Ingénierie des Systèmes d'Information
Vol. 25, No. 6, December, 2020, pp. 771-782
Journal homepage: http://iieta.org/journals/isi

A Hybrid Multi-level Statistical Load Balancer-Based Parameters Estimation Model in Realtime Cloud Computing Environment

Gutta Sridevi[1], Midhunchakkravarthy[2*]

[1] Dept. of CSE, Lincoln University College, Petaling Jaya 47301, Malaysia
[2] Faculty of Computer Science & Multimedia, Lincoln University College, Petaling Jaya 47301, Malaysia

Corresponding Author Email: midhun@lincoln.edu.my

https://doi.org/10.18280/isi.250607

Received: 4 September 2020
Accepted: 22 November 2020

Keywords:
statistical load balancer, cloud computing, virtual machines

ABSTRACT

As the size of the cloud-based applications and its tasks are increasing exponentially, it is necessary to estimate the load balancing metrics in the real-time cloud computing environments. Hybrid load balancing framework play a vital role in the cloud-based applications and tasks monitoring and resource allocation. Most of the conventional load balancing metrics are dependent on limited number of cloud metrics and type of virtual machines. Also, these models require high computational memory and time on large number of tasks. In this paper, an advanced multi-level statistical load balancer-based parameters estimation model is designed and implemented on the real-time cloud computing environment. In this model, a novel statistical load balancing data collector is used to find the best metrics for the load balance computation. In this model, different types of tasks are simulated under different virtual machine types such as small, medium and large instances. Experimental results show that the proposed multi-level based statistical load balancing collector has better efficiency than the conventional models in terms of memory utilization, CPU utilization, runtime and reliability are concerned.

1. INTRODUCTION

With a wide range of services and an increasingly increasing services, cloud computing needs to find methods for resource utilization management and higher performance ensure. It is necessary to optimize the Service Level Agreements (SLAs) during the cloud service allocation and resource optimization process. It requires better load handling mechanisms to achieve SLAs and to ensure customer satisfaction. Load balancing algorithms are improved to balance the load and maximize resource efficiency and resource management optimizations. It is very important to distribute the resources optimally. If more resources are allocated, resource shortages will occur and when less resources are allocated, the workers will struggle to achieve SLAs. Cloud Computing is an up to-the-minute IT environment-affiliated technology with huge resource demand such as efficient use of storage devices, networking devices, computer devices, services and applications, etc. Now a few days, mobile software and utilities have migrated entirely to portable PCs and connected via the Internet to data centers. This is not operated by personal computers or local data servers, which are shared [1].

For cloud computing the infrastructure's buying and operating costs are eliminated [2]. In short, 'Cloud Computing' offers a simple, open, demand-based, and pay-per-use approach to computing resources [3]. Cloud computing is an essential technology that makes efficient use of resources by allocating it to users according to their needs. Load balancing in the cloud is one of the main challenges for managing the workload between the systems and performance. An efficient load balancing algorithm is required to allow effective use of resources. A thorough analysis on the key aspects of cloud computing and by comparing various existing load balancing algorithms is carried out. The cloud-based partitioning approach which considers the related state of different partitions in the cloud [4]. The condition can be OVERLOADED, NORMAL, or IDLE. The load balancing algorithms, as the states suggest, intuitively understand the environment and take balancing decisions. It is capable of handling more workers when every partition is idle. The standard state shows the VMs are in use but can still handle work. The program is able to improve load handling by providing a queue and testing it iteratively. If load degree is lower for each server, new jobs will be allocated to that server upon arrival. Since cloud servers are equipped with a load balancer program, the load balancer system is responsible for scheduling jobs with load in mind. Nonetheless, the state of the partitions needs to be checked periodically. The problem here is to decide load balancing technique based on partition with an ideal refresh time. Assessment of refresh time to provide a refresh threshold is very necessary for optimizing load balancing using cloud partitioning approach. When cloud space is separated, greater control can be exercised over it. CP (Cloud Partition) based model provides for various strategies such as IDLE, Regular, and OVERLOADED.

Those are in reality partition status which is used to make informed decisions. When a partition performs no work, it is said to be in IDLE state. Similarly, when a partition is used for processing but its load is regular, and then it is called Regular. On the other hand, it is known as OVERLOADED state, the partition that exceeds its maximum processing potential and all of its resources are being used. Such states specifically assist with load balancing. They explored various recent

attempts to build an ACO theory, addressed linkages between ACO mechanisms and stochastic gradient ascension and cross-entropy techniques in model-based search construction, and finally discussed the impact of search bias on the functioning of ACO algorithms. They changed the basic ACO algorithm to focus more on the strategy for optimization in the execution of tasks. To simulate their approach, they used CloudSim toolkit. MACO initiates the finest resource allocation in the cloud for task groups and reduces the makespan but is constrained in finding the Load Balancing Element [5].

A cloud computing is an integral technique that is on-demand operation, and cloud-based service providers at a particular time exchange resources, applications, and information according to the user requirement. The cloud vendors automatically balance the service loads, and more CPUs, memory, and resources are required to manage more user demands. The service is focused on customers ' business requirements. The load balancing strategy helps satisfy the two key criteria, initially supports cloud resource availability and also increases server performance. Load balancing is the difficult problem in cloud computing system as it requires more workloads. The fundamental goal of the philosophy of load balancing is to equalize the workload between each and every node by reducing the execution time, communication delays, resource usage increase, and throughput. Load balancing increases network reliability step by step through the shared workload transfer process. Cloud load balancing reduces costs and greatly improves resource availability. An optimal selection of algorithms should ensure that all computer nodes present in the network process the same number of workloads. The load balancing strategy helps to provide the unorganized resources with visualization of tasks that are distributed in the cloud architecture. The cloud platform, load balancing approach is used to distribute a large workload of processing to small workloads of computation to boost general system execution. Static load balancing approach uses the existing considerable knowledge of the system's application behavior and statistical data. Through this balancing technique machine resource knowledge and output of the processor is calculated at execution begging. So the load decision does not depend on the system's current state. Access is made to the acquired output result and subsequently work load is allocated by substantial processor. The processor tests the assigned job at each node and delivers the result to the processor in question. But, as often executed on the processor itself, the static approach is also called a non-preemptive one. The key aspect of static techniques is to lower the execution and its time series along with eliminating delays directed to communication. A general drawback of static approaches is the overall host selection which requires process allocation [6].

In dynamic form, the load is dynamically supplied to any processor present in the network at the time of program execution. It is different from static because it works as a runtime processor which makes every single node participate in information transfer. Instead, the tasks at each node are buffered at the master node in a queue format, and distributed one by one to the related nodes as a result of remote hosts requests. The complex methodology is continuously trying to track the tasks from each processor. Once the tasks and weight disparity increase to a certain predefined amount, the data processing is re-collected. Monitoring the whole cycle arranges the CPU rotations and manages the method as the algorithm evokes it. Cloud computing provides consumers versatile and scalable Internet-based infrastructure. For that a lot of data can be arrived at a time in a cloud. It leads to server failures causing problems with outages. The load balancing is typically a method that distributes the dynamic load across the cloud to all VM. A problem often occurs when some nodes are heavily loaded when others are idle or doing little work. The balancing algorithm for complex loads is implemented either as distributed or non-distributed algorithm. Dynamic load balancer uses protocols to keep track of the details being changed. The suggested technique has implemented a successful algorithm to avoid the server fault. It helps attain a high user satisfaction and resource usage ratio, thereby enhancing the system's overall efficiency and resource usage. It also ensures that every computing power is effectively and equally distributed.

The proposed algorithm will prevent machine bottlenecks from occurring due to unbalance in load. If one or more elements of any operation fail, load balancing assists in the service's continuity at the time of failover. Resource management can be effectively done using appropriate resource allocation algorithms to optimize the number of physical machines, VMs and equally balance the loads among available resources. Both quantitative and qualitative work was carried out to design and improve algorithms for optimizing the allocation of resources in cloud computing. Dynamic and effective allocation of resources algorithms are focused on optimal usage of the datacenter resources. Virtualization plays a significant role in maximizing the usage of resources. In both software and hardware, it is a technology used to build VMs that allow a single physical machine to operate as multiple machines resulting in scalable resources on demand. The use of resource virtualization techniques increases resource efficiency and reduces the waiting time for tasks. The service provider is increasing their resource utilization to support a wide range of flexible workloads. Moving to Intercloud is therefore a good choice for all sizes of organizations. With the support of the architecture, several open access research problems and resource algorithm provisioning were established and the cloud system was inspired. The energy-oriented heuristics of allocation heuristics from the DC method to client set of applications have effectively enhanced efficiency. Through program implementation, the agreed Quality of Service (QoS) way is accomplished. The findings were obtained by performing an early-research survey and comparison. The result is described as: (a) design principles for managing energy-intensive clouds; (b) efficient policy resource allocation and scheduling processes that take into account the QoS needs and electricity characteristics of the devices employed; and (c) a range of open research and challenges, substantial benefits from the application to both power providers and customers. The performance assessment and judgment of the algorithm built is achieved using the CloudSim toolkit and its operations.

The resource assignment undertaking is placed through the network analysis strategy and the Analytic Hierarchy Method giving accessible assets and customer inclinations. In addition, an implemented inclination lattice is used to separate the conflicting components and to improve the consistency ratio when attributing conflicting weights in different tasks. The findings indicate that further measuring of inconsistent data and increasing the consistency ratio and task weight are useful for dynamically allocating computing resources in cloud computing environment. Load balancing is a method used to spread the workload equally in an eventual way between each

node in the cloud organization system. It is defined as the distribution of load from resources so as to enhance the system's overall performance. Therefore, the load needs to be uniformly spread across the cloud-based resource system, and at any time series, each resource is performed approximately the same amount of work. The key benefit is to provide strategies for managing incoming client requests, and to ensure that the client always performs better. Cloud vendors offer automated load balancing services; allow clients to build up the amount of CPUs or resource memory to calculate with increased demands. However this service is voluntary, and relies on customers ' business needs. Thus load balancing serves two important purposes, it promotes cloud resource availability and it also promotes cloud environment efficiency. This is an important goal in understanding a major objective such as cost effectiveness, efficiency, scalability, and timing objectives of load organizing algorithms in order to manage the loads and resource requests. Load management helps in also allocating present resources to make optimal utilization of resources and a high degree of customer satisfaction. The higher achievement of resources and structured load balancing helps increase the scalability and prevent bottlenecks. This helps to achieve the maximum and minimum amount of response time and throughput. Load balancing often separates traffic between servers in a networked environment, so data can be sent and received at a higher rate without delay. Many algorithms are developed, and it helps with proper traffic loading in a cloud environment between the available servers.

2. RELATED WORK

Several researchers have sought to forecast cloud computing workloads so that they can perform better. No predictive algorithm looks true so far. They concentrated on the idea of external optimisation (EO) [7]. In cloud computing, the EO-based algorithm is used to use a two-step stochastic selection process to balance load. It is simply a meta-heuristic that cares for the scheduling of cloud jobs. It consists of a hybrid approach consisting of the genetic process and the principle of fuzziness. To do this, they've improved the Standard Genetic Algorithm (SGA). They offered CloudSim model simulation solution based on scheduling algorithms for load balancing. They considered the duration of work, the frequency of VMs and memory consumed by VMs during decision making.

Gesvindr et al. [8] testified to MapReduce programming for efficient job handling in cloud computing. They concentrated on the study of trade-off between cost of success and cost of optimisation. To order to provide tests of diversified inputs and outputs various benchmarks are used. Using Amazon Elastic Cloud Compute (EC2), Elastic Block Storage (EBS), and Elastic MapReduce (EMR), they tested this. Time for completion of jobs is measured and the complexity of job size is discussed.

Gholipour [9] demonstrated a task-level scheduling algorithm and used Dynamic Programming (DP) to develop a budget-driven scheduling system. They know their algorithms as Global Optimal Scheduling (GOS) and Global Greedy Budget (GGB). GGB designed to optimize scheduling and balancing of loads under the defined budget constraints. The budget for stockpiling and sharable resources is given. Global budget allocation with specified constraints makes an algorithm greedy and produces optimal scheduling performance. Scheduling time and remaining budget are two important metrics used to determine it.

Gill et al. [10] focused on market-oriented resource management techniques which embrace risk management strategies and customer-centered service management approaches to support resource allocation centered on the SLA. In addition, they developed a method called CloudSim to simulate performance metrics in a cloud environment. They clarified the Particle Swarm Optimization (PSO) is an optimization technique that finds the optimal solution to the problem by iteratively optimizing it. The candidate solutions are popularized to solve the problem of optimization. The particles shift in the search space depending on the velocity and location of the particles, based on the mathematical formula. The movement of each particle determines the optimal location. In solution space the best known locations are defined and modified. PSO and ABC algorithms are effective techniques of optimization that have a fairly short iteration time. On the other hand, they may definitely collapse into local extremes. This method significantly decreases the average completion time of the task in the cloud datacenter.

Guo et al. [11] suggested a profiling-based, energy-efficient cloud computing technique. The offline profiling strategy is conducted for the development of energy aware references for the cloud environment infrastructure given. The profile developed is matched with the data in real time and the services are therefore given to reduce energy consumption. It is however difficult to determine the exact number of VMs required for the cloud system to operate properly and effectively.

Habibi et al. [12] defined the resource capacity and cloud workload scheme which, by applying three flexible time scales and scopes, combines different resource restrainers in automated resource allocation. Three types of time scales were used in the study, such as shortest time scale, shorter time scale and longest time scale. The controllers, namely node controller, pod controller and pod set controller, were configured to handle the allocation of the Virtual Machine resources and the migration of the workload. Additionally, they concentrated on applying different threshold techniques to complex variations in cloud workload [13].

Various efforts are being made to scale elastic cloud application frameworks that reduce elastic resource consumption through cloud-based elastic services. These frameworks were developed to manage scalable services based on precepts and proficiencies of autonomous computing. However, there were no statistical techniques used in elastic cloud services for handling resource allocation and managing vast volumes of data. Various dynamic scheduling techniques have been suggested to improve resource efficiency in cloud computing.

Jodayree et al. [14] used the Ant Colony Optimization (ACO) approach to present a strategy for allocating Virtual Machines to the least number of cloud servers based on user requests and workloads. They addressed the Virtual Machine Resource Planning Scheme for Load Balancing based on the Genetic Algorithm. This algorithm was used to minimize dynamic migration and to achieve better load balancing based on current cloud system state and historical data. Liang et al. [15] proposed ACO load balancing based algorithm to schedule and reduce makepan in cloud setting. In cloud computing [16] the emphasis was on allocating virtual machines for efficient resource allocation based on cost and execution time for real-time tasks in the IaaS environment.

The research allowed the user to pick Virtual Machines and cut Cloud usage costs. The distribution of resources was performed on the basis of a fixed number of processors. So, it has contributed to problems of scalability and elasticity.

Mashhadi et al. [17] proposed a Multi-Objective Genetic Algorithm for Cloud Brokering (MOGA-CB) that considered two objectives in the optimization process to reduce response time and costs.

Nyasulu [18] implemented modified RR approach for resource allocation technique in cloud computing. One of the computing paradigm is cloud computing, which allows services on demand. Cloud users can access their data or software from anywhere, at any time. An enterprise may lease Cloud assets for power and other computational purposes with the intention of significantly decreasing their network costs. Through raising the processing time, the altered RR asset allocation approach fulfills the company demands and program needs. This built algorithm consists of various parts such as load prediction, hot spot mitigation, and green computation. The resource allocation is built at two levels within the cloud framework. The method's code is initially inserted into the cloud, and the load balancer is allocated to the computer's requested instances. Finally, the various types of multiple incoming orders or requests are reached, and these requests are allocated in an instance to each particular function or application, and the computational load is balanced.

Premarathne et al. [19] developed a resource allocation model for the scaling of Software-as-a-Service functionality over cloud infrastructures based on tenant features. The cloud infrastructure offers the necessary access to a collection of computing tools along with some of the system's basic business models; fetches centralized providers constantly calculating service levels. Its internal infrastructures build a number of VMs and instances concerning the application's demand. SaaS providers have the key benefit of scaling up or scale down the application capital capability. The system's ability to access or pay is limited, which at a given time is the most significant challenge in cloud computing.

Anton Rafieyan [20] used Genetic Algorithm (GA) to exhibit client-conscious scheduling of tasks and resource allocation in multi cloud. Mapping the approaching demand for occupation to accessible VMs is a non-polynomial finish problem, as the concept of movement is very subjective. The scalability is considerably high for the simulated multi-cloud environment. Data locality costs, latency arbitration, energy usage, and multi-cloud network running costs are beyond the scope of the simulated scenario.

Sahil et al. [21] conducted a taxonomy survey of techniques that are effective in allocating resources to cloud along with minimized capacity. In the cloud computing model, the allocation of different virtualized ICT mode of services are the dynamic problems created by the presence of heterogeneous application types such as networks dependent content delivery nodes, web apps, and Map Reduce, etc. Workloads are extracted and studied which include more contentious allocation requirements with ICT capacity resources such as network bandwidth, response time, processing speed, etc. The vast number of recent researchers have been addressing problems in improving energy efficiency and allocating cloud-resource applications with a particular point of achievement. In any case, there are no centralized processes concerning this phenomenon to the best of our understanding, because it analyzes the problem of discovery and offers a theoretical categorization from current methods. The majority of the algorithms can be implemented with perfect modifications in cloud generation. Load balancing algorithms can be generally categorized as Heuristic Scheduling Algorithms for Batch Mode (BMHA) and Heuristic Algorithms for Online Mode (OMHA). In BMHA, as it comes into the program, jobs are grouped as a batch, and scheduling starts after a default time period. Heuristic scheduling calculations in the online mode are more suitable for cloud environment. It is important to determine valid load, reliability of different systems, frame execution, communication between each of the hubs when constructing a load balancing procedure. The efficient forecasting techniques as well as the host load balance are required for the VM migration. While workload lengths are shorter, host nodes shift more frequently with higher cloud noise. The implementation of the planned plan and the existing order approaches are contrasting. The load forecast for the host is based on genuine Google cluster information gathered. The length of the cloud host load is much shorter than that in grid, meaning the cloud host load is adjusted a lot more frequently. Time means that the load level is constant over a given time period. Virtualization technology is the foundation of distributed computing that allows to efficiently use managing assets by relegating the amount of VMs to the single physical host. In the meantime, workloads for the cloud environment are dynamic in nature, and several needless hosts sometimes run in the background.

3. PROPOSED WORK

In the proposed framework, a novel multi-level cloud-based load balancing metrics are estimated using the advanced statistical data collector. As shown in the Figure 1, initially multiple virtual machines (VMs) are initialized in the AWS EC2 environment in order to find its resources. Here, a randomized set of tasksare assigned to each virtual machine for resource allocation and load balancing. Tk-1,Tk-2….Tk-m1 represents the set of tasks initialized to virtual machine V1. Similarly, Tk-1,Tk-2….Tk-mn represents the set of tasks assigned to virtual machine Vm.

3.1 Improved multi-level load balancing feature estimation

1. Initializing of cloud virtual machines VM_i for n tasks, m resources and k load balancers.
2. To each virtual machine VM_i
3. do

Assign n tasks to each virtual machine.

Compute the best r metrics in the given tasks to each virtual machine by using the cloud IPSO with QoS selection measure as:

$$MV(cf_i+1,i) = \omega.[e^{cf_i}.MV(cf_i,i).\eta_{c1}(pBt_i - MV(cf_i,i) + \eta_{c2}(gBt_i - MV(cf_i,i)]$$

ω is the multi-level metric selection scaling parameter defined as:

$$\omega = \frac{h^*(\eta_{c1}+\eta_2)}{|h^* cf_i - (\eta_{c1}+\eta_{c2}) - \sqrt{(\eta_{c1}+\eta_{c2})^2 - 4(\eta_{c1}+\eta_{c2})}|}$$

where, η_{c1}, $\eta_{c2} \in$ cyclic group randomized elements and h<=2.

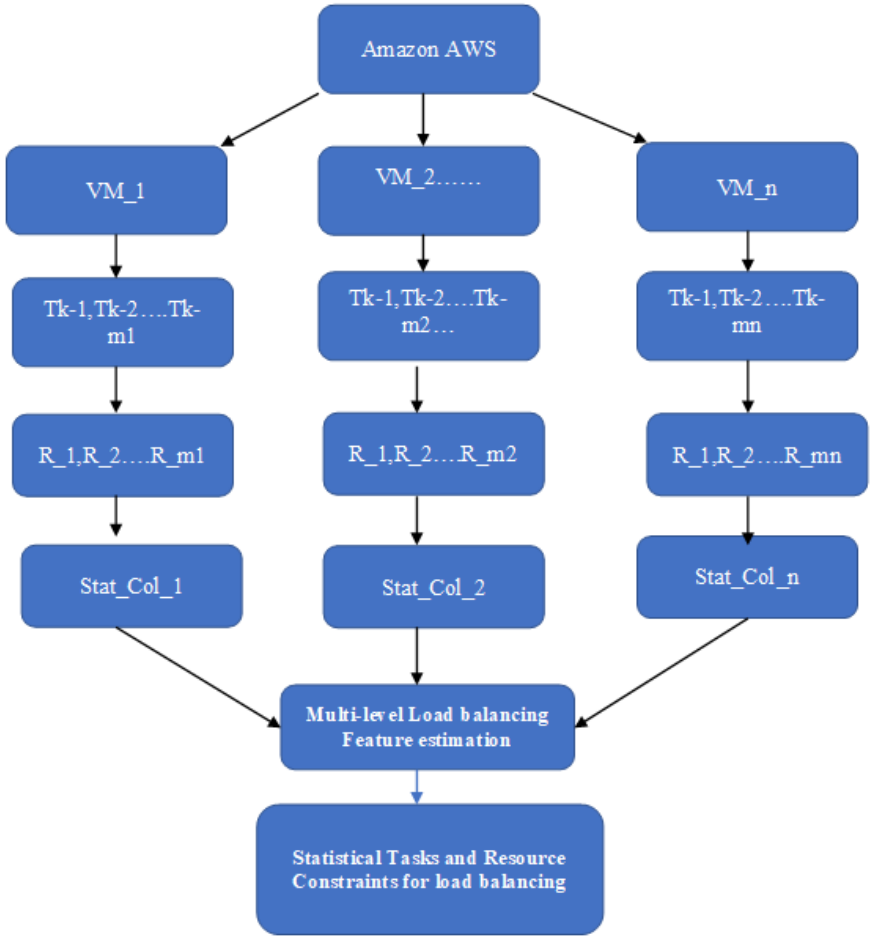

Figure 1. Proposed multi-level load balancing parameter selection

$$\eta_{ci} = \frac{1}{\sqrt{2\pi\sigma_R}} e^{-\max\{cf\}.\frac{(cf_i-\mu_R)^2}{\sigma_R^2}}$$

$$i = 1, 2 \ldots \text{iterations}$$

Here, ω is used to find the local and global best positions for the given PSO model.

Let $R_{t\min}$ and $R_{t\max}$ are the minimum and maximum response time of the task in each instance VM_i.

$$F(c) = MV(cf_i+1,i)*(R_{t\min} \; \eth \; + \; R_{t\max}(1-P_{cf_i}).Wq)$$

$$W_q = \max\{ VM(R_t,i)\}.T_\mu / \lambda_R$$

where,

$|VM|$= Total number of virtual machines.

$|T|$=Total number of tasks.

$$\lambda_R = VM(\mu,i).(1-P_{cf_i})$$

$$P_{cf_i} = (\lambda_R{}^n / |VM|!|T|.\mu_R{}^n)$$

Here, the fitness function is used to select the optimal load balancing features for the prediction purpose.

These selected local and global functions are given to fitness function computation. The fitness measure of the proposed improved PSO is selected based on the minimum and maximum response time of the task in each instance.

Repeat step 3 to each task dynamic response time and the cloud metrics.

In this algorithm, a set of virtual machines, tasks and its response time are taken as input to find the relevant best metrics for load balancing algorithm. In this algorithm, the statistical collector program is implemented in order to extract all the cloud metrics in the AWS environment. In this algorithm, initially, all the virtual machines and its corresponding tasks are initialized in the AWS cloud environment for load balancing metric selection. To each virtual machine, different tasks are assigned to find the best metrics for load balancing property. In this work, an advanced IPSO statistical collector is used to find the probability of selecting the load balancing metric using the QoS response time(R). Steps 5 and 6 are used to find the best load balancing metrics based on the QoS and its corresponding cloud metrics.

3.2 Statistical tasks and resource constraints for load balancing

In this statistical tasks and resource constraints phase, each virtual machine and its resources are monitored in the real-time cloud environment using the hybrid multi-objective functions. In this phase, each cloud virtual instances, capacity and memory bounds are used to define the non-linear objective functions for tasks and resource allocation.

$$\text{Max}\{(\frac{ce^M}{\log(M^2)} + \log(C)) + M\}$$

The above non-linear objective function is used to solve the best computed memory (M) constraints and the task capacity (C).

$$M+\log(C)+\frac{Ce^{M}}{\log(M^{2})} => M+\log(C)+\frac{C\sum_{k=0}^{\infty}\frac{M^{k}}{k!}}{\log(M^{2})}$$

$$M+\log(C)+\frac{C\sum_{k=0}^{\infty}\frac{M^{k}}{k!}}{\log(M^{2})} => M+\log(C)+\frac{Ce^{z_0}\sum_{k=0}^{\infty}\frac{(M-z_0)}{k!}}{\log(M^{2})}$$

$$=>\frac{M\sum_{k=1}^{\infty}\frac{(-1)^{k}(-1+M^{2})^{k}}{k}-C\sum_{k=0}^{\infty}\frac{M^{k}}{k!}-\sum_{k_1=1}^{\infty}\sum_{k_2=1}^{\infty}\frac{(-1)^{k_1+k_2}(-1+C)^{k_1}(-1+M^{2})^{k_2}}{k_1k_2}}{\sum_{k=1}^{\infty}\frac{(-1)^{k}(-1+M^{2})^{k}}{k}}$$

$$=>\frac{1}{\sum_{|k=1}^{\infty}\frac{(-1)^{k}(-1+M^{2})^{k}}{k}}\left(\begin{array}{c}M\sum_{k=1}^{\infty}\frac{(-1)^{k}(-1+M^{2})^{k}}{k}-C\sum_{k=0}^{\infty}\frac{M^{-1+2k}(2k+M)}{(2k)!}\\ \sum_{k_1=1}^{\infty}\sum_{k_2=1}^{\infty}\frac{(-1)^{k_1+k_2}(-1+C)^{k_1}(-1+M^{2})^{k_2}}{k_1k_2}\end{array}\right)$$

$$\text{Let } \psi_i=\frac{1}{\sum_{|k=1}^{\infty}\frac{(-1)^{k}(-1+M^{2})^{k}}{k}}\left(\begin{array}{c}M\sum_{k=1}^{\infty}\frac{(-1)^{k}(-1+M^{2})^{k}}{k}-C\sum_{k=0}^{\infty}\frac{M^{-1+2k}(2k+M)}{(2k)!}\\ \sum_{k_1=1}^{\infty}\sum_{k_2=1}^{\infty}\frac{(-1)^{k_1+k_2}(-1+C)^{k_1}(-1+M^{2})^{k_2}}{k_1k_2}\end{array}\right)$$

Here ψ value is used as to take the decision making based on the task run capacity C and the virtual machine's memory M.

Let the $B_i=\{B_1,B_2,B_3.....B_r\}$ represents the virtual machine allotted to the task with the required resources.

For each virtual machine VM (ψ_i),

$B_i=1$ if $\psi_{VMj}>0$; T assigns to R.

$B_i=0$ otherwise; not assigned

Let $\upsilon_{VM}->\{\upsilon_1,\upsilon_2,\upsilon_3...\upsilon_K\}$ is the set of cloud virtual instances in cloud environment.

Let $r_i \in R_i$ be the set of resources of all cloud virtual instances.

$$r_i \in R_i \rightarrow \{r_1^{CPU}, r_2^{RAM}, r_3^{PORT}, r_4^{TIME}, r_5^{HARDDISK}, ... r_m\}$$

The statistical multi-objective function for the resource bound checking of the cloud virtual machines are defined as below:

$$\text{Min}\sum_{i=1}^{N}B_i.\log\{\psi_{VMj}\}/|R_i| \text{ and}$$

$$\text{Max}\sum_{j=1}^{M}B_j.\exp(\psi_{VMj})/|R_i|$$

Let $r_i \in R_i$ be the set of resources of all cloud virtual instances.

$$r_i \in R_i \rightarrow \{r_1^{CPU}, r_2^{RAM}, r_3^{PORT}, r_4^{TIME}, r_5^{HARDDISK}, ... r_m\}$$

$$\sum_{\upsilon=1}^{|VM|}L(r_1^{CPU}) \le U(r_1^{CPU});$$

$$\sum_{\upsilon=1}^{|VM|}L(r_2^{RAM}) \le U(r_2^{RAM});$$

$$\sum_{\upsilon=1}^{|VM|}L(r_3^{PORT}) \le U(r_3^{PORT});$$

$$\sum_{\upsilon=1}^{|VM|}L(r_4^{TIME}) \le U(r_4^{TIME});$$

$$\sum_{\upsilon=1}^{|VM|}L(r_5^{HARDDISK}) \le U(r_5^{HARDDISK});$$

$$\sum_{\upsilon=1}^{|VM|}L(r_m) \le U(r_m);$$

The above statistical constraints are used to predict the range bounds of the tasks and it resources for load monitoring process. Also, these constraints are considered to find the lower bound and upper bound constraints to each virtual machine in the real-time cloud computing environment.

4. EXPERIMENTAL RESULTS

Experimental results are simulated using the Amazon AWS server and the java programming environment. In this work, AWS JAVA API is used to develop the proposed load balancing parameters estimation. In this work, a novel statistical collector-based load balancing metrics are predicted using the real-time cloud computing environment. In this experimental results, various performance measures such as Memory utilization, CPU utilization, Reliability and runtime computation on the real-time cloud server tasks. In these results, all the tasks are executed in small, medium and large virtual machines. Experimental results are compared using the traditional models such as dynamic load balancing (DLB) [22], load balancing based on bayes clustering (LB-BC) [23], load balancing with resource clustering [24] and particle swarm optimization with firefly approach [25]. In the experimental analysis, different cases of load balancing metrics are evaluated on tasks using small, medium and large instances with 1GB,4GB and 16GB ram capabilities.

Table 1. Sample real-time cloud instances and its metrics for load balance analysis

Instance no	backet name	region	load balancer ID	data req	data res	load time	req time	job idle time	CPU utilization	memreq	job runtime
Inst-1	Buckt-112.0	Asia Pacific (Mumbai)	LoadBal_ID147.0	39	329	39	1431	11	6	29	439
Inst-2	Buckt-113.0	US East (Ohio)	LoadBal_ID198.0	30	277	23	2922	12	6	5	784
Inst-3	Buckt-198.0	US East (Ohio)	LoadBal_ID201.0	27	298	35	2639	20	4	34	540
Inst-4	Buckt-126.0	East (Virginia)	LoadBal_ID126.0	46	285	28	3884	6	6	22	667
Inst-5	Buckt-147.0	Asia Pacific (Mumbai)	LoadBal_ID147.0	40	350	37	2994	18	6	53	416
Inst-6	Buckt-136.0	Asia Pacific (Mumbai)	LoadBal_ID118.0	47	283	20	2985	19	3	31	499
Inst-7	Buckt-133.0	US East (Ohio)	LoadBal_ID192.0	24	262	35	2388	19	5	35	669
Inst-8	Buckt-149.0	East (Virginia)	LoadBal_ID214.0	44	251	35	4301	16	6	45	553
Inst-9	Buckt-105.0	Asia Pacific (Mumbai)	LoadBal_ID163.0	29	252	35	1608	21	4	26	283
Inst-10	Buckt-143.0	East (Virginia)	LoadBal_ID136.0	44	313	35	3343	17	4	44	762
Inst-11	Buckt-141.0	US East (Ohio)	LoadBal_ID124.0	27	341	42	3398	13	6	28	711
Inst-12	Buckt-168.0	Asia Pacific (Mumbai)	LoadBal_ID161.0	27	258	23	1207	17	6	27	755
Inst-13	Buckt-120.0	Asia Pacific (Mumbai)	LoadBal_ID173.0	25	329	30	4279	20	4	35	656
Inst-14	Buckt-199.0	East (Virginia)	LoadBal_ID194.0	33	276	28	2884	14	4	28	738
Inst-15	Buckt-135.0	US East (Ohio)	LoadBal_ID207.0	34	242	29	1828	12	6	15	509
Inst-16	Buckt-177.0	Asia Pacific (Mumbai)	LoadBal_ID114.0	49	249	34	2925	18	5	42	715
Inst-17	Buckt-184.0	US East (Ohio)	LoadBal_ID236.0	36	252	30	2778	14	2	11	357
Inst-18	Buckt-142.0	US East (Ohio)	LoadBal_ID192.0	41	254	38	3014	9	4	46	537
Inst-19	Buckt-130.0	South America	LoadBal_ID235.0	47	270	40	4621	21	5	20	558
Inst-20	Buckt-181.0	Asia Pacific (Mumbai)	LoadBal_ID186.0	41	286	32	1246	17	3	12	483
Inst-21	Buckt-192.0	Asia Pacific (Mumbai)	LoadBal_ID199.0	26	327	31	3842	19	6	25	740
Inst-22	Buckt-118.0	US East (Ohio)	LoadBal_ID201.0	27	267	43	1510	6	2	46	538
Inst-23	Buckt-127.0	Asia Pacific (Mumbai)	LoadBal_ID127.0	29	335	25	2051	23	2	21	527
Inst-24	Buckt-148.0	Asia Pacific (Mumbai)	LoadBal_ID231.0	32	297	20	1601	8	4	28	342
Inst-25	Buckt-162.0	Asia Pacific (Mumbai)	LoadBal_ID221.0	36	306	40	1602	21	5	39	440
Inst-26	Buckt-112.0	US East (Ohio)	LoadBal_ID184.0	26	247	38	1084	16	3	12	746
Inst-27	Buckt-139.0	US East (Ohio)	LoadBal_ID187.0	41	337	37	2176	13	2	43	464
Inst-28	Buckt-123.0	US East (Ohio)	LoadBal_ID189.0	46	343	39	1694	7	3	26	741
Inst-29	Buckt-127.0	Asia Pacific (Mumbai)	LoadBal_ID189.0	47	342	42	4437	16	5	18	755
Inst-30	Buckt-163.0	US East (Ohio)	LoadBal_ID151.0	27	244	34	2225	18	4	32	348
Inst-31	Buckt-190.0	US East (Ohio)	LoadBal_ID208.0	22	239	40	3086	19	3	7	561
Inst-32	Buckt-119.0	US East (Ohio)	LoadBal_ID145.0	27	341	29	1586	16	3	38	460
Inst-33	Buckt-105.0	Asia Pacific (Mumbai)	LoadBal_ID176.0	42	295	42	3274	24	5	14	755

Inst-34	Buckt-153.0	East (Virginia)	LoadBal_ID237.0	26	350	33	2828	12	5	25	365
Inst-35	Buckt-108.0	Asia Pacific (Mumbai)	LoadBal_ID180.0	25	342	32	1998	15	6	17	689
Inst-36	Buckt-157.0	Asia Pacific (Mumbai)	LoadBal_ID151.0	30	268	38	3568	8	3	36	765
Inst-37	Buckt-153.0	US East (Ohio)	LoadBal_ID189.0	27	264	38	3809	18	5	46	440
Inst-38	Buckt-149.0	South America	LoadBal_ID105.0	42	338	37	3794	14	3	11	696
Inst-39	Buckt-168.0	US East (Ohio)	LoadBal_ID199.0	41	289	40	2617	19	4	35	483
Inst-40	Buckt-111.0	East (Virginia)	LoadBal_ID140.0	38	314	36	2850	24	4	31	425
Inst-41	Buckt-189.0	US East (Ohio)	LoadBal_ID156.0	49	251	41	2572	22	3	49	344
Inst-42	Buckt-136.0	US East (Ohio)	LoadBal_ID122.0	37	349	27	3027	11	3	29	290
Inst-43	Buckt-112.0	US East (Ohio)	LoadBal_ID228.0	47	260	21	3821	14	3	12	358
Inst-44	Buckt-160.0	Asia Pacific (Mumbai)	LoadBal_ID102.0	24	238	31	3193	14	6	6	485
Inst-45	Buckt-188.0	Asia Pacific (Mumbai)	LoadBal_ID189.0	33	262	36	1212	17	4	16	439
Inst-46	Buckt-143.0	US East (Ohio)	LoadBal_ID209.0	38	311	30	4369	17	3	21	528
Inst-47	Buckt-166.0	South America	LoadBal_ID230.0	41	266	26	3555	10	6	20	331
Inst-48	Buckt-172.0	East (Virginia)	LoadBal_ID227.0	22	266	35	3149	14	6	50	370
Inst-49	Buckt-185.0	South America	LoadBal_ID209.0	30	285	34	1783	7	4	40	505
Inst-50	Buckt-108.0	Asia Pacific (Mumbai)	LoadBal_ID143.0	25	353	41	3192	13	4	42	595
Inst-51	Buckt-136.0	East (Virginia)	LoadBal_ID153.0	34	335	21	1877	10	2	47	638
Inst-52	Buckt-182.0	South America	LoadBal_ID237.0	49	332	34	2793	23	2	7	720
Inst-53	Buckt-148.0	Asia Pacific (Mumbai)	LoadBal_ID115.0	30	253	40	3752	22	3	40	632
Inst-54	Buckt-126.0	Asia Pacific (Mumbai)	LoadBal_ID188.0	44	322	29	3925	21	3	34	316
Inst-55	Buckt-195.0	Asia Pacific (Mumbai)	LoadBal_ID112.0	46	257	40	2542	14	4	51	495
Inst-56	Buckt-200.0	Asia Pacific (Mumbai)	LoadBal_ID162.0	26	270	23	2076	15	3	29	405
Inst-57	Buckt-191.0	US East (Ohio)	LoadBal_ID172.0	35	313	24	2462	15	3	48	537
Inst-58	Buckt-165.0	Asia Pacific (Mumbai)	LoadBal_ID201.0	33	267	21	1062	10	3	48	654
Inst-59	Buckt-145.0	Asia Pacific (Mumbai)	LoadBal_ID121.0	45	346	33	2933	21	4	32	686
Inst-60	Buckt-112.0	East (Virginia)	LoadBal_ID186.0	44	270	32	3605	15	6	49	387

Table 1 describes the sample virtual machines and its metrics in the real-time cloud servers. In this table, different types of cloud servers and its virtual machine properties are captured from the real-time cloud servers.

4.1 CPU utilization

It is the sum of work processed by the cloud server for task completion.

Table 2 illustrates the performance of the proposed multi-level load balancing based parameters estimation on different small type virtual machines and tasks. In this table, different tasks are experimental tested on the small virtual machines for CPU utilization in cloud computing environment. Also, from the table, it is noted that the proposed statistical multi-level load balancing based parameter estimation is better than the conventional models in small cloud VMs and its tasks are considered. Various figures can be accepted. Here, in this study multiple virtual cloud instances of each size 1GB RAM are used to evaluate the load balancing property in the Table 2.

Figure 2 illustrates the performance of the proposed multi-level load balancing based parameters estimation on different medium type virtual machines and tasks. In this figure, different tasks are experimental tested on the medium virtual machines for CPU utilization in cloud computing environment. Also, from the figure, it is noted that the proposed statistical multi-level load balancing based parameter estimation is better than the conventional models in medium cloud VMs and its

tasks are considered. In the Figure 2, different statistical parameters are estimated using the cloud instances and its CPU utilization are shown in the figure. From the Figure 2, it is noted that the different tasks are assigned to the statistical load data collector and balancer for CPU utilization in percentage. In the proposed scenario, CPU utilization is optimal due to efficient data collection and statistical parameter estimation on the given variables. Here, in this study multiple virtual cloud instances of each size 4GB RAM are used to evaluate the load balancing property in the Table 2.

Table 2. Performance analysis of the multi-level load balancing based parameter estimation by using small type of VMs for CPU Utilization (%)

Small	DLB	LB-BC	LBRC	PSO – Firefly	Proposed
Task-2	63	58	64	48	79
Task-4	58	60	57	60	77
Task-6	54	58	50	61	86
Task-8	55	66	66	59	87
Task-10	65	60	51	64	79
Task-12	62	56	46	51	75
Task-14	62	57	56	53	88
Task-16	58	47	57	58	73
Task-18	67	49	56	52	87
Task-20	51	50	58	45	74

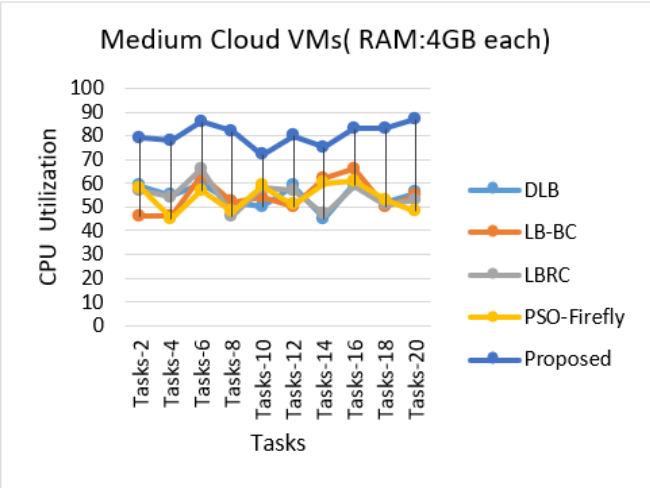

Figure 2. Performance analysis of the multi-level load balancing based parameter estimation by using medium type of VMs for CPU utilization

Figure 3 illustrates the performance of the proposed multi-level load balancing based parameters estimation on different large type virtual machines and tasks. In this figure, different tasks are experimental tested on the large virtual machines for CPU utilization in cloud computing environment. Also, from the figure, it is noted that the proposed statistical multi-level load balancing based parameter estimation is better than the conventional models in large cloud VMs and its tasks are considered. Here, in this study multiple virtual cloud instances of each size 16GB RAM are used to evaluate the load balancing property in the Table 2.

The main relationship between the Table 1, Figure 2 and Figure 3 are:

1. Each task takes different instance RAM size for data collection and parametric estimation.
2. Each task run dynamically in order to predict the CPU utilization for data storage and task execution.

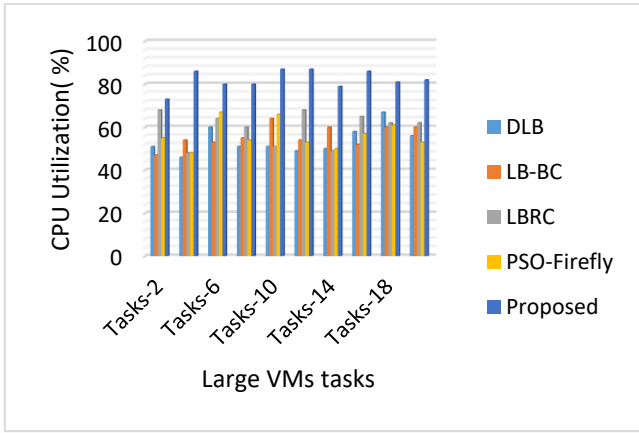

Figure 3. Performance analysis of the multi-level load balancing based parameter estimation by using large type of VMs for CPU utilization

4.2 Memory utilization

It is the amount of memory required to process the task in the cloud server.

Table 3 illustrates the performance of the proposed multi-level load balancing based parameters estimation on different small type virtual machines and tasks. In this table, different tasks are experimental tested on the small virtual machines for memory utilization in cloud computing environment. Also, from the table, it is noted that the proposed statistical multi-level load balancing based parameter estimation is better than the conventional models in small cloud VMs and its tasks are considered. Here, in this study multiple virtual cloud instances of each size 1GB RAM are used to evaluate the load balancing property in the Table 3.

Table 3. Performance analysis of the multi-level load balancing based parameter estimation by using small type of VMs for Memory Utilization (%)

Small	DLB	LB-BC	LBRC	PSO – Firefly	Proposed
Task-2	60	57	67	56	86
Task-4	65	62	45	56	82
Task-6	45	61	48	62	81
Task-8	48	49	60	58	83
Task-10	63	64	66	48	80
Task-12	48	48	50	47	81
Task-14	63	57	63	66	84
Task-16	48	46	48	53	81
Task-18	63	58	55	52	80
Task-20	59	47	48	63	85

Figure 4 illustrates the performance of the proposed multi-level load balancing based parameters estimation on different medium type virtual machines and tasks. In this figure, different tasks are experimental tested on the medium virtual machines for memory utilization in cloud computing environment. Also, from the table, it is noted that the proposed statistical multi-level load balancing based parameter estimation is better than the conventional models in medium cloud VMs and its tasks are considered. Here, in this study multiple virtual cloud instances of each size 4GB RAM are used to evaluate the load balancing property in the Figure 4.

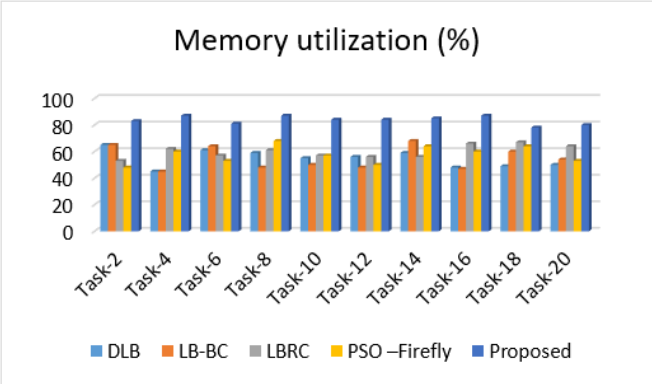

Figure 4. Performance analysis of the multi-level load balancing based parameter estimation by using medium type of VMs for memory utilization (%)

Table 4. Performance analysis of the multi-level load balancing based parameter estimation by using large type of VMs for memory utilization

Large VMs	DLB	LB-BC	LBRC	PSO – Firefly	Proposed
Task-2	52	49	62	67	75
Task-4	62	53	64	56	90
Task-6	67	65	61	46	89
Task-8	49	55	65	51	91
Task-10	59	60	61	53	86
Task-12	67	60	58	46	75
Task-14	49	53	46	60	77
Task-16	47	62	46	55	75
Task-18	60	57	61	46	84
Task-20	52	54	63	58	74

Table 4 illustrates the performance of the proposed multi-level load balancing based parameters estimation on different large type virtual machines and tasks. In this table, different tasks are experimental tested on the large virtual machines for memory utilization in cloud computing environment. Also, from the table, it is noted that the proposed statistical multi-level load balancing based parameter estimation is better than the conventional models in large cloud VMs and its tasks are considered. Here, in this study multiple virtual cloud instances of each size 16GB RAM are used to evaluate the load balancing property in the Table 4.

4.3 Runtime analysis

It is the amount of time required to process the multiple requested jobs in the Queue. In the runtime analysis(ms), different cases of load balancing on tasks are performed based on the small, medium and large instances with 1GB,4GB and 16GB ram capabilities.

Table 5 illustrates the performance of the proposed multi-level load balancing based parameters estimation on different small type virtual machines and tasks. In this table, different tasks are experimental tested on the small virtual machines for runtime analysis in cloud computing environment. Also, from the table, it is noted that the proposed statistical multi-level load balancing based parameter estimation is better than the conventional models in small cloud VMs and its tasks are considered. Here, in this study multiple virtual cloud instances of each size 1GB RAM are used to evaluate the load balancing property in the table.

Table 5. Performance analysis of the multi-level load balancing based parameter estimation by using small type of VMs for runtime analysis (RAM:1GB each)

Small VMs	DLB	LB-BC	LBRC	PSO – Firefly	Proposed
Task-2	5471	4903	4684	4791	4002
Task-4	5181	5806	4846	4945	4141
Task-6	5699	5432	4870	5228	4257
Task-8	4828	5833	5450	5576	4008
Task-10	5742	5417	5659	5275	4359
Task-12	5160	4668	5710	5806	4075
Task-14	5007	5684	5388	5510	4119
Task-16	5816	5396	5511	4989	4291
Task-18	5032	5223	5020	5037	4210
Task-20	4924	4871	5267	4822	4015

Table 6. Performance analysis of the multi-level load balancing based parameter estimation by using medium type of VMs for runtime analysis (4GB RAM)

Medium VMs	DLB	LB-BC	LBRC	PSO – Firefly	Proposed
Task-2	5696	4843	5719	4834	3806
Task-4	5268	5072	5052	4766	4177
Task-6	5550	5669	5744	5780	4188
Task-8	4786	5292	5282	5400	3881
Task-10	4998	4829	5158	5375	4110
Task-12	5607	4956	5647	5471	3688
Task-14	4844	5570	5001	5344	4169
Task-16	5642	5648	5670	5168	4014
Task-18	4979	4976	5214	5039	3870
Task-20	5001	5057	5267	4652	3785

Table 6 illustrates the performance of the proposed multi-level load balancing based parameters estimation on different large type virtual machines and tasks. In this table, different tasks are experimental tested on the medium virtual machines for runtime analysis in cloud computing environment. Also, from the table, it is noted that the proposed statistical multi-level load balancing based parameter estimation is better than the conventional models in medium cloud VMs and its tasks are considered. Here, in this study multiple virtual cloud instances of each size 4GB RAM are used to evaluate the load balancing property in the table.

Table 7. Performance analysis of the multi-level load balancing based parameter estimation by using large type of VMs for runtime analysis (16GB RAM)

Large VMs	DLB	LB-BC	LBRC	PSO – Firefly	Proposed
Task-2	5231	5323	5706	4718	3746
Task-4	5176	4946	5020	5569	3489
Task-6	4786	4702	5359	5277	3794
Task-8	4734	5238	5285	5485	3432
Task-10	5491	5506	5271	4863	3526
Task-12	5250	4791	5388	5170	3821
Task-14	5089	4839	5184	5017	3890
Task-16	5605	5398	4820	4865	3922
Task-18	5603	4927	4875	4886	3703
Task-20	5129	5722	4821	5199	3452

Table 7 illustrates the performance of the proposed multi-level load balancing based parameters estimation on different large type virtual machines and tasks. In this table, different tasks are experimental tested on the large virtual machines for

runtime analysis in cloud computing environment. Also, from the table, it is noted that the proposed statistical multi-level load balancing based parameter estimation is better than the conventional models in large cloud VMs and its tasks are considered. Here, in this study multiple virtual cloud instances of each size 16GB RAM are used to evaluate the load balancing property in the table.

4.4 Reliability analysis

In the reliability analysis, load balancer scheduling and optimization models can increase the reliability by monitoring incoming and outgoing network I/O among the tasks. In the reliability analysis, different cases of load balancing on tasks are performed based on the small, medium and large instances with 1GB,4GB and 16GB RAM capabilities.

Table 8. Performance analysis of the multi-level load balancing based parameter estimation by using large type of VMs for reliability analysis (1GB RAM)

Large VMs	DLB	LB-BC	LBRC	PSO – Firefly	Proposed
Task-2	65	69	70	77	90
Task-4	65	66	78	68	93
Task-6	69	74	77	68	96
Task-8	68	73	70	69	90
Task-10	75	69	72	74	94
Task-12	73	72	70	71	94
Task-14	74	66	67	68	95
Task-16	72	75	78	72	95
Task-18	75	77	75	66	90
Task-20	74	70	68	76	95

Table 8 illustrates the performance of the proposed multi-level load balancing based parameters estimation on different large type virtual machines and tasks. In this figure, different tasks are experimental tested on the large virtual machines for reliability in cloud computing environment. Also, from the figure, it is noted that the proposed statistical multi-level load balancing based parameter estimation is better than the conventional models in large cloud VMs and its tasks are considered. Here, in this study multiple virtual cloud instances of each size 1GB RAM are used to evaluate the load balancing property in the table.

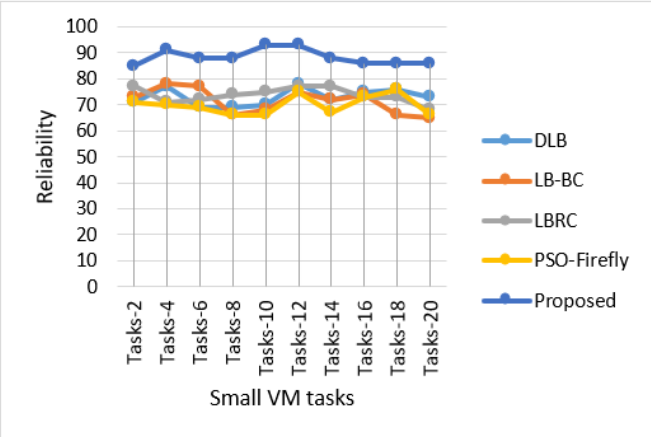

Figure 5. Performance analysis of the multi-level load balancing based parameter estimation by using small type of VMs for reliability analysis (4GB RAM)

Figure 5 illustrates the performance of the proposed multi-level load balancing based parameters estimation on different large type virtual machines and tasks. In this figure, different tasks are experimental tested on the small virtual machines for reliability in cloud computing environment. Also, from the figure, it is noted that the proposed statistical multi-level load balancing based parameter estimation is better than the conventional models in small cloud VMs and its tasks are considered. Here, in this study multiple virtual cloud instances of each size 4GB RAM are used to evaluate the load balancing property in the figure.

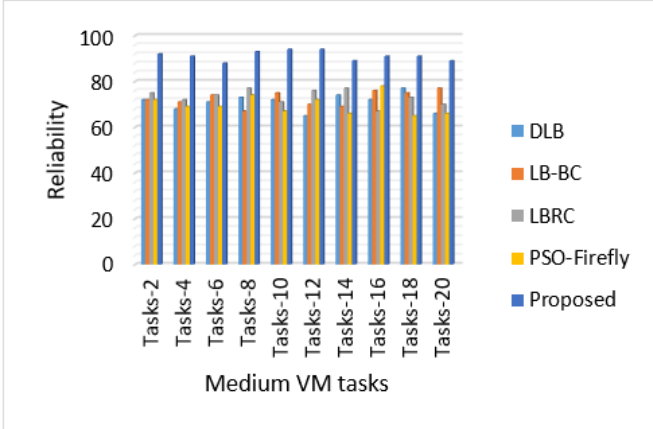

Figure 6. Performance analysis of the multi-level load balancing based parameter estimation by using medium type of VMs for reliability analysis (16 GB RAM)

Figure 6 illustrates the performance of the proposed multi-level load balancing based parameters estimation on different medium type virtual machines and tasks. In this figure, different tasks are experimental tested on the medium virtual machines for reliability in cloud computing environment. Also, from the figure, it is noted that the proposed statistical multi-level load balancing based parameter estimation is better than the conventional models in medium cloud VMs and its tasks are considered. Here, in this study multiple virtual cloud instances of each size 16GB RAM are used to evaluate the load balancing property in the figure.

5. CONCLUSION

In this paper, a novel statistical load balancing metrics are estimated in the real-time cloud computing environment on large number of instances and tasks. Since, most of the conventional models are based on limited number of tasks and VMs in the cloud environment; it is difficult to monitor the different cloud-based applications due to high computational time and memory. In this work, proposed framework is implemented in the real-time amazon AWS instances with a large number of tasks. Experimental results proved that the multi-level statistical load balancing parameters estimation model has nearly 2% optimization in terms of reliability, runtime, memory utilization and CPU utilization.

REFERENCES

[1] Abutayeh, M., Padilla, R.V., Lake, M., Lim, Y.Y., Garcia, J., Sedighi, M., Too, Y.C.S., Jeong, K. (2019). Effect of

short cloud shading on the performance of parabolic trough solar power plants: motorized vs manual valves. Renewable Energy, 142: 330-344. https://doi.org/10.1016/j.renene.2019.04.094

[2] Adhikari, M., Amgoth, T., Srirama, S.N. (2020). Multi-objective scheduling strategy for scientific workflows in cloud environment: A Firefly-based approach. Applied Soft Computing, 93: 106411. https://doi.org/10.1016/j.asoc.2020.106411

[3] Adhikari, M., Nandy, S., Amgoth, T. (2019). Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud. Journal of Network and Computer Applications, 128: 64-77. https://doi.org/10.1016/j.jnca.2018.12.010

[4] Alonso-Monsalve, S., García-Carballeira, F., Calderón, A. (2018). A heterogeneous mobile cloud computing model for hybrid clouds. Future Generation Computer Systems, 87: 651-666. https://doi.org/10.1016/j.future.2018.04.005

[5] Chaudhary, D., Kumar, B. (2018). Cloudy GSA for load scheduling in cloud computing. Applied Soft Computing, 71: 861-871. https://doi.org/10.1016/j.asoc.2018.07.046

[6] Chaudhary, D., Kumar, B. (2019). Cost optimized hybrid genetic-gravitational search algorithm for load scheduling in cloud computing. Applied Soft Computing, 83: 105627. https://doi.org/10.1016/j.asoc.2019.105627

[7] Elrotub, M., Gherbi, A. (2018). Virtual machine classification-based approach to enhanced workload balancing for cloud computing applications. Procedia Computer Science, 130: 683-688. https://doi.org/10.1016/j.procs.2018.04.120

[8] Gesvindr, D., Gasior, O., Buhnova, B. (2020). Architecture design evaluation of PaaS cloud applications using generated prototypes: PaaSArch Cloud Prototyper tool. Journal of Systems and Software, 169: 110701. https://doi.org/10.1016/j.jss.2020.110701

[9] Gholipour, N., Arianyan, E., Buyya, R. (2020). A novel energy-aware resource management technique using joint VM and container consolidation approach for green computing in cloud data centers. Simulation Modelling Practice and Theory, 104: 102127. https://doi.org/10.1016/j.simpat.2020.102127

[10] Gill, S.S., Tuli, S., Xu, M.X., Singh, I., Singh, K.V., Lindsay, D., Tuli, S., Smirnova, D., Singh, M., Jain, U., Pervaiz, H., Sehgal, B., Kaila, S.S., Misra, S., Aslanpour, M.S., Mehta, H., Stankovski, V., Garraghan, P. (2019). Transformative effects of IoT, blockchain and artificial intelligence on cloud computing: Evolution, vision, trends and open challenges. Internet of Things, 8: 100118. https://doi.org/10.1016/j.iot.2019.100118

[11] Guo, J., Li, C., Chen, Y., Luo, Y. (2020). On-demand resource provision based on load estimation and service expenditure in edge cloud environment. Journal of Network and Computer Applications, 151: 102506. https://doi.org/10.1016/j.jnca.2019.102506

[12] Habibi, M., Fazli, M., Movaghar, A. (2019). Efficient distribution of requests in federated cloud computing environments utilizing statistical multiplexing. Future Generation Computer Systems, 90: 451-460. https://doi.org/10.1016/j.future.2018.08.032

[13] Hamid-Lakzaeian, F. (2020). Point cloud segmentation and classification of structural elements in multi-planar masonry building facades. Automation in Construction, 118: 103232. https://doi.org/10.1016/j.autcon.2020.103232

[14] Jodayree, M., Abaza, M., Tan, Q. (2019). A predictive workload balancing algorithm in cloud services. Procedia Computer Science, 159: 902-912. https://doi.org/10.1016/j.procs.2019.09.250

[15] Liang, Y., Qi, G., Zhang, X., Li, G. (2019). The effects of e-Government cloud assimilation on public value creation: An empirical study of China. Government Information Quarterly, 36(4): 101397. https://doi.org/10.1016/j.giq.2019.101397

[16] Mansouri, N., Ghafari, R., Zade, B.M.H. (2020). Cloud computing simulators: A comprehensive review. Simulation Modelling Practice and Theory, 104: 102144. https://doi.org/10.1016/j.simpat.2020.102144

[17] Mashhadi Moghaddam, S., O'Sullivan, M., Walker, C., Fotuhi Piraghaj, S., Unsworth, C.P. (2020). Embedding individualized machine learning prediction models for energy efficient VM consolidation within Cloud data centers. Future Generation Computer Systems, 106: 221-233. https://doi.org/10.1016/j.future.2020.01.008

[18] Nyasulu, M., Haque, M.M., Boiyo, R., Kumar, K.R., Zhang, Y.L. (2020). Seasonal climatology and relationship between AOD and cloud properties inferred from the MODIS over Malawi, Southeast Africa. Atmospheric Pollution Research, 11(11): 1933-1952. https://doi.org/10.1016/j.apr.2020.07.023

[19] Premarathne, U.S., Rajasingham, S. (2020). Trust based multi-agent cooperative load balancing system (TCLBS). Future Generation Computer Systems, 112: 185-192. https://doi.org/10.1016/j.future.2020.01.037

[20] Rafieyan, E., Khorsand, R., Ramezanpour, M. (2020). An adaptive scheduling approach based on integrated best-worst and VIKOR for cloud computing. Computers & Industrial Engineering, 140: 106272. https://doi.org/10.1016/j.cie.2020.106272

[21] Sahil, Sood, S.K. (2019). Smart vehicular traffic management: An edge cloud centric IoT based framework. Internet of Things, 100140. https://doi.org/10.1016/j.iot.2019.100140

[22] Marin, A., Balsamo, S., Fourneau, J.M. (2017). LB-networks: A model for dynamic load balancing in queueing networks. Performance Evaluation, 115: 38-53. https://doi.org/10.1016/ j.peva.2017.06.004

[23] Zhao, J., Yang, K., Wei, X., Ding, Y., Hu, L., Xu, G. (2016). A heuristic clustering-based task deployment approach for load balancing using bayes theorem in cloud environment. IEEE Transactions on Parallel and Distributed Systems, 27(2): 305-316. https://doi.org/10.1109/TPDS.2015.2402655

[24] Priya, V., Sathiya Kumar, C., Kannan, R. (2019). Resource scheduling algorithm with load balancing for cloud service provisioning. Applied Soft Computing, 76: 416-424. https://doi.org/10.1016/j.asoc.2018.12.021

[25] Golchi, M.M., Saraeian, S., Heydari, M. (2019). A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation. Computer Networks, 162: 106860. https://doi.org/10.1016/j.comnet.2019.106860