Systematic Review on Existing Load Balancing Techniques in Cloud Computing

Danlami Gabi
Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia

Abdul Samad Ismail
Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia

Anazida Zainal
Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia

ABSTRACT
Cloud computing requires more reliable, efficient and scalable load balancing algorithm to survive. As one of the main challenges in cloud computing, load balancing facilitate dynamic workload across multiple nodes ensuring that no single node get overloaded. With proper load balancing, resource consumption is maintained at minimum level, enabling scalability, avoiding bottleneck and overprovisioning etc. In this paper, systematic review on existing load balancing techniques currently prevalent in cloud computing was carried out. Load balancing metrics; Response time, Performance, Resource Utilization, Throughput, Cost Overhead, Scalability, Fault Tolerant and Migration Time were used to evaluate the existing techniques. Findings show that the existing techniques mainly focus on reducing response time, completion time, cost and improving throughput. Neither of the techniques was able to unveil efficient load balancing of task scheduling for single and federated cloud environment. However, research such as load balancing of energy consumption, server consolidation, Virtual Machine Migration, are not taken into consideration by the existing techniques. Future research is to unveil efficient multi-objective load balancing of tasks scheduling algorithm with quality of service improvements for homogeneous and federated heterogeneous cloud environment.

Keywords
Cloud Computing, Load Balancing, Task Scheduling, Federated Cloud.

1. INTRODUCTION
Load balancing appears to be the major challenge in cloud computing, due to heterogeneous nature of cloud environment where resource pool is on increase. To further overcome challenges of resource imbalance and deadlock in cloud computing, load balancing is required to share task to appropriate virtual machines in datacentres with actual processing capabilities to ensure immediate processing. Providing an efficient load balancing in cloud computing enables efficient resource utilization, achieved higher user satisfaction and also prioritizes users by applying appropriate scheduling criteria [12]. It helps in preventing bottlenecks that may arise as a result of load imbalance [11] caused by overprovisioning or under-provisioning of computing resources such as memory, processing elements, bandwidth, processor speed etc. Efficient load balancing avoid resource wastage and ensure energy is consumed at minimum. When one or more components of any service fails, load balancing implements fair-over for service continuity [23]. Virtual machine sometimes become over saturated with processing capabilities, most especially during peak hours, while other virtual machines can remain underutilized due to nature that may arise in terms of resource allocation policy. Several research [10,13,19,25,] were conducted to fine-tune load balancing policy to enable sharing of workload across datacentres, so as to leased VMs that are over utilized and engage those that are underutilized. Consideration to the heterogeneous nature of computing task and its high demand in cloud has made selecting virtual nodes to execute task a major problem. Therefore, distributing workload across datacentres to achieve better utilization of resources at the same time, minimize data processing time to avoid overloading, will leased VMs [16] and increase their makespan effectively.

The main objective of this systematic review, is to unveil existing load balancing techniques prevalent in cloud computing, improved on their capabilities and proposed efficient load balancing algorithm that will take in to account the overall network loads, energy efficiency with better quality of service satisfaction. The rest of this paper is organized as follows: Section 2 discussed two types of load balancing algorithm in cloud. Section 3 discussed the need for load balancing in cloud computing. Section 4 provided a systematic review on the existing load balancing techniques and their publication fora. Section 5 discussed the metrics used for evaluating the load balancing techniques. Section 6 is a comparison and analysis carried out on the existing techniques. Section 7 concluded the paper and provided future direction.

2. LOAD BALANCING IN CLOUD COMPUTING
Load balancing is a viable method that eased VMs and datacentre overloaded with computing jobs, through sharing loads across datacentre infrastructures in other to achieve a well organised system performance[4]. Their classification is based on to two type's algorithms:

- ***Static load balancing algorithms****:* These algorithms portray simplicity in term of implementation and overhead. It outperforms better in homogeneous environments with high communication speeds, ignoring communication delays [20]. Constants monitoring of network nodes is ignored by this algorithms. It works properly where variation in loads is minimal to virtual machines. The inability of this algorithms to provide fault tolerance ruled out it implementation in a dynamic cloud environment.
- ***Dynamic load balancing*****:** Dynamic algorithms make changes when distributing workloads amongst nodes at run-time. The distributions of workloads are

performed with the current state of VM information on its capacity to fast track next distribution decision [4]. It functions in reducing communication delays and execution time which makes it more robust to adapts in cloud environments. Several policies necessitate the basis for dynamic load balancing in cloud computing [4] as highlighted in table 1.

Table 1: Load balancing policy for resource provisioning

Serial No	Description
i	*Resource type policy*: This classifies resources based on either server or receiver of tasks according to its availability status.
ii	*Triggering policy*: The triggering policy helps to determine the actual time a load balancing is expected.
iii	*Information policy*: Determines what workload information are to be collected, when is to be source and from where will it be sourced.
iv	*Location policy*: It determines what available resources are there to provide enabling services with a based selection of a suitable server or receiver.
v	*Selection policy*: Specifies the task that should be migrated from over utilized resources to underutilized resources.

3. WHY LOAD BALANCING IN CLOUD COMPUTING?

The benefits of load balancing in cloud computing helps to ensure that customers' needs are met at the appropriate time, keeping the makespan of the system at better state, ensured better throughput, energy consumption and also coordinate the management of resource pool across cloud environment. When datacentre becomes overload with processing task, the benefit of load balancing migrate task to available datacentres that are underutilized for processing. Load balancing helps to boost productivity of cloud datacentre while maintaining service level agreement (SLA). Some benefits of load balancing in cloud computing are to ensure that;

- Available of resources on demand.
- Resources are effectively utilized under condition of high or low load.
- Energy is saved in case of low load.
- Reduce Cost.
- Save processing time.
- Overcome resources wastages.
- Ensure better communication across network nodes.

4. SYSTEMATIC REVIEW ON EXISTING LOAD BALANCING TECHNIQUES IN CLOUD COMPUTING

Current load balancing techniques discussed below are prevalent in clouds:

4.1 A2LB: Autonomous Agent Based Load Balancing

[20], proposed an A2LB algorithm for dynamic load balancing. The algorithm uses three agents; Load agent, Channel agent and Migration agent demonstrating its load balancing. The load and channel agents are static while Migration agents represent an ant. The load agent controls information policy and maintained all details of datacentre and at the same time responsible for calculating the load on every available virtual machine (VM) when new task are allocated in a datacentre. The load agent is supported with a VM_Load Fitness table. The fitness table maintained the list of all details of virtual machines properties in a datacentre such as id, memory, and fitness value and load status of all virtual machines. When the load agent completed the controlling policy, the channel agent controls the transfer policy, selection policy and location policy. Upon request received from the load agents, the channel agent initiates a communication with the migration agents. The migration agent then moved to other datacentres and communicates with load agent of the datacentre to enquire about the status of VMs presents. Upon receiving the status, it communicates to its parent channel agent. This approach reduces service time and overcome the challenge of overloaded virtual machines.

4.2 HBBLBP: Honey Bee Behaviour Load Balancing with Pareto Dominance.

[18], in their attempt to provide a more robust techniques, they proposed load balancing algorithm that balances load across virtual machines (VMs) through mapping tasks to under loaded VMs based on foraging behaviour of honey bees and if there is more than one under loaded VMs, its selects the cost efficient one using Pareto dominance strategy. Their algorithm do not consider priority as a basis for VMs selection, instead, they took the cost of VMs and expected running time of the task for selecting the most optimal VM. In using the concepts of Pareto dominance, they selected the optimal VM by comparing the cost of executing a task on one VM with that of all other VMs. Upon comparing, VM with the minimum value of minimization function is computed based on running time and monitoring cost. Their method portrays significant improvement in reducing cost of using VM and execution time.

4.3 Minsd: Minimized Standard Deviation for Cloud –

[8], unveiled a load balancing algorithm called Minsd that works at three levels of control; Datacentres, Hosts and processing elements (PEs). When new task (cloudlet) comes from user, it submits to VM that are under loaded. The balancing is controlled by the physical resources such as "Datacentres", "Host" and "Pes". In an interval of time, the resource entity checks its load and classifies them into three categories; under loaded, overloaded and normal loaded. Likewise, the resource entity classified the resource type (Datacentre, Host, PE) based on "LowerLimit", "UpperLimit", and Load. If the load on "Datacentre, Host, PE" are more than the "UpperLimit", the state is overloaded.

If the load on Datacentre, Host, PE, is less than the "LowerLimit", the state of "Datacentre, Host, PE" is under loaded; otherwise the state is normal. They further used a standard deviation to calculate the load on resource entity. In their narration, if the standard deviation of a method is small, it means the difference of each load is small. The small standard deviation tells that the load of the entire system is balanced. The adopted method shows good performance in makespan, and communication overhead.

4.4 VM-assign: VM-assign Load Balancing Algorithm –

[5], proposed a static load balancing algorithm called VM-assign. The algorithm focuses on mainly finding the least loaded virtual machine and how incoming jobs are allocated intelligently. The load balancer purported in the algorithm maintains an index assign table of virtual machines and the loads of virtual machines. A round robin selection criteria is followed by the algorithm to select the least loaded VM. The load balancer checks the VM table, if a VM is not used in the previous assignment, it then assigned with request and its id is returned to the datacentre otherwise, the balancer find the next leased loaded VM. Their algorithm balanced load across virtual machine and ensures efficient resource utilization.

4.5 Hybrid Scheduling Algorithm –

[2], proposed a hybrid scheduling algorithm based on throttled and equally spread current execution load balancing algorithm (ESCE). The hybrid algorithm initializes all VMs allocation status to AVAILABLE in the VM state List. With the "hashmap" list size available to determine overloaded and under loaded VM. When a data centre receives a new request to be processed, the DataCenterController queries new load balancer for next allocation. If the "hashmap" list size is less than the state list size, a VM is allocated otherwise the load balancer has to wait until the virtual machine gets free. When the request is been processed by a datacentre, and "DataCenterController" receives the cloudlet response, it notices the load balancer of the VM de-allocation. Finally, the load balancer updates the status of the VM in VMs state list and "hashmap" list. A CloudAnalyst was used to implement the algorithm. The result shows a significant improvement in response time, and the datacentre processing time as well as the cost of processing request compared to the existing load balancing algorithm (RR, ESCE, and TLB).

4.6 Cloudlet Allocation Algorithm –

[3], unveiled a new cloudlet allocation strategy for load balancing that improved quality of service (QoS) of a cloud in association with two basic parameters; completion time and makespan. In the proposed algorithm, all VMs are sorted in descending order according to their MIPS (Million Instruction per Second) and stored in *vm_list* array. The cloudlets are also sorted in descending order according to their Million Instruction (MI) value and placed in *cloudlet_list.* The cloudlets are then assigned to the VMs guided by the First Come First Serve (FCFS) policy following the process in which the highest capacity VM are allocated first and after allocation, the remaining load capacity (RLC) of the VM are compared with the maximum load capacity (MLC) of other VMs in the *vm_list.* If the comparison returns greater RLC value than the MLC value of other VMs, then the VM may continue with further cloudlet allocation until RLC value becomes lesser than the MLC value of other VMs. The process will continue until all cloudlets in *cloudlet_list* get assigned into the VMs. The adopted strategy shows significant improvement in completion time of the cloudlets as well as improved the maksspan of the VMs and the host datacentre when compared to Round Robin and Conductance Algorithm.

4.7 Nature Inspired Preemptive Algorithm

The nature inspired pre-emptive task scheduling for load balancing in cloud datacentre was presented by [21]. Their algorithm considered balancing pre-emptive independent task on virtual machines (VMs) based on honey bee foraging behaviour. To consider the pre-emption of task based on priority, when virtual machine are over loaded, the task with the highest priority is removed to under loaded virtual machine. Upon arrival of the migrated task to the under loaded virtual machine, its priority and the required execution time is then compared with the priority and the execution time of the already running task in the under loaded virtual machine. If the migrated task has higher priority and the execution time is less than the already running executing task, then the executing task is paused and pre-empted until migrated task finishes executing, the pre-empted task now return to the state of execution, otherwise, the running task has to finish executing. This approach works better for a static cloud environment where high speed connection is required.

4.8 Genetic Algorithm –

[6], proposed a genetic algorithm for scheduling and load balancing for Static parallel heterogeneous systems. Their techniques considered five main factors; *Encoding, Generation of Initial Population, Fitness Function, Selection Operator,* and *Crossover Operator.* Each chromosome is considered as a sequence variety of tasks. Each task is considered as a gene. A generation of an initial random population for entry into the first generation was done by the genetic algorithm. Random generator functions of chromosomes are employed. Individual are selected according to their fitness value. Once fitness values have been evaluated for all chromosomes, good chromosomes is selected through rotating roulette wheel strategy. This operator generate next generation by selecting best chromosomes from parents and offspring. Crossover operator randomly selects two parent chromosomes (chromosomes with higher values have more chance to be selected) and randomly chooses their crossover points, and mates them to produce two child (offspring) chromosomes. However, their approach was able to reduce response time and execution time when compared with LPT, SPT and FIFO algorithms.

4.9 CLB: Central Load Balancer

[22], in their attempt to provide a load balancing algorithm, they proposed a "Central Load Balancer" that balances load among virtual machines in cloud datacentre. Two key factors were adopted for load balancing; "DataCenterController" and the "Central Load Balancer". The states of the virtual machine are maintained as "BUSY" and "AVAILABLE". In this technique, every request from user bases arrived at "DataCenterController". The "DataCenterController" queries the Central Load Balancer for allocation of requests. The Central Load Balancer maintains a table that consist of id, states and priority of all virtual machines. It parses the table and find out highest priority virtual machine, then check its states. If the virtual machine state is "AVAILABLE", then the id of the VM (VMid) is returned to the "DataCenterController". If the state of virtual machine is "BUSY" it chooses next less high priority virtual machine. Finally, "DataCenterController" assigns the request to that

VMid that is provided by Central Load Balancer (CLB). The Central Load Balancer (CLB) is connected to all users and virtual machines present in cloud datacentre through the "DatacenterController". The Central Load Balancer calculates the priorities of virtual machines based on CPU speed (MIPS) and memory. The algorithm was implemented using CloudAnalyst and the results shown significant improvement when compared to the existing load balancing techniques (RR, ESCE, TLA) in term of response time.

4.10 Heuristic Based Load Balanced Scheduling.

[9], proposed heuristic based load balanced scheduling model for efficient execution of tasks across virtual machine. In their load balancing techniques, 4 VMs and 16 cloudlets are considered. Initial entries for VMs with their "ids" and their processing power are also considered. The VM that has the lowest value of VM power is considered to be more powerful. Based on FCFS (First Come First Served) policy, the first Cloudlet goes to VM 'n2' where corresponding entry in VM power is marked bold. After this binding, power of n2 is calculated. The power value helps to determine if a virtual machine at that instance can be able to execute a given cloudlets and if not, the VMs table is indexed to find the available VM to execute the available task. The technique shows improvement in term of response time and turnaround time.

4.11 PSO: Particle Swarm Optimization Load Balancing –

[1], proposed a particle swarm optimization (PSO) load balancing algorithm that considered number of incoming jobs and available virtual machines. All incoming jobs are allocated to each virtual machine in such a manner that the load on each virtual machine is distributed among VMs uniformly, VM PSO load balancer maintains the index / assign table of VMs which has the number of requests currently allocated to each VM. The Index table is parsed and least loaded VM is selected and correspondingly, the local best (LB) is compared with the VMCount of the VM. VM PSO load balancer then updates the datacenter by returning the VM "id". Request is assigned to the VM after updating the global best (GB). The global best is used to determine the best available under loaded virtual machine in term of load. Datacenter notifies the proposed PSO load balancer about the allocation. The request held by each VM is updated by PSO load balancer. The technique was later implemented using CloudAnalayst and result shows improvement in average response time.

4.12 STR: Server Throughput Restriction

[24], proposed load balancing policy called "Server Throughput Restriction (STR)", based on M/G/s/s+r queueing model that comprises of two concurrent phases: the calculation phase and the scheduling phase. The calculation phase is responsible for data collecting that includes the deployment of applications and their QoS demands, as well as throughput that updates each server's throughput restriction when its corresponding information has changed. For each incoming request, STR first extracted the application context that the request belongs to from its HTTP head, the STR look up the request weight and the server list that the application is deployed from local configuration file, then dispatch the request to the server that meet throughput restriction and update the server's available throughput. If no server is available, the request is rejected by the load balancer as its admittance would lead to performance deterioration of some applications. This approach provides guarantee to each application's mean response time and also achieve better server throughput. A comparison was made with Round Robin and Least-Work-Remaining, result shows a better response time and improved throughput.

4.13 ALBA: Adaptive Solution Load Balancing

[14], proposed an adaptive solution load balancing algorithm called "ALBA" that introduces intelligent agents at two levels in cloud computing model. Every datacentre comprises of series of virtual machines situated at the physical machines. The virtual machine load balancer agent is responsible for keeping an eye on status of every virtual machine in terms of its load. These enable it to maintain information about the availability of resources on virtual machines, response time and their queue lengths. A repository agent (RA) is set to work at global level, which keeps the record of all available virtual machines in a datacentre and collaborate with available virtual machine load balancer agent of various datacentres. The VMLBA is then provided to maintain a log file, keeping the record of current job executing on a VM and status of previously executed tasks that helps to calculate average waiting time and throughput of a VM. Whenever user request has to be allocated some resources on virtual machine, RA consulted further checks with VMLBA to know present status of VM at a datacentre. VMLBA only indicates nodes having loaded less than predefined threshold value. In this case, the adopted datacentre should have the minimum data transfer time. This approach was implemented using CloudSim and shows significant improvement in term of response time compared to Round Robin and Throttled algorithm.

4.14 TBSLB-PSO: Task based System Load Balancing Using Particle Swarm Optimization

[15], discovered task based system load balancing for cloud computing using particle swarm optimization (TBSLB-PSO). The algorithm achieves system load balancing by transferring extra tasks from an overloaded VM instead of migrating the entire overloaded VM. In this approach, central task scheduler (CTS) is responsible for task transfer from an overloaded VM to new similar VM. The CTS finds appropriate VMs as new hosts for the extra tasks of the overloaded VMs. Task are migrated from the overloaded VM to the selected host. The CTS find an optimal way and allocate extra tasks to the new host VMs with less task execution and task transfer time. The experiment was evaluated using CloudSim simulator and results shows minimum task execution time and task transfer time.

4.15 EFLB: Estimated Finish time Load Balancer

[7], proposed a new load balancing algorithm called estimated finish time load balancer (EFLB) that takes into account, the current load of the virtual machine in a datacenter and the estimated finish time of a task before any allocation. In this technique, tasks are randomly assigned to each virtual machine and their processing time in the allocation table is initialized. At the arrival of new queries, on the basis of their characteristic: the algorithm classifies the tasks and estimates its finish time. With thorough findings among virtual machines, the algorithm finds the machine that gives the shortest estimated time of task. When found, the algorithm

returns the ID of the machine identified to the datacenter controller. Upon receiving the ID by the datacenter controller, a new allocation will immediately starts. The algorithm updates the allocation table by incrementing the number of tasks assigned to the virtual machine.

Table 2 below provided summary of the existing techniques and their authors according to years and the source of publication.

Table 2. Existing Load Balancing Techniques and their Publication Fora

Author (s)	Source	Techniques
[3]	Arab Journal of Science & E ngineering	Cloudlets Allocation Strategy
[20]	Elsevier Journal	A2LB
[14]	ICCICT Conference	Adaptive Solution Load Balan cing (ALBA)
[22]	IEEE Conference	Central Load Balancer
[9]	ICSPCT Conference	Heuristic Based Load Balance d Scheduling
[1]	IEEE Conference	Particle Swarm Optimization (PSO)
[18]	ICCSC Conference	HBBLBP
[24]	ISSOSE Conference	Server Throughput Restriction (STR)
[6]	Elsevier Journal	Genetic Algorithm
[2]	ICGCCE Conference	Hybrid Scheduling
[15]	Parallel Programing Jou rnal	TBSLB-PSO
[5]	COMSNETS Conferen ce	VM-assign
[8]	Grid Distribution Comp uting Journal	Minimized Standard Deviatio n of loud Load Method(Mins d)
[21]	ICICES Conference	Nature Inspired Preemptive Algorithm
[7]	WCCS Conference	Estimated Finish time Load Balancer (EFLB)

5. CLOUDS LOAD BALANCING METRICS

Load balancing metrics are considered in cloud computing for evaluating the performance of an algorithm to determine its effectiveness in surviving cloud challenges. Below are the qualitative metrics used in evaluating the performance of the existing cloud load balancing algorithms.

- **Response time:** This is the amount of time taken between submission of a request and the first responding time that is produced by a load balancing algorithm in a distributed system. However, reduction in waiting time helps in improving the responsiveness of a virtual machines.
- **Performance:** It is used to determine how effective the system when implementing load balancing.
- **Resource utilization:** Checking the utilization of resources in load balancing is required to optimize the efficiency of load balancing algorithm.
- **Throughput:** This is the overall task completion. The throughput required to be high for better performance of system. It comprises of overhead as a result of task migrations from one domain to another.
- **Cost overhead:** It's used to find amount of overhead involved when implementing load balancing techniques. It should be acquired at minimum to enable load balancing works effectively.
- **Scalability:** When an algorithm performs load balancing with any finites number of nodes for a system, is called scalability. Improvement in scalability of an algorithm is paramount to load balancing.
- **Fault tolerance:** An algorithm in this instance can still perform load balancing even when there is a failure in one of the node. A fault-tolerant algorithm is good for load balancing so as to switch to other nodes that will cover failure of the failed nodes.
- **Migration time:** The time taken to migrate computing resources from one particular node to another should be minimized in order to enhance the performance of the entire system.

6. COMPARISON AND ANALYSIS OF THE EXISTING TECHNIQUES

Since cloud environment today are either homogeneous or heterogeneous in nature, collaboration of homogeneous and heterogeneous cloud to form federations, requires efficient task scheduling algorithm to survive. Comparing the existing techniques becomes paramount to determine the best possible way in proposing efficient load balancing algorithms for cloud environment.

Table 3. Comparison of the Existing Load Balancing Techniques Based on Various Cloud Metrics

Environment	Techniques	Fault Tolerant	Response Time	Scalability	Resource Utilization	Throughput	Migration Time	Cost Overhead	Performance
Dynamic	Cloudlets Allocation Strategy	Yes	More	Less	More	Low	Less	Less	Less
Dynamic	A2LB	Yes	More	Less	Less	Low	Less	Less	More
Dynamic	ALBA	Yes	More	Less	More	Low	Less	Less	Low
Static	Central Load Balancer	No	More	No	Less	Low	More	High	Less
Dynamic	Heuristic Based Load Balanced Scheduling	yes	More	Less	More	Low	More	More	Less
Dynamic	PSO	Yes	High	Less	High	Low	Less	More	Low
Dynamic	HBBLBP	Yes	High	Less	Less	Low	Less	More	Low
Static	STR	No	More	Less	More	Low	More	More	Less
Static	Genetic Algorithm	No	More	No	High	Low	More	Less	Less
Static	Hybrid Scheduling	No	More	No	Less	Low	High	More	Less
Dynamic	TBSLB-PSO	Yes	More	Less	More	Low	Less	Less	Low
Static	VM-assign	No	More	No	Less	Low	More	High	Less
Dynamic	Minsd	Less	Less	Less	High	Low	Less	Less	Low
Static	Nature Inspired Preemptive Algorithm	No	More	No	More	Low	More	high	Less
Dynamic	EFLB	Yes	High	Less	More	Low	No	Less	Less

Table 4. Main Attributes Addressed by the Existing Techniques.

Author	Techniques	Environment	Issues addressed
[3]	Cloudlets Allocation Strategy	Dynamic	Completion time,
[20]	A2LB	Dynamic	Service Time,
[14]	ALBA	Dynamic	Scalability, Performance.
[22]	Central Load balancer	Static	Response Time.
[9]	Heuristic Based Load Balanced Scheduling	Dynamic	Response Time, Turnaround Time.
[1]	PSO	Dynamic	Average Response Time.
[18]	HBBLBP	Dynamic	Cost, Execution Time.
[24]	STR	Static	Response Time, Throughput.
[6]	Genetic Algorithm	Static	Response Time, Utilization.
[2]	Hybrid Scheduling	Dynamic	Cost, Response Time, Processing Time
[15]	TBSLB-PSO	Dynamic	Migration Time.

[5]	VM-assign	Static	Response Time.
[8]	Minsd	Dynamic	Performance, Communication Overhead, Through-put.
[21]	Nature Inspired Preemptive Algorithm	Static	Execution Time.
[7]	EFLB	Dynamic	Execution Time

Table 5: Analysis of the Existing Techniques for Load Balancing in Cloud Environment

Author	Techniques	Shortcomings
[3]	Cloudlets Allocation Strategy	✓ Time consuming due to fact that cloudlets have to be arranged in descending order according to MI. ✓ Exhibit low throughput due to higher execution time.
[20]	A2LB	✓ It has high computational complexity. ✓ A migration agent in the algorithm has to go in search of available VM, this result in high service time.
[14]	ALBA	✓ Slow due to fact that, global agent has to communicate with the local agents which in turn communicate with the load balancer for appropriate VM.
[22]	Central Load balancer	✓ Not fault tolerant due to single point of failure. ✓ The Central Load Balancer calculates the priorities of virtual machines based on CPU speed (MIPS) and memory. However reductions in task capabilities are not taken into account. ✓ Only one parameter "response time" is considered, there is need to consider more parameters such as cost, throughput to determine the effectiveness of the algorithm.
[9]	Heuristic Based Load Balanced Scheduling	✓ Inherit the FCFS algorithm to respond to request, resulting in higher waiting time. ✓ Returns high turnaround due to adopted FCFS policy.
[1]	PSO	✓ VM load not taken into account. Allocating tasks can result to load imbalance. ✓ Inherit swarm behaviour resulting in high response time.
[18]	HBBLBP	✓ Exhibit high communication overhead. ✓ Considered task with higher priority, resulting in low priority task waiting or may never execute due to availability of jobs with higher priority arriving.
[24]	STR	✓ Work well only in static cloud environment with high network connection. ✓ Single point of failure. ✓ Not scalable to adapt dynamic changes of user request.
[6]	Genetic Algorithm	✓ High complexity in term of operation, this may eventually affect the overall performance of the system. ✓ Exhibit Round Robin characteristics, hence not fault tolerance.
[2]	Hybrid Scheduling	✓ Did not consider capacity of VM before allocation of task is made. ✓ Exhibit high response time due to the fact that the 'hasmap' has to determine the overloaded and under loaded VM.
[15]	TBSLB-PSO	✓ Too complex in term of implementation. ✓ Only migration time is considered for evaluating the performance of the technique. Other factors such as cost, execution time etc., are also required.
[5]	VM-assign	✓ Only function in homogeneous environment. ✓ Not fault tolerant. ✓ Inherits Round Robin issues, ignoring the loads capabilities of network nodes.
		✓ Complex and time consuming due to the fact that the load on Datacentre, Host, PE, has to be determined for task to be

[8]	Minsd	distributed.
[21]	Nature Inspired Preemptive Algorithm	✓ Static in nature, hence not fault tolerant ✓ Only task with highest priority are pre-empted. ✓ Starvation may occur to tasks with low priority and may never execute if priority jobs keeps coming.
[7]	EFLB	✓ Did not consider the load of a VM before task allocation are carried out. ✓ Only one parameter (finish time) considered. More QoS needed.

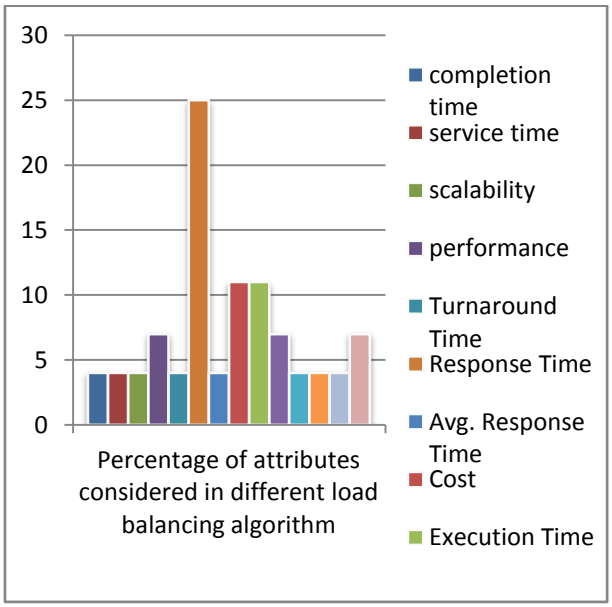

Figure 1. Percentage of each attributes considered in different load balancing algorithm

7. CONCLUSION

Cloud computing today requires more reliable, efficient and scalable load balancing algorithm to survive. As one of the main challenge in cloud computing, load balancing is required to distribute dynamic workload across multiple nodes, ensuring that no single node is overloaded. In this paper, a systematic review on existing load balancing techniques currently prevalent in cloud computing were analysed. Table 3, 4, 5 and figure 1, shows analysis carried out from existing techniques. Although existing techniques concentrated on optimizing cost, response time, execution time, communication overhead, migration time. However, neither of the techniques discussed so far, was able to provide concise approach for load balancing of task scheduling for single and federated heterogeneous cloud environments. Other issues not fully addressed by the techniques that requires further research are; virtual machine migration, server consolidation, energy management and carbon emission factor. Future research is to unveil efficient multi-objective load balancing of tasks scheduling algorithm with quality of service improvements for homogeneous and federated heterogeneous cloud environment.

8. REFERENCES

[1] Ashwin, T. S., Domanal, S. G. and Guddeti, R. M. R. 2014. A Novel Bio-Inspired Load Balancing of Virtual Machines in Cloud Environment. In Proceedings of the IEEE International Conference on Cloud Computing in Emerging Networks (CCEM). 15- 17 October. Bangalore, India, 1-4.

[2] Bagwaiya, V. and Raghuwanshi, S. K. 2014. Hybrid Approach Using Throttled and ESCE Load Balancing Algorithms in Cloud Computing. In Proceedings of the International Conference on Green Computing Communication and Electric Engineering (ICGCCEE). 6-8 March. Coimbatore, India, 1-6.

[3] Bagwaiya, V. and Raghuwanshi, S. K. 2014. Hybrid Approach Using Throttled and ESCE Load Balancing Algorithms in Cloud Computing. In Proceedings of the International Conference on Green Computing Communication and Electric Engineering (ICGCCEE). 6-8 March. Coimbatore, India, 1-6.

[4] Dhinesh, B, L. D. and Krishna, P. V. "Honey Bee Behavior Inspired Load Balancing of Tasks in Cloud Computing Environments", Journal of Applied Soft Computing, 2013, 13(5), pp. 2292-2303.

[5] Domanal, G. S. and Reddy, G. R. M. 2014. Optimal Load Balancing in Cloud Computing by Efficient Utilization of Virtual Machines. In Proceedings of the Sixth International Conference on Communication Systems and Networking (COMSNETS). 6-10 January. Banglore, India, 1-4.

[6] Effatparvara,M. and Garshasbi, M. S. "A Genetic Algorithm for Static Load Balancing in Parallel Heterogeneous Systems", Procedia - Social and Behavioural Sciences,Jornal, 2014, 129, 358 – 364.

[7] Fahim, Y., Lahmar, E. B., Labrlji, E. and Eddaoui, A. 2014. The Load Balancing Based on the Estimated

Finish Time of Tasks in Cloud Computing. In Proceedings of the Second World Conference on Complex Systems (WCCS). 10-12 November. Agadir, Morocco, 594-598.

[8] Hao, Y. Liu, G. and Lu, J. "Three Levels Load Balancing on Cloud", International Journal of Grid Distribution Computing, 2014, 7(3), pp. 71-88.

[9] Haidri, R. A., Katti, C. P. and Saxena, P. C. 2014. A Load Balancing Strategy for Cloud Computing Environment. In Proceedings of the 2014 International Conference on Signal Propagation and Computer Technology (ICSPCT). 12-13 July. Ajmer, India, 636-641.

[10] Hassan, M. M., Song, B. and Huh, E-N. 2011. Distributed Resource allocation Games in Horizontal Dynamic Cloud Federated platform. In Proceedings of the IEEE International Conference on High Performance Computing and Communications. 2-4 September. Banff, Canada, 822-827.

[11] Kansal, N. J. and Chana, A. J. "Existing Load Balancing Techniques in Cloud Computing: A Systematic Review", Journal of Information Systems and Communication, 2012, 3(1), 87-91.

[12] Katyal, M. and Mishra, A. "A Comparative Study of Load Balancing Algorithm in Cloud Computing Environment", International Journal of Distributed and Cloud Computing, 2012, 1(2), 5-14.

[13] Katyal, M. and Mishra, A. "Application of Selective Algorithm for Effective Resource Provisioning In Cloud Computing Environment", International Journal on Cloud Computing: Service and Architecture (IJCCSA), 2014, 4(1), 1-10.

[14] Malhotra, M. and Singh, A. 2014. Adaptive Framework for Load Balancing to Improve the Performance of Cloud Environment. In Proceedings of the IEEE International Conference on Computational Intelligence and Communication Technology (CICT). 13-14 February. Ghaziabad, India, 224-228.

[15] Ramezani, F., Jie Lu, J. and Hussain, F. K. **"**Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization", International Journal of Parallel Programming, 2014, 42, 739-754.

[16] Sharma, T. and Banga, V. K. "Efficient and Enhanced Algorithm in Cloud Computing", International Journal of Soft Computing and Engineering (IJSCE), 2013, 3(1), 385-219.

[17] Shahapure, N. H. and Jayarekha, P. 2014. Load Balancing with Optimal Cost Scheduling Algorithm. In Proceedings of the International Conference on Computation of Power, Energy, Information and communication (ICCPEIC). 16-17 April. Chennai, India, 24-31.

[18] Sheeja, Y. S. and Jayalekshmi, S. 2014. Cost Effective Load Balancing Based on Honey bee Behaviour in Cloud Environment. In Proceedings of the First International Conference on Computational System and Communication (ICCSC). 17-18 December. Trvandrum, India, 214-219.

[19] Silpa, C. S. and Basha, M. S. S. "A Comparative Analysis of Scheduling Policies in Cloud Computing Environment", International Journal of Computer Applications, 2013, 67(20), 16-24.

[20] Singh, A., Juneja, D. and Malhotra, M. 2015. "Autonomous Agent Based Load Balancing Algorithm in Cloud Computing", Procedia Computer Science journal, 2015, 45(1), 832-841.

[21] Shobana, G., Geetha, M. and Suganthe, R. C. 2014. Nature Inspired Preemptive Task Scheduling for Load Balancing in Cloud Datacenter. In Proceeding of the International Conference on Information Communication and Embedded Systems (ICICES). 27-28 February. Chennai, India, 1-6.

[22] Soni, G. and Kalra, M. 2014. A Novel Approach for Load Balancing in Cloud Data Centre. In Proceedings of the IEEE International Conference on Advance Computing Conference (IACC). 21-22 February. Gurgaon, India, 807-812.

[23] Sreenivas, V. Prathap, M. and Kemal, M. 2014. Load Balancing Techniques: Major Challenge in Cloud Computing – A Systematic Review. In Proceedings of the International Conference on Electronic and Communication System (ICECS). 13-14 February. Coimbatore, India, 1-6.

[24] Sun, H., Zhao, T., Tang, Y. and Liu, X. 2014. A QoS-aware Load Balancing Policy in Multi-tenancy Environment. In Proceedings of the 8th International Symposium on Service Oriented System Engineering. 7-11 April. Oxford, United Kingdom, 140-147.

[25] Xu, X., Yu, H. and Cong, X. 2013. A QoS-constrained Resource Allocation Game in Federated Cloud. In Proceedings of the 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. 3-5 July. Asia University, Taichung, Taiwan, 268-275.