Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.777871

HC-MOTS: HEURISTIC CLUSTERING AND MULTI-OBJECTIVE TASK SCHEDULING FOR CLOUD COMPUTING ENVIRONMENTS

Karnam Sreenu
Research Scholar, Department of Computer Science and Engineering, ANU College of Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India.

Sreelatha Malempati
Professor, Department of Computer Science and Engineering, RVR & JC College of Engineering, Guntur, Andhra Pradesh, India, * Corresponding author's Email: karnam.sreenu@gmail.com

Abstract: Load balancing and Task scheduling are the two major research challenges in cloud computing environments. Aiming at these challenges, this paper proposed a new mechanism called as Heuristic Clustering and Multi-Objective Task Scheduling (HC-MOTS). HC aims at load balancing over physical hosts which have totally dynamic resource availability. From the available hosts in cloud data center, HC picks up only optimal set of physical hosts by considering their dynamic characteristics like Posterior Probability, remaining CPU resource and remaining Memory resource. Further, the task scheduling is modeled as multi-objective fitness function formulated based on three objective functions with Execution time, Cost and Resource utilization. For each function, an individual weight is assigned and they are iteratively optimized to get an optimal solution. Simulation experiments have been conducted on datasets namely GOCJ and Synthetic Workload dataset and the performance is assessed through Makespan and Average Throughput. Further, the comparison proves the superiority of proposed approach over state-of-the-art methods.

Keywords: Cloud computing, Execution time, Load balancing, task scheduling, CPU, Memory.

I. Introduction

From the past several years, cloud computing is serving as a major source for several organizations. Cloud computing is initiated to provide services to users by accessing resources from several host entities. On the other hand, it can also be demonstrated as an interface which enables users to purchase the cloud services based on their requirements on demand. Cloud computing has a great flexibility to provide services from different aspects depending on the working platform and services requested by different types of users [1]. Cloud computing can be regarded as combination of parallel and distributed architectures with the capability of resources sharing such as hardware and software those can be consumed on 'pay-as-you-go' basis [2]. To utilize the resources, the user won't require purchasing either any software or hardware, they only must be enabled with an internet connection and pay for their usage.

Cloud computing has different services in software, platform, and infrastructure for users, and provides the services to users through internet [3]. Among these services, Infrastructure as a Service (IaaS) is a infrastructure oriented technology which can also be considered as a basis of Cloud computing. For the provision services to users, the cloud data center places so many physical hosts. In IaaS, the utilization of physical host's resources is accomplished through Virtual Machines (VMs). Users can run any application or operating system on the server without paying any maintenance fee and operating fee for hardware. IaaS is more advantageous because the entire cloud infrastructure runs on the VMs and it can ensure reliable services for the users located in nearby locations.

Majorly, the VM accesses three resources of physical host to provide services to users; they are namely RAM, CPU and Hard Disk. The users request the resources present on the VM to perform their tasks. Hence, the cloud network suffers from non-uniform resources distribution and sometimes some VMs can't get resources because many VMs have non-preemptive and preemptive links to resources [4]. An improper tasks assignment to VMs causes serious delays in the task executions. Alongside, the number of resources present at the physical host is not constant, as it varies frequently. Hence, the tasks cannot be placed on a single host for longer times. If the amount of required resource requested by task is found as more than the available remaining resource amount at host, then the physical host can't handle such tasks and causes an execution failure. Next, If the amount of required resource requested by task is found as almost equal to the available remaining resource amount at host, then the physical host feels much burden and takes more execution time to complete the task. Further, if any cloud network is receiving the task requests continuously, then it suffers from huge load imbalance in executing the tasks and cannot ensure satisfactory and timely results to users which make the cloud network ineffective.

Hence, load balancing and task scheduling are found to be a hot research issue in cloud computing environments. Towards such prospect, this paper proposes a new mechanism for cloud computing which performs both load balancing and task scheduling effectively. The major contributions of this paper are outlined as follows.

Formulate a mathematical model for Task scheduling and load balancing based on the major resource attributes of Physical Hosts and Virtual machines.

Proposes a new Load balancing mechanism by determining the optimal number of physical hosts through heuristic clustering mechanism.

Proposes a new Task Scheduling mechanism based on Multi-Objective function formulated as the combinational model of three objective functions based on Cost, makespan and Resource Utilization.

The rest of the article is organized as follows; section II provides the literature survey details. Section III provides the details of the proposed methodology. Section IV investigates the details of experimental investigations and section V concludes the paper.

II. Literature Survey

Load balancing and Task scheduling are always hot topics of research in cloud computing environments, as their goal is to ensure an efficient handling of user's requests without putting larger computational burden on few hosts and VMs. So, many approaches have been introduced by many researchers in cloud computing in various aspects like resource allocation, task assignment, task scheduling and load balancing etc.

Y. H. Prasanna Raju and Nagaraju D. [5] suggested a clubbing clustering technique to ensure load balancing through biologically inspired algorithms in cloud computing. They proposed a hybrid method called "K-means with Particle Swarm Optimization (PSO) using weights (KPSOW)". KPSOW applied the two algorithms namely K-means and PSO and used their strengths to perform load balancing. Next, A. K. Maurya [6] proposed a new task scheduling approach which applies clustering of resources for cloud computing environment in different workflow applications. Their algorithm is an enhanced version of the most popular "Hybrid scheduling algorithm based on resource clustering (HySARC)" algorithm [7]. Like HySARC, this method also initially clusters all the resources present in the cloud and tasks requested and then applies a task scheduling approach to every cluster for executing the tasks. Compared to the HySARC algorithm, their algorithm shows good performance in terms of makespan and clustering time.

Kekun Hu et al. [8] focused on the parallel processing in cloud and suggested a 3-phase cluster scheduling approach which is aware of heterogeneous features of VMs to perform task scheduling of Directed Acyclic Graph (DAG). At first, they considered different granularities of DAG and clustered them into a linear set of clusters through a parallel scheduling. Next, they designed a load balancing algorithm based on the heterogeneous features of VMs to map the clusters to several computational nodes of the cloud network. In the final phase, a task ordering mechanism is designed which assigns the clusters as early as possible initiation periods.

M. Dong et al. [9] focused on reducing the cost by preserving the deadline constraint and proposed an "efficient task- clustering based cost-effective aware scheduling algorithm (ECOS)". At first, they considered different characteristics like cost model, heterogeneity of cloud and multiple types of workflows and task clustering is formulated to simplify the workflow's structure. Further, they also scheduled the workflow to reduce the cost within the constraint of deadlines. Then they modeled ECOS in two phases; (1) vertical clustering in which the sequential tasks are selectively merged thereby the transferring time of individual tasks will get reduced within the workflow. (2) Greedy allocation and Horizontal Clustering is proposed to accumulate the parallel tasks with an aim of cost minimization within the stipulated deadline.

Vrajesh Sharma, and Manju Bala [10] proposed a credits-based task scheduling algorithm for cloud computing. Totally they considered four parameters; they are namely cost, deadline, task priority and task length and applied a modified K-means algorithm to categorize the cloudlets and VMS. Geetha M., S. R. Chandran [11] suggested a "cluster-based task scheduling framework (CBTS)" approach through K-means clustering algorithm which considers capacity of VM and length of task to cluster the VMs and tasks respectively. VMs are clustered based on their processing capacity and tasks are clustered based on their lengths. After the completion of clustering, each task in cluster is assigned to an individual VM in their groups. Their approach performs load balancing by aiming at the execution time and makespan minimization. Compared the performance with "Grouped Task Scheduling (GTS)", "Dynamic Cloud Task Scheduling (DCTS)", "Online Potential Finish Time (OPFT)" and proven the effectiveness.

With an aim of load balancing and the prediction of execution times of tasks, Mao et al. [12] proposed a task scheduling mechanism based on a Min-max algorithm. Then, the VMs are allowed to take the tasks based on their Min-Max statistics. They experienced efficient utilization of VM resources and decreased execution time. Similarly, Kruekaew and Kimpan

[13] also aimed at load balancing and reduction of execution time and proposed a "Heuristic Task Scheduling with Artificial Bee Colony (HABC)" algorithm for task scheduling and cloud resources management. At optimization they employed a Largest Job First Heuristic (LJF) Algorithm and determined the effectiveness in scheduling and resource management. Madni et al. [14] proposed an approach called as "Multi-objective Cuckoo Search Optimization (MOCSO)" algorithm to handle the resource scheduling issue in IaaS based cloud computing environments. They mainly targeted at the reduction of cloud user cost and reducing the makespan time. Simulation results are done with different heuristic algorithms like PSO, Min-Max, Genetic Algorithm and Ant Colony Optimization and discovered that CSO had shown better performance.

Guo [16] proposed a Multi-objective Task Scheduling approach based on Fuzzy Self-defense approach in cloud computing environments. Here, they considered VMs resources utilization, deadline violation and shortest execution time as multiple objective functions. They employed the ACO algorithm for the optimization of weights in the multi-objective fitness function. Further, they proved that their method is more effective than the "Multi-Objective Optimization Scheduling Method Based on the Ant Colony Algorithm in Cloud Computing (PBACO) [20]" and Reinforcement Learning based Task scheduling algorithm [21].

M. S. Sanaj and P. M. Joe Prathap [17] focused on the efficient task scheduling in cloud computing and proposed a Maximum a Posterior (MAP) reducing approach with the assistance of two heuristic algorithms namely GA and Wolf Optimization Algorithm (WOA). At first, they represented each task with different features and then they were reduced through Multi-Resolution Q-Factor Based Linear Discriminant Analysis (MRQFLDA) algorithm. Then the tasks with larger size are partitioned into sub-tasks through MAP framework. Finally, the GA-WOA algorithm is employed for efficient task scheduling. Li and Han [18] mainly aimed at the minimization of maximum execution time, maximum device workload and overall workload on all devices and proposed a Flexible Task Scheduling algorithm based on Hybrid Discrete Artificial Bee Colony (ABC) algorithm.

Alsadie [19] introduced a multi-objective task scheduling framework called "Meta-Heuristic Framework for Dynamic Virtual Machine Allocation (MDVMA)" in cloud computing environments. MDVMA considers multiple objectives namely energy usage, makespan and cost for modelling the multi-objective fitness function. The experimental investigations showed that MDVMA perform well than the ABC algorithm, WOA algorithm and PSO algorithm in terms of energy, makespan and cost.

III. Proposed Approach

3.1 Overview

The proposed approach mainly focused on task scheduling and load balancing in cloud computing environments. Towards such work, this paper introduced a new clustering mechanism and task scheduling based on resources present in Cloud. The proposed clustering mechanism is mainly intended at load balancing and the task scheduling mechanism intended at the efficient resource utilization. At task scheduling, we consider multiple Objective functions and formulated a final and composite objective function through which scheduling is accomplished. Since each physical host have several virtual machines, we adapt multiple attributes at the selection of number of virtual machines and types of virtual machines At task scheduling, we consider totally three attributes namely execution time, cost and resource utilization. The complete details of load balancing based on clustering and task scheduling are explored in the following sections.

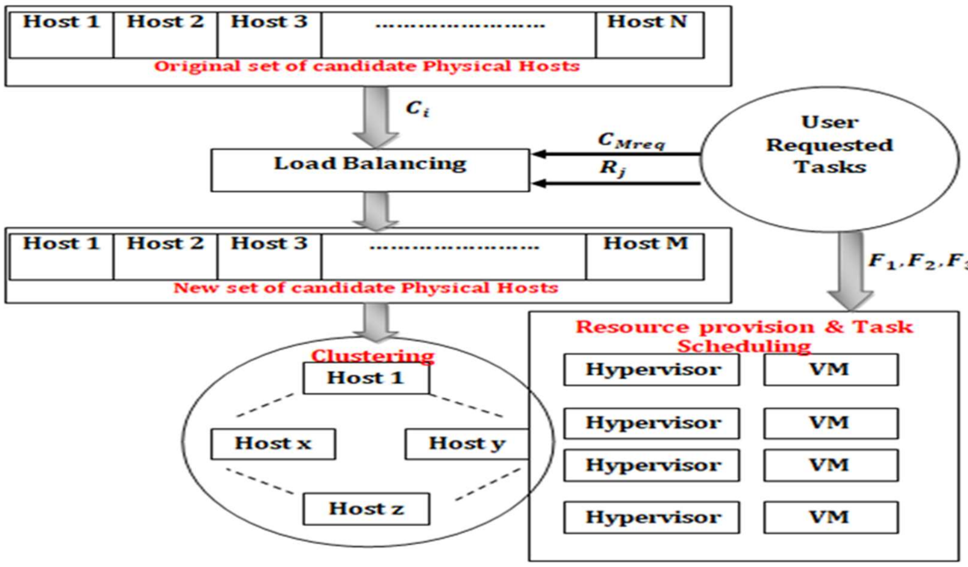

Figure.1 working flow of proposed task scheduling mechanism.

3.2 Load balancing

The load balancing problem is constituted as the determination of optimal physical host to execute tasks requested by users in cloud computing environments. Generally, in the cloud data centre there exist so many physical hosts which are ready to give service. However, the proper selection of physical host is required because a random or improper physical host selection may constitute an unsatisfactory service to the users. Hence, we applied host clustering mechanism which finds optimal number of hosts. The selection process considers the resource requirements of the requested task. So every physical host has a constraint value and it can't provide service beyond its constraint value. Here the constraint value of a host is measured based on available remaining resource amount and let it be denoted as C_i. Mathematically it is expressed as

$$C_i = \alpha \times C_c^i + \beta \times C_m^i \qquad (1)$$

Where C_c^i is the remaining CPU resource of host i and C_m^i is the remaining memory resource of host i. Next α and β are the weight parameters of CPU and memory respectively. In this manner the constraint value of each physical host is computed and then compared with the maximum requested resource amount C_{MReq} mathematically it is expressed as

$$C_{MReq} = \max_j (R_j) \forall j \in n \qquad (2)$$

Where R_j denotes the requested resource amount of the j^{th} task, n denotes total number of tasks. The mathematical expression for R_j is given as

$$R_j = \alpha \times R_c^j + \beta \times R_m^j \qquad (3)$$

Where R_c^j and R_m^j are the requested CPU and memory resources by the task j respectively. Then the C_i of each host is compared with C_{MReq} and if it found as larger, then the host i is placed into a new set and let it be denoted as S_n. Upon comparing every host's C_i with the C_{MReq}, the new set $S_n$is obtained and is represented as $S_n = \{S_n^1, S_n^2, S_n^3, \dots, S_n^m\}, m' \leq m$ where m denotes the total number of hosts in original data centre and m' denotes the total number of hosts in the set S_n. Then hosts present in the new set are called as candidate host and they are used for clustering process.

Next the physical hosts with better performance are processed for clustering. However, we can't ensure that every physical host put in set can process any task requested by user because some physical hosts may have larger C_i value but they can't do our aimed task. Hence, we aimed at the removal of unreasonable hosts. Towards such aspect, we perform clustering which can extract optimal number of hosts through which the data centre can achieve a long-term load balancing. So, the major aim of clustering is to pick up the optimal number of hosts whose relative computing power is more such that they can perform the requested tasks in the task set there by the objective of load balancing is achieved. Here the clustering process considers the posterior probability of each host. The posterior probability of a host i is calculated as

$$P(Y_i|X) = \frac{P(X|Y_i) \times P(Y_i)}{\sum_{i=1}^{m'} P(X|Y_i) \times P(Y_i)} \qquad (4)$$

in the above expression event X is defined as the execution of certain tasks on some physical host and event Y_i is defined as the event that host i is chosen to execute the task. Then the prior probability of host i is defined as the ratio of maximum requested resource amount of all tasks requested in the time interval Δt and the remaining computing power of host i present in the set S_n. When C_{MReq} is closer to C_i then the prior probability is close to 1, however this is a contradictory statement because the host whose computational power is more is if it is found as more suitable host for the exhibition of task. Hence the modified probability is completed as

$$P(X|Y_i) = 1 - \frac{C_{MReq}}{C_i} \qquad (5)$$

Since there exist m' number of physical hosts in S_n, the probability to choose i^{th} host is computed as $P(Y_i) = 1/m'$. So based on this, the posterior probability of host i is expressed as

$$P(Y_i|X) = \frac{(C_i - C_{MReq})C_{1,\dots,}C_{i-1}C_{i+1}}{m'C_{1,\dots,}C_{m'} - C_{MReq}(C_2C_3 \dots C_{m'} + C_1C_2 \dots C_{i-1}C_{i+1} \dots C_{m'})} \qquad (6)$$

Now every physical host in the set S_n has posterior probability. Let $P_i = P(Y_i|X)$ is the posterior probability of host i, it also has two more attributes namely CPU and memory and denoted as C_c and C_m respectively. The posterior probability values of hosts in set S_n are sorted and placed in a descending order. Consider S_{nj} is a physical host with larger posterior probability value, then it is selected as a cluster head (CH). Now the new set is formulated by comparing the similarity degree (SD) between each physical host with cluster head host. Mathematically, the similarity degree is calculated as

$$SD = \frac{1}{\sqrt{(P_d^{ij}+C_c^{ij}+C_m^{ij})^2}} \quad (7)$$

Where $P_d^{ij} = \left(P(S_{nj}|X) - P(S_{ni}|X)\right)^2$is the SD through posterior probability between CH host and *i^th^* host, $C_c^{ij} = \left(C_c^j - C_c^i\right)^2$ is the SD through CPU between CH host and *ith* host and $C_m^{ij} = \left(C_m^j - C_m^i\right)^2$ is the SD through memory between CH host and *ith* host. Then we compute a threshold (SD_{th}) and each SD value is compared withSD_{th}. If the SD value of any host is found as more than the SD_{th}, then the corresponding host is placed in the new set and let it be denoted as S'_n. In the new set S_{nj} is kept at top and remaining hosts or accumulated after comparing their SD values with threshold hence the new set is expressed as $S'_n = \{S'_{n1}, S'_{n2}, \dots, S'_{nm''}\}, m'' \leq m' \leq m$.

3.3 Task Scheduling

Once the physical host is assigned to the required request resource amount for the set of tasks, then the Cloud Service Provider performs task scheduling. Each physical host is monitored by hypervisor as it assigns the resources of host to users through Virtual Machines. This kind of process is called task scheduling. A virtual machine constitutes of resources of host like RAM, Hard Disk and CPU so the users execute their tasks through VM by utilizing the resources of host machines. Task scheduling plays an important role in cloud computing environments which makes efficient resource utilization, improves the response time, reduces latency, and balances the load on each host. Generally, task scheduling is treated as an objective function-based solution in which the resource parameters are formulated as an objective function and solved in an iterative fashion. In our approach we consider multi objective function which has constituted by the integration of three individual objective functions based on execution time, cost, and resource utilization. The multi objective function is formulated as $F(x) = \{f_1(x), f_2(x), \dots f_k(x)\}$as where *k* denotes the number of individual objective functions. Such kind of objective function won't have a single solution and hence it is being solved through a set of non-dominated individual solutions.

A. Execution time

In our approach, the first objective is defined based on the make span or execution time which is defined as the time taken to complete the task. The makespan is a more useful parameter which can help the completion of tasks prematurely and helps in reducing the execution time. The makespan defines individual execution times for individual VMs. For a given VM, if its execution time is found as more, then its makespan is also more and such kind of VM is considered as poorly distributed operator. On the other hand, the lower value of execution time lowers the makespan. Consider a task $t_i \in T$ is assigned to a virtual machine$VM_j \in V$, then the task of is represented as t_{ij}. So, the total execution time is computed as

$$ET(t_{ij}) = \frac{\sum_{t_{ij}} Lengt\ (t_{ij})}{CPU(VM_j)} \quad (8)$$

Where $Length(t_{ij})$ is defined as the length of task that is defined through total number of instructions (million instructions) and $CPU(VM_j)$ is calculated as CPU rate to process in the cloud. Based on these values the makespan is computed as ratio of maximum of execution times of all virtual machines so

$$MS = \max\left(ET(VM_j)\right), 1 \le j \le Q \quad (9)$$

Similarly Minimum Makespan is computed as

$$MinMS = \max\left(ET(VM_j)\right), 1 \le j \le Q \quad (10)$$

Based on these two values, the fitness function is modeled as a ratio of minimum makespan and overall makespan, mathematically it is expressed as

$$F_1 = \frac{MinMS}{MS} \quad (11)$$

B. Cost

Under the second objective we consider the cost of execution of the task. Simply the task cost is computed in terms of three sub costs, they are CPU usage cost, memory usage cost and bandwidth usage cost. For a task t_i processed by virtual machine VM_j the cost is computed as

$$C(t_{ij}) = \left(c_1 \times ET(t_{ij})\right) + \left(c_2 \times ET(t_{ij})\right) + \left(c_3 \times ET(t_{ij})\right) \quad (12)$$

Where c_1, c_2 and c_3 are the bandwidth usage cost for Unit, Memory usage cost per unit and CPU usage cost for unit respectively. The overall cost is computed as the summation of cost of all tasks processing through overall VMs, mathematically it is expressed as

$$TC = \sum_{i=1}^{n} \sum_{j=1}^{m''} C(t_{ij}) \quad (13)$$

Then the fitness function through cost function is modeled as the ratio of minimum total cost and overall total cost of all VMs, mathematically it is expressed as

$$F_2 = \frac{MinTC}{TC} \quad (14)$$

Where $MinTC$ is defined as the minimum total cost when the set of assignment tasks are processed through VM as

$$MinTC = \min\left(TC(VM_j)\right) \forall j \in m'' \quad (15)$$

C. Resource utilization

Third objective function is defined in terms of resource utilization. Under resources we consider majorly CPU and memory as main attributes. For a given task processed through a VM j, the memory load ML_j on VM j is calculated as

$$ML_j = BM_j + \frac{RM_j}{AM_j} \quad (16)$$

Where BM_j is memory amount used before the execution of task at virtual machine, RM_j is the memory need to be used by the virtual machine for the execution of task and AM_j is the total available memory at VM.

Next the CPU parameter is also computed in the same manner. For a given task the CPU load on virtual machine is calculated as

$$CL_j = BC_j + \frac{RC_j}{AC_j} \quad (17)$$

Where BC_j is the CPU amount used before the execution of task at virtual machine, RC_j is the CPU need to be used by virtual machine and AC_j is the total available CPU at VM.

Based on these attributes the VM utilization is computed as follows

$$F_3 = w_1 * \frac{1}{1-ML_j} + w_2 * \frac{1}{1-CL_j} \quad (18)$$

Where w_1 and w_2 denotes the weight of CPU and memory resource usages respectively. The values of w_1 and w_2 are assigned in such a way $w_1 + w_2 = 1$. In our paper, we fixed the values of w_1 and w_2 to 0.5 because both CPU and memory or important.

Finally, the multi objective function is formulated by combining all the three individual objective functions and its mathematical expression is given as

$$F = \alpha \times F_1 + \beta \times F_2 + \gamma \times F_3 \quad (19)$$

Where α, β and γ are the weight coefficients of fitness functions F_1, F_2 and F_3 respectively. The value at which the F gets maximized is considered as best values and such virtual machine is assigned for task scheduling.

IV. Experimental Results

Here in the current section, we discuss the details of experimental investigations conducted on the proposed mechanism. A vast set of experiments are conducted by varying the simulation parameter's, number of tasks and the performance is measured through different performance metrics like Throughput (Tasks/Sec), and Makespan (Sec). Two datasets are considered for simulation experiments, they are namely GOCJ dataset [23] and Synthetic Workload Dataset [24]. In this section, initially we discuss the GOCJ dataset and the obtained results and then the details of Synthetic Workload Dataset and obtained results.

A. GOCJ dataset

This is treated like Realistic Google dataset and was made from the workload behaviors happened in the traces of Google Cluster. A well-known simulation method called Monte Carlo simulation is employed for this dataset creation. The size of tasks in GOCJ dataset ranges from 15k to 900k Million Instructions (MIs). Totally the tasks in this dataset are classified into five categories; they are Small Sized Tasks, Medium Sized Tasks, larger sized tasks, extra larger sized tasks, and Huge larger sized tasks ranging from 15k-55k MIs, 59k-99k MIs, 101k-135k MIs, 150k-337.5k MIs and 525k-900k MIs respectively. This dataset is provided for the evaluation purposes in the form of Different text files in the Mendeley Repository. The data is organized in different rows and columns which consist of numeric values. Each numerical value signifies the cloud task size through MIs.

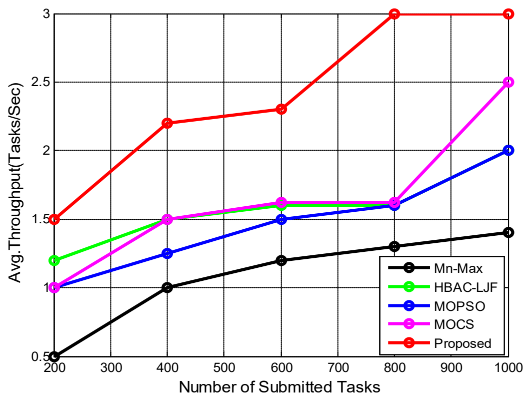

Figure.1 Average Throughput for varying number of submitted tasks.

Figure.1 shows the variations of Average throughput for varying number of submitted tasks in GOCJ dataset. From the results, we can see that as the number of submitted tasks increases, the Average throughput is increasing. Since throughput is linearly related to the number of tasks, the throughput follows a linear relation with task count. Further, it can be noted that the proposed method has gained maximum throughput at every instance of task count because it applied organized task scheduling. Compared to the proposed methods, no method has adapted to the clustering of physical hosts. Hence, they showed limited throughput, especially at the larger number of submitted tasks. On an average, the proposed approach has gained an average throughput of 2.4 tasks/sec while the existing method has gained 1.64 tasks/sec, 1.47 tasks/sec, 1.58 tasks/sec and 1.08 tasks/sec by MOCS [14], MOPSO [15], HBAC_LJF [13] and Min-Max [12] algorithms respectively.

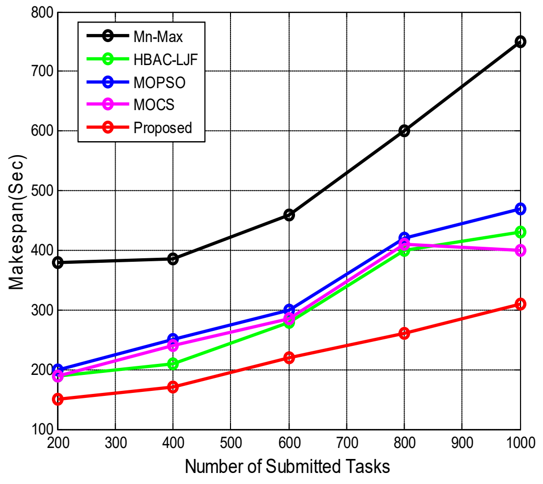

Figure.2 Makespan for varying number of submitted tasks.

Figure.2 shows the variations of Makespan for varying number of submitted tasks in GOCJ dataset. From the results, we can see that the makespan values of request tasks increases along with the increases in the number of requested tasks. The existing methods have applied random task scheduling upon getting the user request and hence the physical hosts can't handle them due to their busy execution with other tasks. In such cases, the new requests need to wait for longer times which increases the delay. Hence, the makespan of existing methods is more compared to the makespan of proposed mechanism. Further, the proposed approach adapted to cluster the hosts based on their remaining resource availability, the hypervisor optimally picks up the physical hosts and assigns tasks to their VMs. Such kind of flexibility is not observed in any of the existing methods. Hence, they experienced a larger makespan than the proposed approach. On average, the proposed approach has gained an average makespan of 222 Seconds while the existing method has gained 3.5 sec, 328 sec, 302 sec and 515 sec by MOCS, MOPSO, HBAC_LJF and Min-Max algorithms respectively.

B. Synthetic Workload dataset

This dataset is generated using two methods, they are Monte-Carlo simulation and a random number generator mechanism. The size of tasks in this dataset ranges from 1 to 45k Million Instructions (MIs). Totally the tasks in this dataset are classified into five categories; they are tiny sized tasks (1-250 MIs), Small Sized Tasks (800-1200 MIs), Medium Sized Tasks (1800-2500 MIs), larger sized tasks (7k-10lk MIs), and extra larger sized tasks (30k-45k MIs). Compared to the GOCJ dataset, the synthetic workload dataset is smaller in size because the length of tasks is smaller as they have a smaller number of instructions. So, the makespan of this dataset is observed as less when compared with the makespan of GOCJ dataset. Further, the Average throughput is observed as more than the throughput of GOCJ dataset. Since the proposed approach can pick up an optimal number of clusters, only a small number of clusters can execute the tasks of this dataset.

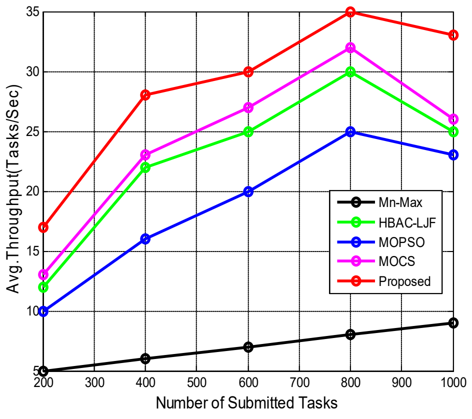

Figure.3 Average Throughput for varying number of submitted tasks.

Figure.3 shows the variations of Average throughput for varying number of submitted tasks in synthetic workload dataset. In this analysis, the Min-Max algorithm has gained very less average throughput than all the methods. Since the min-max algorithm adapted simple Min-Max statistics of VMs, it has taken a lot of time to execute even small sized tasks. Hence, the throughout is observed as very less, the average throughput is found as only 7 tasks/sec. On the other hand, the proposed method has gained maximum average throughput, it is observed as 28.6 tasks per second. The remaining existing methods has experienced a slightly lower throughput of 24.2 tasks/sec, 18.8 tasks/sec and 22.8 tasks/sec by MOCS, MOPSO, HBAC_LJF respectively.

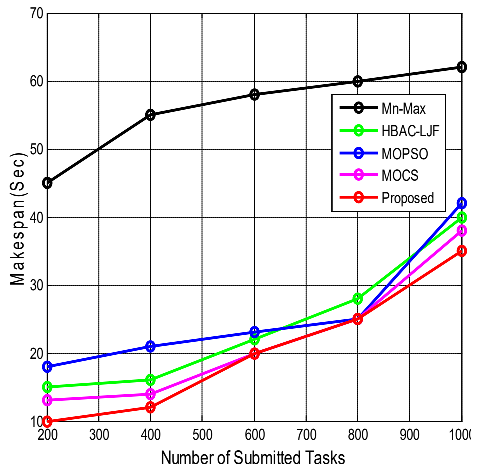

Figure.4 Makespan for varying number of submitted tasks.

Figure.4 shows the variations of Makespan for varying number of submitted tasks in synthetic workload dataset. Compared to the size of tasks in GOCJ dataset, the tasks of synthetic workload dataset are lower in size and hence the methods experienced a less makepsan, it is around tent times lower than GOCJs makespan. Among the methods employed for execution, the proposed method obtained lesser makespan than all the existing methods. The major reason is that it has the flexibility of picking up only optimal set of physical hosts thereby unnecessary hosts and the corresponding VMs are not considered. On average, the proposed approach has gained an average makespan of 20.4 Seconds while the existing method has gained 22 sec, 25.8 sec, 24.2 sec and 56 sec by MOCS, MOPSO, HBAC_LJF and Min-Max algorithms respectively.

V. Conclusion

In this paper, we proposed a task deployment strategy to ensure load balancing and efficient resource utilization in cloud computing environments. Load balancing is attained through the newly proposed heuristic clustering mechanism which can optimally pick up the physical hosts based on their posterior probability, remaining CPU resource and remaining memory resource. Out of huge number of physical hosts, only few hosts are picked up which can perform tasks effectively by assigning the available resources through VMs. Further, for task scheduling, we adapted a multi-objective-based mechanism which is formulated as a combination of three individual objective functions based on execution time, cost, and resource utilization. Simulation experiments reveal the efficacy of the proposed approach in terms of both load balancing and resource utilization for both smaller sized and larger sized tasks. Two standard and publicly available datasets namely GOCJ and Synthetic workload dataset are used for experimental validation and the performance is assessed through makespan and throughput. Finally, the comparison with existing task scheduling methods proves the superiority of the proposed method with state-of-the-art methods.

References

[1] T. S. George and V. P. S. Kumar, “Multicloud computing for on-demand resource provisioning using clustering” in Proc. 3rd Int. Conf. Sustain. Energy Intell. Syst. (SEISCON), 2012, pp. 435_440.

[2] S. Yang, L. Pan, Q. Wang, S. Liu, and S. Zhang, ``Subscription or pay-asyou-go: Optimally purchasing IaaS instances in public clouds," in Proc. IEEE Int. Conf. Web Services (ICWS), Jul. 2018, pp. 219-226.
[3] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang, ”Toward Cloud-based Vehicular Networks with Efficient Resource Management,” IEEE Network Magazine, vol. 27, no. 5, pp. 48-55, 2013.
[4] K. Psychas and J. Ghaderi, “On non-preemptive VM scheduling in the cloud” Proc. ACM Meas. Anal. Comput. Syst., vol. 1, no. 2, pp. 1-29, Dec. 2017.
[5] Y. Home Prasanna Raju and Nagaraju Devarakonda, “ Cluster based Hybrid Approach to Task Scheduling in Cloud Environment”, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 10, No. 4, 2019.
[6] Ashish Kumar Maurya, “Resource and Task Clustering based Scheduling Algorithm for Workflow Applications in Cloud Computing Environment”, 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC),
[7] M.A. Vasile, F. Pop, R.I. Tutueanu, V. Cristea, “HySARC 2: Hybrid scheduling algorithm based on resource clustering in cloud environments,” In International conference on algorithms and architectures for parallel processing,Springer, pp. 416-425, December 2013.
[8] K. Hu, G. Zeng, S. Ding and H. Jiang, "Cluster-Scheduling Big Graph Traversal Task for Parallel Processing in Heterogeneous Cloud Based on DAG Transformation," in IEEE Access, vol. 7, pp. 77070-77082, 2019.
[9] Minggang Dong, Lili Fan a, Chao Jing, “ECOS: An efficient task-clustering based cost-effective aware scheduling algorithm for scientific workflows execution on heterogeneous cloud systems”, The Journal of Systems and Software 158 (2019) 110405.
[10] Vrajesh Sharma, Manju Bala, “An Improved Task Allocation Strategy in Cloud using Modified K-meansClustering Technique”, Egyptian Informatics Journal 21 (2020) 201–208.
[11] Geetha Muthusamy, and Suganthe Ravi Chandran, “Cluster-based Task Scheduling Using K-Means Clustering for Load Balancing in Cloud Datacenters”, Journal of Internet Technology, vol. 22, no. 1 , pp. 121-130, Jan. 2021.
[12] Y. Mao, X. Chen, and X. Li, “Max-min task scheduling algorithm for load balance in cloud computing” in Proc. Int. Conf. Comput. Sci. Inf. Technol., 2014, pp. 457-465
[13] B. Kruekaew and W. Kimpan, “Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing” Int. J. Comput. Intell. Syst., vol. 13, no. 1, pp. 496_510, 2020.
[14] S. H. H. Madni, M. S. A. Latiff, J. Ali, and S. M. Abdulhamid, ``Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds," Arabian J. Sci. Eng., vol. 44, no. 4, pp. 3585-3602, 2019,
[15] H. Saleh, H. Nashaat, W. Saber, and H. M. Harb, ‘IPSO task scheduling algorithm for large scale data in cloud computing environment” IEEE Access, vol. 7, pp. 5412-5420, 2018.
[16] X. Guo, “Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm," Alexandria Eng. J., vol. 60, no. 6, pp. 5603-5609, Dec. 2021.
[17] M. S. Sanaj and P. M. Joe Prathap, “An efficient approach to the map reduce framework and genetic algorithm-based whale optimization algorithm for task scheduling in cloud computing environment” Mater. Today, Process., vol. 37, pp. 3199-3208, Oct. 2021.

[18] J.-Q. Li and Y.-Q. Han, “A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing systems” Cluster Comput., vol. 23, no. 4, pp. 2483-2499, Dec. 2020. [19] D. Alsadie, ``A metaheuristic framework for dynamic virtual machine allocation with optimized task scheduling in cloud data centers," IEEE Access, vol. 9, pp. 74218_74233, 2021.
[20] L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, ``A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing," IEEE Access, vol. 3, pp. 2687-2699, 2015.
[21] B. Kruekaew and W. Kimpan, "Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm With Reinforcement Learning," in IEEE Access, vol. 10, pp. 17803-17818, 2022.
[22] A. Hussain and M. Aleem, “GoCJ: Google cloud jobs dataset for distributed and cloud computing infrastructures”, Data, Vol. 3, No. 4, pp. 38, 2018.
[23] A. Hussain, M. Aleem, A. Khan, M. A. Iqbal, and M. A. Islam, “RALBA: A computation-aware load balancing scheduler for cloud computing”, Cluster Comput., vol. 21, no. 3, pp. 1667-1680, 2018.