It categorizes VMs based on their characteristics. A simple template of a DcMonitor_table is shown as below (Table 2).

**Table 1** VM Local table store the state of a vm

| VM ID | RAM | CPU | BW | OS | Load status |
|---|---|---|---|---|---|
| $VM_1$ | $\mu_1$ | $\lambda_1$ | $\gamma_1$ | Type 1 | Status $VM_1$ |
| $VM_2$ | $\mu_2$ | $\lambda_2$ | $\gamma_2$ | Type 2 | Status $VM_2$ |
| $VM_n$ | $\mu_n$ | $\lambda_n$ | $\gamma_n$ | Type n | Status $VM_n$ |

**Table 2** Dc Monitor table keeps the information about vms

| Status | Under-loaded; balanced; over loaded | List of VMs |
|---|---|---|
| OS | OS Type 1; OS type 2; OS Type n | List of VMs |
| Number of PE | Number 1; Number 2; Number p | List of VMs |
| Ram | Ram Type 1; Ram type 2; Ram Type m | List of VMs |
| CPU | Type 1 GHz; type 2 GHz; Type m GHz | |
| Bandwidth | Type 1 M; Type 2 M; Type m M; | |

### Negotiator ant agent (NA)

These agents are initiated by DCM agents. It will move to other datacenters and communicate with DCM agent of that datacenter to enquire the status of VMs present there, looking for the desired configuration. On receiving the required information, it communicates the same to its parent DCM agent. Afterwards, it will stay at destination location, waiting for self-destroy message from parent. The status of the NA agent may be alive or destroyed.

## MA load balancing algorithm scenario

(Figure 1) Provides high level view of proposed mechanism we should first be aware of the load status of any VM. VMM agents determine own VM's state based on its load by eq. 1. There are overloaded, under loaded and balanced states. If the load of a VM is more than the upper threshold ($\sigma$) then the system is overloaded. If the load is less than a lower threshold ($\nu$), the system is under loaded. Otherwise the system is in a balanced state. In the overloaded state a heavily loaded vm is the one who initiates load balancing. Hence, this implies that this strategy is coupled with a sender-initiated transfer strategy. In the under loaded state a lightly loaded vm is the one who initiates load balancing. Hence, this implies that this strategy is coupled with a receiver-initiated transfer strategy (Figure 1). MA load balancing scenario after finding the state of a vm as overloaded, VMM send a load balancing request to its DCM agent. DCM look ups its DcMonitor_table to find an under loaded VM with desired configuration. If found, the task moved from overloaded VM to under loaded VM to perform. If not, DCM agent initiates some NA agents per each datacenter in the system to communicate with other datacenters. The NAs go straightforward to the given datacenter and request the desired VM from their DCM agents. If not found, NA send back an unsuccessful TTL-based message to its parent and wait for destroy message. In the case that the desired VM found, the characteristics of the VM send back to its DCM agent and wait for parent message.

In the other hands, DCM agent receives reports from its NAs. If DCM agent don't receive message, resend another NA agent to the datacenter. Then, it sends destroy message to any NA which couldn't found a desired VM. If all received message are negative, send a negative reply to requested VM in which the load balancing is not feasible. Otherwise, it select the appropriated VM from all received options based on the load and response time and send a transfer message to that NA and a destroy message to all other ants. Now, it transfers task from requested VM to given VM and after receive new load status, DCM agent will update the DcMonitor_table. Algorithms of various agents deployed in proposed framework are given as follow (Figure 2). VMM, DCM and NA Agents algorithms.

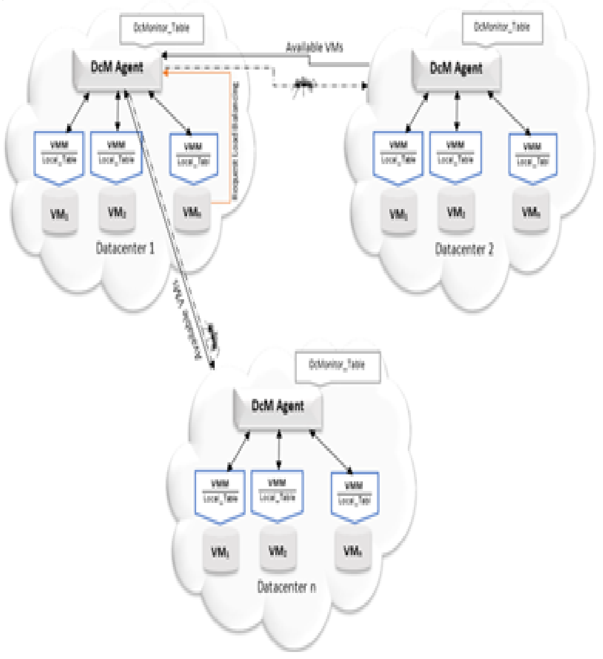

**Figure1** MA load balancing scenario.

Copyright: ©2016 Keshvadi et al.

**VMM agent algorithm**

```
locally calculate the load of own VM based on equation (1)
if (load > upper-Threshold)
 status=overloaded;
if (load< lower-threshold)
 status=under_loaded;
else
 Status=load_balanced;
if(load_Changed(Status))
 request_DcMA (status, characteristics-list[], load)
```

**DcM agent algorithm**

```
Accept load-state from VMMA_i
case Overloaded:
{
 Update vm_i status
 Look-up (DcTable, under_loaded_VM[], desired configuration)
 if found {
Transfer task from overloaded to under-loaded
Receive new status of both
Update Dctable}
 else {
 for j=2 to number_of_Datacenter do
 initiate NA_j (desire configuration, datacenter)
 if (negotiation_ result != feasible)
 send(Self destroy)
 NA_j.statuse = destroyed
 else
 Add information and response time to temp-table
 destination = SelectBestFit(temp_table, desire configuration)
 Transfer task form VM_i to destination)
 Receive new status of both
 Update Dctable
 for each NA that status==alive
 send(Self_destroy)
}
case Under_loaded:
{
 Look-up (DcTable, over_loaded_VM[], desired configuration)
 if found
 Transfer task from one overloaded_VM to under-loaded_VM
 Receive new status of both
 Update Dctable
}
NA Agent Algorithm
```

**NA agent**

```
Receive VM_configuration and Datacenter_ID from DcMA
Search_datacenter(Datacenter_ID)
ask (VM_configuration)
if (found)
 reply(feasible)
else
 reply(!feasible)
waitForDestroyMessage()
destroy()
```

**Figure 2** VMM, DCM and NA agents algorithms.

## Simulation result

In this section, we will show simulation results for our proposed algorithm i.e. MA load balancing algorithm compared with other existing algorithms. Experimenting new techniques or strategies in repeatable, dependable, and scalable environments using real-world Cloud environments is not practically possible as such experiments will compromise the end users Qos requirements like security, cost, and speed. There is a need for a good simulator for experimental purposes. One such a simulator is CloudSim.[14,15] This simulator is a generalized simulation framework that allows modeling, simulation and experimenting the cloud computing infrastructure and application services. We have extended the classes of CloudSim simulator to simulate our algorithm. Our agents are implemented using java programming. All simulations are collected using a core-i3 pc with 2.4 GHz CPU and 4 GB memory. We compare the simulation results of our proposed algorithm with different low and over loaded ratios with these existing algorithms:

**First in first out (FIFO):** a general scheduling algorithm that implements Fisrt-In-First-Out policy to allocate the VM requests to the PM that can provide the resource required.

**Weighted round robin (WRR):** a common routing policy offered in cloud load balancers, which allocates the task to each VM in a sequentially turn.

**List scheduling algorithm (LS):** a generic greedy algorithm. Whenever a machine becomes available, process any unprocessed job.

### Measurement parameters

In this paper we study three parameters: makespan, response time and, degree of imbalance. These parameters are used to evaluate the effect of load balancing approach. In the following, we introduce them briefly and then we study the impact of algorithms on these parameters. Makespan can be defined as the overall task completion time. We denote completion time of task $T_i$ on $VM_j$ as $CT_{ij}$. Hence, the makespan is defined as the following function.[12]

$$makespan = \max\{CT_{ij} \;\; i \in T \;, i = 1, 2, \ldots, n \; and \; j \in VM \;, j = 1, 2, \ldots, m\} \quad (2)$$

Response time: Response time is the amount of time taken between submission of a request and the first response that is produced. The reduction in waiting time is helpful in improving responsiveness of the VMs.

Degree of imbalance: Imbalance can be calculated as standard deviation of the load:

$$\sigma = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (PT_i - PT)^2}$$

Where $PT_i$ is processing time of $VM_i$:

$$(PT_i) = \frac{load \; vM_i}{Capacity_i}$$

### Make span

Figure 3 illustrates the comparison of makespan between MA, FIFO, WRR and LS Algorithms. The X-axis represents number of tasks and the Y-axis represents the Makespan. MA is more efficient and has a lower Makespan when compared with other three algorithms.

### Response time

Figure 4 illustrates the response time of VMs in seconds for MA, FIFO, WRR and LS Algorithms. The X-axis represents number of tasks and the Y-axis represents time in seconds. It is evident that MA is more efficient compared with other three methods.

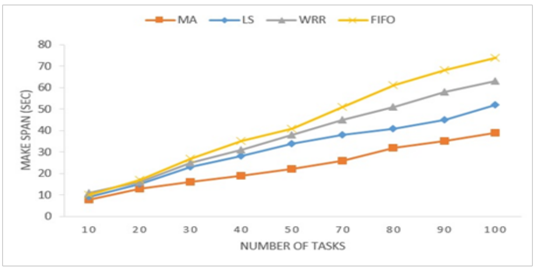

**Figure 3** Comparison of makespan.

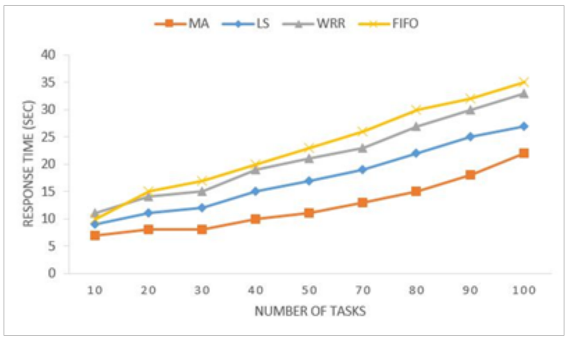

**Figure 4** Comparison of response time.

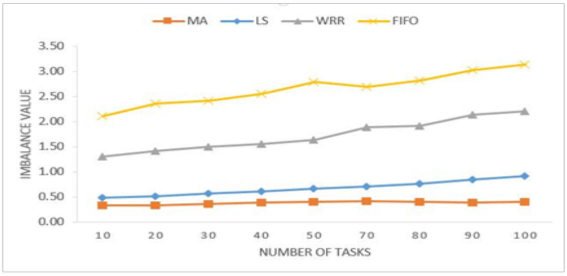

**Figure 5** Comparison of degree of imabalance.

### Degree of imbalance

Figure 5 shows the comparison of degree of imbalance between MA, Random, WRR and LS Algorithms. The X-axis represents number of tasks and the Y-axis represents the degree of imbalance. It is clearly evident that by load balancing with MA, the degree of imbalance is greatly reduced.

## Conclusion

Rapid growth in number of cloud users has raised demand for load balancing mechanisms. It is essential to monitor the IaaS to avoid over and under estimation of resource levels and guarantee the SLA. We have introduced a dynamic and integrated resource scheduling algorithm for IaaS Cloud datacenters that presents a scheduling strategy on VM load balancing by using multiple monitor and mobile agents. In this way, the method achieves the best load balancing and reduces or avoids dynamic migration thus resolves the problem of load imbalancing and high migration cost caused by traditional scheduling algorithms. The experimental results show that this method can better realize load balancing and proper resource utilization.

## Acknowledgements

None.

## Conflict of interest

The author declares no conflict of interest.

## References

1. Meeraa A, Swamynathan S. Agent based resource monitoring system in iaas cloud environment. *International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA)*. 2013;10:200–207.

2. Tian WH, Zhao Y, Zhong YL, et al. Dynamic and integrated load-balancing scheduling algorithms for cloud data centers. *Cloud Computing and Intelligence Systems (CCIS)*. 2011;8(6):117–126.

3. Ding Y, Qin X, Liu L, et al. Energy efficient scheduling of virtual machines in cloud with deadline constraint. *Quality of Service in Grid and Cloud Netherlands*. 2015;50(c):62–74.

4. Singha A, Junejab D, Malhotra M. Autonomous agent based load balancing algorithm in cloud computing. *International Conference on Advanced Computing Technologies and Applications India*. 2015;45:832–841.

5. Wang SC, Yan KQ, Liao WP, et al. Towards a load balancing in a three-level cloud computing network. in proc, 3rd International Conference on. *Computer Science and Information Technology (ICCSIT)*. 2010;1:108–113.

6. Alakeel A M. A guide to dynamic load balancing in distributed computer systems. *International Journal of Computer Science and Network Security (IJCSNS)*. 2010;10(6):153–160.

7. Janhavi AB, Surve SK, S Prabhu. Comparison of load balancing algorithms in a grid. *Data Storage and Data Engineering (DSDE), International Conference*. 2010;20–23.

8. Sreenivas V, Pratha M, Kemal M. Load balancing techniques: Major challenge in cloud computing - a systematic review. Electronics and Communication Systems (ICECS), International Conference; Africa: IEEE; 2014. p. 1–6.

9. Tawfeek MA, El Sisi A, Keshk AE, et al. Cloud task scheduling based on ant colony optimization. Computer Engineering & Systems (ICCES). 8th International Conference; Egypt,: IEEE; 2013. p. 64–69.

10. Hu J, Gu J, Sun G, et al. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. *Proc PAAP*. 2010. p. 89–96.

11. Moradi M, Dezfuli MA, Safavi MH. *A new time optimizing probabilistic load balancing algorithm in grid computing IEEE*. Iran: Springer; 2010.

12. Dhinesh Babu LD, Krishna PV. Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Applied Soft Computing*. 2013;13(5):2292–2303.

13. Wood T, Prashant S, Arun V, et al. Black-box and Gray-box Strategies for Virtual Machine Migration in the proceedings of Symp. On Networked Systems Design and Implementation (NSDI) Portland. USA: USENIX Association; 2007. p.1–14.

14. Calheiros RN, Ranjan R, Beloglazov A, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Softw Pract Exper*. 2011;41(1):23–50.

15. Buyya R, Ranjan R, Calheiros RN. *Modeling simulation of scalable cloud computing environments and the cloudsim toolkit: challenges and opportunities*. Australia: University of Melbourne; 2009. p. 21–24.