International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 02, February-2015

Efficient Load Balancing and Dynamic Resource Allocation in Cloud Environment

Sridevi S
Ramanujan Computing Center
Anna University
Chennai, India

Chitra Devi D
Ramanujan Computing Center
Anna University
Chennai, India

Dr. V. Rhymend Uthariaraj
Professor and Director, Ramanujan Computing Center
Anna University
Chennai, India

***Abstract* — The overall performance of cloud is influenced by the scheme adopted to balance the load among the Virtual Machines. An efficient way to handle both dependent and independent tasks is the need of the hour. The problem is to optimize cloud utilization by devising a strategy which handles task scheduling and load balancing effectively. Various algorithms exist for load balancing and scheduling in cloud. The existing algorithms are studied. An algorithm which includes parameters such as processing capabilities of Virtual Machines, current load on the Virtual Machines, job lengths and job interdependencies are considered to propose an algorithm which outperforms the other existing algorithms. Results prove that the proposed algorithm performs better than the existing ones in terms of execution time, number of tasks delayed before getting executed in a VM and the number of task migrations.**

Keywords— cloud; computing; load balancing; resource allocation

I. INTRODUCTION

Cloud computing is an on demand elastic service in which shared resources, information, software and other devices are provided according to the clients requirement at specific time. It can be viewed as solution where data storage and any processing take place without the user being able to pinpoint the specific computer carrying out the task. In Cloud computing, the availability and performance of services are two important aspects to be raised, because users require a certain level of quality service in terms of timeliness of their duties in a lower cost.

Cloud computing uses the concepts of scheduling, load balancing, distributed computing and migrate the tasks to under-utilized Virtual Machines (VM) for effectively sharing the resources like hardware, software and other devices on demand. The goal of scheduling algorithms in distributed systems is spreading the load on processors and maximizing their utilization while minimizing the total task execution time. Job scheduling, one of the most famous optimization problems, plays a key role to improve flexible and reliable systems. Customers are primarily expecting in the reduction of the overall completion of time of tasks on the machines. Virtual Machine is the execution unit of the Cloud and it forms the foundation of the cloud technology. Virtualization can be applied to variety of computer resources: Infrastructure such as Storage, Network, Compute (CPU/Memory etc), Platform (such as Linux/Windows OS) and Software as Services.

II. LITERATURE SURVEY

A. Related Work

The work by Junwei Cao, Keqin Li and Ivan Stojmenovic provides new insights into power management and performance optimization. [1] For multiple heterogeneous multi core server processors across clouds and data centers, the aggregated performance of the cloud of clouds can be optimized by load distribution and balancing. Energy efficiency is one of the most important issues for large scale server systems in current and future data centers. The multi core processor technology provides new levels of performance and energy efficiency. The paper aims to develop power and performance constrained load distribution methods for cloud computing in current and future large-scale data centers. In particular, it addresses the problem of optimal power allocation and load distribution for multiple heterogeneous multi core server processors across clouds and data centers.

The elasticity of Cloud infrastructures makes them a suitable platform for execution of deadline-constrained workflow applications, because resources available to the application can be dynamically increased to enable application speedup. [2] To mitigate effects of performance variation of resources on soft deadlines of workflow applications, an algorithm that uses idle time of provisioned resources and budget surplus to replicate tasks is proposed. Simulation experiments with four well-known scientific workflows show that the proposed algorithm increases the likelihood of deadlines being met and reduces the total execution time of applications as the budget available for replication increases.

The algorithm of Honey Bee Behavior inspired load balancing [3] (HBB-LB), which aims to achieve well balanced load across virtual machines for maximizing the throughput. This algorithm balances the priorities of tasks on the machines in such a way that the amount of waiting time of the priority tasks in the queue is minimal,. Whenever a VM is heavily loaded with multiple tasks, these tasks have to be removed and submitted to the under-loaded VMs of the same data center. In this case, when the removal of more than one task from a heavily loaded VM and if there is more than one VM available to process these tasks, the tasks have to be submitted to the VM such that there will be a good mix of priorities.

Many of the touted gains of cloud computing comes from resource multiplexing through virtualization technology. Here the system uses virtualization technology to allocate data center resources dynamically based on application demands. They introduced the concept of "Skewness" [4] to measure the unevenness in the multidimensional resource utilization of a server. By minimizing skewness, the different types of workloads have been combined to improve the overall utilization of server resources,. The significant contributions of this work are that they developed a resource allocation system that can avoid overload in the system effectively while minimizing the number of servers used. The introduction of "skewness" concept is to measure the uneven utilization of a server. By minimizing skewness, they improved the overall utilization of servers in the face of multidimensional resource constraints. They designed a load prediction algorithm that can capture the future resource usages of applications accurately without looking inside the VMs. The algorithm can capture the rising trend of resource usage patterns and help reduce the placement churn significantly.

A better load balance model has been introduced for the public cloud based on the cloud partitioning [5] concept with a switch mechanism to choose different strategies for different situations. The algorithm applies the game theory to the load balancing strategy to improve the efficiency in the public cloud environment. The load balancing strategy is based on the cloud partitioning concept. After creating the cloud partitions, the load balancing starts, when a job arrives at the system, with the main controller deciding which cloud partition should receive the job. The partition load balancer then decides how to assign the jobs to the nodes. When the load status of a cloud partition is normal, this partitioning can be accomplished locally. If the cloud partition load status is not normal, this job should be transferred to another partition. The cloud partition balancer gathers load information from every node to evaluate the cloud partition status.

III. Scheduling and Load Balancing

The Scheduler has the logic to find the most suitable VM and assign the tasks to VMs based on the algorithm used. The scheduler places the jobs in the most suitable VMs based on the least utilized VM at that particular job arrival time.

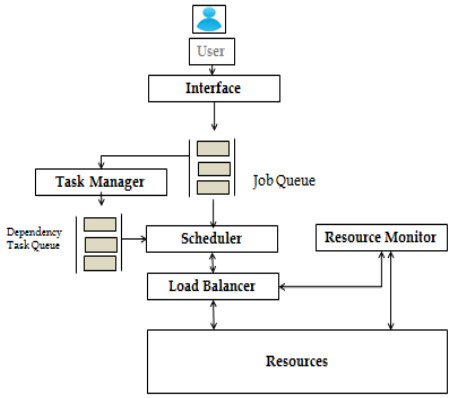

Fig 1. Scheduling and Load Balancing Design

Load Balancer decides the migration of task from a heavily loaded VM to an idle VM or least loaded VM at run time, whenever it finds an idle VM or least loaded VM by utilizing the resources current status information. Resources monitor is available as part of Load Balancer and it communicates with all the VMs resources prober and collects the VM capabilities, current load on each VM, number of jobs in execution/waiting queue in each VM. The Task Requirement is provided by the user which includes the length of the tasks to be executed and transfer the requirements to the scheduler for its operative decisions.

A. *Scheduling and Load Balancing design*

The various modules of the Scheduling and Load Balancing design are discussed below:

- *Job Queue:* All the client's Job requests are reaching the Request Queue in the order of its arrival. The priority or the length of the jobs has not been considered in the Queue. When a job has been taken-out for VM assignment by the Scheduling Controller using the algorithm of First-In-First-Out (FIFO), it will be moved out of the Request Queue.
- *Dependency Task Queue:* This queue will contain the tasks, which depends on the other tasks present in the VMs. Once all the child tasks of the tasks present in this queue got completed its execution the this parent task will be taken for the execution by assigning it to the VM.
- *Task Manager*: This module receives the Job and verifies the job whether it is a complete independent task or it contains multiple tasks. In case, if it contains multiple tasks, then it verifies the inter-dependency between the multiple tasks. Now, all the independent tasks will be directly assigned to the VMs. The dependent tasks will be notified to the scheduler so that parent tasks are scheduled after child tasks are executed.
- *Scheduler:* The scheduler selects the appropriate VMs based on the configured algorithms. This Scheduler collects the resources information through the Load Balancer from the Resource Monitor. It calculates the processing capacity of each of the VMs and then it applies the configured algorithm to find the appropriate VM for the given job.

- *Load Balancer:* Load Balancer (LB) calculates the ratio between the number of jobs running and the number of VMs. If the ratio is less than 1, then it communicates the scheduler to identify a VM for the job else it will calculate the load on each of the VM using the Job Execution List of the VMs. If the utilization is less than the 20% then the least utilized VM will be allotted else the scheduler will be communicated to identify the most suitable VM for the job. Once the appropriate VM has been identified the Job will be assigned to that VM.
- *Resources:* The configured datacenters, hosts and their VM and their Processing Elements form the set of resources available for computing. The resources are probed for idleness and for heavy load so that the job requests are effectively allocated to an appropriate resource.

B. Algorithms

The following scheduling and load balancing algorithms are implemented for functionalities such as scheduling and load balancing. WRR++ is the proposed algorithm. The RR and WRR are existing algorithms which are implemented for carrying out a comparative study.

1) RR: The Round Robin algorithm allocates task to the next VM in the queue irrespective of the load on that VM. RR works well in most configurations, but could be more effective if the VMs are of roughly equal processing capacity, speed and memory.

2) WRR: Here the VMs are ordered in a circular queue based on weightage assigned to them. The incoming requests are then allocated to the ordered VMs in a circular fashion. WRR does not take into account of the load on the VMs or the length of the tasks allocated to the VMs. Hence, small tasks may be assigned to a VM with high processing capability and vice versa. WRR becomes equivalent to RR when the VM configuration, VMs processing capacity and speed are similar.

3) WRR++: WRR++ is an algorithm which is proposed as an improvement over the existing WRR algorithm. Here the WRR++ considers processing capabilities of the VMs, current load on the VMs and estimated job execution time. The algorithm works with coordination among three major modules of the system. The modules are,

a) Static Scheduler: This module does the functionality of initial job placements by considering the total number of VMs provisioned and the number of job requests.

b) Dynamic Scheduler: The dynamic scheduler takes care of run time job placements by taking into account of current load on the VMs, the nature of the task arrival and the instance at which the task requests are submitted.

c) Load Balancer: The load balancer checks for the current load and remaining time estimated for task completion and balances the load on the VMs by migrating a task from heavily loaded VM to a lightly loaded VM.

C. Mathematical Model

The problem is to assign dynamically arriving dependent / independent tasks to VMs and balance the load on the VMs to achieve reduced response latency and maximize resource utilization. Let us consider there are *n* number of VMs and *m* number of tasks. The set of all VMs are represented as VM_j where j varies from 1 to *n* and the set of all task requests are represented as T_i where i varies from 1 to *m*.

Processing time of all tasks in a VM_j can be defined as,

$$\varphi_j = \sum_{i=1}^{m} PT_{ij} \quad (1)$$

where PT_{ij} is the processing time of i^{th} task T_i on j^{th} virtual machine VM_j. The factor gives the minimum time the VM is required to be provisioned and is running to execute all the tasks assigned to it.

Processing capacity of a VM can be denoted as follows,

$$\mu_j = n_{(pe)} * mips_{(pe)} \quad (2)$$

where μ_j is the processing capacity of the VM_j, $n_{(pe)}$ is the number of processing elements in the VM and $mips_{(pe)}$ is the Million Instructions per second of a PE.

The earliest start time and latest finish time of a task T_i are represented as $t_{es(i)}$ and $t_{lf(i)}$ respectively. $t_{es(i)}$ is the earliest time a task is able to start, which happens when all its child tasks complete execution as early as possible. It is represented as follows:

$$t_{es(i)} = \begin{cases} max\,(t_{es(children(i))} + t_{burst(i)}),\ if\ dependent \\ t_{es(a)} + t_{burst(a)},\ if\ independent \end{cases} \quad (3)$$

where *children(i)* denotes the set of all child tasks for task T_i, $t_{burst(i)}$ is the burst time needed to complete the task T_i and T_a represents the preceding independent task.

$T_{lf(i)}$ is the latest time a task can finish without exceeding the maximum provisioning time. This happens when any one of its child tasks complete execution as late as possible. It is represented as follows:

$$t_{lf(i)} = \begin{cases} max\,(t_{lf(children(i))}),\ if\ dependent \\ t_{ls(a)} + t_{burst(a)},\ if\ independent \end{cases} \quad (4)$$

where $t_{ls(a)}$ is the latest time a task can be scheduled to a VM without leading to starvation.

The Schedule time of task T_i is $t_{sch(i)}$. It is the time at which the task has been scheduled for execution. This parameter can assume any value between $t_{es(i)}$ and $t_{lf(i)}$. Our problem is to identify the right VM and right time to schedule the task such that the VM utilization is high and load on the VMs are balanced well. Formally,

$$t_{es(i)} \leq t_{sch(i)} \leq t_{lf(i)} \quad (5)$$

The expected task execution time requirement is given by γ_i. A practical approach is to predict the length of the next burst, based on some historical measurement of recent burst times for this process. One simple, fast, and relatively accurate method is the *exponential average*, which can be defined as follows.

$$\gamma_{i+1} = \alpha * \gamma_{i+1} + (1.0 - \alpha) * \gamma_i \quad (6)$$

The VM utilization $util_j$ is calculated as,

$$util_j = \frac{VM\ CPU\ usage\ (in\ MHz)}{n * core\ frequency\ (in\ MHz)} \quad (7)$$

where n is the number of VMs running.

Our optimal load balancing problem for multiple heterogeneous multi core VMs can be specified as follows: Given the number of VMs *n*, the number of tasks *m*, the capacity of a VM μ_j, expected task execution time requirement γ, the task arrival rate to the VMs λ and the load threshold τ, find the best VM for the jobs in the job queue with VM utilization $util_j$, such that the average task response time $T(\lambda 1, \lambda 2, \ldots, \lambda m,)$ is minimized and overall VM utilization $util_{(VM1,VM2,\ldots VMj)}$ is maximized, subject to the constraints, $F(\lambda 1, \lambda 2, \ldots, \lambda m) = \lambda$, where $F(\lambda 1, \lambda 2, \ldots, \lambda m) = (\lambda 1 + \lambda 2 + \ldots + \lambda m)$, λi follows poisson distribution of arrival, the overall VM utilization varies between $0 \leq util_j \leq 1$, processing capacity of the VMs μ_j varies between $0 \leq \mu_j \leq 1$ and $t_{es(i)} \leq t_{sch(i)} \leq t_{lf(i)}$ as per (5)

IV. Experimental Results

The performance of the algorithm has been analyzed based on the results of simulation done using the CloudSim. The CloudSim framework is studied and the framework is modified to include the various LB algorithms and perform a comparative study.

In the following illustrations, the overall execution time in Time Shared (TS) and Space Shared (SS) environments are analyzed in the RR, WRR, WRR++ algorithms under the combination of heterogeneous & homogenous resource conditions and heterogeneous & homogenous job nature with cloudlet allocation policy being time and space shared.

Table 1 gives the cloud configuration details based on which the following results are obtained and performance analysis is done. The number of VMs is varied from 10 to 100 by increments of 10 to analyze the parameters such as execution time in TS and SS execution modes.

TABLE I. Entities And their Configurations

Sl No	Entity	Quantity
1	Data Center	1
2	Hosts in DC	200
3	Processing Elements (PE)	8/16
4	PE Processing Capacity	125/355/455 MIPS
5	Host RAM Capacity	2/8 GB RAM
6	VM	10 to 100 incremented by 10
7	No of PE to VM	1
8	VMs PE Processing Capacity	150/300/90/120/93/112/105/225
9	VM RAM capacity	1000 MB
10	VM Manager	Xen

A. Performance Analysis

The algorithms are implemented and an extensive comparative study is conducted by executing them in time shared and space shared execution environments. The analysis is as follows:

The time taken for a given number of tasks to complete on a given configuration is analysed by varying the number of VMs for all the given algorithms. The analysis proves that the WRR++ outperforms existing LB algorithms as it includes VM and task status information for resource allocation unlike WRR and RR as shown in Fig. 2 and 3.

1) Based on Overall Execution Time (Time Shared): Results proved that WRR++ delivers a faster completion time than the other load balancing algorithms in the heterogeneous resources. The WRR++ algorithm considers the estimated job execution time along with processing capacity of the heterogeneous VMs to assign the job. So, the lengthy jobs get assigned to the higher capacity VMs in the heterogeneous environments. This helps to complete the job in a shorter time.

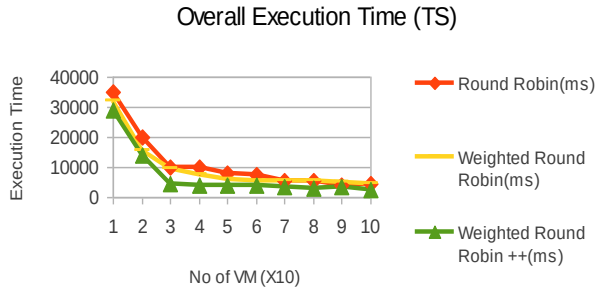

Fig 2. Overall Execution Time (TS)

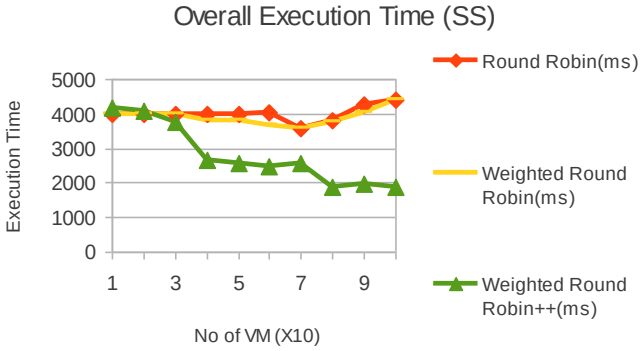

Fig 3. Overall Execution Time (SS)

2) Based on Overall Execution Time (Space Shared): The overall execution time for SS mode of execution is given in Fig 3. WRR++ performs better than WRR and RR even in SS mode. SS mode of execution is the way of task execution in VMs where the core is entirely allocated for a specific task before any other task is allowed to get executed. Whereas TS is the VM's task execution strategy where tasks are executed parallely with a specific time slot for each task. As the number of VMs are increasing the WRR++ performs better than the other algorithms in discussion.

V. CONCLUSION AND FUTURE WORK

Cloud based applications are influenced heavily by the way LB is handled. For end users, the LB capability adopted is one of the major factors based on which they select a cloud service provider. For cloud providers, LB capability directly relates to the service quality with response time as QoS parameter. And this in turn directly influences the revenue. Thus an efficient LB strategy is inevitably required to build any cloud architecture.

In this paper the algorithms RR, WRR and WRR++ are discussed for scheduling, load balancing and task independence and dependency scenario. These algorithms are having three different stages to handle the three different scenarios in the environment life cycle. The performance analysis and experiment results of this algorithm proved that the WRR+ algorithm is most suitable to the heterogeneous / homogenous jobs with heterogeneous resources (VMs) than the other Round Robin, Weighted Round Robin algorithms.

As part of future work, other performance analysis parameters such as number of task migrations between VMs, Million Instructions wasted due to migration can be analyzed for the various algorithms. Many more tests are required to guarantee high system availability and responsiveness. The load threshold is to be evaluated and fixed comprehensively in order to migrate tasks at the right time when the VM starts to get heavily loaded.

ACKNOWLEDGMENT

We wish to express my whole hearted thanks to Mr. K. GokulNath, Anna University, Chennai for his valuable suggestions, constant encouragement and invaluable inputs which was the driving force to complete the project.

We are extremely indebted to our family members and friends for their adorable support throughout the development of the project.

REFERENCES

[1] Junwei Cao, Keqin Li, Ivan Stojmenovic, "Optimal Power Allocation and Load Distribution for Multiple Heterogenous Multicore Server Processors across Clouds and Data Centers" IEEE Transactions On Computers, Vol. 63, No. 1, January 2014.

[2] Rodrigo N. Calheiros, Rajkumar Buyya, "Meeting Deadlines of Scientific Workflows in Public Clouds with Tasks Replication" IEEE Transactions on Parallel and Distributed Systems, Vol. 25, No. 7, July 2014.

[3] L.D.Dhinesh Babu P. Venkata Krishna, "Honey Bee Behavior inspired Load Balancing of tasks in Cloud Computing Environments" Applied Soft Computing, January 2013.

[4] Zhen Xiao, Weijia Song, Qi Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment" IEEE Transactions on Parallel and Distributed Systems, Vol. 24, No. 6, June 2013.

[5] Gaochao Xu, Junje Pang, Xiaodong Fu, "A Load Balancing Model on Cloud Partitioning for the Public Cloud" Tsinghua Science and Technology, Vol. 18, No. 1, February 2013.

[6] Xia Junjie Ni, Yuanqiang Huang, Zhongzhi Luan, Juncheng Zhang, Depei Qian, "Virtual Machine Mapping Policy Based on Load Balancing in Private Cloud Environment", 2011 International Conference On Cloud And Service Computing.

[7] M.Arumbrust et al., "Above the clouds: A Berkeley View of Cloud computing, " technical report, Univ.of Califoronia, Berkeley, Feb.2009.

[8] L.Siegele, "Let It Rise; A Special Report on Corporate IT,"The Economist, Vol.389, PP. 3 – 16, Oct.2008.

[9] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A.Ho, R. Neugebauer, I. Pratt, and A.Warfield, "Xen and the Art of Virtualization, " Proc.ACM Symp. Operating Systems Principle (SOSP' 03), Oct 2003.

[10] "Amazon elastic compute cloud (Amazon EC2)," http://aws.amazon.com/ec2/, 2012.

[11] C.Clark, K.Fraser, S.Hand, J.G.Hansen, E.Jul, C.Limpach, I.Pratt, and A.Warfield, "Live Migration of Virtual Machines, " Proc.Symp.Networked Systems Design and Implementation (NSDI '05), May 2005.

[12] M.Nelson, B-H. Lim, and G.Hutchins, "Fast Transparent Migration for Virutual Machines, " Proc. USENIX Ann. Technical Conf., 2005.

[13] Rodrigo N.Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A.f.De Rose, and RajkumarBuyya "Cloudsim: A Toolkit for Modeling and Simulation of cloud Computing Environments and Evaluation of Resource Provisioning Algorithms" Cloud Computing and Distributed systems (CLOUDS) Laboratory, Department of Computer Science and software Engineering, The University of Melbourne, Australia.