Indian Journal of Science and Technology, *Vol 9(11), DOI: 10.17485/ijst/2016/v9i11/89290, March 2016* *ISSN (Print) : 0974-6846* *ISSN (Online) : 0974-5645*

A Novel Approach for Dynamic Load Balancing with Effective Bin Packing and VM Reconfiguration in Cloud

Dinesh Komarasamy* and Vijayalakshmi Muthuswamy

Department of Information Science and Technology, Anna University, Chennai - 600025, Tamil Nadu, India; dinesh@auist.net, vijim@annauniv.edu

Abstract

Background/Objectives: In cloud computing, the existing job scheduling algorithms were focused either on efficient job scheduling or optimal load balancing among the Virtual Machines. **Methods/Statistical Analysis:** This paper introduces a novel approach called Dynamic Load Balancing with effective Bin Packing and VM Reconfiguration (DLBPR) in the cloud. In the proposed work, the jobs are initially classified using the deadline based job scheduler and stored in a different job queue based on the expected processing speed of the job. After classification, the jobs in the various queues are prioritized using their attributes. **Findings:** The proposed approach dynamically splits and coalesces the VMs based on the required processing speed of the job. The VMs in the data center are dynamically clustered based on their processing speed with the support of VM live migration and the jobs are processed using the VMs in the cluster. **Applications/Improvements:** The proposed work is experimented in a cloudsim that minimizes the physical machine nearly 22% compared to other existing algorithms.

Keywords: Bin Packing, Cloud Computing, Load Balancing, VM Reconfiguration

1. Introduction

A large number of naïve users are moving towards cloud computing for processing their job in a cost-effective manner. In the cloud, the user submitted jobs are categorized into two types, namely dependent and independent jobs[1,2]. Among these, the independent jobs do not require the output of other jobs to perform their execution. But, the dependent jobs need the results of the some other job to complete their execution. In cloud computing, the job response time has become the most important factor while running the jobs. Hence, scheduling plays an essential role to minimize the response time of the job and to complete the jobs within their deadline. Along with job scheduling, load balancing plays a vital role to improve the resource utilization and to minimize the network congestion. Load balancing can be done either statically or dynamically[3]. In static load balancing, the jobs distribute among the VMs based on the predefined set of rules that is not suitable for the cloud system. So, dynamic load balancing is preferable for cloud computing due to the dynamic nature of cloud computing.

Several job scheduling algorithms were proposed for scheduling and processing the jobs. Still, job scheduling is an NP problem due to the dynamic behavior of the cloud system[4]. From the state of the art, First Come First Serve (FCFS) is one of the most famous existing scheduling algorithms to process the jobs based on the arrival time[5]. Shortest Job First (SJF) and Round Robin algorithms were introduced to process the jobs using the job length and time slice respectively[6]. The above algorithms cannot optimally process the deadline based jobs, so Earliest Deadline First (EDF) algorithm was introduced to schedule the deadline based jobs which minimize the waiting time of the job by changing the order of job execution[7,8]. Moreover, some of the jobs were processed using the arrival time of the jobs. In the above scenario, FCFS algorithm cannot optimally schedule the jobs and so EASY backfilling has

**Author for correspondence*

been proposed to minimize the waiting time of the job[9]. Conservative Migration support Backfilling (CMBF) was proposed to improve the job responsiveness[10].

Several static scheduling techniques like FCFS, Round robin, Max-min were introduced to process the jobs. But, these algorithms are not suitable for cloud computing because some nodes might be heavily loaded and some are not in the data center. To overcome these issues, an LBMM (Load Balancing Max-Min) algorithm has been proposed that balances the load and reduces the execution time of each node[11]. The static load balancing algorithm is not preferable for cloud computing because the workload varies drastically in the cloud environment. So, the dynamic load balancing algorithms has been evolved in the cloud for balancing the load. Dynamic load balancing algorithms distributes the workload among Physical Machines (PMs) at the run time effectively. Moreover, Equally Spread Current Execution algorithm[12] was proposed that distributes the load to the lightly loaded server. Further, a Hierarchical Load Balanced Algorithm (HLBA) was developed for scheduling the jobs using network utilization, memory utilization and idle CPU processing. It scheduled the incoming jobs to the cluster having the fastest idle computing power[13]. Further, the Resources Attribute Selection (RAS) was introduced which process the jobs based on the resource computing capacity, storage space and network utilization of the node to increase the resource utilization[14]. But, arrival rate and service rate are not easy to predict in large-scale cloud environment and so Blind Online Scheduling Algorithm (BOSA) were proposed to forward the jobs to the server having large free slot[15].

The servers have huge computing as well as storage capacities that are issued as a service concurrently to the end-user using virtualization technology. The cloud provides various services such as compute node, web service, a storage node and so on. Amazon provides three different services based on the computing capacity such as small, medium and large. Here, the VMs group into a particular server based on the computing capacity of the VM. The utilization of the VMs is improved using live migration[16]. The VM processing speed cannot be fully utilized while processing the jobs and so the VM computing capacity was partitioned into foreground and background VM. The VM processing speed dynamically varies between foreground and background VM[17].

To the best of our knowledge, none of the existing scheduling algorithms concurrently addresses the job scheduling and load balancing. Thus, the ultimate objectives of the proposed work are to process the jobs within their deadline and to balance the load among the resources. In the proposed work, the VMs are dynamically clustered as small, medium and large based on the processing speed of the VM and the jobs are mapped with the suitable VM persisting in the cluster. In the proposed work, the clusters are sometimes overloaded due to the arrival of similar kind of jobs. At that moment, the VMs may either split or integrate the VMs in the data center based on the request of the job using receiver-initiated approach. After reconfiguration, the VMs will dynamically regroup based on the processing speed of the VMs.

2. Problem Definition

Due to pay per usage model[16], the colossal end-users move toward the cloud computing. The users request is considered as jobs. The jobs are processed using the VMs that persist in the Physical Machine. In the existing algorithms, the VMs are statically grouped in the physical machine based on the VM size in terms of processing power. The load of the cloud system dynamically varies with respect to time. So, the clustering of the VM statically is not suitable for cloud computing. So, the VM can be dynamically split and integrate based on the requesting processing speed of the job. The processing speed of the VM is not entirely utilized while running the jobs in a VM. So, the computing capacity of the VM is partitioned into foreground and background VM and so utmost two jobs process in one VM. Thus, this paper proposes a novel approach for Dynamic Load Balancing with effective Bin Packing and VM Reconfiguration that will split and merge the processing speed of the VM at the runtime based on the arrival rate of the job.

3. Design of Proposed System

Cloud computing is generally deemed as a layered approach. Therefore, the proposed methodology also composed in a layered manner like web tier, schedule tier and resource allocation tier. The overall design of the proposed work is described in Figure 1.

Among these tiers, the jobs are submitted by the users through the web interface which is termed as web tier. In the web tier, *m* jobs are submitted by the end-user at any arbitrary time, which are forwarded to the job scheduling tier. Afterwards, the submitted jobs are

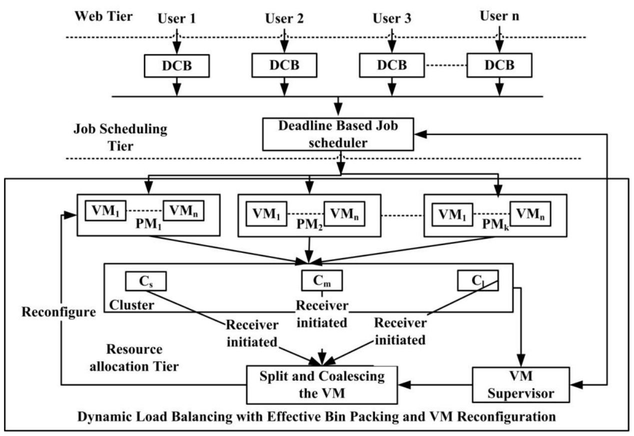

Figure 1. Proposed system design.

scheduled effectively in the job scheduling tier. Here, the deadline based job scheduler will classify and prioritize the jobs. The prioritized jobs are effectively processed in VMs that exist in resource allocation tier. The VM allocation tier consists of numerous Virtual Machines (VMs) which are allocated to suitable hosts or Physical Machines (PMs) based on their size. The novel DLBPR approach has been proposed that will map the VM based on the required processing speed of the job. It forms the clusters as small, medium and large size based on the processing speed of the VMs. Due to the dynamic variation of arrival jobs, any of the clusters may either overloaded or underload which initiates the receiver-initiated problem to improve the utilization of the Physical Machines.

3.1 Deadline based Job Scheduler

During the job submission, the end-user must offer three necessary parameters of the job such as length (*ls*), deadline (*d*) and arrival time (*a*) of the jobs. The jobs are stored in j_q. Afterwards, the jobs are gathered in the multi-job queue (m_q) of deadline based job scheduler that exists in the job scheduling tier. The multi-job queue embeds with three queues as small, medium and large. The m_{sq}, m_{mq}, m_{lq} represents the small, medium and large queue respectively. Initially, the deadline based job scheduler classifies the incoming jobs based on the expected processing speed of the job and stored in the respective queue (m_q). The expected processing speed of the job is denoted as E_p and computed as given below:

$$E_{pj} = \frac{ls_j}{d_j};\ where \forall_j\ in\ m_q$$

The jobs are classified and stored in the multi-job queue. The classified jobs are prioritized to complete the jobs within deadline. The priority of the job is represented as P_{qj} and computed as:

$$P_{qj} = d_j - a_j\ ; where \forall_j\ in\ m_{sq,} m_{mq}, m_{lq}$$

where *a* represents the arrival time of the job. The jobs are sorted based on their deadline along with the arrival time of the job. Sometimes, one or more jobs have equal deadline. At that point, the jobs are sorted based on the FCFS manner. Afterwards, the prioritized jobs are executed using the VM in VM allocation tier. Several VMs are embedded in a particular PMs.

3.2 Dynamic Load balancing with effective Bin Packing and VM Reconfiguration (DLBPR)

In the proposed DLBPR approach, the PMs and VMs are considered as bin and items respectively. Here, the items are packed as bins. Similarly, the VMs are packed in a PMs. In the proposed work, the VM supervisor contains the detailed information of the VM. The VMs are initially clustered based on the processing speed of the VM. Initially, the equal number of VMs is classified with different size of processing speed that exists in the VM supervisor. The job mapped with the VM based on the required processing speed of the job using the VM information in the VM supervisor. The VMs grouped based on the processing speed of the VM as small, medium and large. c_s, c_m, c_l represent the small, medium and large cluster respectively, depending on the processing speed of the VM. The jobs are arriving randomly with the different processing speed request. Due to random and an uneven number of the job request, the VMs in PMs are either underutilized or over utilized during the execution of the jobs. The incoming jobs utmost need small VM. At that point, the VM in c_s cannot sufficient to process the jobs in m_{sq} and so the receiver initiated the problem of overloading VM in particular c_s. At this juncture, the VM that belongs to the c_m will process two jobs in m_{sq} concurrently instead of migrating the VM to another cluster. On the other hand, the incoming jobs sometimes request large VM and so the VM in c_l cannot adequately process the jobs in m_{lq}. In this scenario, the c_l cluster initiates the problem of receiver-initiated overloading and hence, the receiver-initiated sends message to the remaining clusters c_s and c_m. Here, the VMs in c_s and c_m cluster may be underutilized and so the VMs in c_s and c_m cluster will integrate the processing speed of the idle VMs in order to provide the VM for m_{lq} and that are reform the clusters with the support of live migration through virtualization.

3.3 Scheduling Algorithm

```
• Begin
• Cloudlet list ← holds the list of incoming jobs
• For every job in $j_q$ do
• Classify the jobs based on $E_p$ as $E_{pj} = \frac{ls_j}{d_j}$; where $\forall_j$ in $m_q$
• Group the jobs in $m_q$ as $m_{sq}$, $m_{mq}$, $m_{lq}$ based on $E_p$
• Prioritize the jobs in $m_{sq}$, $m_{mq}$, $m_{lq}$ as given as $P_{qj} = d_j - a_j$; where $\forall_j$ in $m_{sq}$, $m_{mq}$, $m_{lq}$
• Map the job in $m_{sq}$, $m_{mq}$, $m_{lq}$ with $c_s$, $c_m$, $c_l$ respectively
• If $c_s$ overloaded then \\ receiver-initiated
• Jobs in $m_{sq}$ allocates to $c_m$, $c_l$
• Else if $c_m$ overloaded then \\receiver-initiated
• Jobs in $m_{mq}$ allocates to $c_l$ only if $c_l$ not fully utilized
• Otherwise $m_{mq}$ allocates to $c_s$ by live migration of VM
  through virtualization.
• Else \\ receiver-initiated
• Jobs in $m_{lq}$ allocates to $c_s$ or $c_m$ by live migration of VM
  through virtualization.
• End if          End if
• Cluster to corresponding VM using online bin packing
  approach
• End for
• End
```

The jobs process with the available VM in the cluster using the above algorithm.

4. Experimental Setup and Result Analysis

CloudSim is a simulation framework that provides cloud platform simulation[18]. It has the power to manage services and modeling of cloud infrastructure. In a real cloud computing environment, it is very difficult to test the incoming jobs and so simulation model is necessary to evaluate the results. In CloudSim, the computing capacity of the data center is partitioned into several VMs using the virtualization technique. Hence, this paper proposes a novel DLBPR approach that will efficiently utilize the VMs existing in the data center or PMs. Table 1 represents the VMs with PMs cluster. The VMs are grouped based on the processing capacity of the VM in the PMs. Here, the cloudlet processes in the VM based on the expected processing speed of the job.

4.1 PM Requirement Comparison Graph

The proposed work focuses to minimize the number of PMs by avoiding the over and under utilization of the PMs and thereby it also minimizes the energy consumption of the data center. Initially, the jobs increase linearly in the proposed system. Figure 2 represents the comparison of PM requirements. Figure 2 shows only a less number of PMs required for processing the jobs compared with the other algorithm. The proposed DLBPR approach cluster the VMs based on their processing speed. Here, the VM are reconfigured based on the processing speed of the job when the cluster is over or under utilized that will minimize the number of PMs.

4.2 Resource Utilization

The utilization of the resource in DLBPR increased compared to the other algorithm. Initially, the equal

Table 1. PM requirement comparison

No of VMs	No of PMs		
	Cloudsim Allocation	**Online bin packing allocation**	**DLBPR**
4	4	3	2
8	8	7	5
11	10	8	7
13	11	10	8
16	14	12	8

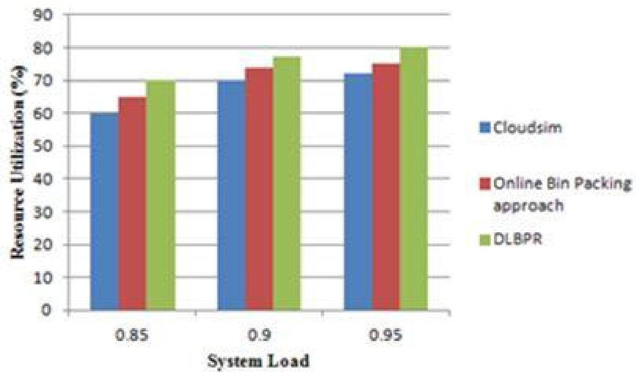

Figure 2. PM requirement comparison.

number of VMs clustered into the PMs. Here, the VMs are dynamically reconfigured based on the arrival rate of the jobs with the support of the VM live migration. The reconfigured VM is clustered at the runtime. Figure 3 represents the utilization of the resources. Here, the jobs are grouped based on the expected processing speed of the job that mapped with the VM existing in the cluster. The number of VM migration is reduced by running two small jobs concurrently in the medium VM cluster. The resource utilization has been increased and thereby increases the throughput of the data center which reduces the waiting time of the job.

4.3 Waiting time

Initially, the jobs in m_{sq}, m_{mq}, m_{lq} process using the c_s, c_m, c_l respectively. The jobs in m_{sq}, m_{mq}, m_{lq} cannot wait for executing in the VMs exist in c_s, c_m, c_l when the number of jobs in m_{sq}, m_{mq}, m_{lq} less than c_s, c_m, c_l. Otherwise, the job has been waiting in the queue when the cluster is overloaded. Figure 4 represents the waiting time of the job. The proposed DLBPR approach reduces the waiting time of the job by migrating the VMs from one cluster to another cluster whenever the cluster is overloaded. The migrated VM is dynamically grouped into the cluster and so the waiting time of the job is reduced in the proposed algorithm.

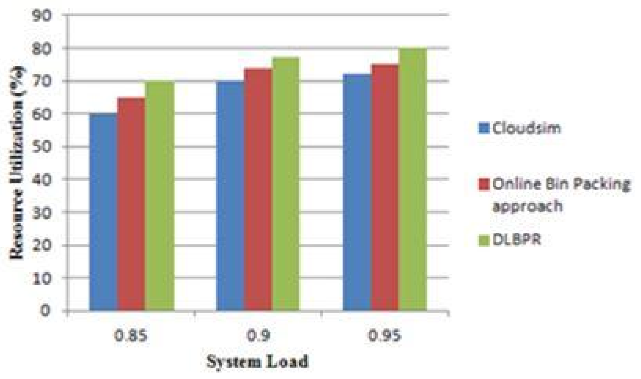

Figure 3. Resource utilization.

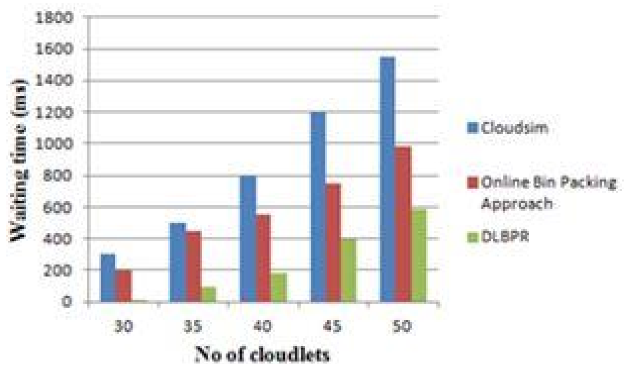

Figure 4. Waiting time of the job.

5. Conclusion and Future Work

The proposed system reviewed the challenges of the existing scheduling and load balancing algorithms during the execution of deadline based jobs. Initially, the deadline based job scheduler categories and prioritizes the jobs to complete within the deadline. Thus, this work proposed a novel Dynamic Load balancing with effective Bin Packing and VM Reconfiguration (DLBPR) approach that splits and integrates the VM based on the requirements of the arrival job. After reconfiguration, the proposed DLBPR approach clusters the VM at the runtime using the receiver-initiated approach. In receiver-initiated approach, the VM live migration reconfigures the VM based on the required processing speed of the job. Thus, the proposed DLBPR approach outperforms the existing scheduling algorithm by migrating the VMs from one cluster to another. It mainly focused on load balancing that automatically improves the throughput and also increases the utilization of the resources. In future, the proposed DLBPR work can be extended to develop the energy efficient system for the dependent and independent jobs.

6. References

1. Zhao C, Zhang S, Liu Q, Xie J, Hu J. Independent tasks scheduling based on genetic algorithm in cloud computing. 5th International Conference on Wireless Communications, Networking and Mobile Computing. WiCom'09; Beijing. 2009. p. 1–4.
2. Maguluri ST, Srikant R. Scheduling jobs with unknown duration in clouds. IEEE/ACM Transactions on Networking. 2014 Dec; 22(6):1938–51.
3. Ajit M, Vidya G. VM level load balancing in cloud environment. 4th International Conference on Computing, Communication and Networking; Tiruchengode. 2013 Jul 4-6. p. 1–5.
4. Van den Bossche R, Vanmechelen K, Broeckhove J. Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. IEEE 3rd International Conference on Cloud Computing; Miami, FL. 2010 Jul 5-10. p. 228–35.

5. Silberschatz A, Galvin PB, Gagne G. Operating System Concepts. 9th edn. John Wiley and Sons; 2011.
6. Garala K, Goswami N. A performance analysis of load balancing algorithm in cloud environment. International Conference on Computer Communication and Informatics; Coimbatore. 2015. p. 1–6.
7. Van den Bossche R, Vanmechelen K, Broeckhove J. Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds. Future Generation Computer Systems. 2013 Jun; 29(4):973–85.
8. Shyamala K, Rani TS. An analysis on efficient resource allocation mechanisms in cloud computing. Indian Journal of Science and Technology. 2015 May; 8(9):814–21.
9. Alem AWM, Feitelson DG. Utilization, predictability, workloads and user runtime estimates in scheduling the IBM SP2 with backfilling. IEEE Transactions on Parallel and Distributed Systems. 2001 Jun; 12(6):529–43.
10. Liu X, Wang C, Zhou BB, Chen J, Yang T, Zomaya AY. Priority-based consolidation of parallel workloads in the cloud. IEEE Transaction on Parallel and Distributed Systems. 2013 Sep; 24(9):1874–83.
11. Wang SC, Yan KQ, Liao WP, Wang SS. Towards load balancing in a three-level cloud computing network. IEEE Intenational Conference on Computer Science and Information Technology; Chengdu. 2010 Jul 9-11. p. 108–13.
12. Nitika M, Shaveta M, Raj MG. Comparative analysis of load balancing algorithms in cloud computing. International Journal of Advanced Research in Computer Engineering and Technology. 2012; 1(3):34–8.
13. Lee YH, Leu S, Chang RS. Improving job scheduling algorithm in a grid environment. Future Generation Computer Systems. 2011 Oct; 27(8):991–8.
14. Bin Z, Li D, Guowei D, Lei W, Weiming W, Julong L. Resource scheduling algorithm and ecnomic model in ForCES networks. Chine Communications. 2014 Mar; 11(3):91–103.
15. Zhou L, Wang H. Toward blind scheduling in mobile media cloud: Fairness, simplicity and asymptotic optimality. IEEE Transaction on Multimedia. 2013 Jun; 15(4):735–46.
16. Mishra M, Das A, Kulkarni P, Sahoo A. Dynamic resource management using virtual machine migrations. IEEE Communications Magazine. 2012 Sep; 50(9):34–40.
17. Dinesh K, Vijayalakshmi M. Deadline constrained adaptive multilevel scheduling system in cloud environment. KSII Transactions on Internet and Information Systems. 2015 Apr; 9(4):1302–20.
18. Buyya R, Ranjan R, Calheiros RN. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. International Conference on High Performance Computing and Simulation, HPCS '09; Leipzig. 2009. p. 1–11.