DOI 10.12694/scpe.v20i2.1538
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 433–456. http://www.scpe.org

DYNAMIC TASK SCHEDULING USING BALANCED VM ALLOCATION POLICY FOR FOG COMPUTING PLATFORMS

SIMAR PREET SINGH[*] ANAND NAYYAR[†] HARPREET KAUR[‡] AND ASHU SINGLA[§]

Abstract. The fog computing models are getting popular as the demand and capacity of data processing is rising for the various applications every year. The fog computing models incorporate the various task scheduling algorithms for the resource selection among the given list of virtual machines (VMs). The task scheduling models are designed around the various task metrics, which include the task length (time), energy, processing cost etc. for the various purposes. The cost oriented scheduling models are primarily built for the customer's perspectives, and saves them a handful amount of money by efficiently assigning the resources for the tasks. In this paper, we have worked upon the multiple task scheduling models based upon the Local Regression (LR), Inter Quartile Range (IQR), Local Regression Robust (LRR), Non-Power Aware (NPA), Median Absolute Deviation (MAD), Dynamic Voltage and Frequency Scheduling (DVFS) and The Static Threshold (THR) methods using the ifogsim simulation designed with the 50 nodes and 50 virtual machines, i.e. 1 virtual machine per node. All of the models have been implemented using the standard input simulation parameters for the purpose of performance assessment in the various domains, specifically in the time domain and effective consumption of energy. The results obtained from the experiments have shown the overall time of 86,400 seconds during the simulation, where the DVFS has been recorded with the 52.98 kWh consumption of energy, which shows the efficient processing in comparison to the 150.68 kWh of energy consumption in the NPA model. Also, there are no SLA violations recorded during both of the simulation, because no VM migration model has been utilized among both of the implemented models, which clearly shows that the VM migrations are the major cause of SLA violation cases. The LRR (2520 VMs) has been observed as best contender on the basis of mean of number of VM migrations in comparison with LR (2555 VMs), THR (4769 VMs), MAD (5138 VMs) and IQR (5352 VMs).

Key words: VM allocation, VM selection, fog computing, task scheduling, ifogsim simulator.

AMS subject classifications. 68M14, 90B35

1. Introduction.

In this era, the cloud computing applications are getting popular and more online applications are opting for the cloud computing platforms to effectively execute, manage and optimize the applications [1, 2]. The cloud computing environments provide the flexible application hosting plans, which are primarily divided in three infrastructural variants: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [3, 4, 5, 6].

The SaaS plans offer the hosting of software or application without worrying about the platform and infrastructure related operations, whereas PaaS plans enable the user to take full control over the operating system environment, and can effectively optimize the application performance on the platform level [7, 8]. On the other hand, the IaaS service includes the internal network of various systems (particularly VMs in this case) altogether, which are used to run the applications with high user count. Cloud computing grids are owned by the cloud operators, and is implemented in few grids across the world [3, 9, 10]. Because the cloud computing infrastructure is quite expensive, it is always implemented in form of small number of grids across the globe and provides a high-performance service with abundance of processing resources, i.e. CPU, RAM and storage. When cloud computing is known for a processing powerhouse, it has one primary disadvantage, which is associated with communication cost (i.e. the extra time delay to transfer the request and request-reply between the cloud & end user) [11, 12, 13, 14].

As described the primary disadvantage of cloud computing in the form of communication cost is the preference of extending the cloud computing services on the edge (the computing on the edge). There are several extensions of the cloud computing services, which forms fog computing, edge computing and content delivery networks (CDNs) [15, 16, 17, 18, 19]. The CDNs offer frequent data caching services, which enables the rapid delivery of frequently accessed data from the cloud resources. The frequently requested data is saved in the

[*]Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, India – corresponding author (dr.simarpreetsingh@gmail.com)
[†]Graduate School, Duy Tan University, Da Nang, Vietnam (anandnayyar@duytan.edu.vn)
[‡]Computer Science and Engineering Department, Chandigarh University, Mohali, Punjab, India (harpreet8307@gmail.com)
[§]Computer Science and Engineering Department, Sant Longowal Institute of Engineering and Technology, Longowal, Punjab, India (ashusinglaoct@gmail.com)

caching memory on the internet service providers (ISP) network, which does not offer any additional service [20, 21, 22]. For example, when you browse Facebook website on your PC or smart phone, most of the data associated with your profile and friends is loaded from the local CDN offered by ISP. The edge computing, on the other hand provides the distributed smart grid services, which enables the use of user end nodes to compute the data [23, 24]. The Search for Extraterrestrial Intelligence (SETI) project uses distributed smart grid over the internet by enabling the user nodes to process the satellite data in small chunks per node, and pretty well describes the concept of edge computing. On the contrary, the fog computing is the semi-centralized processing paradigm, which extends the cloud computing close to edge nodes [25, 26, 27]. The semi-centralized infrastructure is owned by cloud operators or its business associates to effectively offer the services with optimized and reduced communication cost, as well as extends the overall processing power of the cloud computing. Unlike, the edge computing and CDN, the fog computing offers the complete service package, which hosts the computing resources and offers computing, storage and event-based or need-based synchronization with primary cloud using synchronous or asynchronous archetypes [28, 29, 30, 31, 32].

In this paper, the proposed model is design and developed to effectively schedule the user tasks on the fog computing resources by combining the VM allocation and VM selection methods in the perfect arrangement. Various methods associated with VM allocation & VM selection are evaluated and combined in suitable combination to discover the best task scheduling combination for the effective and optimized user data processing.

The paper structure is as follows: This section (Sect. 1) discusses the introduction of cloud and fog computing technologies. Next section (Sect. 2) covers the literature review. Section 3 explains the decision parameters (Sect. 3.2) and the proposed algorithm (Sect. 3.3). Section 4 describes the results that are computed using the proposed approach. Finally, Sect. 5 describes the conclusion and future directions.

2. Related Work. Zhuo Tang et al. [33] proposed DVFS enabled Energy Efficient Workflow Task Scheduling algorithm (DEWTS). They used the scheduling order of all the tasks to obtain the makespan in their algorithm. The authors used different algorithms for computation of deadlines. In overall process, their proposed algorithm was able to reduce total power consumption by upto 46.5% for parallel applications. The authors worked on randomly generated workflows in their research work.

Yuan Fang et al. [34] discussed Cyber-Physical Systems (CPS) and proposed Simple and Proximate Time Model (SPTimo) framework. In addition to this, the authors also presented Mix Time Cost and Deadline First (MTCDF) time task scheduling algorithm, which was based on computation model of SPTimo framework. Their research provides an optimal scheduling solution in total time required and time cost parameters.

Zhao, Qing et al. [35] has implemented the energy-aware scheduling of the user tasks over the cloud computing resources. This scheme generates the task binary tree based upon task correlation, which is used to prioritize the user tasks. The authors proposed the Task Requirement Degree (TRD) based calculation method for proficient scheduling, where it also considers the bandwidth to optimize the communication cost.

Nidhi Bansal et al. [36] designed the QoS enabled optimized cost-based scheduling methodology. The authors have focused upon the cost of computing resources (virtual machines) to schedule the given pool of the tasks over the cloud computing model. The cost optimization has been performed over the QoS-task driven task scheduling mechanism, which did not encounter the cost optimization problem earlier. The authors have shown that the earlier QoS-driven task scheduling based studies has been considered the makespan, latency and load balancing. The QoS-based cost evaluation model evaluates the resource computing cost for the scheduling along with the other parameters as in their secondary precedence.

Gaurang patel et al. [37] have worked upon enhancement in the existing algorithm of Min-Min (Minimum-minimum methodology) for scheduling on cloud platform. The authors have proposed the use of active load balancing in processing the tasks over the cloud environments. The authors have proposed the new method for the efficient processing of tasks over the given cloud environment known as the Enhanced Load Balanced Min-Min (ELBMM) algorithm. The authors have recovered the major drawback of the existing model of Min-Min algorithm, where sometimes the makespan and current resource utilization is not properly considered and the tasks is scheduled over the slow resource which causes the latency. In their research, they have effectively overcome the problem concerned with the Min-Min algorithm. The authors have proved their model better than the Min-Min and ELBMM model for the task scheduling. Also, the execution times has been reduced to the optimum levels, and better than the existing model.

Weiwei Chen et al. [38] have proposed the imbalanced metrics for the optimization of task clustering on scientific workflow data executions. The authors have examined the imbalanced nature of the task clustering during the runtime evaluation for the purpose of task clustering in depth. The authors have proposed the improvement to effectively evaluate the problem of runtime task imbalance. The authors have proposed an horizontal and vertical method for the evaluation of series of task clustering for the widely used scientific workflows. Their proposed model has utilized the in-depth metric values for the real time evaluation of their research model.

Xu et al. [39] has worked towards the load balancing of the user tasks, which considers the task partitioning on cloud. The load balancing methods are known to be effective for efficient user task processing on cloud resources, because clouds generally receive high volumes of user data. Y. Tan et. al. [40] worked on a novel scheduling technique for cloud models. The authors complimented the use of particle swarm optimization (PSO) model to analyze the scheduling performance in the terms of delay and resource consumption. An optimized weight based mutation criteria with adaptable indolence oriented methodology is deployed to optimize the scheduling performance. Additionally, this scheme offers the load balancing schema to effectively schedule the user tasks.

K. Li et al. [41] described the feasible resource expansion for centralized, de-centralized and semi-centralized computing platforms, which also involve the parallel processing paradigm. The scheduling problem is described as NP-hard problem, and suggested several feasible solutions to effectual scheduling of the allocated computing resources. The authors proposed the swarm optimization (ACO oriented solution) to deploy the load balancing as effective meta-heuristic scheduling elucidation for the cloud platforms by reducing the individual load and effectively distributing the tasks of multiple users altogether.

X. Luo et al. [42] proposed an algorithm for resource scheduling under cloud computing environment. It is different from the under conventional distributed computing domain because of the high scalability and heterogeneity of computing resources in cloud computing domain. In this paper, based on dynamic load balance, the authors has proposed a resource-scheduling algorithm. The different statistic transferring power and retard between nodes in cloud as well statistic-processing power of nodes in cloud is considered in this algorithm. To increase the efficiency of cloud computing and reduce the median response time of tasks, the algorithm selects the best node to fulfill the task. The simulation results show that the algorithm reduces the average response time of tasks.

N. Bessis et al. [43] discussed in their paper about the new technologies develop fast and their complexity becomes a crucial concern. One proven way to deal with improved complexity was to engage a self-organizing strategy. The many different strategies exists that deal with the load balancing problem but most of the problem are task oriented and it is, therefore, hard to differentiate. So, the researchers of the paper developed and implemented a generic architectural pattern, called self-initiative load balancing agents. It allocates the exchange of different algorithms, both sightful and dense ones, through plugging. In placing at different levels, different algorithms can be tested in combination. The objective was simplicity in the selection of optimal algorithm for a definite problem. Self-initiative load balancing agent was the concern and domain independent, and can be collected towards inconsistent network topologies.

A. Jain and R. Singh [44] described grid computing for classification of non-identical resources that are cast off as virtual resource to a user and impart superior grid domain. Now-a-days, large amount of resource management in peer-to-peer grid environment is used. Load balancing is crucial concern to balance the overall load of the nodes. There are numbers of solutions to achieve load equality state. ACO is used to provide optimal solution for solving a problem of load balancing. In the paper, the authors has proposed Master-Ant Colony Optimization algorithm (M-ACO), and it is used in peer-to-peer environment. The proposed algorithm gives better results in peer-to-peer environment. MATLAB simulation tool was used, which provides different kinds of functions to bloom heuristic algorithms with new notions.

R. Chaukwale et al. [45] discussed the complication of efficiently scheduling jobs on several devices, it is a vital consideration when operating the Job Shop Production (JSP) scheduling system. JSP was a NP hard difficulty. The procedures that focus on fabricating an exact solution of the problem can evince insufficiency in discovering an optimal solution of the problem to Job Shop Production system (JSP). Hence, in such conditions, heuristic methods can be developed to discover a good solution of the problem within reasonable time period.

In their paper, the authors studied the traditional ACO algorithm and has proposed a load balancing ACO algorithm for JSP. The paper also presented the observed results. It was noticed that the proposed algorithm showed better outcomes when compared to traditional ACO. Many researches [46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 70, 71] discussed about scheduling and allocation methods in fog and cloud environments.

After going through the related work, it was found that with the increase of Internet of Things (IoT) devices, sensors, fog devices, actuators etc., lots of data is getting generated. This will lead to network congestion in coming future. Thus, there is a huge need to schedule and allocate the tasks, that are dynamic in nature, in a proper planned/optimal manner. This research work tries to simplifies the future arising problems in the area of fog computing.

3. Methods and Materials. The fog scheduling solution proposed in this paper is implemented using the ifogsim simulator considering the fog environment. Ifogsim simulator is based upon cloudsim platform for cloud infrastructure simulations. The proposed scheme combines VM allocation & VM selection procedures with performance optimization methods to boost the cloud's capability for user task processing. An idle process sequencing algorithm should be aimed at reducing the overall tasking overhead, tasking time (task completion time) and communication overhead by the whole task considering the incoming and outgoing information. The task management faces the major challenges from the bias-free dynamic resource allocation while keeping the cloud performance to the maximum in terms of execution time and computational overheads. This scheme offers the load balanced paradigm over user task stack, coupled with environmental parameter optimization, and enhances the endurance and general capability of the cloud environment.

3.1. Proposed Approach. The link optimization algorithm is designed as an intelligent solution influenced by behavior of the real Internet of Things (IoT) inter-nodal relations in scenario of increasing number of IoT nodes. A collaboration of IoT nodes in finding the appropriate paths and doing other tasks has been prioritized to achieve the link behavior in cloud systems. The fog resources store the usability for path devising and following while taking a movement from source node to the destination computing resource on cloud environment. With the raise in the number of requests on a singular path, the strength of connection increases on that particular path. The requests of that group select the shortest path on the basis of this usability index. The IoT connection request province optimization method has been applied for resolving the problem of rising number of requests, with the target of discovering the shortest path. The algorithm fully depends upon the history of usability index to take further judgments for optimal solutions for any of the computational requirement. The use of artificial links for the state of development rule and for the selection of optimal resources beyond the grid computation or the cloud environments has been proposed in the prospective work. The artificial links have been used for the purpose of cloud computing scheduling and shortest path identification. The link province system adopts the arbitrary-proportional rule, which is the state of transition rule used for link optimization system and works on the basis of probability or a chance to choose the optimal resource out of k-resources for task assignment in the cloud. The usability index of a resource depends upon the number of available resources, processing cost and estimated time. The VM load has been selected as the prime factor out of all these three factors; hence the computing decision is computed after verifying the cumulative and individual runtime VM load. The usability indexes are regularly updated using particular cloud resources or VMs selected for the act of scheduling. The shortest path is computed after analyzing the runtime parameters, which effectively analyzes the load, availability, communication cost and processing delay of a virtual machine. The VM runtime parameters are procured and continuously updated, and helps the scheduling decision on the cloud systems. Fig. 3.1 describes the shortest path in distributed and/or segmented sub-paths, and explains the Eqs. 3.1 and 3.2.

$$Prob_A = \frac{(k + A_i)^n}{(k + A_i)^n + (k + B_i)^n}, \qquad (Prob_A + Prob_B = 1) \tag{3.1}$$

$$A_{i+1} = A_i + \delta, \qquad B_{i+1} = B_i + (1 - \delta) \qquad (A_i + B_i = i), \tag{3.2}$$

where δ describes a binary object and carries only 0 or 1 as value, which is computed over the runtime probability values (described as $Prob_A$ & $Prob_A$). The variable stacks A assigns the primary shortest path and B denotes the optimized shortest path over A.

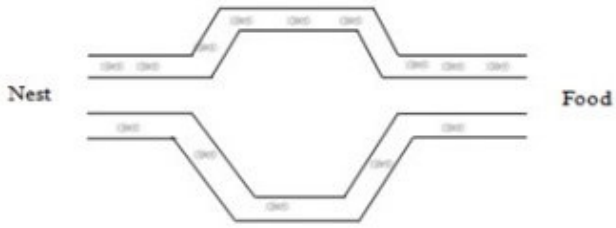

FIG. 3.1. *Shortest path in distributed sub-paths*

3.2. Decision Parameters. The VM load and failure rate has been assigned as the main parameters to take the scheduling decisions. Both of the parameters has been used for the purpose of data scheduling over the given cloud resources. The virtual machine load is the parameter which defines the overall utilization of the resources of the given virtual machine. The VM load can be used to signify the runtime availability in order to process the given task t on the given time t. The tasks running over the given VM, utilizes the certain amount of resources. The total percentage of the resources being used during the time t is considered as the VM load.

When the virtual machines are ordered in the workload allocation pool for process sequencing in the given cloud environment, the load monitoring on each virtual machine becomes very important step to correctly perform the data scheduling tasks. The virtual machine load or overhead is calculated on the basis of different parameters like CPU size, memory size etc. Each VM load must be calculated on the basis of its local parameters. Any use of general parameter values can result the biased load over the given VMs. The CPU and memory overhead or usage on the given VM considerably influences the performance of VMs in the process sequencing practices. The workload on VM can be evaluated on the basis of formula represented in Eq. 3.3. To calculate the total load over the virtual machine in the cloud environment is more than or equal to its capability, the Eq. 3.3 gives the result.

$$\sum_{i}^{[v]} Load_i \times X_{ik} \lesneqq Capacity, \qquad \forall k, P_k \in P \tag{3.3}$$

Finally, to justify the virtual machine load, Eq. 3.4 is used.

$$X_{ik} = x_{ik} \tag{3.4}$$

where X_{ik} is considered as the components of assignment to the non-overloaded VMs. The overloading or non-overloading defines the current state of the VM calculated after computing overall load and percentage of resources and measuring them against the threshold level.

The failure rate is described in the form of percentage of scheduling failures in processing the assigned tasks over the given VMs in the cloud environment. The failure rate signifies the trust of virtual machine. The VM with the lowest failure rate can be considered as the highly trusted VM and vice-versa. The probability of processing of the task can be increased by assigning the tasks over VMs with optimized & reduced failure rate (FR). The FR can be computed by using the Eq. 3.5.

$$FR = (\frac{T_p}{T_t}) \cdot 100 \tag{3.5}$$

where T_p is the sum of processed tasks and T_t is total amount of tasks assigned over the given VM.

3.3. Link Optimization Based Optimal VM Allocation (Link Optimization-OVA). In this work, the optimal load sharing approach based on the link optimization has been introduced for the load offset approach over the cloud environment in the case of data scheduling. The path A defines the first resource and path B defines the second resource. The other resources can be assigned with the further alphabets with the assumption that all of the resources or assets are logically able to executing all the processes in the cloud environment. The resource selection must be done on the basis of availability of RAM and CPU processing

powers, which must make the whole process efficient in terms of response time. The traditional methods are known to allot the random resources for the given task, which effect the performance of cloud scheduling model and hence slow down the query processing procedure resulting with higher response time. The link optimization is the probability-based procedure to choose the appropriate resource in the available list of VMs. The proposed model is aimed at lower task response time for maximizing the number of jobs processing in the span of one second. The proposed model has been made capable of subdividing the task, which facilitates the quicker process and processes the smaller tasks faster than the hefty ones to reduce the overall load and to increase the number of successful requests processing every second. The subdivision of tasks is based on the length of the task. A task is usually divided into $'t'$ slots, where t is smallest time unit available for the task length calculation in our proposed model. A task smaller than or equal to t will be processed in one round, where the tasks larger than t can be scheduled in queue or on different VMs according to the load and time calculation for the faster processing. The arbitrary proportional rule is applied to recognize the ratio of processes in processing the given resource, and has been presented in the Eqs. 3.6 and 3.7.

$$P_1 = \frac{(R_1 + K)^k}{(R_1 + K)^k + (R_2 + K)^h}, \tag{3.6}$$

$$B_1 = P_1 \cdot TR_i, \tag{3.7}$$

where A_1 is the count of assigned tasks on the resource P_1 & A, involves the resource probability, R_1 denotes the usability index based on the available ratio of RAM and CPU on VM under consideration, TR_i depicts the resource availability required to process task i. The k and h are the coefficients used for the choice of probability among the available resources for sequencing of the processes among accessible resources. The value of k and h is calculated on the basis of VM load and resource availability on all of the available VMs. The variation in the values of k and h will define the variability on the basis of current processing load on different VMs, which inspires the task assignment decision of the link optimization algorithm. The used rule for the probability calculation has been represented in the Eq. 3.8.

$$P_j = \frac{(R_i + K)^k}{\sum_{i=1}^{n}((R_i + K)^k)}, \tag{3.8}$$

In the proposed work, the meta tasks are used for testing of the proposed model. The meta tasks does not carry any dependency on other tasks in the processing queue, which means the response time will be calculated for each individual task by evaluating the variation between finish time and start time. The waiting time is also considered as the response time delay, which is caused due to the waiting period spent in the queue.

Figure 3.2 represents the basic flow of Algorithm 1.

4. Results and Discussion. In this research, there are total seven VM allocation and selection policies are described. All seven models are programmed to utilize the different aspects into consideration in order to take the final decision on VM allocation and VM selection for the completion of job assignments. The VM allocation models used in this simulation are Local Regression (LR), Inter Quartile Range (IQR), Local Regression Robust (LRR), Non-Power Aware (NPA), Median Absolute Deviation (MAD), Dynamic Voltage and Frequency Scheduling (DVFS) and Static Threshold (THR) models. The following figures elaborates all of the models implemented under this research paper.

Each of the VM allocation model is further amalgamated with the VM selection models. The NPA and DVFS models are not primarily designed for specific VM selection or allocation policy. The NPA and DVFS models are designed to select all of the available VMs, and allocate sub-tasks or tasks on the optimal resource selected from the list.

Each of the VM allocation model (IQR, LR, LRR, MAD & THR) is combined with all VM selection models including Minimum Migration Time (MMT), Maximum Correlation (MC), Random Selection (RS) and Maximum Utilization (MU). All of the VM selection policies are described in the Fig. 4.1. There are total 22 combinations, which are produced using the combination of VM allocation and selection policies. The Fig. 4.1 shows all of the possible combinations of VM allocation and selection models.

Algorithm 1 Link Optimization - OVA Algorithm

1: Acquire the environmental parameters for task scheduling
2: Analyze & acquire the list of available VMs in the VM stack over cloud segment
3: Analyze & acquire the runtime performance of available VMs in the form of CPU, RAM, storage capacity, power consumption, etc.
4: Represent the acquired parameter list obtained on Step 2 & 3;

$$VM_l = V_1, V_2, V_3, V_4, ...V_n, \tag{3.9}$$

where VM is the virtual machine list and V_1 to V_n represents the virtual machine IDs
5: Obtain cumulative & independent list of resources in the form of computing capacity

$$VM_r = VM_1, VM_2, VM_3, ...VM_n, \tag{3.10}$$

where VM_r represents the resource capacity of each resource VM_1 to VM_n to represent the virtual machine IDs
6: Begin the iterative structure to process tasks with every effective resource
   a. Obtain & acquire the resource availability from every VM on availability stack

$$VM_E = \int_{i=1}^{N} VM_i, \tag{3.11}$$

   where VM_E gives the resource availability after calculating the resource load using Eq. 3.12.

$$L = \frac{VCPU_u}{VCPU_T}, \tag{3.12}$$

   where L represents the overall resource load on the particular VM, whereas the $VCPU_u$ and $VCPU_T$ gives the currently used resources and total resources available respectively.

$$L_i = L_1, L_2, L_3, ...L_n, \tag{3.13}$$

   where L_i represents the list of resource load for all the VMs in simulation.
   b. The fundamental utilization factor is computed for individual resource
7: Terminate the iterative structure initiated on step 5
8: Assign the task stack to runtime cloud environment

$$T = t_1, t_2, t_3, ...t_n, \tag{3.14}$$

where T vector represents the task vector and t_1 to t_n represents the individual tasks
9: Determine the workflow's task stack and compute the length of each independent task in the stack

$$t_c(t_i) = (EST_{finishtime} - EST_{starttime}), \tag{3.15}$$

where t_c and t_i gives the overall time length for each of the task by subtracting the estimated start time from estimated finish time
10: In case a task is dependent or multivariate, sub-divide it into sub-stacks involving minor tasks recognizable with index i
11: Initialize the iterative structure to process each sub-task on sub-task stack indexed with index i
   a. Obtain the resource availability factors for each VM on the VM list
   b. Compute and validate the task duration (estimated) against the computational capacity (resource availability) against each available VM

c. Determine the current load of each VM on the list by analyzing the resource engagement

$$A_j = P_j \cdot TR_i, \tag{3.16}$$

where A_j depicts the availability of the VMs

d. Observe and accumulate failure events of each VM on the list and prepare the FR value to evaluate its endurance

e. Confiscate all the VMs on the list with FR below threshold to process current task of sub-task (t) to prepare the allocated VM resource list ($aVMrl$)

f. Finally select the appropriate resource based upon best combination of time (estimated) and resource engagement from $aVMrl$

$$IfT_c(i) \ln VC(j), \tag{3.17}$$

$$VM_E(K) = V(V_{c(i)}), \tag{3.18}$$

where $VM_E(K)$ resource availability after calculating the resource load for particular machine with id K, where K any can be any value from the given VM IDs. VM represents the virtual machine list and $V_{c(i)}$ gives the capacity of the VM with ID as i.

g. Revise resource allocation record accordingly and also update total load of allocated VM after task assignment

h. Further, revise the utilization record enlisting resource availability

$$R_i = R_j + 1, \tag{3.19}$$

where R_i is the usability and this equation shows the incremental usability index with the movement of each VM.

i. Go the step 9(a) if not end of task list

12: Terminate the iterative structure and exit the program

The simulation results of all the unique combinations are acquired in the form of various performance parameters. These performance parameters are included to analyze the performance on the basis of time, VM migrations, Service Level Agreements (SLA) related parameters, Energy consumption, Host Shutdowns etc. Detailed statistical analysis of host shutdowns, VM & host migrations, VM & host selections and overall time-based analysis in the terms of mean and standard deviation is also computed. The Table 4.1 shows the detailed list of performance parameters.

The simulation of all results, based on the parameters discussed in Table 4.1, are obtained and listed in this section for each of the VM allocation and VM selection models. The only exceptions are Dynamic Voltage Frequency Scaling (DVFS) and Non-Power Aware (NPA) models. For these two exceptions, total 15 parameters are recorded in contrast to the 23 parameters for all other models.

The DVFS model has been described with the random nature, where all of the available VM are used in the random order without any qualitative based allocation parameters. The Fig. 4.2 shows the results obtained for the random DVFS.

In this sub-section, the VM allocation model of Inter Quartile Range (IQR) has been used along with the Maximum correlation (MC) method. Fig. 4.3 the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Inter Quartile Range (IQR) has been used along with the Minimum Migration Time (MMT) method. Fig. 4.4 represents the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Inter Quartile Range (IQR) has been used along with the Maximum Utilization (MU) method. Fig. 4.5 shows the results obtained from this model for all of the enlisted

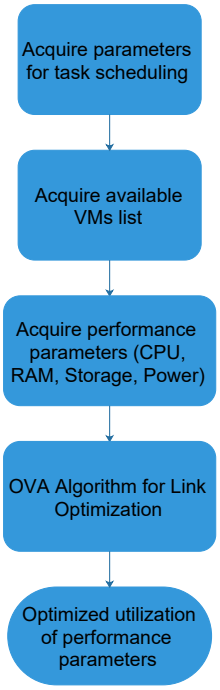

FIG. 3.2. *Basic flow of Proposed Algorithm*

VM ALLOCATION	VM SELECTION
Inter Quartile Range (IQR)	Maximum Correlation (MC)
Local Regression (LR)	Minimum Migration Time (MMT)
Local Regression Robust (LRR)	Maximum Utilization (MU)
Median Absolute Deviation (MAD)	Random Selection (RS)
Non-Power Aware (NPA)	
Dynamic Voltage Frequency Scaling (DVFS)	
The Static Threshold (THR)	

FIG. 4.1. *Possible combinations of VM allocation and VM selection models*

parameters.

In this sub-section, the VM allocation model of Inter Quartile Range (IQR) has been used along with the Random Selection (RS) method. The results shown in Fig. 4.6 is obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Local Regression (LR) has been used along with the Maximum Correlation (MC) method. Fig. 4.7 represents the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Local Regression (LR) has been used along with the Minimum Migration Time (MMT) method. The results represented in Fig. 4.8 are obtained from this model for all of the enlisted parameters.

```
Name of Experiment: random_dvfs
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 52.98 kWh
Migration counts (VM): 0
Service Level Agreement (SLA): 0.00000%
SLA (Performance Degradation): 0.00%
SLA (Per host Elapsed Time): 0.00%
Total violations (SLA): 0.00%
Average violations (SLA): 0.00%
Host Shutdown Count: 29
Time before shutdown (Mean): 300.10 sec
Time before shutdown (StDev): 0.00 sec
VM Migration Delay (Mean): NaN sec
VM Migration Delay (StDev): NaN sec
```

FIG. 4.2. *Results obtained for random DVFS*

```
Name of Experiment: random_iqr_mc_1.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 46.86 kWh
Migration counts (VM): 5085
Service Level Agreement (SLA): 0.02113%
SLA (Performance Degradation): 0.26%
SLA (Per host Elapsed Time): 8.14%
Total violations (SLA): 1.13%
Average violations (SLA): 10.81%
Host Shutdown Count: 1517
Time before shutdown (Mean): 1002.30 sec
Time before shutdown (StDev): 1214.40 sec
VM Migration Delay (Mean): 20.33 sec
VM Migration Delay (StDev): 7.93 sec
VM Selection (Mean of execution delay): 0.00663 sec
VM Selection (StDev of execution delay): 0.09327 sec
Selection of Host (Mean of execution delay): 0.00102 sec
Selection of Host (StDev of execution delay): 0.00079 sec
VM Reallocation (Mean of execution delay): 0.00317 sec
VM Reallocation (StDev of execution delay): 0.00494 sec
Total Execution Delay (Mean): 0.01952 sec
Total Execution Delay (StDev): 0.09417 sec
```

FIG. 4.3. *Results obtained for Inter Quartile Range (IQR)*

```
Name of Experiment: random_iqr_mmt_1.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 47.85 kWh
Migration counts (VM): 5502
Service Level Agreement (SLA): 0.01770%
SLA (Performance Degradation): 0.23%
SLA (Per host Elapsed Time): 7.82%
Total violations (SLA): 1.05%
Average violations (SLA): 10.44%
Host Shutdown Count: 1549
Time before shutdown (Mean): 1004.52 sec
Time before shutdown (StDev): 1178.23 sec
VM Migration Delay (Mean): 17.62 sec
VM Migration Delay (StDev): 7.89 sec
VM Selection (Mean of execution delay): 0.00017 sec
VM Selection (StDev of execution delay): 0.00044 sec
Selection of Host (Mean of execution delay): 0.00100 sec
Selection of Host (StDev of execution delay): 0.00144 sec
VM Reallocation (Mean of execution delay): 0.00393 sec
VM Reallocation (StDev of execution delay): 0.01149 sec
Total Execution Delay (Mean): 0.01308 sec
Total Execution Delay (StDev): 0.02002 sec
```

FIG. 4.4. *Results obtained for Inter Quartile Range (IQR) with Minimum Migration Time (MMT) method*

TABLE 4.1
List of performance parameters for the results evaluation

Parameter Name	Units
Host Count	Count (default 50)
VM Count	Count (default 50)
Simulation Length (Total)	Seconds (default 86400 seconds)
Consumed Energy Levels	kWh (kilo Watt per hour)
Migration counts (VM)	Count
Service Level Agreement (SLA)	Percentage
SLA (Performance Degradation)	Percentage
SLA (Per host Elapsed Time)	Percentage
Total violations (SLA)	Percentage
Average violations (SLA)	Percentage
Host Shutdown Count	Counts
Time before shutdown (Mean)	Seconds
Time before shutdown (StDev)	Seconds
VM Migration Delay (Mean)	Seconds
VM Migration Delay (StDev)	Seconds
VM Selection (Mean of execution delay)	Seconds
VM Selection (StDev of execution delay)	Seconds
Selection of Host (Mean of execution delay)	Seconds
Selection of Host (StDev of execution delay)	Seconds
VM Reallocation (Mean of execution delay)	Seconds
VM Reallocation (StDev of execution delay)	Seconds
Total Execution Delay (Mean)	Seconds
Total Execution Delay (StDev)	Seconds

```
Name of Experiment: random_iqr_mu_1.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 49.32 kWh
Migration counts (VM): 5789
Service Level Agreement (SLA): 0.02148%
SLA (Performance Degradation): 0.26%
SLA (Per host Elapsed Time): 8.24%
Total violations (SLA): 0.98%
Average violations (SLA): 10.71%
Host Shutdown Count: 1622
Time before shutdown (Mean): 997.96 sec
Time before shutdown (StDev): 1119.87 sec
VM Migration Delay (Mean): 20.38 sec
VM Migration Delay (StDev): 8.02 sec
VM Selection (Mean of execution delay): 0.00021 sec
VM Selection (StDev of execution delay): 0.00049 sec
Selection of Host (Mean of execution delay): 0.00094 sec
Selection of Host (StDev of execution delay): 0.00053 sec
VM Reallocation (Mean of execution delay): 0.00428 sec
VM Reallocation (StDev of execution delay): 0.00420 sec
Total Execution Delay (Mean): 0.01346 sec
Total Execution Delay (StDev): 0.00926 sec
```

FIG. 4.5. *Results obtained for Inter Quartile Range (IQR) with Maximum Utilization (MU) method*

In this sub-section, the VM allocation model of Local Regression (LR) has been used along with the Maximum Utilization (MU) method. Fig. 4.9 shows the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Local Regression (LR) has been used along with the Random Selection (RS) method. Fig. 4.10 shows the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Local Regression Robust (LRR) has been used along with the Maximum Correlation (MC) method. The results represented in Fig. 4.11 is obtained from this model for all of the enlisted parameters.

```
Name of Experiment: random_iqr_rs_1.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 47.43 kWh
Migration counts (VM): 5032
Service Level Agreement (SLA): 0.02059%
SLA (Performance Degradation): 0.25%
SLA (Per host Elapsed Time): 8.32%
Total violations (SLA): 1.05%
Average violations (SLA): 10.42%
Host Shutdown Count: 1526
Time before shutdown (Mean): 1009.40 sec
Time before shutdown (StDev): 1191.37 sec
VM Migration Delay (Mean): 20.29 sec
VM Migration Delay (StDev): 7.95 sec
VM Selection (Mean of execution delay): 0.00019 sec
VM Selection (StDev of execution delay): 0.00049 sec
Selection of Host (Mean of execution delay): 0.00098 sec
Selection of Host (StDev of execution delay): 0.00060 sec
VM Reallocation (Mean of execution delay): 0.00277 sec
VM Reallocation (StDev of execution delay): 0.00271 sec
Total Execution Delay (Mean): 0.01110 sec
Total Execution Delay (StDev): 0.01006 sec
```

FIG. 4.6. *Results obtained for Inter Quartile Range (IQR) with Random Selection (RS) method*

```
Name of Experiment: random_lr_mc_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 34.35 kWh
Migration counts (VM): 2203
Service Level Agreement (SLA): 0.02124%
SLA (Performance Degradation): 0.14%
SLA (Per host Elapsed Time): 15.63%
Total violations (SLA): 3.17%
Average violations (SLA): 12.45%
Host Shutdown Count: 685
Time before shutdown (Mean): 1484.67 sec
Time before shutdown (StDev): 2719.41 sec
VM Migration Delay (Mean): 20.35 sec
VM Migration Delay (StDev): 7.95 sec
VM Selection (Mean of execution delay): 0.00266 sec
VM Selection (StDev of execution delay): 0.02902 sec
Selection of Host (Mean of execution delay): 0.00081 sec
Selection of Host (StDev of execution delay): 0.00197 sec
VM Reallocation (Mean of execution delay): 0.00133 sec
VM Reallocation (StDev of execution delay): 0.00235 sec
Total Execution Delay (Mean): 0.01283 sec
Total Execution Delay (StDev): 0.03109 sec
```

FIG. 4.7. *Results obtained for Local Regression (LR) with Maximum Correlation (MC) method*

In this sub-section, the VM allocation model of Local Regression Robust (LRR) has been used along with the Minimum Migration Time (MMT) method. Fig. 4.12 shows the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Local Regression Robust (LRR) has been used along with the Maximum Utilization (MU) method. The results obtained from this model for all of the enlisted parameters is shown in Fig. 4.13.

In this sub-section, the VM allocation model of Local Regression Robust (LRR) has been used along with the Random Selection (RS) method. The results obtained from this model for all of the enlisted parameters are represented in Fig. 4.14.

In this sub-section, the VM allocation model of Median Absolute Deviation (MAD) has been used along with the Maximum Correlation (MC) method. Fig. 4.15 represents the results obtained from this model for all of the enlisted parameters.

```
Name of Experiment: random_lr_mmt_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 35.37 kWh
Migration counts (VM): 2872
Service Level Agreement (SLA): 0.01912%
SLA (Performance Degradation): 0.13%
SLA (Per host Elapsed Time): 14.31%
Total violations (SLA): 3.16%
Average violations (SLA): 12.89%
Host Shutdown Count: 806
Time before shutdown (Mean): 1330.63 sec
Time before shutdown (StDev): 2212.70 sec
VM Migration Delay (Mean): 16.60 sec
VM Migration Delay (StDev): 7.70 sec
VM Selection (Mean of execution delay): 0.00013 sec
VM Selection (StDev of execution delay): 0.00039 sec
Selection of Host (Mean of execution delay): 0.00087 sec
Selection of Host (StDev of execution delay): 0.00355 sec
VM Reallocation (Mean of execution delay): 0.00133 sec
VM Reallocation (StDev of execution delay): 0.00208 sec
Total Execution Delay (Mean): 0.00943 sec
Total Execution Delay (StDev): 0.00991 sec
```

FIG. 4.8. *Results obtained for Local Regression (LR) with Minimum Migration Time (MMT) method*

```
Name of Experiment: random_lr_mu_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 35.38 kWh
Migration counts (VM): 2808
Service Level Agreement (SLA): 0.02047%
SLA (Performance Degradation): 0.13%
SLA (Per host Elapsed Time): 15.21%
Total violations (SLA): 3.39%
Average violations (SLA): 13.13%
Host Shutdown Count: 816
Time before shutdown (Mean): 1293.22 sec
Time before shutdown (StDev): 2183.88 sec
VM Migration Delay (Mean): 20.06 sec
VM Migration Delay (StDev): 8.11 sec
VM Selection (Mean of execution delay): 0.00018 sec
VM Selection (StDev of execution delay): 0.00078 sec
Selection of Host (Mean of execution delay): 0.00105 sec
Selection of Host (StDev of execution delay): 0.00523 sec
VM Reallocation (Mean of execution delay): 0.00155 sec
VM Reallocation (StDev of execution delay): 0.00324 sec
Total Execution Delay (Mean): 0.01002 sec
Total Execution Delay (StDev): 0.01019 sec
```

FIG. 4.9. *Results obtained for Local Regression (LR) with Maximum Utilization (MU) method*

In this sub-section, the VM allocation model of Median Absolute Deviation (MAD) has been used along with the Minimum Migration Time (MMT) method. The results, shown in Fig. 4.16, are obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Median Absolute Deviation (MAD) has been used along with the Maximum Utilization (MU) method. Fig. 4.17 represents the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Median Absolute Deviation (MAD) has been used along with the Random Selection (RS) method. The results obtained from this model for all of the enlisted parameters are shown in Fig. 4.18.

In this sub-section, the VM allocation model of Non-Power Aware has been used with no method for VM selection. The VM selection policy is simple random method like DVFS, which is unlike the random selection (RS) method for other VM allocation policies. Fig. 4.19 shows the results obtained from this model for all of

```
Name of Experiment: random_lr_rs_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 34.33 kWh
Migration counts (VM): 2338
Service Level Agreement (SLA): 0.02269%
SLA (Performance Degradation): 0.14%
SLA (Per host Elapsed Time): 16.17%
Total violations (SLA): 3.16%
Average violations (SLA): 12.78%
Host Shutdown Count: 692
Time before shutdown (Mean): 1459.61 sec
Time before shutdown (StDev): 2639.05 sec
VM Migration Delay (Mean): 20.37 sec
VM Migration Delay (StDev): 7.94 sec
VM Selection (Mean of execution delay): 0.00008 sec
VM Selection (StDev of execution delay): 0.00049 sec
Selection of Host (Mean of execution delay): 0.00088 sec
Selection of Host (StDev of execution delay): 0.00375 sec
VM Reallocation (Mean of execution delay): 0.00111 sec
VM Reallocation (StDev of execution delay): 0.00256 sec
Total Execution Delay (Mean): 0.01036 sec
Total Execution Delay (StDev): 0.01202 sec
```

FIG. 4.10. *Results obtained for Local Regression (LR) with Random Selection (RS) method*

```
Name of Experiment: random_lrr_mc_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 34.35 kWh
Migration counts (VM): 2203
Service Level Agreement (SLA): 0.02124%
SLA (Performance Degradation): 0.14%
SLA (Per host Elapsed Time): 15.63%
Total violations (SLA): 3.17%
Average violations (SLA): 12.45%
Host Shutdown Count: 685
Time before shutdown (Mean): 1484.67 sec
Time before shutdown (StDev): 2719.41 sec
VM Migration Delay (Mean): 20.35 sec
VM Migration Delay (StDev): 7.95 sec
VM Selection (Mean of execution delay): 0.00137 sec
VM Selection (StDev of execution delay): 0.00630 sec
Selection of Host (Mean of execution delay): 0.00132 sec
Selection of Host (StDev of execution delay): 0.00720 sec
VM Reallocation (Mean of execution delay): 0.00139 sec
VM Reallocation (StDev of execution delay): 0.00254 sec
Total Execution Delay (Mean): 0.01081 sec
Total Execution Delay (StDev): 0.01231 sec
```

FIG. 4.11. *Results obtained for Local Regression Robust (LRR) with Maximum Correlation (MC) method*

the enlisted parameters.

In this sub-section, the VM allocation model of Static Threshold (THR) has been used along with the Maximum Correlation (MC) method. Fig. 4.20 shows the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Static Threshold (THR) has been used along with the Minimum Migration Time (MMT) method. The results obtained from this model for all of the enlisted parameters are shown in Fig. 4.21.

In this sub-section, the VM allocation model of Static Threshold (THR) has been used along with the Maximum Utilization (MU) method. Fig. 4.22 shows the results obtained from this model for all of the enlisted parameters.

In this sub-section, the VM allocation model of Static Threshold (THR) has been used along with the Random Selection (RS) method. Fig. 4.23 represents the results obtained from this model for all of the enlisted

```
Name of Experiment: random_lrr_mmt_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 35.37 kWh
Migration counts (VM): 2872
Service Level Agreement (SLA): 0.01912%
SLA (Performance Degradation): 0.13%
SLA (Per host Elapsed Time): 14.31%
Total violations (SLA): 3.16%
Average violations (SLA): 12.89%
Host Shutdown Count: 806
Time before shutdown (Mean): 1330.63 sec
Time before shutdown (StDev): 2212.70 sec
VM Migration Delay (Mean): 16.60 sec
VM Migration Delay (StDev): 7.70 sec
VM Selection (Mean of execution delay): 0.00024 sec
VM Selection (StDev of execution delay): 0.00088 sec
Selection of Host (Mean of execution delay): 0.00112 sec
Selection of Host (StDev of execution delay): 0.00541 sec
VM Reallocation (Mean of execution delay): 0.00205 sec
VM Reallocation (StDev of execution delay): 0.00331 sec
Total Execution Delay (Mean): 0.01088 sec
Total Execution Delay (StDev): 0.01140 sec
```

Fig. 4.12. *Results obtained for Local Regression Robust (LRR) with Minimum Migration Time (MMT) method*

```
Name of Experiment: random_lrr_mu_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 35.38 kWh
Migration counts (VM): 2808
Service Level Agreement (SLA): 0.02047%
SLA (Performance Degradation): 0.13%
SLA (Per host Elapsed Time): 15.21%
Total violations (SLA): 3.39%
Average violations (SLA): 13.13%
Host Shutdown Count: 816
Time before shutdown (Mean): 1293.22 sec
Time before shutdown (StDev): 2183.88 sec
VM Migration Delay (Mean): 20.06 sec
VM Migration Delay (StDev): 8.11 sec
VM Selection (Mean of execution delay): 0.00022 sec
VM Selection (StDev of execution delay): 0.00087 sec
Selection of Host (Mean of execution delay): 0.00099 sec
Selection of Host (StDev of execution delay): 0.00556 sec
VM Reallocation (Mean of execution delay): 0.00220 sec
VM Reallocation (StDev of execution delay): 0.00332 sec
Total Execution Delay (Mean): 0.01037 sec
Total Execution Delay (StDev): 0.00992 sec
```

Fig. 4.13. *Results obtained for Local Regression Robust (LR) with Maximum Utilization (MU) method*

parameters.

Table 4.2 shows the summary of the results for each experiment. This table represents the experiment name and the result obtained by that particular experiment with respect to each parameter. This summary will help us to evaluate and analyze the conducted experiments in much easier way.

All the experiments were conducted keeping the host count and VM count fixed (as 50) so as to compute the results on the same platform. This helps us in comparison with the different algorithms. From this, it is seen that experiment name: random_npa consumes the highest energy levels than all the experiments.

```
Name of Experiment: random_lrr_rs_1.2
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 34.30 kWh
Migration counts (VM): 2196
Service Level Agreement (SLA): 0.02350%
SLA (Performance Degradation): 0.14%
SLA (Per host Elapsed Time): 16.35%
Total violations (SLA): 3.60%
Average violations (SLA): 13.29%
Host Shutdown Count: 701
Time before shutdown (Mean): 1451.49 sec
Time before shutdown (StDev): 2789.53 sec
VM Migration Delay (Mean): 20.52 sec
VM Migration Delay (StDev): 7.93 sec
VM Selection (Mean of execution delay): 0.00008 sec
VM Selection (StDev of execution delay): 0.00053 sec
Selection of Host (Mean of execution delay): 0.00099 sec
Selection of Host (StDev of execution delay): 0.00491 sec
VM Reallocation (Mean of execution delay): 0.00133 sec
VM Reallocation (StDev of execution delay): 0.00281 sec
Total Execution Delay (Mean): 0.00981 sec
Total Execution Delay (StDev): 0.01107 sec
```

FIG. 4.14. *Results obtained for Local Regression Robust (LRR) with Random Selection (RS) method*

```
Name of Experiment: random_mad_mc_2.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 44.99 kWh
Migration counts (VM): 4778
Service Level Agreement (SLA): 0.02504%
SLA (Performance Degradation): 0.26%
SLA (Per host Elapsed Time): 9.81%
Total violations (SLA): 1.53%
Average violations (SLA): 10.96%
Host Shutdown Count: 1468
Time before shutdown (Mean): 980.23 sec
Time before shutdown (StDev): 1213.20 sec
VM Migration Delay (Mean): 20.35 sec
VM Migration Delay (StDev): 7.95 sec
VM Selection (Mean of execution delay): 0.00202 sec
VM Selection (StDev of execution delay): 0.00782 sec
Selection of Host (Mean of execution delay): 0.00117 sec
Selection of Host (StDev of execution delay): 0.00247 sec
VM Reallocation (Mean of execution delay): 0.00323 sec
VM Reallocation (StDev of execution delay): 0.00397 sec
Total Execution Delay (Mean): 0.01353 sec
Total Execution Delay (StDev): 0.01212 sec
```

FIG. 4.15. *Results obtained for Median Absolute Deviation (MAD) with Maximum Correlation (MC) method*

```
Name of Experiment: random_mad_mmt_2.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 45.61 kWh
Migration counts (VM): 5265
Service Level Agreement (SLA): 0.01967%
SLA (Performance Degradation): 0.23%
SLA (Per host Elapsed Time): 8.61%
Total violations (SLA): 1.31%
Average violations (SLA): 10.91%
Host Shutdown Count: 1528
Time before shutdown (Mean): 965.45 sec
Time before shutdown (StDev): 1253.17 sec
VM Migration Delay (Mean): 17.17 sec
VM Migration Delay (StDev): 7.77 sec
VM Selection (Mean of execution delay): 0.00020 sec
VM Selection (StDev of execution delay): 0.00081 sec
Selection of Host (Mean of execution delay): 0.00144 sec
Selection of Host (StDev of execution delay): 0.00498 sec
VM Reallocation (Mean of execution delay): 0.00378 sec
VM Reallocation (StDev of execution delay): 0.00360 sec
Total Execution Delay (Mean): 0.01324 sec
Total Execution Delay (StDev): 0.00997 sec
```

FIG. 4.16. *Results obtained for Median Absolute Deviation (MAD) with Minimum Migration Time (MMT) method*

```
Name of Experiment: random_mad_mu_2.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 47.36 kWh
Migration counts (VM): 5628
Service Level Agreement (SLA): 0.02529%
SLA (Performance Degradation): 0.26%
SLA (Per host Elapsed Time): 9.73%
Total violations (SLA): 1.53%
Average violations (SLA): 11.11%
Host Shutdown Count: 1632
Time before shutdown (Mean): 944.32 sec
Time before shutdown (StDev): 1137.05 sec
VM Migration Delay (Mean): 20.18 sec
VM Migration Delay (StDev): 8.03 sec
VM Selection (Mean of execution delay): 0.00025 sec
VM Selection (StDev of execution delay): 0.00090 sec
Selection of Host (Mean of execution delay): 0.00117 sec
Selection of Host (StDev of execution delay): 0.00519 sec
VM Reallocation (Mean of execution delay): 0.00471 sec
VM Reallocation (StDev of execution delay): 0.00437 sec
Total Execution Delay (Mean): 0.01504 sec
Total Execution Delay (StDev): 0.01110 sec
```

FIG. 4.17. *Results obtained for Median Absolute Deviation (MAD) with Maximum Utilization (MU) method*

```
Name of Experiment: random_mad_rs_2.5
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 44.95 kWh
Migration counts (VM): 4882
Service Level Agreement (SLA): 0.02485%
SLA (Performance Degradation): 0.26%
SLA (Per host Elapsed Time): 9.66%
Total violations (SLA): 1.69%
Average violations (SLA): 11.16%
Host Shutdown Count: 1489
Time before shutdown (Mean): 970.18 sec
Time before shutdown (StDev): 1185.94 sec
VM Migration Delay (Mean): 20.29 sec
VM Migration Delay (StDev): 7.98 sec
VM Selection (Mean of execution delay): 0.00028 sec
VM Selection (StDev of execution delay): 0.00097 sec
Selection of Host (Mean of execution delay): 0.00127 sec
Selection of Host (StDev of execution delay): 0.00538 sec
VM Reallocation (Mean of execution delay): 0.00348 sec
VM Reallocation (StDev of execution delay): 0.00418 sec
Total Execution Delay (Mean): 0.01263 sec
Total Execution Delay (StDev): 0.01028 sec
```

FIG. 4.18. *Results obtained for Median Absolute Deviation (MAD) with Random Selection (RS) method*

```
Name of Experiment: random_npa
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 150.68 kWh
Migration counts (VM): 0
Service Level Agreement (SLA): 0.00000%
SLA (Performance Degradation): 0.00%
SLA (Per host Elapsed Time): 0.00%
Total violations (SLA): 0.00%
Average violations (SLA): 0.00%
Host Shutdown Count: 29
Time before shutdown (Mean): 300.10 sec
Time before shutdown (StDev): 0.00 sec
VM Migration Delay (Mean): NaN sec
VM Migration Delay (StDev): NaN sec
```

FIG. 4.19. *Results obtained for Non-Power Aware (NPA)*

```
Name of Experiment: random_thr_mc_0.8
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 40.85 kWh
Migration counts (VM): 4392
Service Level Agreement (SLA): 0.03726%
SLA (Performance Degradation): 0.27%
SLA (Per host Elapsed Time): 13.79%
Total violations (SLA): 3.09%
Average violations (SLA): 12.93%
Host Shutdown Count: 1389
Time before shutdown (Mean): 924.72 sec
Time before shutdown (StDev): 1363.51 sec
VM Migration Delay (Mean): 20.47 sec
VM Migration Delay (StDev): 7.94 sec
VM Selection (Mean of execution delay): 0.00152 sec
VM Selection (StDev of execution delay): 0.00632 sec
Selection of Host (Mean of execution delay): 0.00047 sec
Selection of Host (StDev of execution delay): 0.00119 sec
VM Reallocation (Mean of execution delay): 0.00201 sec
VM Reallocation (StDev of execution delay): 0.00370 sec
Total Execution Delay (Mean): 0.00868 sec
Total Execution Delay (StDev): 0.01095 sec
```

FIG. 4.20. *Results obtained for Static Threshold (THR) with Maximum Correlation (MC) method*

```
Name of Experiment: random_thr_mmt_0.8
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 41.81 kWh
Migration counts (VM): 4839
Service Level Agreement (SLA): 0.03048%
SLA (Performance Degradation): 0.23%
SLA (Per host Elapsed Time): 12.99%
Total violations (SLA): 3.25%
Average violations (SLA): 12.81%
Host Shutdown Count: 1424
Time before shutdown (Mean): 929.70 sec
Time before shutdown (StDev): 1348.87 sec
VM Migration Delay (Mean): 16.82 sec
VM Migration Delay (StDev): 7.67 sec
VM Selection (Mean of execution delay): 0.00011 sec
VM Selection (StDev of execution delay): 0.00060 sec
Selection of Host (Mean of execution delay): 0.00038 sec
Selection of Host (StDev of execution delay): 0.00110 sec
VM Reallocation (Mean of execution delay): 0.00249 sec
VM Reallocation (StDev of execution delay): 0.00502 sec
Total Execution Delay (Mean): 0.00839 sec
Total Execution Delay (StDev): 0.00835 sec
```

FIG. 4.21. *Results obtained for Static Threshold (THR) with Minimum Migration Time (MMT) method*

```
Name of Experiment: random_thr_mu_0.8
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 44.08 kWh
Migration counts (VM): 5404
Service Level Agreement (SLA): 0.03546%
SLA (Performance Degradation): 0.28%
SLA (Per host Elapsed Time): 12.69%
Total violations (SLA): 2.73%
Average violations (SLA): 12.73%
Host Shutdown Count: 1578
Time before shutdown (Mean): 900.54 sec
Time before shutdown (StDev): 1253.98 sec
VM Migration Delay (Mean): 20.23 sec
VM Migration Delay (StDev): 8.09 sec
VM Selection (Mean of execution delay): 0.00017 sec
VM Selection (StDev of execution delay): 0.00075 sec
Selection of Host (Mean of execution delay): 0.00033 sec
Selection of Host (StDev of execution delay): 0.00103 sec
VM Reallocation (Mean of execution delay): 0.00262 sec
VM Reallocation (StDev of execution delay): 0.00388 sec
Total Execution Delay (Mean): 0.00886 sec
Total Execution Delay (StDev): 0.00900 sec
```

FIG. 4.22. *Results obtained for Static Threshold (THR) with Maximum Utilization (MU) method*

```
Name of Experiment: random_thr_rs_0.8
Host count: 50
VM count: 50
Simulation Length (total): 86400.00 sec
Consumed Energy Levels: 41.12 kWh
Migration counts (VM): 4442
Service Level Agreement (SLA): 0.03592%
SLA (Performance Degradation): 0.27%
SLA (Per host Elapsed Time): 13.16%
Total violations (SLA): 3.03%
Average violations (SLA): 13.18%
Host Shutdown Count: 1391
Time before shutdown (Mean): 934.82 sec
Time before shutdown (StDev): 1404.86 sec
VM Migration Delay (Mean): 20.52 sec
VM Migration Delay (StDev): 7.96 sec
VM Selection (Mean of execution delay): 0.00007 sec
VM Selection (StDev of execution delay): 0.00044 sec
Selection of Host (Mean of execution delay): 0.00045 sec
Selection of Host (StDev of execution delay): 0.00106 sec
VM Reallocation (Mean of execution delay): 0.00251 sec
VM Reallocation (StDev of execution delay): 0.00535 sec
Total Execution Delay (Mean): 0.00877 sec
Total Execution Delay (StDev): 0.01113 sec
```

FIG. 4.23. *Results obtained for Static Threshold (THR) with Random Selection (RS) method*

TABLE 4.2
Result Summary for Each Experiment

Experiment Name/ Parameter	Host count	VM count	SimulationLength	Consumed Energy Levels	Migration counts	Service Level Agreement	Performance SLA	Per Host Elapsed Time SLA	Total violations	Average violations	Host shutdown count	Time before shutdown_Mean	Time before shutdown_StDev	Mean VM Migration Delay	StDev VM Migration Delay	Mean VM Selection	StDev VM Selection	Mean Host Selection	StDev Host Selection	VM Reallocation Mean	VM Reallocation StDev	Total Execution Delay Mean	StDev Total Execution Delay
random_dvfs	50	50	86400	52.98	0	0	0	0	0	0	29	300.1	0	NaN	NaN								
random_iqr_mc_1.5	50	50	86400	46.86	5085	0.02113	0.26	8.14	1.13	10.81	1517	1002.3	1214.4	20.33	7.93	0.00663	0.09327	0.00102	0.00079	0.00317	0.00494	0.01952	0.09417
random_iqr_mmt _1.5	50	50	86400	47.85	5502	0.0177	0.23	7.82	1.05	10.44	1549	1004.52	1178.23	17.62	7.89	0.00017	0.00044	0.001	0.00144	0.00393	0.01149	0.01308	0.02002
random_iqr_mu_1.5	50	50	86400	49.32	5789	0.02148	0.26	8.24	0.98	10.71	1622	997.96	1119.87	20.38	8.02	0.00021	0.00049	0.00094	0.00053	0.00428	0.0042	0.01346	0.00926
random_iqr_rs_1.5	50	50	86400	47.43	5032	0.02059	0.25	8.32	1.05	10.42	1526	1009.4	1191.37	20.29	7.95	0.00019	0.00049	0.00098	0.0006	0.00277	0.00271	0.0111	0.01006
random_lr_mc_1.2	50	50	86400	34.35	2203	0.02124	0.14	15.63	3.17	12.45	685	1484.67	2719.41	20.35	7.95	0.00266	0.02902	0.00081	0.00197	0.00133	0.00235	0.01283	0.03109
random_lr_mmt_1.2	50	50	86400	35.37	2872	0.01912	0.13	14.31	3.16	12.89	806	1330.63	2212.7	16.6	7.7	0.00013	0.00039	0.00087	0.00355	0.00133	0.00208	0.00943	0.00991
random_lr_mu_1.2	50	50	86400	35.38	2808	0.02047	0.13	15.21	3.39	13.13	816	1293.22	2183.88	20.06	8.11	0.00018	0.00078	0.00105	0.00523	0.00155	0.00324	0.01002	0.01019
random_lr_rs_1.2	50	50	86400	34.33	2338	0.02269	0.14	16.17	3.16	12.78	692	1459.61	2639.05	20.37	7.94	0.00008	0.00049	0.00088	0.00375	0.00111	0.00256	0.01036	0.01202
random_lrr_mc_1.2	50	50	86400	34.35	2203	0.02124	0.14	15.63	3.17	12.45	685	1484.67	2719.41	20.35	7.95	0.00137	0.0063	0.00132	0.0072	0.00139	0.00254	0.01081	0.01231
random_lrr_mmt_1.2	50	50	86400	35.37	2872	0.01912	0.13	14.31	3.16	12.89	806	1330.63	2212.7	16.6	7.7	0.00024	0.00088	0.00112	0.00541	0.00205	0.00331	0.01088	0.0114
random_lrr_mu_1.2	50	50	86400	35.38	2808	0.02047	0.13	15.21	3.39	13.13	816	1293.22	2183.88	20.06	8.11	0.00022	0.00087	0.00099	0.00556	0.0022	0.00332	0.01037	0.00992
random_lrr_rs_1.2	50	50	86400	34.3	2196	0.0235	0.14	16.35	3.6	13.29	701	1451.49	2789.53	20.52	7.93	0.00008	0.00053	0.00099	0.00491	0.00133	0.00281	0.00981	0.01107
random_mad_mc_2.5	50	50	86400	44.99	4778	0.02504	0.26	9.81	1.53	10.96	1468	980.23	1213.2	20.35	7.95	0.00202	0.00782	0.00117	0.00247	0.00323	0.00397	0.01353	0.01212
random_mad_mmt_2.5	50	50	86400	45.61	5265	0.01967	0.23	8.61	1.31	10.91	1528	965.45	1253.17	17.17	7.77	0.0002	0.00081	0.00144	0.00498	0.00378	0.0036	0.01324	0.00997
random_mad_mu_2.5	50	50	86400	47.36	5628	0.02529	0.26	9.73	1.53	11.11	1632	944.32	1137.05	20.18	8.03	0.00025	0.0009	0.00117	0.00519	0.00471	0.00437	0.01504	0.0111
random_mad_rs_2.5	50	50	86400	44.95	4882	0.02485	0.26	9.66	1.69	11.16	1489	970.18	1185.94	20.29	7.98	0.00028	0.00097	0.00127	0.00538	0.00348	0.00418	0.01263	0.01028
random_npa	50	50	86400	150.68	0	0	0	0	0	0	29	300.1	0	NaN	NaN								
random_thr_mc_0.8	50	50	86400	40.85	4392	0.03726	0.27	13.79	3.09	12.93	1389	924.72	1363.51	20.47	7.94	0.00152	0.00632	0.00047	0.00119	0.00201	0.0037	0.00868	0.01095
random_thr_mmt_0.8	50	50	86400	41.81	4839	0.03048	0.23	12.99	3.25	12.81	1424	929.7	1348.87	16.82	7.67	0.00011	0.0006	0.00038	0.0011	0.00249	0.00502	0.00839	0.00835
random_thr_mu_0.8	50	50	86400	44.08	5404	0.03546	0.28	12.69	2.73	12.73	1578	900.54	1253.98	20.23	8.09	0.00017	0.00075	0.00033	0.00103	0.00262	0.00388	0.00886	0.009
random_thr_rs_0.8	50	50	86400	41.12	4442	0.03592	0.27	13.16	3.03	13.18	1391	934.82	1404.86	20.52	7.96	0.00007	0.00044	0.00045	0.00106	0.00251	0.00535	0.00877	0.01113

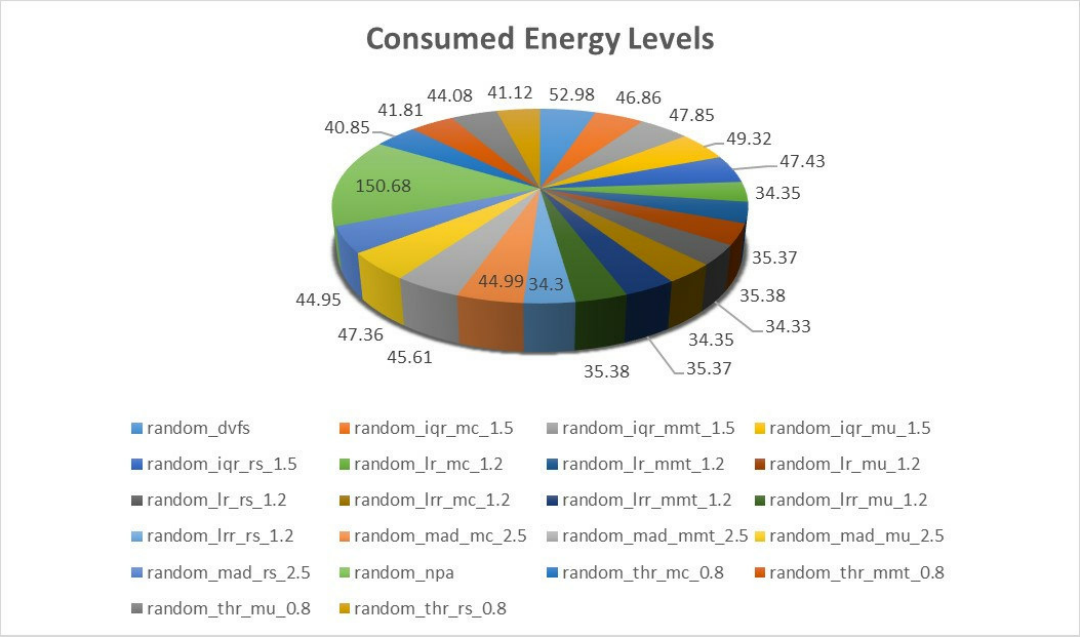

FIG. 4.24. *Consumed Energy Level per Experiment*

Fig. 4.24 shows consumed energy levels along with their experiment names.

From the Table 4.2, migration counts can also be computed and it is seen that experiment name: random_iqr_mu_1.5 involves maximum number of migration counts. Fig. 4.25 shows the migration counts for each experiment.

5. Conclusion and Future Directions. The fog computing resource allocation methods proposed in this paper combines the allocation and selection techniques altogether with optimal parameter stack to make scheduling decisions. This paper primarily focused to reduce the task load by implementing the rapid task processing, while also incorporating the sub-group oriented scheduling on available resources. This scheme is believed to improve the user contentment by improving the cost to operation length ratio, which eventually reduces the customer churn, and can effectively boost the operational revenue. The failure event tracking also plays a vital role in scheduling operations by avoiding the computing resources with high failure probability. The proposed model is learnt to reduce the queue size by effectively allocating the resources, which resulted in the form of quicker completion of user workflows. The prospective method results are evaluated against the state of the art scene with non-power aware based task scheduling mechanism. Out of the random VM allocation and selection policy, the DVFS (52.98 kWh) scheme outperforms NPA (150.68 kWh) model for the cloud task processing. Out of the particular VM allocation and selection models, which includes IQR, LR, LRR, MAD & THR. The results have obtained and analyzed using the energy, SLA infringement and workflow execution delay. The performance of the proposed schema has been analyzed in various experiments particularly designed to analyze various aspects for workflow processing on given fog resources. The LRR (35.85 kWh) model has been found most efficient on the basis of average energy consumption in comparison to the LR (34.86 kWh), THR (41.97 kWh), MAD (45.73 kWh) and IQR (47.87 kWh). The LRR model has been also observed as the leader when compared on the basis of number of VM migrations. The LRR (2520 VMs) has been observed as best contender on the basis of mean of number of VM migrations in comparison with LR (2555 VMs), THR (4769 VMs), MAD (5138 VMs) and IQR (5352 VMs).

In future, this work may not only confine to task allocation and task scheduling, but can be extended towards various load balancing algorithms that compute the load that gets generated on each VM. Moreover, this work of allocation and scheduling can be extended to the emerging technologies like bigdata to solve problems arising due to huge data in daily routine. This work may also be extended towards machine learning and deep learning

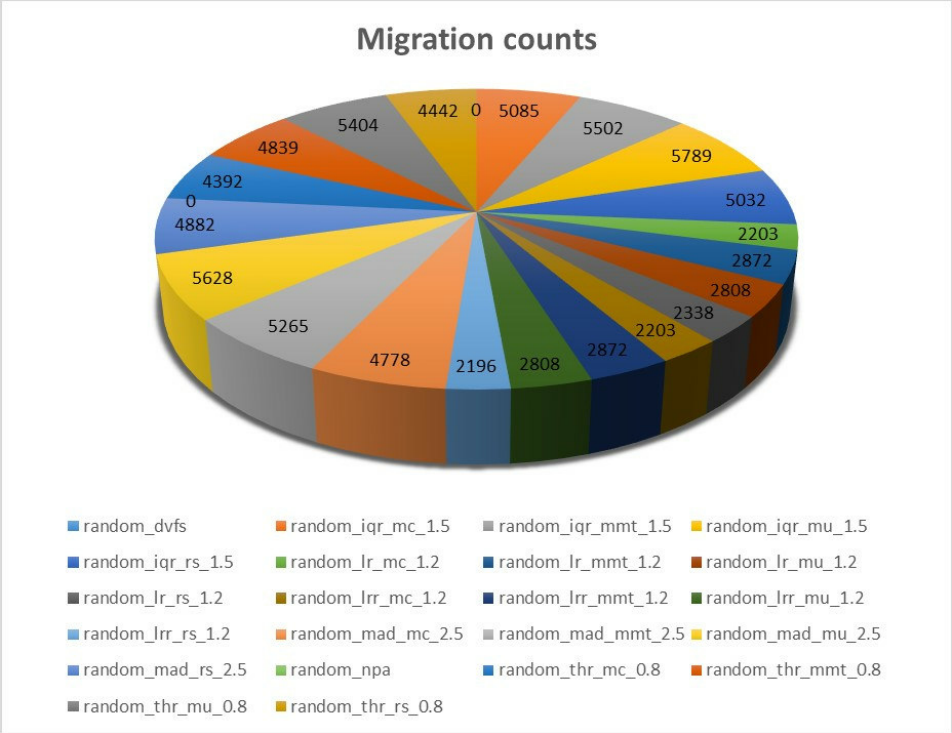

FIG. 4.25. *Migration counts per Experiment*

for pre-judgement of the upcoming difficulties and can set up a recovery/maintenance module accordingly.

REFERENCES

[1] W. A. JANSEN, T. GRANCE, ET AL., Guidelines on security and privacy in public cloud computing (2011).
[2] Basic concept and terminology of cloud computing (Jan. 2015).
[3] E. A. PANSOTRA, E. S. P. SINGH, Cloud security algorithms, International Journal of Security and Its Applications 9 (10) (2015) 353–360 (2015).
[4] A. T. VELTE, T. J. VELTE, R. C. ELSENPETER, R. C. ELSENPETER, Cloud computing: a practical approach, McGraw-Hill New York, 2010 (2010).
[5] S. P. SINGH, A. SHARMA, R. KUMAR, Analysis of load balancing algorithms using cloud analyst, International Journal of Grid and Distributed Computing 9 (9) (2016) 11–24 (2016).
[6] M. RAHMAN, S. IQBAL, J. GAO, Load balancer as a service in cloud computing, in: 2014 IEEE 8th International Symposium on Service Oriented System Engineering, IEEE, 2014, pp. 204–211 (2014).
[7] Gartner highlights five attributes of cloud computing (Dec. 2013).
[8] A. BALA, I. CHANA, Fault tolerance-challenges, techniques and implementation in cloud computing, International Journal of Computer Science Issues (IJCSI) 9 (1) (2012) 288–293 (2012).
[9] Cloud computing: A delicate balance of risk and benefit (Feb. 2015).
[10] PRERAKMODY, Cloud computing (Jan. 2015).
[11] L. M. VAQUERO, L. RODERO-MERINO, J. CACERES, M. LINDNER, A break in the clouds: towards a cloud definition, ACM SIGCOMM Computer Communication Review 39 (1) (2008) 50–55 (2008).
[12] S. PATEL, A. S. SINGH, Fault tolerance mechanisms and its implementation in cloud computing–a review, International Journal 3 (12) (2013).
[13] A. M. ALAKEEL, ET AL., A guide to dynamic load balancing in distributed computer systems, International Journal of Computer Science and Information Security 10 (6) (2010) 153–160 (2010).
[14] W. ZHAO, P. MELLIAR-SMITH, L. E. MOSER, Fault tolerance middleware for cloud computing, in: 2010 IEEE 3rd International Conference on Cloud Computing, IEEE, 2010, pp. 67–74 (2010).
[15] Q. LI, J. ZHAO, Y. GONG, Q. ZHANG, Energy-efficient computation offloading and resource allocation in fog computing for internet of everything, China Communications 16 (3) (2019) 32–41 (2019).
[16] S. P. SINGH, A. NAYYAR, R. KUMAR, A. SHARMA, Fog computing: from architecture to edge computing and big data

processing, The Journal of Supercomputing (2018) 1–36 (2018).
[17] M. Dorigo, Optimization, learning and natural algorithms, PhD Thesis, Politecnico di Milano (1992).
[18] M. Aazam, K. A. Harras, S. Zeadally, Fog computing for 5g tactile industrial internet of things: Qoe-aware resource allocation model, IEEE Transactions on Industrial Informatics (2019).
[19] Y. Jie, M. Li, C. Guo, L. Chen, Game-theoretic online resource allocation scheme on fog computing for mobile multimedia users, China Communications 16 (3) (2019) 22–31 (2019).
[20] X. Li, J. Wan, H.-N. Dai, M. Imran, M. Xia, A. Celesti, A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing, IEEE Transactions on Industrial Informatics (2019).
[21] S. Dam, G. Mandal, K. Dasgupta, P. Dutta, An ant colony based load balancing strategy in cloud computing, in: Advanced Computing, Networking and Informatics-Volume 2, Springer, 2014, pp. 403–413 (2014).
[22] A. Tasiopoulos, O. Ascigil, I. Psaras, S. Toumpis, G. Pavlou, Fogspot: Spot pricing for application provisioning in edge/fog computing, IEEE Transactions on Services Computing (2019).
[23] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J. P. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, Journal of Systems Architecture (2019).
[24] G. Yoon, D. Choi, J. Lee, H. Choi, Management of iot sensor data using a fog computing node, Journal of Sensors 2019 (2019).
[25] Z. Ning, J. Huang, X. Wang, Vehicular fog computing: Enabling real-time traffic management for smart cities, IEEE Wireless Communications 26 (1) (2019) 87–93 (2019).
[26] S. K. Goyal, M. Singh, Adaptive and dynamic load balancing in grid using ant colony optimization, International Journal of Engineering and Technology 4 (4) (2012) 167–174 (2012).
[27] B. Donassolo, I. Fajjari, A. Legrand, P. Mertikopoulos, Fog based framework for iot service provisioning, in: 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2019, pp. 1–6 (2019).
[28] S. K. Dhurandher, M. S. Obaidat, I. Woungang, P. Agarwal, A. Gupta, P. Gupta, A cluster-based load balancing algorithm in cloud computing, in: 2014 IEEE International Conference on Communications (ICC), IEEE, 2014, pp. 2921–2925 (2014).
[29] V. S. Kushwah, S. K. Goyal, P. Narwariya, A survey on various fault tolerant approaches for cloud environment during load balancing, Int J Comput Netw Wirel Mobile Commun 4 (6) (2014) 25–34 (2014).
[30] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, R. Rastogi, et al., Load balancing of nodes in cloud using ant colony optimization, in: 2012 UKSim 14th International Conference on Computer Modelling and Simulation, IEEE, 2012, pp. 3–8 (2012).
[31] R. Mishra, A. Jaiswal, Ant colony optimization: A solution of load balancing in cloud, International Journal of Web & Semantic Technology 3 (2) (2012) 33 (2012).
[32] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, S. Dam, A genetic algorithm (ga) based load balancing strategy for cloud computing, Procedia Technology 10 (2013) 340–347 (2013).
[33] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, K. Li, An energy-efficient task scheduling algorithm in dvfs-enabled cloud environment, Journal of Grid Computing 14 (1) (2016) 55–74 (2016).
[34] F. Yuan, S. E. OOI, L. Yuto, T. Yasuo, Time task scheduling for simple and proximate time model in cyber-physical systems, in: Computational Science and Technology, Springer, 2019, pp. 185–194 (2019).
[35] Q. Zhao, C. Xiong, C. Yu, C. Zhang, X. Zhao, A new energy-aware task scheduling method for data-intensive applications in the cloud, Journal of Network and Computer Applications 59 (2016) 14–27 (2016).
[36] N. Bansal, A. Maurya, T. Kumar, M. Singh, S. Bansal, Cost performance of qos driven task scheduling in cloud computing, Procedia Computer Science 57 (2015) 126–130 (2015).
[37] G. Patel, R. Mehta, U. Bhoi, Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing, Procedia Computer Science 57 (2015) 545–553 (2015).
[38] W. Lin, C. Liang, J. Z. Wang, R. Buyya, Bandwidth-aware divisible task scheduling for cloud computing, Software: Practice and Experience 44 (2) (2014) 163–174 (2014).
[39] G. Xu, J. Pang, X. Fu, A load balancing model based on cloud partitioning for the public cloud, Tsinghua Science and Technology 18 (1) (2013) 34–39 (2013).
[40] Z. Liu, X. Wang, A pso-based algorithm for load balancing in virtual machines of cloud computing environment, in: International conference in swarm intelligence, Springer, 2012, pp. 142–147 (2012).
[41] K. Li, G. Xu, G. Zhao, Y. Dong, D. Wang, Cloud task scheduling based on load balancing ant colony optimization, in: 2011 Sixth Annual ChinaGrid Conference, IEEE, 2011, pp. 3–9 (2011).
[42] H. Chang, X. Tang, A load-balance based resource-scheduling algorithm under cloud computing environment, in: International Conference on Web-Based Learning, Springer, 2010, pp. 85–90 (2010).
[43] V. Šešum-Čavić, E. Kühn, Self-organized load balancing through swarm intelligence, in: Next Generation Data Technologies for Collective Computational Intelligence, Springer, 2011, pp. 195–224 (2011).
[44] A. Jain, R. Singh, An innovative approach of ant colony optimization for load balancing in peer to peer grid environment, in: 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), IEEE, 2014, pp. 1–5 (2014).
[45] R. Chaukwale, S. S. Kamath, A modified ant colony optimization algorithm with load balancing for job shop scheduling, in: 2013 15th International Conference on Advanced Computing Technologies (ICACT), IEEE, 2013, pp. 1–5 (2013).
[46] S. Razzaq, A. Wahid, F. Khan, N. ul Amin, M. A. Shah, A. Akhunzada, I. Ali, Scheduling algorithms for high-performance computing: An application perspective of fog computing, in: Recent Trends and Advances in Wireless and IoT-enabled Networks, Springer, 2019, pp. 107–117 (2019).

[47] J. Wang, D. Li, Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing, Sensors 19 (5) (2019) 1023 (2019).
[48] J. Luo, L. Yin, J. Hu, C. Wang, X. Liu, X. Fan, H. Luo, Container-based fog computing architecture and energy-balancing scheduling algorithm for energy iot, Future Generation Computer Systems (2019).
[49] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, W. Dai, Energy efficient task allocation and energy scheduling in green energy powered edge computing, Future Generation Computer Systems 95 (2019) 89–99 (2019).
[50] C. Li, J. Tang, H. Tang, Y. Luo, Collaborative cache allocation and task scheduling for data-intensive applications in edge computing environment, Future Generation Computer Systems 95 (2019) 249–264 (2019).
[51] C. Li, J. Bai, J. Tang, Joint optimization of data placement and scheduling for improving user experience in edge computing, Journal of Parallel and Distributed Computing 125 (2019) 93–105 (2019).
[52] Y. Deng, Z. Chen, X. Yao, S. Hassan, J. Wu, Task scheduling for smart city applications based on multi-server mobile edge computing, IEEE Access 7 (2019) 14410–14421 (2019).
[53] S. Javaid, N. Javaid, T. Saba, Z. Wadud, A. Rehman, A. Haseeb, Intelligent resource allocation in residential buildings using consumer to fog to cloud based framework, Energies 12 (5) (2019) 815 (2019).
[54] L. Li, Q. Guan, L. Jin, M. Guo, Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system, IEEE Access 7 (2019) 9912–9925 (2019).
[55] R. Mahmud, R. Buyya, Modeling and simulation of fog and edge computing environments using ifogsim toolkit, Fog and Edge Computing: Principles and Paradigms (2019) 433–465 (2019).
[56] B. Singh, O. Gupta, S. P. Singh, Performance evaluation of dns based load balancing techniques for web servers (2011).
[57] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, M. Song, Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing, IEEE Transactions on Wireless Communications 18 (1) (2019) 695–708 (2019).
[58] P. T. I. M. C. COMPUTING, An efficient job sharing strategy for prioritized tasks in mobile cloud computing environment using acs-js algorithm, Journal of Theoretical and Applied Information Technology 97 (4) (2019).
[59] L. Yang, B. Liu, J. Cao, Y. Sahni, Z. Wang, Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds, IEEE Transactions on Services Computing (2019).
[60] S. Kim, Novel resource allocation algorithms for the social internet of things based fog computing paradigm, Wireless Communications and Mobile Computing 2019 (2019).
[61] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, J. Rodriguez, Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach, IEEE Transactions on Vehicular Technology (2019).
[62] B. Mondal, K. Dasgupta, P. Dutta, Load balancing in cloud computing using stochastic hill climbing-a soft computing approach, Procedia Technology 4 (2012) 783–789 (2012).
[63] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and experience 41 (1) (2011) 23–50 (2011).
[64] K. Braekers, R. F. Hartl, S. N. Parragh, F. Tricoire, A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience, European Journal of Operational Research 248 (2) (2016) 428–443 (2016).
[65] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future generation computer systems 28 (5) (2012) 755–768 (2012).
[66] S. Zaman, D. Grosu, A combinatorial auction-based mechanism for dynamic vm provisioning and allocation in clouds, IEEE Transactions on Cloud Computing 1 (2) (2013) 129–141 (2013).
[67] S. Zaman, D. Grosu, Combinatorial auction-based dynamic vm provisioning and allocation in clouds, in: 2011 IEEE Third International Conference on Cloud Computing Technology and Science, IEEE, 2011, pp. 107–114 (2011).
[68] Z. Cao, S. Dong, Dynamic vm consolidation for energy-aware and sla violation reduction in cloud computing, in: 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE, 2012, pp. 363–369 (2012).
[69] K. Kumar, J. Feng, Y. Nimmagadda, Y.-H. Lu, Resource allocation for real-time tasks using cloud computing, in: 2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN), IEEE, 2011, pp. 1–7 (2011).
[70] A. Beloglazov, R. Buyya, Energy efficient allocation of virtual machines in cloud data centers, in: 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, IEEE, 2010, pp. 577–578 (2010).
[71] A. Beloglazov, R. Buyya, Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints, IEEE Transactions on Parallel and Distributed Systems 24 (7) (2013) 1366–1379 (2013).

Edited by: Pijush Kanti Dutta Pramanik
Received: Mar 18, 2019
Accepted: Apr 2, 2019