RESEARCH REVIEW JOURNALS
Volume-04
Issue-06
June-2019

ISSN: 2455-3085 (Online)
RESEARCH REVIEW International Journal of Multidisciplinary
www.rrjournals.com[UGC Listed Journal]

Efficient Load Balancing in Cloud Computing

Joshi Narayan

Professor, Department of MCA, Dharmsinh Desai University, Nadiad, Gujarat (India)

ARTICLE DETAILS

Article History
Published Online: 10 June 2019

Keywords
Cloud computing, load balancing, task migration, resource allocation.

***Corresponding Author**
Email: narayan.joshi@yahoo.com

ABSTRACT

The cloud computing stage has instigated noteworthy job behind the exponential development of present day IT industry. An ever increasing number of associations and people from different areas of society are catalogue or in a roundabout way expending distributed computing stage. Such rising interest for cloud based figuring assets has created necessity of enhanced asset board and burden management. Broad research work is occurring for effective resource and workload allocation in this area. An improved version of the multi-threaded dynamic load balancing technique for cloud computing environment is suggested here. The results obtained by applying the suggested mechanism in open source cloud computing platform indicate noteworthy improvement in workload of overloaded virtual machines after execution of load balancing.

1. Introduction

Cloud computing technology offers high availability, reliability, transparency and flexibility. Apart from the benefits and applications of cloud computing models available, the booming need for diversified cloud computing environments is creating various challenges for stake holders. Cost effective management of cloud computing based services and resources is one of the key factor for delivering efficient cloud based services. Efficient resource management and resource allocation is one of the critical operational task in administration of underlying physical and virtual cloud infrastructure [2].

With help of different approaches, noteworthy research work is being carried out in domain of load balancing in cloud computing platform. Ongoing progressions in cloud load balancing are more towards resource allocation and management in hybrid cloud computing environments. An enhanced load balancing technique of the suggested mechanism is presented here. Section 2 presents the related work in domain of resource allocation and management in cloud computing domain. The proposed mechanism is presented in section 3. Outcomes are presented in section 4.

2. Related Work

A TB-LB approach based technique using clustering and the features of three heuristic algorithms is proposed by R. Sajjan et al. [10]. The technique tries to keep the least makespan. A load balancing technique is suggested by Zhu et al. The technique is based on improved Particle Swarm Optimization [15]. A technique suggested by Vasudevan et al. is based on honey bee theory for assignment of existing resources to cloud network by means of bringing down the service request makspan [11]. An EcoPower algorithm is proposed by Deng et al. The algorithm is for accomplishing improved rules of power and resource management and works according to the Lyapunov optimization [14]. Efficient resource allocation may result into better performance to various cloud computing applications [6].

A load balancing technique suggested by Chien et al. depends on the completion time of processes for the sake of overall performance improvement [8]. For load balancing in cloud computing environment, N. Joshi et. al. have suggested a technique which is based on centralized load balancing aspect. In spite of migrating whole virtual machine, the suggested prototype mechanism is based on shifting of workload among machines in cloud environment [4]. A resource management mechanism based on control-based self-adaptive randomized optimization has been suggested by Papadopoulos [1]. Another technique suggested by Nan S. et al. is based on dynamic placement, optimized annealing and optimized resource utilization [12]. The mechanism operates for betterment of annealing algorithms for efficiently employment virtual machines.

In order to bring down the response time, a resource management strategy has been proposed by Ajit et. al.. The technique gathers workstations' load assignment value. The technique is based on weighted signature for mapping virtual machines to physical machines [3]. In an another load balancing technique which is suggested by Azar et al. is based on f-restricted algorithm and first fit based policy [13]. N. Joshi has proposed resource management techniques for dynamic load balancing for cloud computing environments [5]. Another load balancing technique is proposed by Kapur [9]. The technique works for the on-demand and reserved cloud resources.

Diversified methods have been made functional for efficient utilization of cloud resources and workload sharing. Few techniques are based on shifting of highly loaded virtual machine to other physical machines. Few practices work on shifting of inward requests to other virtual machines having lighter work load. Few techniques also offer load balancing in collaborated cloud computing environments. An enhanced version of load balancing mechanism suggested in [5] and [7] is suggested here. The mechanism is based on shifting of task from highly loaded virtual machine to lightly loaded virtual machine.

3. Mechanism

Two more virtual machine states are introduced here: `UNDERLOAD_PASSIVE` and `OVERLOAD_PASSIVE` for keeping track about the virtual machines which have been designated for load balancing. Moreover, the suggested technique also records the time of designating such virtual machines so that later on such virtual machines may be brought back to the `ACTIVE` state. The efficient mechanism for our load balancer [5] and [7] is presented here. Block diagram is shown in figure 1. The thread `UnderloadVM` works on identification of under loaded virtual machines and maintaining their information in the data structure `qUnderloaded`. On other hand, the thread `OverloadVM` works on identification of overloaded virtual machines and maintaining their information in the data structure `qOverloaded`.

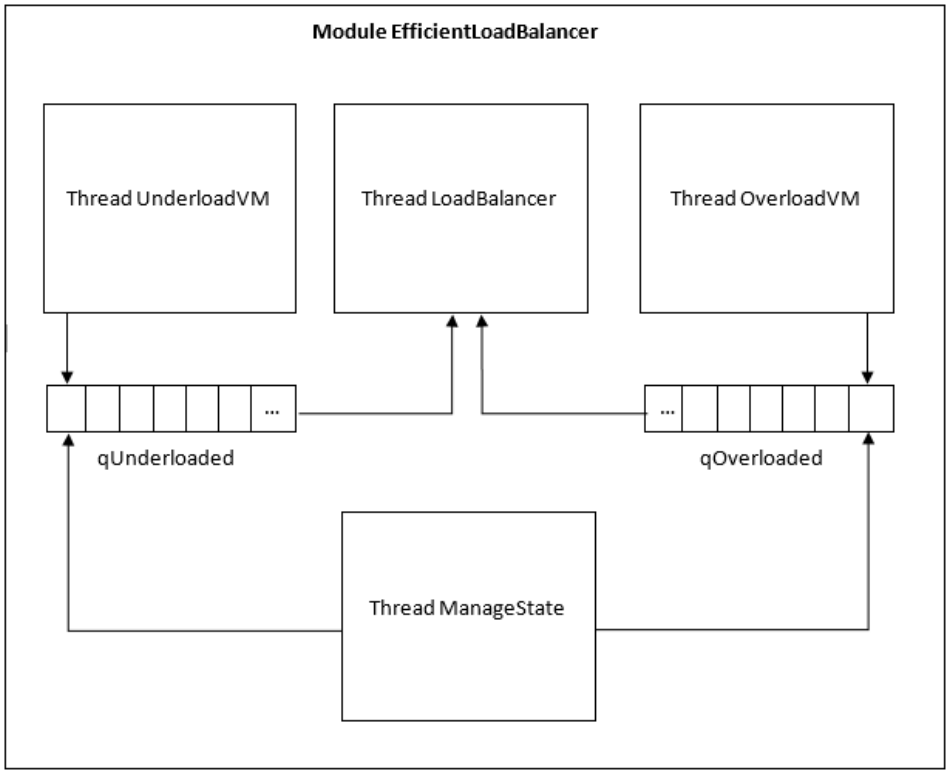

Figure 1: Improved Load balancing module

```
Module EfficientLoadBalancer
{
	int Tu, To;
	enum state {INACTIVE=0, ACTIVE=1, UNDERLOAD_PASSIVE=2, OVERLOAD_PASSIVE=3};
	struct VM // VM's current information
	{
		float load, TOver, TUnder; // workload & threshold values
		unsigned short cores;
		unsigned long passive_set_time,tot_mem,free_mem,bandwidth;
		state vm_state;   // VM's current state
		char vmIP[40];
		…
	};
	Queue<VM*> qUnderloaded, qOverloaded;
	Thread: thread_UnderLoadedVM
	{
		VM* pVM_U; //pointer to underloaded VM
		void fetch_Underloaded_VM() //thread method
		{
			while(true)
			{
				pVM_U = null;
				pVM_U = determine_underloaded_VM();
```

```
                        if(qUnderloaded.find(pVM_U) || pVM_U->load > TUnder)
                              continue;
                        if(!pVM_U)
                        {
                              sleep(Tu);  continue;
                        }
                        qUnderloaded.append(pVM_U);
                  }
            };
      }; //end of thread_UnderLoadedVM
      Thread: thread_OverLoadedVM
      {
            VM* pVM_O; //pointer to overloaded VM
            void fetch_Overloaded_VM() //thread method
            {
                  while(true)
                  {
                        pVM_O = null;
                        pVM_O = determine_overloaded_VM();
                        if(qOverloaded.find(pVM_O) || pVM_O->load < TOver)
                              continue;
                        if(!pVM_O)
                        {
                              sleep(To);  continue;
                        }
                        qOverloaded.append(pVM_O);
                  }
            };
      }; //end of thread_OverLoadedVM
      Thread: thread_LoadBalancer
      {
            VM* pVM_O, pVM_U; //pointers to overloaded and underloaded VMs
            while(true)
            {
                  pVM_O = qOverloaded.fetchAndRemove();
                  if(!pVM_O) continue;
                  if(pVM_O.timeSincePassive()>MaxOverloadThresholdTime ||
                        !pVM_O.isOverloaded() ||
                        pVM_O.state == OVERLOAD_PASSIVE)
                        continue;
                  pVM_U = qUnderloaded.fetchAndRemove();
                  if(!pVM_U) continue;
                  if(pVM_U.timeSincePassive()>MaxUnderloadThresholdTime
                     || !pVM_U.isUnderloaded() ||
                     pVM_U.state == UNDERLOAD_PASSIVE)
                        continue;
                  pVM_U->vm_state = UNDERLOAD_PASSIVE;
                  pVM_O->vm_state = OVERLOAD_PASSIVE;
                  set passive_set_time for pVM_O and pVM_O
                  balance(pVM_O,pVM_U);
                  …
            }
      }; //end of thread_ LoadBalancer
      Thread: thread_ManageState //VM state management
      {
            …
```

```
            if(pVM->state == UNDERLOAD_PASSIVE &&
               pVM->passive_set_time >= Tu)
                  //reset passive_set_time for pVM
                  pVM->state = ACTIVE;
            else if(pVM->state == OVERLOAD_PASSIVE &&
                    pVM->passive_set_time >= To)
                  //reset passive_set_time for pVM
                  pVM->state = ACTIVE;
            …
      }
      void start() //initiate & setup environment
      {
            …
            Initialize Tu and To intervals
            Initialize qOverloaded<VM*>
            Launch thread thread_OverLoadedVM
            Initialize qUnderloaded<VM*>
            Launch thread thread_UnderLoadedVM
            Launch thread thread_LoadBalancer
            Launch thread thread_ManageState
            …
      }
};
```

4. Outcomes and Discussion

Often, it may happen that the virtual machines which were previously designated as participants for load balancing, now at the time of actual load balancing, they may no more be in a position to execute their role. The mechanism presented here takes care about such cases.

Moreover, after certain time intervals, the suggested mechanism keeps on periodical reviewing the state information of virtual machines with help of one more thread of execution `thread_ManageState`. If required, the thread carries out updation in the state information of such virtual machines. The thread helps to protect from starvation.

The thread `thread_ManageState` works in coordination with the threads `thread_LoadBalancer`, `qOverloaded` and `qUnderloaded`. The thread `thread_LoadBalancer` assures that the load balancing must not be burdensome for virtual machines which have already started participating in the load balancing operation. It also helps to stay away from over balancing. The multithreaded mechanism enriched with mutual exclusion features enables its threads to function independently for the sake of overall better performance. Some of the load balancing key results are shown in Table1. For implementation, we have used open source cloud computing platform.

Sr.	Workload Before load balancing		Workload After load balancing	
	Source VM : load	Destination VM : load	Source VM : load	Destination VM : load
1	192.168.10.2 : 89%	192.168.10.9 : 34%	192.168.10.2 : 64%	192.168.10.9 : 61%
2	192.168.10.3 : 90%	192.168.10.1 : 39%	192.168.10.3 : 63%	192.168.10.1 : 59%
3	192.168.10.4 : 88%	192.168.10.8 : 22%	192.168.10.4 : 59%	192.168.10.8 : 44%
4	192.168.10.5 : 84%	192.168.10.9 : 20%	192.168.10.5 : 61%	192.168.10.9 : 41%
5	192.168.10.7 : 91%	192.168.10.1 : 14%	192.168.10.7 : 66%	192.168.10.1 : 38%

Table 1: Workload on virtual machines before and after load balancing

Table 1 shows that the workload of heavily loaded virtual machines comes down after balancing their load with lightly loaded virtual machines.

5. Conclusion

In cloud computing environments, optimized resource allocation and load sharing performs significant job. An efficient load balancing technique is suggested in this paper. Four simultaneously working threads work for providing efficient and hassle free workload sharing among virtual machines in cloud computing platforms. Moreover, the mechanism addresses issues such as instantaneous over burdening and under loading of virtual machines. The technique is flexible about passive machine state for overloaded and under loaded virtual machines.

References

1. V. Papadopoulos, C. Klein, M. Maggio, J. Drango, M. Dellkrantz, F. Hernndez and K. E. Arzn, Control-based load-balancing techniques: Analysis and performance evaluation via a randomized optimization approach. Control Engineering Practice, 52, 2016
2. F. F. Moges and S. L. Abebe, Energy Aware VM Placement Algorithms for the OpenStack Neat Consolidation Framework, Journal of Cloud Computing Advances, Systems and Applications, January, 2019
3. M. Ajit and G. Vidya, VM level load balancing in cloud environment, IEEE Fourth International Conference on Computing, Communications and Networking Technologies, July 2013
4. N. Joshi, Load Balancing in Cloud Computing Using Process Migration, International Journal of Advanced Research in Engineering and Technology, 5(4), April, 2014
5. N. A. Joshi, Dynamic Load Balancing In Cloud Computing Environments, International Journal of Computer Engineering and Technology, 4(3), 2013, pp. 70–76
6. N. A. Joshi, Performance-Centric Cloud-Based e-Learning, The IUP Journal of Information Technology, ISSN 0973-2896, Vol. 10, No. 2, pp. 7-16, June 2014
7. N. A. Joshi, Dynamic Load Balancing in Cloud Computing Platform, International Journal of Computer Engineering and Technology 10(3), May, 2019
8. N. K. Chien, N. H. Son and H. D. Loc, Load balancing algorithm based on estimating finish time of services in cloud computing, 18th IEEE International Conference on Advanced Communication Technology, January, 2016
9. R. Kapur, A workload balanced approach for resource scheduling in cloud computing, Eighth IEEE International Conference on Contemporary Computing, August, 2015
10. R. S. Sajjan and R. Y. Biradar, Task Based Approach Towards Load Balancing in Cloud Computing Environment, International Journal of Computer Applications, 179 (31), April, 2018
11. S. K. Vasudevan, S. Anandaram, A. J. Menon and A. Aravinth, A novel improved honey bee based load balancing technique in cloud computing environment, Asian Journal of Information Technology, 15 (9), 2016
12. S. Nan, A. Shi, C. Chen, E. Chen and Y. Wang, Research on Virtual Machine Placement in the Cloud Based on Improved Simulated Annealing Algorithm, World Automation Congress, 2016
13. Y. Azar, I. R. Cohen, A. Fiat and A. Roytman, Packing small vectors, Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 2016
14. Y. Deng and R.W. Lau, Dynamic load balancing in distributed virtual environments using heat diffusion, ACM Transactions on Multimedia Computing, Communications, and Applications, 10 (2), 2014
15. Y. Zhu, D. Zhao, W. Wang and H. He, A Novel Load Balancing Algorithm Based on Improved Particle Swarm Optimization in Cloud Computing Environment. International Conference on Human Centered Computing, Springer, January, 2016.

