Future Technologies Conference (FTC) 2017
29-30 November 2017| Vancouver, Canada

Optimization of Multi-Dimensional Metrics through Task Scheduling in Cloud Computing Systems

Sambit Kumar Mishra, Saurabh Kumar Choudhary, Bibhudatta Sahoo, Mahardhika Pratama, Mohammad S. Obaidat
Fellow of IEEE & Fellow of SCS and Deepak Puthal
National Institute of Technology, Rourkela, India
Nanyang Technological University, Singapore
Fordham University, USA and University of Jordan, Jordan
University of Technology Sydney, Australia,
Corresponding Email: m.s.obaidat@ieee.org and Deepak.Puthal@uts.edu.au

***Abstract*—Cloud-based data centers consume a considerable amount of energy, which is an expensive system. The virtualization technique helps to overcome various issues including the energy issue. Because of the dynamic nature of workload, task consolidation is an effective technique to decrease the total number of servers and unnecessary migrations and consequently optimize energy. Effective task allocation techniques act as a key issue to optimize several performance parameters in the cloud system. This paper presents a novel task consolidation technique to achieve energy-makespan-throughput optimally balanced in the cloud data center. We evaluate the performance of our proposed algorithm using simulation analysis in Java-based CloudSim simulator environments. Results of performance evaluation certify that our proposed algorithm has reduced the energy consumption as compared to existing standard algorithms, and optimized the makespan and throughput of the cloud data center.**

Keywords—Cloud computing; task scheduling; energy consumption; makespan; throughput; simulation

I. INTRODUCTION

Cloud Computing is such a trending technology that has gained immense popularity and acceptance among users worldwide. This gain has led to a considerable increase in the number of data centers. A data center is a major component of cloud computing. It includes shared resources whose processing power can meet the requirements of many users' computing. A cloud service provider (CSP) is a company that is responsible for providing services such as Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS) [18]. A CSP constitutes of either privately owned or third party owned data centers, which help them in providing required services to their users. Cloud users only have to obtain the required amount of resources from cloud infrastructure for the execution of the task. These services are provided by the CSP on a pay-per-use or rental basis. The important activity behind the service delivery in the cloud is the estimation of efficiency and effectiveness of the system.

A constant intensity to enhance efficiency and effectiveness of data centers has led to ignoring energy consumption. A quick view of the energy consumption in data centers for the year 2014 in the US, which was 70 billion kilowatt-hours of electricity (which amounts to 4% gain in total data center power consumption from 2010 to 2014), clearly validates the fact that energy consumption parameters have always been somehow neglected [1]. With the rise in energy consumption by the datacenters, there is a rise in operational cost. Apart from that, greater energy consumption also leads to increase in temperature, reduction in reliability and longevity of resources. These results in the increase in the emission of CO_2 and causes the greenhouse effect.

The energy consumption in a data center depends upon various sources. These sources include CPU, RAM, storage, network and many others. Among these sources, the energy consumption of CPU amounts greater than 50%. Fig. 1 shows a comparison of energy consumption among various sources [2]. Keeping a future outlook in mind, while the concept of energy consumption seems to be of great importance, the efficiency and effectiveness of data center resources cannot be compromised. Various researchers have identified the problem of maintaining a balance between efficiency and energy consumption, and have proposed various solutions [3], [10], [12].

In this paper, in order to maintain a perfect balance between efficiency and energy consumption, we have presented a different approach as studied in the literature. This paper uses the deadline as a major parameter, which helps in setting the task priority. Our approach is to minimize task failure rate and energy consumption along with maximizing resource utilization while selecting the best possible resource available in the data center for task processing.

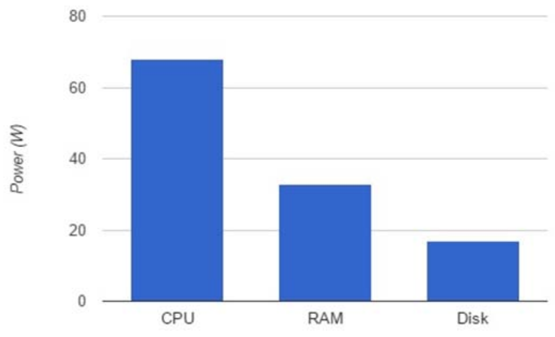

Fig. 1. Power consumption by different sources.

This work has the following key contributions:

- It presented a cloud system model that shows the service delivery process.
- It proposed an energy-aware algorithm to increase the stability of the system regarding the energy consumption, makespan, and throughput of the system.
- A comparative analysis has been made between the proposed algorithm and major competing existing algorithms, (i.e., First Come First Serve (FCFS) algorithm, modified Round-Robin algorithm, and Random algorithm).

The remaining of the paper is arranged as follows. Section II describes an overview of some related work. Section III presents the cloud system model along with service delivery process. Section IV illustrates our proposed work to model and minimize the energy consumption accompanying with makespan and throughput in a heterogeneous computing environment. Section V presents the performance evaluation and shows the effectiveness of our algorithm. Section VI concludes the paper.

II. Related Work

There has been great progress in data center efficiency and utilization over the recent years by researchers and IT specialists to investigate various aspects of cloud. The performance metrics of the cloud system are optimized through various approaches like task consolidation [3]-[5], VM consolidation [2], [7], and Dynamic Voltage Frequency Scaling (DVFS) [6]. In this paper, we have optimized the energy consumption, makespan, and threshold of the system through task consolidation. Cao and Dong [7] have tried to balance the relationship between energy and performance by proposing a new energy-efficient framework for VM consolidation. In the framework, they have defined an algorithm for SLA violation to determine when a host is overloaded by considering some performance metrics.

Zhao et al. [3] have proposed a Tree-to-Tree task scheduling method based on Task Requirement Degree estimation to enhance the energy performance of the cloud system. This approach reduces the number of active machines, thereby, decreasing the time expenditure in data transmission and optimizing the utilization of its computing resources and bandwidth. The authors in [8] have developed an algorithm to estimate the performance gap between cloud and local resources. The algorithm aims to prove the fact that even if a cloud has a slower network speed than a local resource, the cloud can still give better overall performance than the local resource for bursty jobs. The algorithm focuses on three major factors; these are job queue waiting time, execution time and relative performance of the cloud compared to that of a local resource. The algorithm predicts these factors and combines them to make the allocation decision.

Keshanchi and Navimipour in [4] have proposed a task scheduling algorithm using various priority chains and memetic algorithms. The authors have used a genetic algorithm besides hill climbing to designate priority to each subtask and then, adopted a heuristic based earliest termination time approach for the task to processor mapping. The authors in [9] have aimed to reduce uncertainty propagation in real-time workflow scheduling. The authors have presented an uncertainty-aware scheduling design to minimize the influence of uncertainty factor on the quality of workflow schedules. They have presented a dynamic workflow scheduling algorithm that can employ the proactive and reactive scheduling methods dynamically.

Yassi et al. [6] have addressed the general optimization problem of cloud workflow scheduling that requires examining multiple criteria so as to meet a great number of QoS (Quality of Service) requirements. The authors have designed a hybrid PSO algorithm to optimize scheduling performance along with the usage of DVFS technique to depreciate energy usage. The authors in [5] have suggested a task scheduling algorithm for mapping tasks to VMs in order to improve the throughput of the data center without any violation of Service Level Agreement (SLA).

TABLE I. Summary of Related Works

Task Scheduling Technique	**Objective**	**Resource Considered**
[3]	Energy consumption	CPU, Network Bandwidth
[4]	Makespan	CPU, RAM
[6]	Makespan, Cost, Energy	CPU
[10]	Makespan, Cost, and Energy	CPU, RAM
[11]	Makespan, Resource Cost, preserving the fault tolerance	CPU, Memory
[12]	Optimizing application, Energy efficiency	CPU
[13]	Energy consumption. Increase the performance	CPU
[14]	Makespan	CPU

From this related work, we observe that most of the researchers considered CPU time as the resource of the cloud system to optimize various parameters as shown in Table 1. Table 1 shows some specific related works and their primary objectives with resource consideration briefly. In order to address the limitations of existing standard scheduling solutions, we proposed a new efficient algorithm. In summary, the proposed algorithm: 1) has a well-organized structure to efficiently map tasks to cloud resources (VMs), and 2) enables cloud users to optimize the energy consumption, total execution time (makespan) and throughput of the system.

III. System Model and Architecture

This section explores the proposed cloud system model and architecture, which consists of data centers with hosts which in turn contains multiple virtual machines as shown in Fig. 2. All of these components and sub-components possess a different level of hardware characteristics. Besides this, the top layer of our model comprises users, who are responsible for submitting their service request for processing. These service requests are submitted to Task Manager (or CSP), whose job is to process every task and map it to the best performable virtual machine. The task manager after completing its job submits the final mapped service request to the broker. The broker then takes the responsibility of submitting the mapped service request to the data center for processing. Following this paragraph, we have briefly defined every entity of the proposed model along with its characteristics.

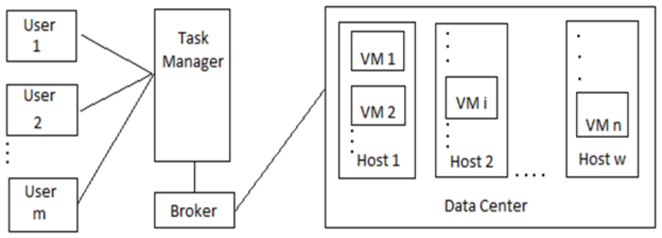

Fig. 2. Cloud system model.

A. User

Users are the top most layer of our system model. They are responsible for submission of service requests along with their deadline. These service requests require different amount of resources for processing and all these service requests constitute the tasks set *T*. Each task (T_i in *T*) is an independent, individual task (T_i), which has its identification, length, and deadline, i.e., $T = \{T_1, T_2, ..., T_m\}$, where m is the total number of service requests or tasks.

B. Data Center

A data center is the composition of the network of computers that allow storage and has the power to process users' service request. Every data center has its own characteristics. These characteristics include an operating system, virtual machine manager, host list, and much more.

C. Host

A data center can consist of one or more physical host. These hosts are responsible for hosting the virtual machines in a data center. Every host in a data center has an id, RAM (main memory), bandwidth, storage, and a list of processing elements depending upon the core of the machine. The host set (*H*) in a data center can be defined as, $H= \{H_1, H_2, ..., H_w\}$, where *w* is the finite number of hosts.

D. Virtual Machine

A set of virtual entities deployed in a data center over the physical host is identified by the VM set *V*. The set *V* represents the finite number of VMs, i.e., $\{VM_1, VM_2, ..., VM_n\}$, where *n* is the total number of VMs. Every virtual machine (VM_i) present in the set (*V*) is hosted by some host in *H*. These virtual machines are responsible for processing service requests of users. Every virtual machine has its id, speed (MIPS), RAM, bandwidth, a number of cores, and a virtual machine manager (VMM).

E. Task Manager

The users' task (with task length) and its corresponding deadline once obtained, has to be mapped with the best performable virtual machine available in VM set *V*. The task manager is responsible for this mapping and hence it is responsible for the success of the proposed algorithm.

F. Broker

Once every task in set *T* is mapped to its best performable virtual machine in the set *V*, it is submitted to the broker. It is the responsibility of broker to retrieve resource information and assign every task T_i to its mapped VM_i, and finally service back to the end use. It has complete information about the available resources of the system.

IV. Proposed Work

An attempt to reduce the consumption of CPUs' can directly help in reducing the total energy consumption as shown in Fig 1. The main focus of this paper is the utilization of all virtual machines present in an activated host in order to minimize task failure and maximize efficiency with the use of minimum amount of energy or power. The proposed approach takes deadline along with task length to set priorities for the task. Once the tasks are sorted according to their priorities, from high to low, the best possible virtual machine is chosen for them whose processing power allows the task to meet its deadline requirement (or SLA). The calculation of task priority along with different performance metrics is described in following sub-sections.

A. Task Priority

By task priority, we mean the importance of consideration of a task. In order to determine the priority of a task, we consider a task priority factor. A task with high priority represents high concern and therefore must be processed first, on the other hand, a task with low priority represents low concern and must be processed after the completion of higher priority tasks. A larger task with fewer deadlines has higher priority value. Generally, the processing of these high preference tasks is more productive for the service providers.

Task priority factor (P_i) is calculated as the ratio of task length (L_i) and deadline (d_i) as in (1).

$$P_i = {}^{L_i}/_{d_i} \tag{1}$$

Clearly, a task with higher task priority factor represents higher priority, and sorting the tasks according to this factor arranges the tasks from highest priority to lowest.

B. Throughput

Throughput is an important performance metric for any system. The high throughput value of the cloud system results in more profit for the service providers. The deadline parameter that is input along with task also assists in minimizing the number of failures, thereby maximizing the throughput. A task is rejected only when the data center has no resources, or the resources are busy with other tasks, and hence the required deadline of the task cannot be met. The throughput value of the system is calculated using (2).

$$\tau = m - c \tag{2}$$

C. Makespan

Makespan of the system is the maximum time required to complete all input service requests by the system as in (3). Reducing the makespan has always remained a challenging job for researchers, let alone maximizing throughput, minimizing power consumption alongside minimizing makespan. This paper presents an approach to meet all the three desired constraints. The throughput maximization can be achieved as discussed earlier. In order to minimize the makespan, we consider a task and allow the best performable resource to execute that task. It is an interesting fact that the best performable resource is not the resource with the best host or VM characteristics. The major reason behind this is that the resource with best possible characteristic might be busy in executing some other task of higher priority and hence is not the best performable resource for the considered task.

$$ɱ = Max\{ST_i\} \quad (3)$$

This approach of consistently providing the most efficient resource to every task always helps us in improving the makespan of a sequence of tasks.

D. Energy Consumption

There is no point in minimizing makespan and maximizing the throughput if the energy consumption of a data center is beyond control. The energy consumed by a resource (VM) in an active state, which depends on the processing speed [15], [16] and inactive (or idle) state can be calculated by using (4) and (5), respectively.

$$\xi_{a_i} = 10^{-8} \times §_i^2 \quad (4)$$

$$\xi_{in_i} = 0.6 \times \xi_{a_i} \quad (5)$$

With the formula in (5), it is clear that even if a resource is inactive, it still consumes 60% of the energy as it would have consumed while remaining active [17]. Therefore, it is not sufficient to keep few resources active and the remaining inactive, as the inactive resources would also consume a considerable amount of energy. The total energy consumption of the system is the sum of energy consumption of all VMs during the active and inactive state for the specified amount of time as in (6).

$$\xi = \sum_{i=1}^{n}\{(\xi_{a_i} \times ST_i) + (\xi_{in_i} \times (ɱ - ST_i))\} \quad (6)$$

This being the case, the paper proposes an algorithm that makes the best possible utilization of the maximum number of available resources in a data center.

TABLE II. SYMBOL TABLE

Symbols	Description
T	A set of task.
T_i	ith task of set T
m	Total number of input task
L_i	Length of ith task
V	A set of VM
VM_i	ith VM of set V
n	Total number of VM
D	A set of deadline
d_i	Deadline of ith task
M_i	VM allocated for ith task
$ɱ$	Makespan of the system
ξ	Total energy consumption of the system
τ	Throughput of the system
P_i	Priority of ith task
$§_i$	Speed of ith VM in MIPS
ξ_a	Energy consumed by the VM in active state
ξ_{in}	Energy consumed by the VM in inactive (idle) state
sr	Stores the VM that has been allocated to ith task
ST	An array of size n to store the schedule time ith VM would take for the processing of allocated task.
LI	Store Lowest Index
c	A counter variable to store number of rejected tasks

E. Algorithm for Scheduling of Tasks

We have considered the cloud system has a sufficient number of hosts. The VM set, V has multiple heterogeneous VMs each of which is hosted by some host in the data center. The task set, T, has heterogeneous resource requirements with different task length. Every task in T has a deadline or the maximum amount of time that it can wait before being processed completely. The proposed algorithm also assumes that if the deadline, d_i, of any ith task is not met, the task, T_i, is never submitted to the broker. Those tasks are strictly rejected due to the lack of time and SLA violation.

Algorithm 1: Proposed Algorithm
Input: V, T, D
Output: M, ɱ, ξ, τ

```
1.  for i←1 to m do
2.      $P_i \leftarrow L_i / d_i$
3.  end
4.  $T \leftarrow$ sort ($T$ using $P_i$)
5.  $c \leftarrow 0$
6.  for i ←1 to $m$ do
7.      for j ←1 to $n$ do
8.          $ST_j \leftarrow ST_j + L_i / §_j$
9.      end
10.     lowest ← Min($ST$)
11.     $LI \leftarrow$ IOM($ST$)
12.         reset $ST$
13.     if lowest <= $d_i$ then
14.         $sr \leftarrow LI$
15.         $M_i \leftarrow sr$
16.         $ST_{sr} \leftarrow ST_{sr} + L_i / §_{sr}$
17.     else do
18.         $c \leftarrow c + 1$
19.         remove $T_i$
20.         remove $d_i$
21.     end
22. end
23. $ɱ \leftarrow$ Max{$ST$}
24. $\tau \leftarrow m - c$
25. for i←1 to $n$ do
26.     $\xi_a \leftarrow 10^{-8} \times §_i^2 \times ST_i$
27.     $\xi_{in} \leftarrow 0.6 \times 10^{-8} \times §_i^2 \times (ɱ - ST_i)$
28.     $\xi \leftarrow \xi + \xi_a + \xi_{in}$
29. end
```

Our algorithm for maximization of throughput and alongside minimization of makespan and energy assumes that VM has already been created in the data center. It takes users' service request or task set, T, and its corresponding deadline (D) as input. For every ith task in T, the algorithm calculates priority factor, P_i, which helps us in determining the priority of a task. Any task, T_i, with higher P_i, is considered to have higher priority and must be attended first. Similarly, a task, T_i, with lower P_i, must be attended after all tasks with higher P_i have been attended.

We maintain a schedule time array, ST, of size n, which stores the running time of all VMs required for the processing of all higher priority tasks that have been allocated to it. For every jth VM in *V*, we update the value of ST_j, signifying the total amount of time jth VM would take for the completion of the ith task. We calculate the minimum value in *ST* (using Min function in step-10 of the algorithm), which indicates the minimum amount of time the data center requires for the processing of the ith task. We also calculate the index of the minimum value of *ST* (using IOM function in step-11 of the algorithm). If this minimum time does not meet the deadline, the task is rejected and consequently removed from the task set *T*. If the minimum time meets the deadline, then the previously updated schedule time array is reset, and only the schedule time of lowest index VM is updated. After every task has been allocated, the array schedule time is consequently updated, and hence the maximum of *ST* gives us the value of makespan. For the calculation of throughput, we maintain a counter (*c*) that counts the total number of rejected task. Hence, throughput is measured using (2) in step-24 of the algorithm. The energy parameter can be calculated using (6) from step-25 to step-29 of the algorithm.

V. PERFORMANCE EVALUATION

We have simulated our proposed algorithm in CloudSim 3.0.3 simulator environment. During the simulation, the cloud resources (VMs), as well as the user requests, are considered heterogeneous. A comparison has been made among our proposed algorithm, First Come First Serve (FCFS), Random, and Modified Round-Robin algorithm. The explanations of all other compared algorithms are as follows.

In FCFS we have allowed the task scheduling on the basis of the order of task input by the users. Every task or service request that is given as input by the user is checked in the VM set (*V*), from least VM speed to the most. If any VM is found to meet the tasks' deadline, the task is allocated to that VM, and the next task is allowed to perform the check operation. In case no VM in *V* is able to meet the deadline of the task, that task is strictly rejected.

For the Random algorithm, we have performed task allocation in order of task input by the user except that instead of searching the VM from least speed to most, we randomly choose a VM from set *V* and check if that VM is able to meet the deadline of the task. If it meets the deadline, then the task is allocated to that VM; else it is checked for another VM in the set *V*. In case all the VMs in *V* have been checked and there is no VM that can meet the deadline of a particular task, then that task is strictly rejected.

In the case of modified Round-Robin, priority factors are determined for every task, and the task set (*T*), is sorted according to its priority, arranging from highest priority factor to lowest. The task set (*T*) is then passed to scheduling algorithm, which takes the help of learning automata to allocate tasks to VMs. A probability array of size n is defined, initially containing equal probabilities for each VM in set *V*, signifying the chance of any VM being allocated to the task. For each task in the arranged set *T*, a VM is selected at random and is checked whether the deadline for that particular task can be met or not. If the deadline is met, that particular VM is rewarded while the other VMs in the set *V* is penalized; else if the deadline is not met, that particular VM is penalized and other VMs are rewarded. This process is continued until the probability of a VM is equal to 1 or there is no VM in the set *V* that can meet the deadline of the task. In case we find a VM suitable for a task, we allocate the task to that VM, and in case there is no VM suitable, the task is strictly rejected.

The simulation for a specific result runs for 20 times and the average of those are shown as the result. We have two separate simulation scenarios to estimate the behavior of all four algorithms including our proposed one as follows.

Scenario-1: In this scenario, the number of tasks is fixed to be 100, and the number of virtual machines varies from 10 to 50 in the interval of 5. The comparison graphs for the calculation of energy consumption, makespan, and throughput are shown in Fig. 4, 6, and 7, respectively. From the figures, it is inferred that the optimization parameters in Y-axis for the three cases is less in the case of our proposed algorithm.

Scenario-2: In the second scenario, the number of VMs is fixed to be 50, and the number of service requests or tasks is varied from 20 to 100 in the interval of 10. The comparison graphs for the calculation of energy consumption, makespan, and throughput are shown in Fig. 3, 5, and 8, respectively. It is observed that the optimization parameters in Y-axis for the three cases are less in the case of our proposed algorithm.

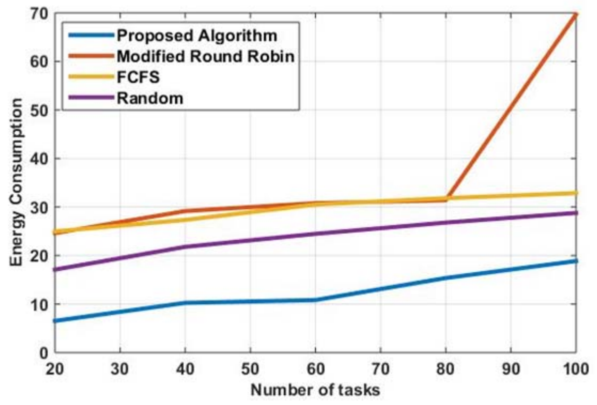

Fig. 3. Comparison graph of algorithms for energy consumption where the number of VM is fixed, and task varies.

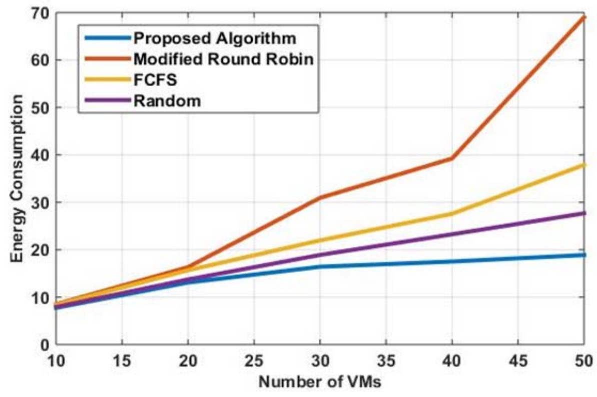

Fig. 4. Comparison graph of algorithms for energy consumption where the number of tasks is fixed and number of VM varies.

From Fig. 3 and 4, it is clearly evident that our proposed algorithm performs the best when it comes to energy consumption, among all three other competing algorithms. This is due to the fact that our algorithm tries to minimize the number of inactive VMs, as it would also consume a considerable amount of energy. Making use of every possible VM in the set *V* makes our proposed algorithm perform better.

For the purpose of minimization of makespan, the proposed algorithm turns out to be a better alternative than the other three of them. This is so because, every task in set *T* is allowed to map with the best performable VM in set *V*, thus making every task time efficient. This approach of ours' thus helps us in reducing the makespan.

When throughput is the matter of concern, our algorithm does not prove itself to be the best among the three, but certainly, has the capability of achieving around 90% of maximum throughput. This lag can be acceptable, because of the alongside optimization of energy consumption and makespan of the system. Moreover, it is clearly understood the fact that if the deadline of the rejected tasks is increased, then the throughput of the algorithm can also be increased. Up to 80 numbers of tasks, the number of VMs or cloud resources is enough for all algorithms as shown in Fig 8.

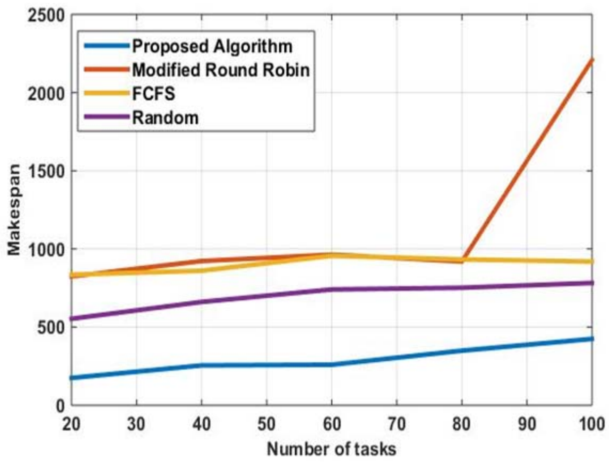

Fig. 5. Comparison graph of algorithms for makespan where the number of VM is fixed, and task varies.

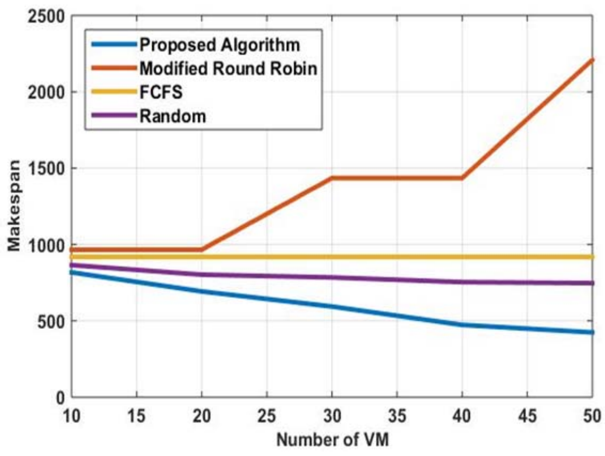

Fig. 6. Comparison graph of algorithms for makespan where the number of tasks is fixed, and number of VM varies.

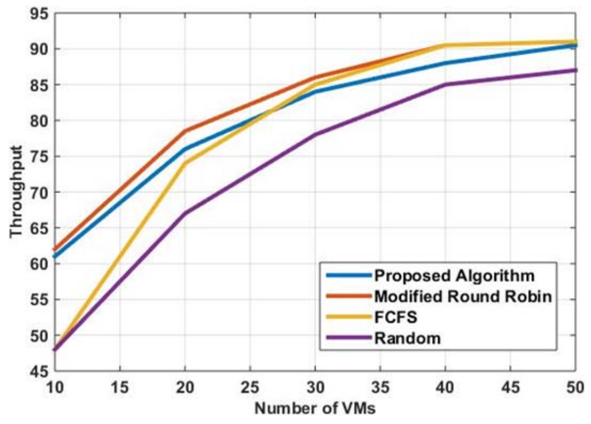

Fig. 7. Comparison graph of algorithms for throughput where the number of tasks is fixed, and number of VM varies.

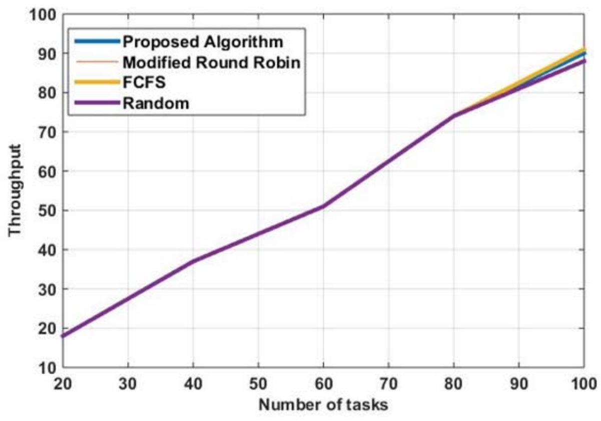

Fig. 8. Comparison graph of algorithms for throughput where the number of VM is fixed, and task varies.

VI. CONCLUSION

To maximize the benefits of the service providers, it is necessary to reduce the energy consumption of the system without any degradation in the QoS. The optimal allocation of heterogeneous tasks to computing resources is one of the most important and basic problems to optimize various parameters in cloud computing. Based on the earlier works on task consolidation, this paper introduces a new model considering the task priorities and after that suggests an efficient algorithm. The simulation results show that our proposed algorithm is significantly better in performance metrics such as energy, makespan, and throughput of the system over the other competing algorithms. The derived results of this paper will be valuable in dimensioning of cloud systems. Future work incorporates an intensive study on dynamic task consolidation in cloud system considering all other resources like main memory, bandwidth, and others.

REFERENCES

[1] Data Center Knowledge: http://www.datacenterknowledge.com/archives/2016/06/27/heres-how-much-energy-all-us-data-centers-consume/ 2016.

[2] A. Kansal, F. Zhao, J. Liu, N. Kothari, A. Bhattacharya, "Virtual Machine Power Metering and Provisioning," Proceedings of the 1st ACM symposium on Cloud computing, pp. 39-50, 2010.

[3] 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, pp. 14-27, 2016.

[4] B. Keshanchi, & N. J. Navimipour, "Priority-Based Task Scheduling in the Cloud Systems Using a Memetic Algorithm," Journal of Circuits, Systems and Computers, 25(10), pp. 1650119, 2016.

[5] A. V. Lakra, & D. K. Yadav, "Multi-objective tasks scheduling algorithm for cloud computing throughput optimization," Procedia Computer Science, 48, pp. 107-113, 2015.

[6] S. Yassa, R. Chelouah, H. Kadima, & B. Granado, "Multi-objective approach for energy-aware workflow scheduling in cloud computing environments," The Scientific World Journal, 2013.

[7] Z. Cao, & S. Dong, "An energy-aware heuristic framework for virtual machine consolidation in Cloud computing," The Journal of Supercomputing, 69(1), pp. 429-451, 2014.

[8] R. L. Cunha, E. R. Rodrigues, L. P. Tizzei, & M. A. Netto, "Job placement advisor based on turnaround predictions for HPC hybrid clouds," Future Generation Computer Systems, 67, pp. 35-46, 2017.

[9] H. Chen, X. Zhu, D. Qiu, & L. Liu, "Uncertainty-Aware Real-Time Workflow Scheduling in the Cloud," In IEEE 9th International Conference on Cloud Computing (CLOUD), pp. 577-584, 2016.

[10] S. K. Panda, & P. K. Jana, "A multi-objective task scheduling algorithm for heterogeneous multi-cloud environment," In IEEE International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV), pp. 82-87, 2015, January.

[11] F. Zhang, J. Cao, K. Li, S. U. Khan, & K. Hwang, "Multi-objective scheduling of many tasks in cloud platforms," Future Generation Computer Systems, 37, pp. 309-320, 2014.

[12] J. J. Durillo, V. Nae, & R. Prodan, "Multi-objective workflow scheduling: An analysis of the energy efficiency and makespan tradeoff," In 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 203-210, (2013, May).

[13] J. Mei, & K. Li, "Energy-Aware Scheduling Algorithm with Duplication on Heterogeneous Computing Systems," ACM/IEEE 13th International Conference on Grid Computing, pp. 122-129, 2012.

[14] K. B. Bey, F. Benhammadi, A. Mokhtari, & Z. Guessoum, "Independent task scheduling in heterogeneous environment via makespan refinery approach," International Conference on Machine and Web Intelligence, pp. 211-217, 2010.

[15] E. Grochowski, & M. Annavaram, "Energy per instruction trends in Intel microprocessors," Technology@ Intel Magazine, 4(3), pp. 1-8, 2006.

[16] T. Shi, M. Yang, X. Li, Q. Lei, & Y. Jiang, "An energy-efficient scheduling scheme for time-constrained tasks in local mobile clouds," Pervasive and Mobile Computing, 27, pp. 90-105, 2016.

[17] S. M. Sampaio, J. G. Barbosa, & R. Prodan, PIASA: "A power and interference aware resource management strategy for heterogeneous workloads in cloud data centers," Simulation Modelling Practice and Theory, 57, pp. 142-160, 2015.

[18] D. Puthal, B. Sahoo, S. Mishra, and S. Swain. "Cloud computing features, issues, and challenges: a big picture." In Computational Intelligence and Networks (CINE), 2015 International Conference on, pp. 116-123, 2015.