International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-2, December, 2019

Optimized Mechanism for Resource Sharing in Cloud

N. A. Joshi

Abstract*: The cloud computing paradigm has settled to a stable stage. Due to its enormous advantages, services based on cloud computing are getting more and more attraction and adoption by diversified sectors of society. Because of its pay per use model, people prefer to execute various data crunching operations on high end virtual machines. Optimized resource management however becomes critical in such scenarios. Poor management of cloud resources may affect not only customer satisfaction but also wastage of available cloud infrastructure. An optimized resource sharing mechanism for collaborated cloud computing environments is suggested here. The suggested resource sharing technique solves starvation issue in inter cloud load balancing context. In case of occurrence of starvation problem, the suggested technique resolves the issue by switching under loaded and overloaded virtual machines between intra cloud and inter cloud computing environment.

Index Terms*: cloud collaboration, cloud computing, virtualization, resource allocation, load balancing.

I. INTRODUCTION

Technological progressions in domain of high end computing systems, communication systems and storage units have boosted up growth of multiple computing paradigms. Cloud computing is one such paradigm. Nowadays, many computing solutions are based on cloud computing in one or many aspects by means of utilizing some of the cloud based services. Among the various cloud services available, the IaaS is used often for processing data crunching operations. People often procure high-end virtual machines to execute large scale CPU-centric data processing jobs. Service providers justify the resource hungry jobs with help of collaborating with other cloud computing environments. Collaborated cloud computing platforms offer advantages such as better resource availability and fault tolerance [7].

Efficient cloud management in terms of allocation and sharing of cloud based resources is important while the requirement for cloud based resources is growing exponentially. Improper and imbalanced resource management and utilization often results in poor consumption and wastage of cloud resources causing dissatisfaction and poor return on investments. An improved resource sharing technique in domain of collaborated cloud computing environments is presented here. The suggested technique help to overcome issues such as starvation and on demand overburdening and under loading.

Comments about study on related work in cloud resource sharing are given in section 2. Section 3 represents the suggested mechanism. Implementation details are given in section 4. Outcomes are mentioned in section 5. Concluding remarks are presented in section 6.

II. RELATED WORK AND CHALLENGES

A technique S. Gond et. al. is works on allocation of cloud resources on basis of teacher learning based optimization [10]. A load balancing technique suggested by Manasrah works on resource sharing for dependent jobs over heterogeneous infrastructure. The technique is based on genetic pso algorithm. The task allocation technique works on the hybrid genetic pso algorithm and aims for reducing makespan and allocation cost [3]. A workflow scheduling technique suggested by H. Ji et.al. works on multi objective scheduling. The technique is based on adaptive priority and works on varying objective weights. The technique self-regulates job priorities for adoption of multiple objectives. N. Joshi proposed a dynamic technique for load balancing. The technique is based on job relocation among virtual machines having varying workloads [8]. A load balancing technique suggested by A. Kaur et.al. is optimization based on hybridization of heuristic techniques with metaheuristic algorithm for attaining optimal performance on makespan and cost [1].

For workflow scheduling in IaaS cloud load balancing, I. Gupta et.al. have suggested a technique. The technique avoids slow convergence rate of genetic algorithm and problem of local optimization. The technique is based on meta-heuristic approach and hybridization of particle swarm optimization and genetic algorithm [6]. A collaborated cloud load balancing technique proposed by N. Joshi works on allocation of inter cloud resources [9]. S. Javanmardi et.al. have presented a hybrid job scheduling approach. The technique performs job allocation by considering job length and virtual machine MIPS. The approach is based on genetic algorithm and fuzzy theory and looks for reduces total execution time and execution cost in cloud load balancing by altering the genetic algorithm [11]. Another resource sharing technique proposed by A. Manasrah, is based on cloud analyst virtual machine environment. The technique works on achieving improved performance through processing time and response time [2]. A technique proposed by N.

Revised Manuscript Received on December 30, 2019.
* Correspondence Author
N. A. Joshi, Department of MCA, Dharmsinh Desai University, Nadiad, India.

Joshi deals with collaborated cloud load balancing by means of shifting workload among virtual machines [9].

A load balancing technique suggested by A. Tripathi et.al. forms a new aco method is based on bee colony and implemented on cloud analyst [4]. Another load balancing mechanism suggested by R. Awatif et.al. works on the Cloud Analyst platform.

The technique suggested works on basis of ensuring efficient response time. The technique aims to reduce overall cost [12]. A technique presented by S. Vasudevan et.al. assigns the available cloud infrastructure to relevant jobs for the sake of reducing the service makespan. The load balancing technique is based on honeybee theory [13].

Few techniques run in simulation environment such as cloudsim and cloud analyst. Some of the techniques discussed here are in form of closed source solutions. Some of the techniques deal with the load balancing issue by means of genetic algorithm. On other hand few techniques work for bringing down response time and cost. Some techniques relocate jobs to the virtual machines which have less workload. More or less of the discussed mechanisms deal with resource sharing in intra cloud computing environment. Whereas, the existing inter cloud resource management technique [9] may cause starvation problem at intra cloud computing environments. Hence, a need for starvation free solution for load balancing in inter cloud computing environments is felt.

III. MECHANISM

A resource sharing mechanism presented in [8] is applicable to load balancing in intra cloud computing environment. On other side, a load balancing technique suggested in [9] is related to resource sharing in inter cloud computing environment. However, the technique may cause resource starvation at local cloud level while the resources may simply go wasted unused. An optimized mechanism of the work presented in [9] is presented here with help an additional thread of execution. The additional thread of execution helps in managing the availability of virtual machines for load balancing at either inter cloud environment or intra cloud environment. The technique suggested here addresses the starvation problem at the level of local cloud computing environment. Block diagram for the proposed mechanism is shown in Fig. 1. Some of the foremost additional steps for the intra cloud load balancing and inter cloud load balancing algorithms are described here.

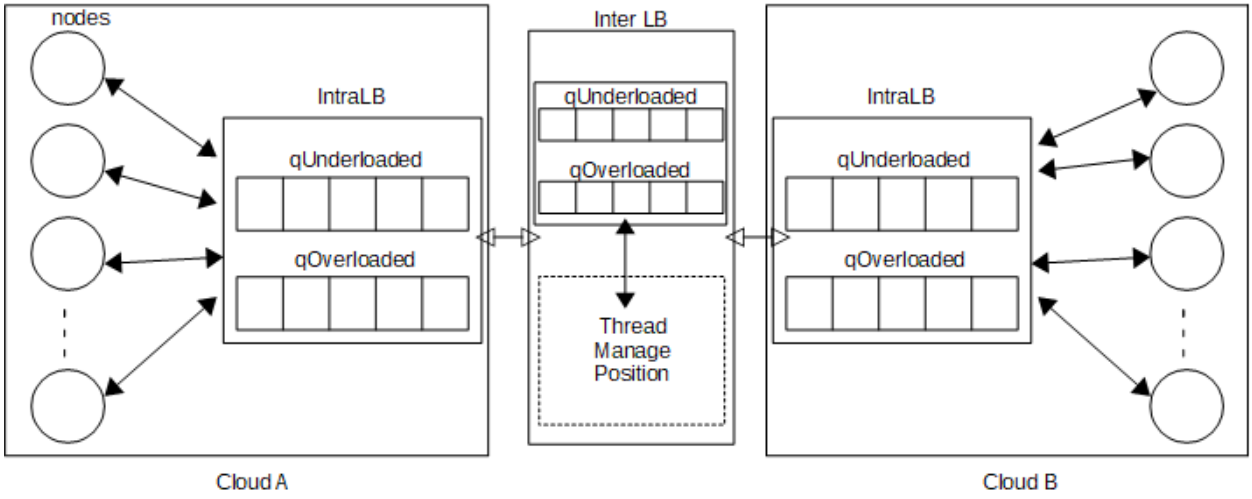

Figure 1: Optimized Resource Sharing Model

1. In addition to the `vm_state` enumerated data type, the mechanism also maintains one more enumerated data type position which is used to represent the type of load balancing a virtual machine is available for: `intra` or `inter`.
2. For keeping track of available over loaded and under loaded virtual machines at inter cloud load balancing layer, two more queues `qUnderloaded` and `qOverloaded` respectively are maintained at central level. Purpose of the two queues `qUnderloaded` and `qOverloaded` is same as their counterparts at local cloud level data structures.
3. The thread `IntraLB:thread_LoadBalancer`, appends information about local level under loaded virtual machines to central queue if there are no more over loaded virtual machines existing at local cloud level. Moreover, the thread appends information about local level over loaded virtual machines to central level queue if there are no more under loaded virtual machines existing at local cloud level. Moreover, the thread sets the position of such virtual machines by toggling their `position` as `INTER`.
4. Contrary to the situation described in step 3, the thread `IntraLB:thread_LoadBalancer`, executes load balancing if the over loaded and under loaded virtual machines are available.
5. The central load balancer thread `InterLB:thread_LoadBalancer`, is responsible for inter cloud load balancing. After issuing the load balance instruction, the thread releases such virtual machines' information from their respective central level queues.
6. Moreover, the central thread `InterLB:thread_LoadBalancer`, resets position of such threads to `INTRA` in order to indicate availability of such virtual machines for intra cloud load balancing.

7. In order to stay away from situations leading towards starvation at local cloud environment, one more thread of execution `InterLB:thread_ManagePosition`, ensures that no virtual machine should settle more than T_I time units at a time in none of the central level virtual machine queues. The thread `InterLB:thread_ManagePosition` runs at central level. The thread sends such virtual machines back to local cloud environment by resetting their position to `INTRA`. Such a behavior helps in keeping concerned virtual machines to be available for intra cloud load balancing.

```
Module ResourceSharing
{
      ..
      enum position {INTRA=0, INTER=1}; // Level of VM availability
      unsigned int TI; // Maximum time a VM can lie in central queue
      struct VM // VM's current information
      {
            position pos; // indicates intra or inter cloud load balancing
            … // other information such as CPU load, memory information
      };

      InterLB: Queue<VM*> qUnderloaded, qOverloaded; //central queues
      IntraLB: Queue<VM*> qUnderloaded, qOverloaded; //local cloud queues

      IntraLB: thread_LoadBalancer //local intra cloud load balancer
      {
            VM* pVMO, pVMU; //over loaded and under loaded VMs
            while(true)
            {
                  pVMO = qOverloaded.fetchAndRemove();
                  //if(!pVMO) continue;
                  if(pVMO && pVMO.timeSincePassive()< MaxOverloadThresholdTime
                      || !pVMO.isOverloaded()
                      || pVMO.state == OVERLOAD_PASSIVE)
                        pVMO = NULL;  //such VM can't participate in Load Balancing

                  pVMU = qUnderloaded.fetchAndRemove();
                  if(pVMU && pVMU.timeSincePassive()< MaxUnderloadThresholdTime
                      || !pVMU.isUnderloaded()
                      || pVMU.state == UNDERLOAD_PASSIVE)
                        pVMU = NULL;  //such VM can't participate in LB

                  if(!pVMU && pVMO) //no under loaded VM is available
                  {
                        InterLB.qOverloaded.append(pVMO);
                        pVMO.vm_state = INTER;
                  }
                  if(pVMU && !pVMO) //no over loaded VM is available
                  {
                        InterLB.qUnderloaded.append(pVMU);
                        pVMU.vm_state = INTER;
                  }

                  //both overloaded & underloaded VMs are available
                  if(!pVMU && !pVMO)
                  {
                        pVMU->vm_state = UNDERLOAD_PASSIVE;
```

```
                pVMO->vm_state = OVERLOAD_PASSIVE;
                set passive_set_time for pVMO and pVMO
                balance(pVMO,pVMU);
          }
          else
               continue;
          …
     }
}; //end of IntraLB:thread_LoadBalancer
InterLB:thread_LoadBalancer //central load balancer
{
     VM* pVMO, pVMU;
     while(true)
     {
          pVMO = qOverloaded.fetchAndRemove();
          if(!pVMO) continue; //no over loaded VM is available
          if(pVMO.timeSincePassive() < MaxOverloadThresholdTime
              || !pVMO.isOverloaded()
              || pVMO.state == OVERLOAD_PASSIVE)
          {
               qOverloaded.remove(pVMO);
               continue; //such VM can't participate in LB
          }
          pVMU = qUnderloaded.fetchAndRemove();
          if(!pVMU) //no under loaded VM is available
               continue;

          if(pVMU.timeSincePassive() < MaxUnderloadThresholdTime
              || !pVMU.isUnderloaded()
              || pVMU.state == UNDERLOAD_PASSIVE)
               continue; //such VM can't participate in LB

          set pVMU->vm_state to UNDERLOAD_PASSIVE;
          set pVMO->vm_state to OVERLOAD_PASSIVE;
          set pVMU->pos to INTRA; //remove from INTER LB
          set pVMO->pos to INTRA; //remove from INTER LB
          set passive_set_time for pVMO and pVMU;
          balance(pVMO,pVMU);
          …
     }
}; //end of InterLB:thread_LoadBalancer
InterLB:thread_ManagePosition //VM position management
{
     …
     //Toggle VM periodically to intra-cloud level
     if(pVM->pos == INTER &&
          pVM->inter_set_time >= TI)
            reset inter_set_time for pVM
            set pVM->pos to INTRA;
     …
} //end of InterLB:thread_ManageState
void start() //initiate & setup environment
{
     …
     Initialize Tu, To and TI intervals
     Initialize IntraLB.qOverloaded
```

```
            Launch thread IntraLB.thread_OverLoadedVM
            Initialize IntraLB.qUnderloaded
            Launch IntraLB.thread_UnderLoadedVM
            Launch IntraLB.thread_LoadBalancer
            Launch IntraLB.thread_ManageState
            Launch InterLB.thread_LoadBalancer
            …
      } //end of start()
}; //end of module ResourceSharing
```

IV. IMPLEMENTATION

The testbed prototype inter cloud load balancing environment was setup with help of an open source Fedora Linux Release 27 operating system. Whereas the target scenario based on cloud environments were setup with help of an open source OpenStack Queens cloud computing software. The open source operating system was deployed on the virtual machine instances. The inter cloud load balancer and relevant queues were implemented on an intermediate proxy server. The intra cloud load balancers in both the cloud computing environments were connected to the intermediate proxy server in order to enable inter cloud load balancing.

V. RESULTS AND DISCUSSION

In local cloud environment, situations may arise such that an overloaded virtual machine might be waiting for availability of suitable destination virtual machine for load balancing. On other side at central level however, the required relevant destination virtual machines from the same cloud might be simply waiting for arrival of suitable request for load balancing request. Vice versa, it may happen that on other side some under loaded virtual machines might be already waiting for suitable load balancing request. Whereas, the over loaded virtual machines from the same local cloud might be waiting at central level for availability for appropriate destination. Such situations if not properly handled, may cause starvation. The suggested mechanism appropriately handles the starvation issue. Based on the scenario, the mechanism looks for such virtual machines. Based on the status of such starving virtual machines, the mechanism either pushes them to central level or pulls them back to local cloud level. Here, the starving virtual machine could be either over loaded or under loaded. Moreover, after issuing the resource sharing request, the technique sends back the source and destination virtual machines back to their local cloud environment. Such a behavior hands over the load balancing control of such virtual machines back to their native cloud controller.

VI. CONCLUSION

An enhanced technique for optimized utilization and management of inter cloud and intra cloud resources is presented here. The suggested resource management mechanism cleverly switches the under loaded and overloaded virtual machines between intra cloud and inter cloud computing environment for resource sharing.

The presented technique also helps in resolving the issue of resource starvation by pulling back the idle virtual machines from central queues back to their local cloud computing environment. The technique can be used for resource sharing in both intra cloud and inter cloud computing environments. The mechanism may be further extended for examining relevant security aspects in the inter cloud resource sharing context.

REFERENCES

1. A. Kaur and B. Kaur, Load balancing optimization based on hybrid heuristic-metaheuristic techniques in cloud environment, Journal of King Saud University – Computer and Information Sciences, 2019
2. A. Manasrah, Dynamic weighted VM load balancing for cloud-analyst, International Journal of Information and Computer Security, 9 (1-2), 2017
3. A. Manasrah and H. Ali, Workflow Scheduling Using Hybrid GA-PSO Algorithm in Cloud Computing, Wireless Communications and Mobile Computing, Vol. 2018, 2018
4. A. Tripathi, S. Shukla and D. Arora, A Hybrid Optimization Approach for Load Balancing in Cloud Computing, Advances in Computer and Computational Sciences, 2017
5. H. Ji, W. Bao, and X. Zhu, Adaptive workflow scheduling for diverse objectives in cloud environments, Transactions on Emerging Telecommunications Technologies, 28 (2), 2017
6. I. Gupta, S. Gupta, A. Choudhary and P. Jana, A Hybrid Meta-heuristic Approach for Load Balanced Workflow Scheduling in IaaS Cloud, International Conference on Distributed Computing and Internet Technology, 2019
7. N. Joshi, Performance-Centric Cloud-Based e-Learning, The IUP Journal of Information Technology, 10 (2), 2014
8. N. Joshi, Efficient Load Balancing in Cloud Computing, Research Review International Journal of Multidisciplinary, 4 (6), 2019
9. N. Joshi, Performance Centric Model for Resource Sharing in Cloud, Research Review International Journal of Multidisciplinary, 4 (6), 2019
10. S. Gond and Shailendra Singh, Dynamic load balancing using hybrid approach, International Journal of Cloud Applications and Computing, 9 (3), 2019
11. S. Javanmardi, M. Shojafar, D. Amendola, N. Cordeschi, H. Liu and A. Abraham, Hybrid Job Scheduling Algorithm for Cloud Computing Environment, Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA, 2014
12. R. Awatif, E. Omri Amina, A. Nouredine, M. Khalid and R. Mohammed, A Performed Load Balancing Algorithm for Public Cloud Computing Using Ant Colony Optimization, Recent Patents on Computer Science, 11(3)
13. S. Vasudevan, S. Anandaram, A. Menon, A. Aravinth, A novel improved honey bee based load balancing technique in cloud computing environment, Asian Journal of Information Technology, 15 (9), 2016

AUTHOR'S PROFILE

Dr Narayan A Joshi is serving as Professor & Head at Dharmsinh Desai University, Nadiad. He was felicitated with prestigious 'Drona Award' by IBM – India.