A Survey on Scheduling and Load Balancing Techniques in Cloud Computing Environment

Subhadra Bose Shaw
PhD Scholar, CSED, MNNIT
Allahabad, India
e-mail: subhadra.shaw@gmail.com

Dr. A.K. Singh
Associate Professor, CSED, MNNIT
Allahabad, India
asbhadoria@gmail.com

Abstract— **Now-a-days cloud computing is the most emerging technology due to its elasticity of resource provisioning and the pay-as-you-go pricing model which enables users to pay only according to their need. As cloud can be accessed anytime and anywhere through commodity hardware only its demand is increasing day by day. So it must provide high performance gain to the user and at the same time must be beneficial for the Cloud Service Provider (CSP). To achieve this goal many challenges have to be faced. Load balancing is one of them which helps the CSP to meet the QoS requirements of the users and at the same time maximize his profit by optimum use of the resources. To balance the load in cloud the resources and workloads must be scheduled in an efficient manner. A variety of scheduling algorithms are used by load balancers to determine which backend server to send a request to. The selected server allocates resources and schedules the job dynamically on some virtual machine (VM) located on the same physical machine. It is also the responsibility of the provider to dynamically reallocate or migrate the VM across physical machines for workload consolidation and to avoid over utilization or under utilization of resources. In this paper, we have discussed different algorithms proposed to resolve the issue of load balancing and task scheduling in Cloud Computing. We have mentioned some of their shortcomings for further development. VM migration issues involved in load balancing are also described briefly.**

Keywords- Cloud Computing, Load Balancing, Migration, Task Scheduling, Virtualization

I. INTRODUCTION

Cloud computing is an internet based distributed computing where computing resources are made available to the users in a pay-as-you-go model. As defined by NIST [1] *"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"*. The three main types of services provided by cloud are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Virtualization is the key technology behind cloud computing that allows the simultaneous execution of diverse tasks over a shared hardware platform [2]. Cloud provides computing resources in the form of virtual machine, which is an abstract machine that runs on physical machine [3]. It gives the user a feeling of working in an isolated environment since the co-located tasks do not interact with each other and access only its own data. The isolation between each virtual machine is maintained by Hypervisor, Virtual Machine Manager, which manages multiple operating systems by allocating required resources and provides an identical abstraction of the underlying hardware to the virtual machines in full virtualization [4]. Virtualization provides the potential for on-the-fly and on-demand configuration of physical machines to run diverse tasks, hence avoiding resource waste [2].

Though virtualization tries to balance the load of the whole system dynamically [27] there is always a chance of over utilization or under utilization of resources. Overloaded servers lead to performance degradation whereas underloaded servers cause poor utilization of resources. Due to inefficient distribution of load more heat will be generated by the overloaded servers which in turn increase the cost of cooling system and substantial emission of CO_2 contributing to greenhouse effect [7]. In most cases, the cooling systems require more power than the core IT equipment [6]. On the other hand underutilized servers increase power consumption of the overall system which is not at all environment friendly and moreover it increases the operational cost. So it is necessary to provide right amount of resource dynamically to the applications running in virtual machines in order to meet the QoS requirement as well as to balance overall load of the system. It also helps in energy-efficient utilization of resources by switching off underutilized resources using DPM (Dynamic Power Management) or operating them in a low performance level using DVFS (Dynamic Voltage and Frequency Scaling) [6].

Load balancing is one of the main challenges in cloud computing. It is a mechanism that distributes the dynamic workload evenly across all the nodes in the whole cloud to avoid a situation where some nodes are heavily loaded while others are idle or doing little work [5]. By optimal utilization of resources it helps to achieve less response time, high throughput, improved fault tolerance, scalability, high user satisfaction, less heat generation, optimum power consumption, reduced emission of CO_2 and less operational cost.

The load balancing problem can be divided in two sub problems:

1. Admission of new request for VM provisioning and placement of VMs on host.
2. Reallocation/migration of VMs.

978-1-4799-6758-2/14/$31.00 ©2014 IEEE

To solve the first sub-problem different load balancing algorithms are there in the literature which we will discuss shortly. The second sub problem faces many challenges like when and where to migrate, which VM should be selected for migration etc.

The rest of the paper is organized as follows: Section II briefly describes types of load balancing algorithms. Different load balancing algorithms with their pros and cons are discussed in Section III. Section IV presents an overview of different VM migration techniques. Finally, Section V concludes the paper with final remarks and scope for future research on this topic.

II. Types of Load balancing Algorithms

Load balancing algorithms can be broadly classified into two types: Static algorithms and Dynamic algorithms. In Static Scheduling the assignment of tasks to processors is done before program execution begins i.e. in compile time. Scheduling decision is based on information about task execution times, processing resources etc. which is assumed to be known at compile time [8]. Static scheduling methods are non-preemptive. The goal of static scheduling methods is to minimize the overall execution time. These algorithms cannot adapt to load changes during run-time [9].

Dynamic scheduling (often referred to as dynamic load balancing) is based on the redistribution of processes among the processors during execution time. This redistribution is performed by transferring tasks from the heavily loaded processors to the lightly loaded processors with the aim of improving the performance of the application. It is particularly useful when the requirement of process is not known a priori and the primary goal of the system is to maximize the utilization of resources. The major drawback of dynamic load balancing scheme is the run-time overhead due to the transfer of load information among processors, decision-making for the selection of processes and processors for job transfers and the communication delays associated with the task relocation itself.

The dynamic load balancing algorithms can be centralized or distributed depending on whether the responsibility for the task of global dynamic scheduling should physically reside in a single processor (centralized) or the work involved in making decisions should be physically distributed among processors [10]. The most important feature of making decisions centrally is simplicity [8]. However centralized algorithms suffer from the problem of bottleneck and single point failure. Distributed load balancing algorithms are free from these problems.

Again distributed dynamic scheduling can be cooperative or non cooperative. The last one is simple where individual processors act alone as autonomous entities and arrive at decisions regarding the use of their resources independent of the effect of their decision on the rest of the system. In the former one each processor has the responsibility to carry out its own portion of the scheduling task to achieve a common system wide goal [8][10].

If information regarding the state of the system as well as the resource needs of a process are known then the cooperative scheduling can be optimal i.e. it can minimize total process completion time, maximize utilization of resources or maximize system throughput. But in real cloud scenario optimal solutions are not possible so suboptimal solutions can be tried [10].

Within the realm of suboptimal solutions to the scheduling problem if instead of searching the entire solution space for the best solution, we are satisfied when we find a "good" one, then it can be categorized as suboptimal-approximate solution [10]. Another category of suboptimal solution is the heuristic method which as the name indicates, relies on rules-of-thumb to guide the scheduling process in the right direction to reach a "near" optimal solution.

III. Literature Survey of Load Balancing Algorithms

The cloud facility is a network of geographically distributed datacenters, each consisting of hundreds of servers. When a user submits a task (popularly known as cloudlet) it is handled by the datacenter controller. The Data Center Controller [11], uses a VmLoadBalancer to determine which VM should be assigned the next request for processing. The Vmloadbalancer can use the following algorithms for load balancing.

- Round Robin - The datacenter controller assigns the requests to a list of VMs on a rotating basis. The first request is allocated to a VM picked randomly from the group and then the subsequent requests are assigned in a circular order [12]. Once the VM is assigned a request, it is moved to the end of the list. Though the work load distributions between processors are equal but the job processing time for different processes are not same. So at any point of time some nodes may be heavily loaded and others remain idle [5].
- Weighted Round Robin – It is the modified version of Round Robin in which a weight is assigned to each VM so that if one VM is capable of handling twice as much load as the other, the powerful server gets a weight of 2. In such cases, the Data Center Controller will assign two requests to the powerful VM for each request assigned to a weaker one. The major issue in this allocation is same as that of Round Robin that is it also does not consider the advanced load balancing requirements such as processing times for each individual requests [13].
- Dynamic Round Robin [14][21]- This algorithm mainly works for reducing the power consumption of physical machine. The two rules used by this algorithm is as follows:

i) If a virtual machine has finished its execution and there are other virtual machines hosted on the same physical machine, this physical machine will accept no more new virtual machine. Such physical machines are called to be in "retiring" state, i.e. when rest of the virtual machines

finishes their execution, then this physical machine can shut down.

ii) The second rule says that if a physical machine is in retiring state for a long time then instead of waiting, all the running virtual machines are migrated to other physical machines. After the successful migration, we can shut down the physical machine. This waiting time threshold is called "retirement threshold".

The algorithm reduces the power consumption cost but it does not scale up for large data centers.

- Randomized – It randomly assigns the selected jobs to the available VM. The algorithm is very simple but it does not take into consideration whether the VM is overloaded or underloaded. Hence, this may result in the selection of a VM under heavy load and the job may require a long waiting time before being serviced [17].
- Equally Spread Current Execution (ESCE) Algorithm - In this method the load balancer continuously scans the job queue and the list of virtual machines. If there is a VM available that can handle the request then the VM is allocated to that request [15]. If there is an overloaded VM that needs to be freed of the load, then the load balancer distributes some of its tasks to the VM having least load to make every VM equally loaded [16]. The balancer tries to improve the response time and processing time of a job by selecting it whenever there is a match. But it is not fault tolerant and has the problem of single point of failure [14].
- Throttled - The Throttled Load Balancer (TLB) maintains a record of the state of each virtual machine (busy/idle) [12]. When a request arrives it searches the table and if a match is found on the basis of size and availability of the machine, then the request is accepted otherwise -1 is returned and the request is queued [16]. During allocation of a request the current load on the VM is not considered which can in turn increase the response time of a task.
- Modified Throttled - Like the Throttled algorithm it also maintains an index table containing list of virtual machines and their states. The first VM is selected in same way as in Throttled. When the next request arrives, the VM at index next to already assigned VM is chosen depending on the state of VM and the usual steps are followed, unlikely of the Throttled algorithm, where the index table is parsed from the first index every time the Data Center queries Load Balancer for allocation of VM [18]. It gives better response time compare to the previous one. But in index table the state of some VM may change during the allocation of next request due to deallocation of some tasks. So it is not always beneficial to start searching from the next to already assigned VM.
- Central Load Balancer – It is basically an updated version of Throttled algorithm. Like Throttled it also maintains a table containing the state of each VM along with their priority. The priority is calculated based on the CPU speed and capacity of memory [38]. The VM assignment policy is similar to that of Throttled except that in this algorithm the VM with highest priority will get the first preference. If it is busy then the VM with next highest priority is checked and the process continues until a VM is found or the whole table is searched. The algorithm efficiently balances load in a heterogeneous environment but it suffers from bottleneck as all the requests will come to central load balancer. Moreover the algorithm is based on the priority of VMs which is calculated in a static way and is not updated during job allocation.
- Active Monitoring Load Balancing (AMLB) Algorithm – It maintains information about each VM and the number of requests currently allocated to which VM. When a request to allocate a new VM arrives, it identifies the least loaded VM. If there are more than one, the first identified is selected. Load Balancer returns the VM id to the Data Center Controller. It sends the request to the VM identified by that id and notifies the Active VM Load Balancer of the new allocation [19]. During allocation of VM only importance is given on the current load of VM, its processing power is not taken into consideration. So the waiting time of some jobs may increase violating the QoS requirement.
- VM-Assign Load Balancing Algorithm – It is a modified version of Active Monitoring Load Balancing algorithm. The first allocation of VM is similar to the previous algorithm. Then if next request comes it checks the VM table, if the VM is available and it is not used in the previous assignment then, it is assigned and id of VM is returned to Data Center, else we find the next least loaded VM and it continues, unlikely of the Active load balancer, where the least loaded VM is chosen but it will not check for the previous assignments [20].

According to Shridhar G. Domanal et. al this algorithm will utilize all the VMs completely and properly unlike the previous one where few VMs will be overloaded with many requests and rest will remain under utilized [20]. But it is not clearly mentioned in the paper that how it happens. This algorithm will not use the VM if it is already allocated in the last round. But there is no logic behind it. Because it may still be the least loaded VM having good processing speed. So more tasks can be assigned to it. Finding the next least loaded VM will distribute the tasks evenly only when there are multiple VMs which are equally loaded or the next least loaded VM has a high processing speed compare to the previous one. But the algorithm only considers the load and if the VMs are equally loaded then the task can be assigned to any of them irrespective of the fact that whether the VM is used in the last iteration or not. Since allocation of a task change the state of VM so in the previous algorithm least loaded VM will be found automatically and even task distribution will take place.

- Weighted Active Monitoring Load Balancing Algorithm - Jasmin James et. al proposed this method [12] which is a combination of Weighted Round Robin and Active Monitoring Load Balancing Algorithm. In this algorithm different weights are assigned to VMs depending on the available processing power. Among the least loaded

VMs the tasks are assigned to the most powerful one according to their weights. In this way it removes the shortcomings of Active Monitoring Load Balancing Algorithm by not only considering the load but also the processing power of available VMs.

- Biased Random Sampling [22] – In this approach, the load on a server is represented by its connectivity in a virtual graph. The initial network is constructed with virtual nodes to represent each server node, with each in-degree mapped to the server's free resources. When a node executes a new job, it removes an incoming edge indicating available resources are reduced. Conversely, when the node completes a job a new inward edge is created indicating available resources are increased again. In a steady state, the rate at which jobs arrive equals the rate at which jobs are finished. So the network would have a static average number of edges.

The increment and decrement process is performed via Random Sampling. The sampling walk starts at a specific node and moves to a randomly chosen neighbour. The last node in the sampling walk is selected for the load allocation. The effectiveness of load distribution is considered to increase with walk length whose threshold is around log(n) steps, where n is the network size [23].

When a node receives a job it will execute it if the job's current walk length is greater than or equal to the walk length threshold. If the walk length is less than the threshold, the job's w value is incremented and it is sent to a random neighbour, thus continuing the Random Sampling approach. As the method is decentralized it is suitable for large network systems and can be easily implemented using standard networking protocols.

- Min-Min Scheduling Algorithm [24] - It starts with a set of tasks. Then the resource which has the minimum completion time for all tasks is found. Next, the task with the minimum size is selected and assigned to the corresponding resource (hence the name Min-Min). Finally, the task is removed from set and the same procedure is repeated by Min-Min until all tasks are assigned. The method is simple but it does not consider the existing load on a resource before assigning a task. So proper load balance is not achieved.
- Load Balance Improved Min-Min Scheduling Algorithm (LBIMM) [24] - It starts by executing Min-Min algorithm at the first step. At the second step it chooses the smallest size task from the heaviest loaded resource and calculates the completion time for that task on all other resources. Then the minimum completion time of that task is compared with the makespan produced by Min-Min. If it is less than makespan then the task is reassigned to the resource that produce it, and the ready time of both resources are updated. The process repeats until no other resources can produce less completion time for the smallest task on the heavy loaded resource than the makespan. Thus the overloaded resources are freed and the underloaded or idle resources are more utilized. This makes LBIMM to produce a schedule which improves load balancing and also reduces the overall completion time. But still it does not consider priority of a job while scheduling.
- User-Priority Awared Load Balance Improved Min-Min Scheduling Algorithm (PA-LBIMM) [24] - User priority is incorporated with LBIMM algorithm to develop PA-LBIMM. This algorithm will first divide all the tasks into two groups G1 and G2. G1 is for the VIP users' tasks having higher priority requirement. G2 is for the ordinary users' tasks. The higher priority tasks in G1 are scheduled first using the Min-Min algorithm to assign the tasks to the VIP qualified resources set. Then the tasks with lower priority are scheduled to assign them to all the resources by Min-Min algorithm. At the end, the load balancing function is processed to optimize the load of all resources to produce the final schedule. The algorithm is only concerned with the makespan, load balancing and user-priority. It does not consider the deadline of each task.
- Max Min Algorithm - It works as the Min-Min algorithm. But it gives more priority to the larger tasks. The jobs that have large execution time or large completion time are executed first. The problem is that smaller jobs have to wait for long time [14].
- Opportunistic Load Balancing (OLB) - OLB is a static load balancing algorithm whose goal is to keep each node in the cloud busy so does not consider the current load on each node. It attempts to dispatch the selected job to a randomly selected available VM [17]. However, OLB does not consider the execution time of the task in that node. This may cause the task to be processed in a slower manner increasing the whole completion time (makespan) [25] and will cause some bottlenecks since requests might be pending waiting for nodes to be free [9].
- Load Balance Min-Min (LBMM) [25]- This method uses Min-Min Scheduling algorithm as its base. It uses a three level hierarchical framework. Request manager which is in the first level of the architecture is responsible for receiving the task and assigning it to one service manager in the second level of LBMM. After receiving the request, service manager divides it into subtasks to speed up the processing. Then service manager assigns the subtask to a service node for execution based on different attributes such as the remaining CPU space (node availability), remaining memory and the transmission rate. This algorithm improves the load unbalance of Min-Min and minimizes the execution time of each node but does not specify how to select a node for a complicated task requiring large-scale computation.
- 2-Phase Load Balancing Algorithm - [25] proposed this algorithm combining OLB and LBMM to have a better execution time and to balance the load more efficiently. A queue is used to store tasks that need to be carried out by manager. In the first phase OLB scheduling manager is used to assign job to the service manager. In the second phase LBMM algorithm is used to choose the suitable service node to execute the subtask by the service manager. The

problem associated with this approach is that it applicable only in static environment.

- Honey bee Foraging algorithm [22] - It is a decentralized honeybee-based nature-inspired load balancing technique for self-organization. It achieves global load balancing through local server action [26]. This algorithm is derived from the behavior of honey bees for foraging and harvesting food. Forager bees search for food sources and after finding advertise this using waggle dance to present quality of nectar or distance of food source from hive [29]. Harvester bees then follow the foragers to the location of food to harvest it.

In this approach the servers are grouped under virtual servers (VS) each having its own virtual service queues. Each server processing a request from its queue calculates a profit, which is analogous to the quality that the bees show in their waggle dance. One measure of this reward can be the CPU time spends on processing a request. The dance floor in case of honey bees is analogous to an advert board which is used to advertise the profit of the entire colony.

Each of the servers takes the role of either a forager or a harvester. The server after processing a request can post their profit with a probability pr. A server can randomly choose a VS's queue with a probability px. A server serving a request, calculates its profit and compare it with the colony profit and then sets its px. If this profit was high, then the server stays at the current virtual server. If it was low, then the server returns to the idle/waiting behavior. This approach works well under heterogeneous types of resources but it does not show equivalent improvement in throughput while increasing number of resources.

- Active Clustering [22] – It is a self-aggregation algorithm that works on the principle of grouping similar nodes together and working on these groups. The process consists of iterative execution of the following steps:

• A node initiates the process (initiator node) and selects another node called the matchmaker node from its neighbors satisfying the criteria that it should be of a different type than the former one.

• The matchmaker node then creates a link between one of its neighbour which is of the same type as the initiator node.

• The matchmaker node then removes the link between itself and the initiator node.

If variety of nodes is increased then the algorithm performs poorly compare to the Honeybee foraging approach.

- Double Threshold Energy Aware Load Balancing Algorithm (DT-PALB)[21] – The algorithm has three basic sections. The first one is the balancing section which is responsible for determining where VMs will be instantiated. If all the nodes have utilization between 25%-75% then it finds the least loaded node to assign a new VM.

The second one is the upscale section which is used to power on additional compute nodes (as long as additional nodes are available) during peak period. It is done if all currently active nodes are in operation.

The last one is the downscale section which is responsible for switching off idle compute nodes. If any nodes have utilization less than 25% then it migrates all its VMs to another node so that it can be switched off.

The algorithm balances load efficiently but when the system load is very low there may be many nodes having utilization less than 25%. So the algorithm will start migrating the VMs of these nodes to the most unutilized nodes. But it may happen that after migration the destination nodes still have utilization less than 25% and again the VMs of this node will be migrated to another unutilized node. So this process may initiate unnecessary dynamic migrations of VMs which may lead to performance degradation and high operational cost. Moreover while migration the algorithm finds the destination node based on the current load only. It ignores the associated communication delay which occurs due to increase in the distance between customer and the new location of the VM.

- Join-Idle-Queue - Y. Lua et al. [30] proposed a Join-Idle-Queue load balancing algorithm for dynamically scalable web services. This algorithm provides large scale load balancing with distributed dispatchers by, first load balancing idle processors across dispatchers and then, assigning jobs to processors to reduce average queue length at each processor. By removing the load balancing work from the critical path of request processing, it effectively reduces the system load, incurs no communication overhead at job arrivals and does not increase actual response time.
- Weighted Least Connection - The WLC algorithm [31] assigns tasks to the node based on the number of connections that exist for that node. This is done based on a comparison of the SUM of connections of each node in the cloud and then the task is assigned to the node with least number of connections. However, WLC does not take into consideration the capabilities of each node such as processing speed, storage capacity and bandwidth.
- Exponential Smoothing Forecast based on Weighted-Least Connection (ESBWLC) – This algorithm is proposed by Ren et. al [28] based on the existing WLC algorithm. It improves WLC by taking into account the time series and trials. That is ESBWLC builds the conclusion of assigning a certain task to a node after having a number of tasks assigned to that node and getting to know the node capabilities. ESBWLC builds the decision based on the experience of the node's CPU power, memory, number of connections and the amount of disk space currently being used. It takes advantage of all historical data and distinguishes them through the smoothing factor to let recent data make a greater impact on the predictive value than long-term data. ESBWLC then predicts which node is to be selected based on exponential smoothing.
- Fractal-based Load Balancing Algorithm – [33] presents a dynamic load balancing algorithm based on VM migration. The algorithm proposed the trigger strategy based on the fractal methods which determines the timing of the

VM migration through forecasting, which can avoid the problem of the peak instantaneous load trigger.

The trigger strategy of traditional load balancing algorithm is based on the threshold. When any of the node load indicators vector exceeds specified thresholds, it is defined as overloading. The trigger strategy will create a virtual machine migration to balance the load. Instantaneous peak load is triggered once migration which leads to frequent migration and adds unnecessary overhead. In order to ensure that a small peak instantaneous load does not trigger unnecessary migration, we use the load forecasting program. When an indicator of node exceeds the threshold, instead of immediately triggering the migration, it predicts its future n load value according to its historical load recorded. When at least k values in the prediction are bigger than the threshold, the migration begins.

S.No.	Algorithm	Description	Pros	Cons
1	Round Robin [5][12]	First request is allocated to a randomly picked VM. Subsequent requests are assigned in circular order.	Equal distribution of work load.	Job processing time is not considered.
2	Weighted Round Robin [13]	Weight is assigned to each VM based on its processing capacity. More requests are assigned to the powerful VM.	Better resource utilization.	Processing time of each request is not considered.
3	Dynamic Round Robin [14][21]	This algorithm is based on two rules : i) Retiring State of VM and ii) Retirement Threshold and migration of VMs.	Reduces the power consumption cost	Does not scale up for large data centers
4	Randomized [17]	Randomly assigns the selected jobs to the available VM.	Very simple Algorithm	Current load is not considered.
5	ESCE Algorithm [14][15][16]	Request is assigned to any available VM that can handle it. If there is an overloaded VM then the balancer distributes some of the tasks to some idle VM to balance the load.	Response time and processing time of a job is improved.	Not fault tolerant because of single point of failure.
6	Throttled Load Balancing Algorithm [12]	Record of the state of each VM (busy/idle) is maintained. Request is accepted if a match is found in the table otherwise -1 is returned and the request is queued.	TLB tries to distribute the load evenly among the VMs.	Does not consider the current load on VM.
7	Modified Throttled [18]	Unlike Throttled here the index table is searched from the next to already assigned VM.	Gives better response time.	State of index table may change during next allocation.
8	Central Load Balancer [38]	VM allocation is like Throttled but based on priority which calculated using CPU speed and memory capacity of VM.	Suitable for heterogeneous environment.	Priority is fixed and bottleneck problem.
9	AMLB Algorithm [19]	The number of requests currently allocated to each VM is maintained. Request is allocated to the least loaded VM.	Consider both load and availability of VM.	Processing power of VM is not considered.
10	VM-Assign LB Algorithm [20]	Similar to the above algorithm but unlike it VM is assigned if it is available and not used in the previous assignment.	Utilizes all the VMs completely and properly.	It is not clearly mentioned in the paper that how it happens.
11	Weighted Active Monitoring LB Algorithm [12]	Weights are assigned to VMs depending on their available processing power. Task is assigned to the least loaded and most powerful VM.	Considers both the load as well as the processing power of available VMs.	Assigning weight increases the complexity of the algorithm.
12	Biased Random Sampling [22][23]	The sampling walk starts at a specific node of a virtual graph and moves to a randomly chosen neighbour. The last node in the walk is selected for the load allocation.	Decentralized method so suitable for large network systems.	Not suitable for dynamic environment.
13	Min-Min Algorithm [24]	The smallest task is assigned to the fastest resource. The task is removed from the set and same process is repeated.	The method is simple.	Does not consider the existing load on a resource.
14	Load Balance Improved Min-Min Scheduling Algorithm (LBIMM) [24]	First step is same as Min-Min algorithm. The completion time of the smallest task from the most heavily-loaded resource is calculated on all other resources. If it is less than makespan of Min-Min then the task is reassigned to the resource that produces it. The same process is repeated.	Produces a schedule which improves load balancing and also reduces the overall completion time.	Does not consider priority of a job while scheduling.
15	User-Priority Awared LB Improved Min-Min Scheduling[24]	To incorporate priority it first divides all the tasks into two groups G1 (higher priority) and G2. G1 is scheduled first using the Min-Min algorithm to assign the tasks to the VIP qualified resource set. Then G2 is scheduled.	Not only balances the load but also consider makespan and user-priority of each task.	Does not consider the deadline of each task.
16	Max Min Algorithm [14]	Works as the Min-Min algorithm. But jobs having large execution time are executed first.	Reduces the makespan.	Smaller jobs have to wait for long time.
17	OLB [9][17][25]	Static load balancing algorithm. Attempts to dispatch the selected job to the available VM chosen randomly.	Keeps each node in the cloud busy.	Does not consider the execution time of the task.
18	Load Balance Min-Min (LBMM) [25]	It uses a three level hierarchical framework. Request manager which is in the first level of the architecture receives task and assigns it to one service manager in the second level. Service manager divides the task into subtasks and assigns them to service nodes for execution.	Improves the load unbalance of Min-Min and minimizes the execution time of each node.	Does not specify how to select a node for a complicated task requiring large-scale computation.
19	2-Phase Load Balancing Algorithm [25]	Combination of OLB and LBMM. In the first phase OLB scheduling manager is used to assign job to the service manager. In the second phase LBMM algorithm is used to choose the suitable service node to execute the subtask.	Better execution time and balance the load more efficiently.	Applicable only in static environment.

20	Honey bee Foraging[22][26]	Decentralized honeybee-based nature-inspired load balancing technique for self-organization.	Works well under heterogeneous resources	Increase in resources does not improve throughput equally.
21	Active Clustering [22]	Self-aggregation algorithm that works on the principle of grouping similar nodes together and working on them.	Balances load efficiently.	Performs poorly in heterogeneous environment.
22	Double Threshold Energy Aware Load Balancing Algorithm [21]	The algorithm has three basic sections: Balancing section, determines where VMs will be instantiated. Upscale section, powers on additional compute nodes during peak period and downscale section, switches off idle nodes.	Balances load efficiently.	May initiate unnecessary migrations of VMs and while migration communication delay is not considered.
23	Join-Idle-Queue [30]	First load balancing idle processors across dispatchers and then, assigning jobs to processors to reduce average queue length at each processor.	Reduces system load. No communication overhead at job arrivals.	More power consumption.
24	Weighted Least Connection [31]	Assigns tasks to the node having least number of connections.	Balances load efficiently.	Processing speed and storage capacity are not considered.
25	Exponential Smoothing Forecast based on WLC (ESBWLC) [28]	Predicts assignment of task based on the experience of the node's CPU power, memory, number of connections and the amount of disk space currently being used. It uses smoothing factor to let recent data make a greater impact on the predictive value than long-term data.	Capabilities of each node are considered.	Complex calculations are involved.
26	Fractal-based Load Balancing Algorithm [33]	Dynamic load balancing algorithm based on VM migration. Timing of VM migration is determined through load forecasting method.	Peak instantaneous load does not trigger unnecessary migrations.	Based on a threshold value.

Table I. Comparison of Different Load Balancing Algorithms

IV. BALANCING LOAD USING VM MIGRATION TECHNIQUES

VM migration mechanisms enable cloud providers to adjust the load at each server. Due to underload or overload of the host VM may be migrated to another server and may continue execution there. This process is called VM migration and may occur multiple times during task execution for efficient load balancing. Migration is transparent to the applications. VM is migrated along with its current state of execution so that it can resume its execution there. After the completion of task the result is transferred to the user by the data center where the user has submitted his task [2].

Migration helps in load balancing throughout the system. There are two types of live migration mechanisms, precopy and postcopy. Postcopy migration transfers a VM's memory contents after its processor state has been sent to the target host. It differs from the precopy approach, which first copies the memory state to the destination, through a repetitive process, after which its processor state is transferred to the target [32]. However, there is a delay associated with each migration comprising of the time required for the VM [2]

- To stop execution at the current server
- To move the accompanying data to the new one
- To initialize the new VM.

Cloud provider wants to fulfill its users QoS requirements by minimizing latency and execution time. At the same time it wants to reduce its operating cost by fully utilizing the available resources and turning off the underutilized server. To achieve all these migration of VM is necessary. So we have to decide: I) When to migrate II) Which VM to migrate and III) Where to migrate [7].

In [34] it is proposed that a migration should be performed only when the QoS requirement of task is violated for a long enough period. As a rule of thumb, a VM migration is beneficial if the anticipated execution time at the new server is less than that at the current server.

There are many challenges in designing efficient migration mechanisms for the cloud [2], such as –

- Workload Uncertainty – The demand of resource varies in the cloud environment due to unpredictable arrival of new tasks and completion of old ones.
- Unpredictability of Multitenancy Effects – Due to continuous change in workload, resource availability also varies dynamically. Hence, multitenancy unavoidably leads to unpredictable contention of physical resources.
- Unknown Evolution of Accompanying Data Volume – While moving a task from one server to another, knowledge about its data evolution pattern is required because the task must be moved along with its data. So a task having little amount of data associated with it is preferred to a task having higher volume of data. Migration within the same data center is always better since it is faster and less costly due to high speed LAN connection.
- Partial availability of Cloud-Related Information – The cloud provider may not have information about complexity and data volume evolution of each task (which is generally known by the user). But for an efficient migration the provider requires information about the cloud infrastructure as well as the details of all tasks.

Modes of VM Migration –

- Centralized Migration Policies - Migration leads to improved performance for the remaining tasks in the source node while due to increase in multitenancy overhead the destination server may suffer from performance degradation. However, an efficient migration scheme should

increase the overall performance of the system. Selecting optimal migration requires a search over a three-dimensional space; for each server of the system and for each of its active tasks the performance gain of a migration to each candidate server has to be estimated. However, instead of an exhaustive search specific characteristics of cloud can be considered to reduce the search space by any efficient migration strategy.

- Data Volume – Data evolution pattern may be increasing, decreasing or constant. If two tasks have generated same amount of data then priority must be given to the task having increasing data volume pattern since the volume of the data generated will increase as its processing advances.
- Residual processing burden – Task which is very close to completion should not be migrated. In contrast, a task having more remaining processing time should be migrated so that it can better use the resources of the destination server.
- Multitenancy – Preferably migrate task causing significant multitenancy cost. e.g. Multiplexing CPU-intensive tasks with network intensive ones generally reduces the multitenancy overhead.

- Server-Initiated Migration – It minimizes the information that has to be circulated within the cloud. A migration may be initiated whenever a server detects that it is overloaded or an SLA agreement is about to be violated. Here a two-dimensional search is performed; for each active task it calculates the anticipated gain for each possible migration and pick the one with maximum gain.
- Task-Autonomic Migration – The key underlying idea is that a migration should only occur if it is beneficial for the anticipated execution time of the task itself including the delay incurred in the migration. Here each task has to perform a one-dimensional search over the set of candidate servers.

Different VM Migration Techniques –

The following three steps are executed for migrating a VM from one host to another.

i) Host Selection – To distribute the load evenly the server selected should be either overloaded or underloaded. Single or double threshold policy can be used for overloaded or underloaded host detection. But these static threshold policies do not work well in a highly dynamic heterogeneous environment having different types of workload. So Beloglazov et. al. have proposed [35] algorithms like Median Absolute Deviation (MAD), Interquartile Range (IQR), Local Regression and Robust Local Regression to find an adaptive utilization threshold. These methods help in overloaded host detection. The hosts with minimum utilization compare to other hosts are considered as underloaded by the author of [35].

ii) VM Selection – After selection of overloaded host the next step is to find suitable VMs for migration. Different VM selection policies exist in the literature. Minimization of Migration (MM) Policy, Highest Potential Growth Policy, and Random Choice Policy are proposed by Beloglazov et. al. in [36]. In their more recent paper [35] they have proposed two more VM Selection methods, Minimization Migration Time Policy (MMT) and Maximum Correlation (MC) Policy.

iii) Destination Host Selection (VM Placement) – The destination host must not become overloaded after the placement of the migrated VM. Authors of [36] have used a modified version of Best Fit Decreasing (BFD) Algorithm which places the VM in the host which provides least increase in power consumption after this allocation. Other metrics like fulfillment of QoS requirement can also be considered during VM Placement.

VM migrations can continue to significant savings in energy consumption by:

- Reducing the number of active servers by appropriate concentration of tasks on fewer physical machines with the aid of virtualization (consolidation).
- Reducing the energy consumption of individual servers by moving the processes from heavily-loaded to less loaded servers (load balancing).

The main challenge is that migration has contradicting consequences. By reducing number of active servers load is concentrated on fewer servers. Again load balancing activate more servers to accommodate the load properly. So the final choice will depend upon the relative amount of energy consumption by the servers when they are idle or loaded.

V. CONCLUSION

In this paper, we surveyed multiple algorithms for load balancing in Cloud Computing. Amazon EC2 uses Elastic Load Balancing to automatically distribute incoming application across multiple Amazon EC2 instances [37]. It not only balances the load but also provides greater levels of fault tolerance and high scalability.

Load balancing has two meanings: first, it puts a large number of concurrent accesses or data traffic to multiple nodes respectively to reduce the time users waiting for response; second, it put the calculation from a single heavy load to the multiple nodes to improve the resource utilization of each node [33]. Allocation strategy is different for different application environments, such as e-commerce sites need CPU idle nodes because these calculations are big, while database applications to read and write frequently will need to be allocated some I/O idle node [33]. We have discussed the pros and cons associated with several load balancing algorithms. The challenges of these algorithms are addressed so that more efficient load balancing techniques can be developed in future. VM migration is an essential component of load balancing because VMs have to be moved to deal with the problem of over-utilization and under-utilization of resources. The algorithms described in this paper not only balance the load but also help in efficient

utilization of resources, increase overall throughput and decrease the response time. All these will reduce the operational cost and will attract more users towards cloud computing.

REFERENCES

[1] Peter Mell, Timothy Grance. The NIST Definition of Cloud Computing (Draft). NIST. 2011.

[2] Lazaros Gkatzikis, Iordanis Koutsopoulos, "Migrate or Not? Exploiting Dynamic Task Migration in Mobile Cloud Computing Systems", IEEE Wireless Communications 2013, pp. 24-32.

[3] Raghavendra Achar, P. Santhi Thilagam, Nihal Soans, P. V. Vikyath, Sathvik Rao and Vijeth A. M., "Load Balancing in Cloud Based on Live Migration of Virtual Machines ", 2013 Annual IEEE India Conference (INDICON) .

[4] Akshay Jain, Anagha Yadav, Lohit Krishnan, Jibi Abraham , "A Threshold Band Based Model For Automatic Load Balancing in Cloud Environment ".

[5] Amandeep Kaur Sidhu, Supriya Kinger , "Analysis of Load Balancing Techniques in Cloud Computing ", International Journal of Computers & Technology , Volume 4 No. 2, March-April, 2013, pp. 737-741.

[6] Dzmitry Kliazovich, Sisay T. Arzo, Fabrizio Granelli, Pascal Bouvry and Samee Ullah Khan3, "e-STAB: Energy-Efficient Scheduling for Cloud Computing Applications with Traffic Load Balancing ", IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing 2013, pp. 7-13.

[7] Anton Beloglazov, Rajkumar Buyya "Energy Efficient Resource Management In Virtualized Cloud Data Centers," 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010, pp. 826-831.

[8] X. Evers , "A Literature Study on Scheduling in Distributed Systems ", 1992.

[9] Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi and Jameela Al-Jaroodi , "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms ", IEEE Second Symposium on Network Cloud Computing and Applications , 2012, pp. 137-142.

[10] Thomas L. Casavant, Jon G. Kuhl, "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems ", IEEE Trans, on Software Eng., vol. 14, no. 2, Feb. 1988, pp. 141-154.

[11] Bhathiya Wickremasinghe, Rodrigo N. Calheiros, Rajkumar Buyya,"CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications", 20-23, April 2010, pp. 446-452.

[12] Jasmin James, Dr. Bhupendra Verma, "Efficient VM Load Balancing Algorithm for a Cloud Computing Environment ", International Journal on Computer Science and Engineering (IJCSE) , Vol. 4 No. 09 Sep 2012 , pp. 1658-1663.

[13] Qi Zhang, Lu Cheng, Raouf Boutaba; Cloud computing: sate-of-art and research challenges; Published online: 20th April 2010, Copyright : The Brazillian Computer Society 2010.

[14] M.Aruna , D. Bhanu, R.Punithagowri , "A Survey on Load Balancing Algorithms in Cloud Environment ", International Journal of Computer Applications, Volume 82 – No 16, November 2013 , pp. 39-43.

[15] Tanvee Ahmed, Yogendra Singh "Analytic Study Of Load Balncing Techniques Using Tool Cloud Analyst" , International Journal Of Engineering Research And Applications, 2012, pp. 1027-1030.

[16] Subasish Mohapatra, K.Smruti Rekha, Subhadarshini Mohanty , "A Comparison of Four Popular Heuristics for Load Balancing of Virtual Machines in Cloud Computing ", International Journal of Computer Applications, Volume 68– No.6, April 2013 , pp. 33-38.

[17] Isam Azawi Mohialdeen , "Comparative Study of Scheduling Algorithms in Cloud Computing Environment ", Journal of Computer Science , 2013, pp. 252-263.

[18] Shridhar G. Domanal, G. Ram Mohana Reddy, "Load Balancing in Cloud Computing Using Modified Throttled Algorithm", IEEE, International conference. CCEM 2013.

[19] Hemant S. Mahalle, Parag R. Kaveri, Vinay Chavan, "Load Balancing On Cloud Data Centres", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, issue 1, January 2013.

[20] Shridhar G.Damanal and G. Ram Mahana Reddy , "Optimal Load Balancing in Cloud Computing By Efficient Utilization of Virtual Machines ", IEEE 2014.

[21] Jayant Adhikari, Prof. Sulabha Patil, "Double Threshold Energy Aware Load Balancing in Cloud Computing", 4th ICCCNT, July 2013.

[22] Martin Randles, David Lamb, A. Taleb-Bendiab , "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing ", IEEE 24th International Conference on Advanced Information Networking and Applications Workshops , 2010, pp. 551-556.

[23] O. Abu- Rahmeh, P. Johnson and A. Taleb-Bendiab, "A Dynamic Biased Random Sampling Scheme for Scalable and Reliable Grid Networks", INFOCOMP - Journal of Computer Science, VOL.7, N.4, December, 2008, pp. 01-10.

[24] Huankai Chen, Professor Frank Wang, Dr Na Helian, Gbola Akanmu, "User-Priority Guided Min-Min Scheduling Algorithm For Load Balancing in Cloud Computing", IEEE, 2013.

[25] Shu-Ching Wang, Kuo-Qin Yan, Wen-Pin Liao, Shun-Sheng Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network", 3rd International Conference on Computer Science and Information Technology (ICCSIT), Vol. 1, IEEE, 2010, pp. 108-113.

[26] Nidhi Jain Kansal, Inderveer Chana, "Cloud Load Balancing Techniques : A Step Towards Green Computing", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012, pp. 238-246.

[27] Jinhua Hu, Jianhua Gu, Guofei Sun, Tianhai Zhao, "A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment", 3rd International Symposium on Parallel Architectures, Algorithms and Programming, IEEE, 2010, pp. 89-96.

[28] Xiaona Ren, Rongheng Lin,Hua Zou, "A Dynamic Load Balancing Strategy for Cloud Computing Platform Based on Exponential Smoothing Forecast", Proceedings of IEEE CCIS2011, pp. 220-224.

[29] Seyed Mohssen Ghafari, Mahdi Fazeli, Ahmad Patooghy, Leila Rikhtechi, "Bee-MMT: A Load Balancing Method for Power Consumption Management in Cloud Computing", IEEE 2013, pp. 76-80.

[30] Y. Lua, Q. Xiea, G. Kliotb, A. Gellerb, J. R. Larusb, and A. Greenber, "Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services", An international Journal on Performance evaluation, August 2011.

[31] Lee, R. and B. Jeng, "Load-balancing tactics in cloud," International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), October 2011, IEEE, pp. 447-454.

[32] VIOLETA MEDINA, JUAN MANUEL GARCIA, "A Survey of Migration Mechanisms of Virtual Machines", ACM Computing Surveys, Vol. 46, No. 3, Article 30, January 2014, pp. 30:1-30:33.

[33] Haozheng Ren, Yihua Lan, Chao Yin, "The Load Balancing Algorithm in Cloud Computing Environment", 2nd International Conference on Computer Science and Network Technology, IEEE, 2012, pp. 925-928.

[34] T. Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif et, "Black-Box and Gray-Box Strategies for Virtual Machine Migration," Proc. 4th USENIX Conf. Networked Systems Design & Implementation, NSDI, 2007, pp. 229–42.

[35] Anton Beloglazov and Rajkumar Buyya, "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers", Concurrency and Computation: Practice and Experience, 2012, vol.24, pp. 1397-1420.

[36] Anton Beloglazov, Jemal Abawajy and Rajkumar Buyya, "Energy-aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing", Future Generation Computer Systems, Elsevier,2011, pp. 755-768.

[37] http://aws.amazon.com/elasticloadbalancing

[38] Gulshan Soni, Mala Kalra, "A Novel Approach for Load Balancing in Cloud Data Center", International Advance Computing Conference (IACC) IEEE , 2014, pp.807-812.