Time Efficient Task Allocation in Cloud Computing Environment

Sambit Kumar Mishra
Department of Computer Science & Engineering
National Institute of Technology
Rourkela, India
Email: skmishra.nitrkl@gmail.com

Md Akram Khan
Department of Computer Science & Engineering
National Institute of Technology
Rourkela, India
Email: akram.sgsits@gmail.com

Bibhudatta Sahoo
Department of Computer Science & Engineering
National Institute of Technology
Rourkela, India
Email: bibhudatta.sahoo@gmail.com

Sanjay Kumar Jena
Department of Computer Science & Engineering
National Institute of Technology
Rourkela, India
Email: skjena@nitrkl.ac.in

***Abstract*—Cloud computing is an evolution of Distributed system that has been adopted by worldwide scientifically and commercially. For optimal use of cloud's potential power, effective and efficient algorithm are required, which will select best resources from available cloud resources for different applications. This allocation of user requests to the cloud resource can optimize various parameters like energy consumption, makespan, throughput, etc. This task allocation or mapping problem is a well-known NP-Complete problem. In this paper, we have proposed an algorithm, Task Based allocation to minimize the makespan of the cloud system and also to increase the resource utilization. We have simulated our algorithm, TBA in CloudSim Simulator in a heterogeneous environment. CloudSim is one of the simulation tools of cloud environment which provides evaluation and testing of cloud services and infrastructure before the development of the real world. During the comparison of the algorithm, we provide the sorted tasks to the TBA algorithm once and un-sorted tasks in the second time. We have compared sorted-TBA, unsorted-TBA and random algorithm where the sorted-TBA algorithm performs better.**

***Keywords*—** Cloud Computing, Makespan, PM, Task Scheduling, TBA algorithm, VM.

I. Introduction

Cloud computing is an evolution of Grid Computing. In the current scenario, Cloud computing is a buzzword and getting more and more attention from users. Cloud computing provides a wide pool of shareable resources (like physical resources: CPU, Memory, Storage, Workstations, etc. logical resources: Operating System, Energy, Network throughput/bandwidth, Network loads and delays etc.) which delivers scalable on-demand resources as a service over the Internet with the help of virtualization technique. In a simple way, Cloud is a collection of parallel and distributed system which is interconnected and virtualized. These virtualized resources allocated to the consumer according to their respective Service Level Agreement (SLA) between consumer and supplier. These services can be PaaS (Platform-as-a-service), IaaS (Infrastructure-as-a-service) or SaaS (Software-as-a-service). The virualization technique virtualize the actual resources of the physical hosts in the form of virtual machines (VMs).

In Technology world, every user wants to get their services in few fraction of the time. Therefore, the task allocation plays an important role [1]. In a competitive world, the number of users working over the Internet is much more and increasing with day-by-day. To provide services to these large number of users is a challenging task and one of the best solutions is Cloud Computing [2], [3]. Task allocation is a combinatorial optimization problem in the fields computer science. Allocation of the task in cloud computing is an NP-Hard optimization problem [1]. The main aim of task allocation is to distribute the tasks for cloud resources in such a way that it will approach to minimize makespan, minimize the execution cost and maximize the utilization of resources. In this paper, we are concentrating on minimization of makespan and maximization of utilization of resources.

Resource allocation algorithm is of two kinds: static and dynamic [4]. In dynamic allocation algorithm, cloud resources may be requested by the cloud user meanwhile running the applications. Here, under-utilization and over-utilization of resources are avoided as much as possible. But, in the static allocation algorithm, the cloud user has to make a prior request for the cloud resources (VMs). It leads to under-utilization or over-utilization of resources depending on the time the application is run. Here, elasticity and the on-demand feature will be not consider. The resource management in the cloud environment will performed with the help of virtualization technique [5].

Contributions:

- Design a system model for cloud environment.
- Desing an algorithm sorted TBA (Task Based Allocation) to reduce the makespan.
- Simulation between available no of algorithm.

The main aim of writing this paper is to propose a heuristic algorithm for minimizing the makespan and maximizing the

utilization of CPU-time. We are considering resources are heterogeneous in nature. The rest of the paper is organized as follows. Section 2 outlines an overview of some related work, Section 3 describes the cloud computing architecture, Section 4 describes a brief idea about generic cloud system model which includes physical machine (PM) model, virtual machine (VM) model and task model. The section 5 describes our proposed work to model the heterogeneous computing environment. In Section 6, explains about simulation and results, utilization of VMs and effectiveness of our algorithm. Section 7, concludes the paper.

II. Related Works

In [2], authors mainly focus on preamble task. The author proposed an adaptive resource allocation algorithm which adjusts the cloud resources adaptively on the basis of actual task executions update. For this, they are using two algorithms: ALS (Adaptive List Scheduling) and AMMS (Adaptive Min-Min Scheduling) algorithm which uses static task scheduling paradigm to allocate the cloud resources statically. The online adaptive procedure is also used for re-evaluating the remaining task execution and update the task execution list. The author proposed another algorithm PBSA (Priority Based Scheduling Algorithm) giving an advance reservation of higher priority task over lower priority task. If lower priority process is under execution and a higher process is coming to execute, then this case it will pre-empt the lower priority process and give the chance to higher priority process to execute. During the pre-emption it will also consider the multiple SLA parameter i.e. network bandwidth, required CPU time, required main memory spaces. The real-time task execution in the cloud environment [9] by using MapReduce framework along with the system model for scheduling of tasks is discussed in [8].

Cloud computing is gaining lots of interest in several domains such as disaster management, health-care, military [10], [11], [12]. Still there is a scope to contribute security solutions to protect data in cloud and intermediate edge data center [10]. In current era data turns to a new term as big data, which need a unique solutions to support 4Vs properties to process big data in cloud [11], [12].

In [7], dynamic resource allocation on various distributed multiple criteria is used. The author proposed a PROMETHEE method, that is used to take the follow decisions:
(1) VM placement: Aim is to find the best suitable Physical machine (PM) which is capable of hosting particular Virtual machine can run without interfering, try to assign that VM to PM.
(2) Monitoring, in this phase algorithm, has to monitor total utilization of resources by hosted VM.
(3) VM Selection: if the current physical machine is not capable of running particular VM then try to migrate that VM to some another Physical machine.

In [13], Tiwari et al. have proposed a reasonable pricing strategy for cloudlets-services for mobile cloud users and also tried to optimize the computational tasks fired by the mobile cloud users. In [14], the objective is to maximize the total expected profit with the help of considering SLA-multidimensional resource allocation scheme for multi-tier services. Basically, the author is considering three dimensions of a server :(1) Processing power, (2) Memory usage, (3) Bandwidth. Here author is giving more concentration on SLA; they are defining two types of SLA:
(1) Gold SLA: response time is guaranteed and if cloud server provider is violating the constraint then they have to pay penalty.
(2) Bronze SLA: some limited amount of time response time may be the delay, then this case no penalty has to pay.

In [15], the author introduces the concept of "skewness" which is to identify the unevenness utilization of a server. Authors main aim is to minimize the skewness with the help of this we can maximize the utilization of a server. They designed a Load prediction algorithm, which is used to identify the current workload of a VM. The main aim of authors is to achieve the following two things:
(1) Overload Avoidance: Load prediction algorithm is used to identify the current workload of Physical machines. If any physical machine is going to overloaded, then try to migrate that VM to another available of PMs.
(2) Green Computing: if any physical machine is under loaded then try to transfer that VM, is running on that PM, to some another PM and turned off that PM, with the help of this number of PM is minimized.

III. Cloud Computing Architecture

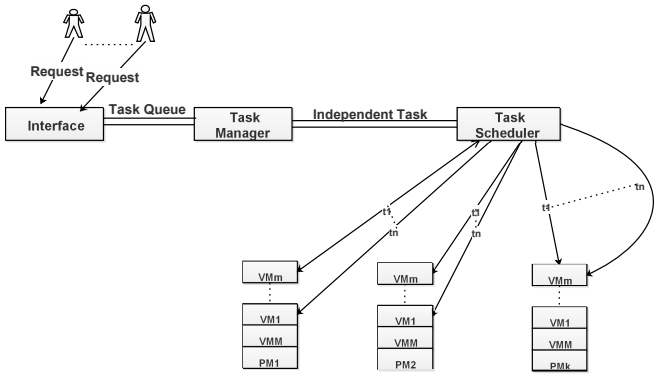

Fig. 1. Cloud Computing Architecture

In this section, we are going to explain the cloud computing architecture with the help of Fig. 1. This model will consist of completely interconnected set of resources. These resources are physical machines (PMs) (i.e., $\{PM_1, PM_2, ..., PM_k\}$), memory, CPU time, and network bandwidth, etc. Each PM has a VMM (Virtual Machine Manager). Over the VMM a finite number of the virtual machines (i.e., $\{VM_1, VM_2, ..., VM_m\}$) will run.
The cloud system model will consist the following module:

1) Task Manager
2) Task Scheduler

Interface nothing but a GUI (graphical user interface), a web-page where cloud user can give their requests. These tasks

TABLE I
I. RELATED WORK

Author (Year)	Objective	VM Environment	VM Allocation	Resource Consideration
Krishna et al. (2013) [1]	To reduce the load of the overloaded VMs considering priority also	Heterogeneous	Dynamic	CPU
Chitra et al. (2016) [2]	Objective is to assign the computational tasks to the most suitable virtual machines from the dynamic pool of the VMs by considering the requirements of each task and the load of the VMs.	Homogeneous	Dynamic & Static	CPU
Chandrasekhar et al. (2012) [6]	To reduce the makespan of VMs with the help of Priority Based schedulling algorithm(PBSA)	Homogeneous	Dynamic	CPU
Goudarzi et al. (2011) [14]	objective is to maximize the total expected profit with the help of considering SLA- multidimensional resource allocation scheme for multi tier services	Homogeneous	Dynamic	CPU, Memory,Network Bandwidth
Song et al. (2015) [16]	To reduce the load of the overloaded hosts and decreasing the communication cost among hosts	Homogeneous	Dynamic	CPU
Ni et al. (2011) [17]	VM mapping algorithm based on multi-resource load balancing and aimed to easing load crowding.	Homogeneous	Static	CPU & Memory
Yang et al. (2011) [18]	To avoid the unnecessary costs caused by the instantaneous peak of resource utilization.	Heterogeneous	Dynamic	CPU
Jonathan et al. (2010) [19]	Its objectives are reducing the load on a single host, moving a VM to a new host with more resources or with specialized resources	Heterogeneous	Dynamic	CPU

will submitted to the Task Queue.

Task Manager: It will take a task from Task Queue, and the Task Queue is nothing but a priority Queue. Here, we are giving higher priority to higher task length or task size. Our idea is to give the higher length of the task to the high speed of VM, so that it will take less time to execute, so somehow it will reduce the makespan.

Task Scheduler: It is the core module of our system. Its main responsibility is to allocate available VMs for the input tasks [24]. For minimizing the makespan, here we are applying some heuristic techniques to allocate the task to available VMs. For allocating the task to VMs, we are calling sorted TBA(Task Based Allocation algorithm) to find suitable available VM to the execution of task so that it will take less time to execute the task. After execution of the task, it is the responsibility of the task scheduler to release the VMs and corresponding resources, and updates the available number of VMs and resources.

IV. PROBLEM STATEMENT

Allocation of tasks in the cloud is an NP-hard optimization problem [1]. Load balancing of non-preemptive independent tasks on VMs is an important aspect of task allocation in the cloud [6]. The main objective of our work is to balance the load and speed up the execution of applications or minimizes the makespan of the VMs. Makespan is the required time to complete all tasks.

makespan=max$\{$ CT_{ij} $i\varepsilon T$, $i = 1, 2, ..., n$ and $j\varepsilon VM$, $j = 1, 2, ..., m\}$

Objective Function = min$\{$makespan$\}$

We define our problem is a 3-tuple

S=$\{$PM, VM, TM$\}$

to represent the problem. PM is a set of Physical machines. VM is a set of virtual machines. TM is the task model that consists task-id,task-length and task-type.

Our first contribution in this paper is to proposed a generalized system model which contains PM model, VM model and task model. It is a generic model which can be used in different scenario depends on the application requirements. The PM contains all the resources i.e. network resources, computational resources, memory requirements. The problem of minimizing the makespan where the resource utilization should be maximum. The problem is containing multiple objectives: (1) Minimize the makespan, (2) Maximize the resource utilization (3) Increase the throughput. Our prime objective is to minimize the makespan by applying sorted TBA (Task Based Allocation) heuristic algorithm. The proposed heuristic algorithm also tries to maximize the resource utilization, memory requirements and VMs requirements, etc. which may require during the execution of tasks [20]. The System model is explained as below:

A. Physical Machine Model

A PM has its own hardware: Memory, network, processing and storage resources. On this hardware, the VMM is loaded [20]. It is a 6-tuples entity i.e.

$PM_k = \{ID_k, CPU_k, MM_k, SS_k, BW_k, VMM_k\}$ Where,

- ID_K is the identification of k^{th} physical machine
- CPU_K is the computation processing speed (in MIPS)of k^{th} physical machine
- MM_K is the capacity of main memory of k^{th} physical machine
- SS_K is the capacity of secondary storage of k^{th} physical machine
- BW_K is the bandwidth capacity of k^{th} physical machine
- VMM_K is the Virtual Machine Monitor(VMM) running on the k^{th} physical machine

B. Virtual Machine Model

We have m number of virtual machines running on i^{th} physical machine [20], $VM_i = \{VM_{i1}, VM_{i2}, ..., VM_{im}\}$ i.e.

A virtual machine can be modeled as 5-tuples,

$VM_{ij} = \{ID_{ij}, CPU_{ij}, MM_{ij}, SS_{ij}, BW_{ij}\}$ Where ,

- ID_{ij} is the identification of j^{th} virtual machine running on i^{th} PM.

- CPU_{ij} is the computation processing speed (in MIPS)of j^{th} virtual machine running on i^{th} PM
- MM_{ij} is the capacity of main memory of j^{th} VM on PM_i
- SS_{ij} is the capacity of secondary storage of i^{th} physical machine
- BW_{ij} is the bandwidth capacity of j^{th} VM on PM_i

C. Task Model

A job is a collection of task in the cloud [20]. A task in the cloud is a service request which the cloud has to provide. We have n number of independent computing tasks running on some virtual machines $T_{ij}^k = \{T_{ij}^1, T_{ij}^2, ..., T_{ij}^n\}$ i.e.
A task model can be modeled as 5-tuples,
$T_{ij}^k = \{ID_{ij}^k, L_{ij}^k, R_{ij}^k\}$ Where ,

- ID_{ij}^k is the identification of k^{th} task running on VM_{ij}.
- L_{ij}^k is the length of the k^{th} task running on VM_{ij}.
- R_{ij}^k represents the service-type $0 \rightarrow CPU - intensive, 1 \rightarrow Data - intensive, 2 \rightarrow Communication - intensive.$ running on VM_{ij}

V. Used Techniques

A. Task Based Allocation algorithm

As mentioned earlier, our aim is to minimize the makespan and maximize the resource utilization as it possible. The key of this is to give the maximum task-length task to the higher speed of VMs so that it will take less time to execute. It is obvious if we are assigning higher task length to higher speed VMs then it will take less time. For fetching higher task-length form queue, we are storing the task into a Max-Heap data structure. It will return higher task length first, then we are calculating ETC (Expected Time to Complete) matrix by calling the ETC_generation algorithm. Since, we are allocating the task dynamically, so we have to consider current waiting time of VMs because at that time that particular VMs also processing some another task. For this, we called sorted TBA (Task Based Allocation) algorithm. Here, we have to pass ETC, $VM\_waiting$. sorted TBA algorithm considers the ETC as well as $VM\_waiting$ time then it will calculate ETC matrix. TBA algorithm returns the makespan.

Algorithm 1: ETC_generation algorithm

Input: $No\_of\_VMs, List\_of\_VMs\_Speed, No\_of\_Task, List\_of\_Task\_Size$
Output: ETC
begin
  for $i = 0$ *to* $(No\_of\_Task - 1)$ **do**
    for $j = 0$ *to* $(No\_of\_VMs - 1)$ **do**
      $ETC[i][j] =$ List_of_Task_Size[i]/List_of_VMs_Speed[j];
    end
  end
  return ETC
end

Algorithm 2: Sorted Task Based Allocation (TBA) algorithm

Input: $ETC\_matrix, VM\_waiting\_list, No\_of\_Task,$ No_of_VMs
Output: $makespan$
begin
  for $i = 0$ *to* $(No\_of\_Task - 1)$ **do**
    $ind = 1, max = \infty, makespan = 1;$
    for $j = 0$ *to* $(No\_of\_VMs - 1)$ **do**
      if $wttime\_of\_VM + ETC[i][j] < max$ **then**
        $ind = index\_of\_currentVM;$
        max=wttime_of_VM + ETC[i][j]
      end
    end
    $task_assigned[i] = ind;$
    $wttime\_of\_VM+ = ETC[i][ind]$;
  end
  return $makespan = max\{wttime\_of\_VM\}$
end

B. Flowchart

Fig. 2 shows the flow chart of our proposed algorithm. Firstly, it will accept the task, and then, it will store into the Max-Heap queue. Task manager picks the task from the queue and it will send the task to the task Scheduler. Task scheduler check the availability of VMs in the system. If VM is not available then create the new VM instance. If VM is available then choose appropriate VM and assigned that task to that VM. After execution of the task, it is the responsibility of the task scheduler to release the VMs and corresponding resources, updates the available no of VMs and resources.

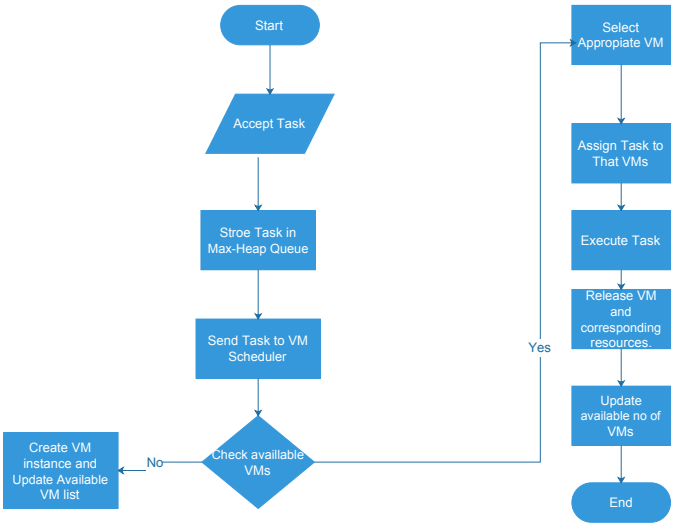

Fig. 2. Flowchart

C. Task Allocation

In Table II, we have services with their sizes, there are total ten tasks. In Table III, VMs with their corresponding

speed, there are total 4-VMs. In Table -IV shows the result of ETC_generation algorithm, it will store in the form of a matrix and the size of the matrix would be No_of_Task X No_of_VM, so, here matrix size would be 10×4. Then, we call sorted TBA algorithm and it will allocate the task to VM shows in Table V. Since makespan is a maximum time to complete the whole task, so here makespan is 2.255 by VM3.

TABLE II
SERVICES WITH THEIR SIZE

Task Id	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9
Task Size (MI)	1000	3000	2000	1500	2200	1200	1400	5000	1900	4000

TABLE III
VIRTUAL MACHINES WITH THEIR SPEED

VM Id	VM0	VM1	VM2	VM3
VM Speed (MIPS)	1000	2000	3000	4000

MI : million instructions.
MIPS :Million Instructions per Second.
The ETC matrix of size 10X4 matrix is as follows

TABLE IV
ETC MATRIX

Task Length/VM	VM0	VM1	VM2	VM3
	1000	2000	3000	4000
1000	1.0	0.5	0.33	0.25
3000	3.0	1.5	1.0	0.75
2000	2.0	1.0	.66	0.5
1500	1.5	0.75	0.5	0.375
2200	2.2	1.1	.73	.55
1400	1.4	0.7	.466	.35
1900	1.9	0.95	.633	.475
1200	1.2	.6	0.4	0.3
5000	5.0	2.5	1.66	1.25
4000	4.0	2.0	1.33	1.0

ETC : Expected Time to Complete.
Task Based Allocation (TBA) allocates the task to VM following way

TABLE V
ALLOCATION OF TASK TO VMS BY SORTED TBA ALGORITHM

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9
VM3	VM2	VM1	VM3	VM0	VM2	VM3	VM1	VM2	VM3

Makespan in Task Based Allocation : 2.425

VI. SIMULATION RESULTS

The simulation is performed on the cloud computing environment tools: CloudSim [21]. In CloudSim, there is two way of execution of a task on the VM or VMs on the PMs. They are Time-Shared and Space-Shared allocation. The space-shared policy allows the multiple tasks to execute on VMs or VMs to PMs simultaneously and they are isolated with the help available spaces. The Time-Shared policy allows multiple tasks to be executed on VMs or VMs on PMs to be multi-task and run simultaneously. One data center is created with default properties as it mentioned by the CloudSim designer. All PMs are running on one data center. To simulate our algorithm, we have some assumptions as follows:

1) Number of VMs will be proportion to a number of the available core in a PM.
2) PMs are heterogeneous.
3) Every PMs have an m-finite number of VMs.
4) VMs are heterogeneous.
5) Tasks are non-preemptive.
6) Each Task is independent in nature.
7) Resource requirement of each task is independent of each other.

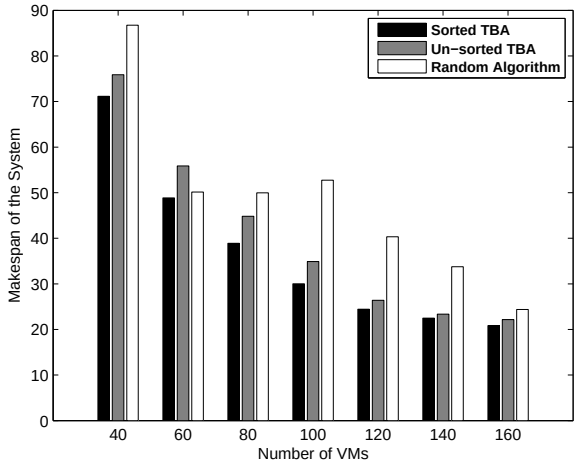

Fig. 3. A bar chart to compare makespan value for Simulation-1

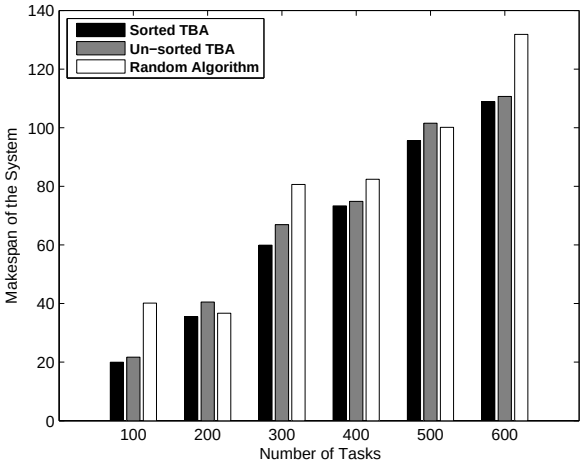

Fig. 4. A bar chart to compare makespan value for Simulation-2

In Simulation-1, we are fixing the number of tasks with 400 and varying the number of VMs from 40 up to 160. We have plotted the bar-chart between makespan and the number of VMs. From Fig. 3, it is shown that our proposed algorithm sorted TBA with sorted task gives minimum makespan.

In simulation-2, we are fixing VMs with 80 and varying the number of tasks from 100 up to 600. We have plotted the bar-chart between makespan and the number of tasks. From Fig. 4, it is shown that our proposed algorithm TBA with sorted task gives minimum makespan.

VII. Conclusion

In this paper, we have studied various task scheduling approaches in homogeneous and heterogeneous cloud environments proposed by different researchers. We have proposed a task based heuristic algorithm for dynamic task allocation in heterogeneous cloud computing environment. We have introduced the system model consists of PM model, VM model, and task model. We have used the ETC (Expected Time to Complete) [22] to implement TBA. The proposed sorted TBA algorithm, giving higher priority to higher task-size, but there is no concept of preemption, and the simulation shows that sorted TBA algorithm has improved makespan.

References

[1] P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments." *Applied Soft Computing*, pp. 2292-2303, 2013.

[2] C. Devi, and V. Uthariaraj, "Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Non preemptive Dependent Tasks." *The Scientific World Journal*, 2016.

[3] S. Kumar, and R. H. Goudar, "Cloud Computing-Research Issues, Challenges, Architecture, Platforms and Applications: A Survey." *International Journal of Future Computer and Communication*, 1(4), pp. 356, 2012.

[4] S. K. Mishra, B. Sahoo, K. S. Sahoo, S. K. Jena, "Metaheuristic Approaches to Task Consolidation Problem in the Cloud." *Resource Management and Efficiency in Cloud Computing Environments*, pp. 168-189, 2016.

[5] K. S. Sahoo, B. Sahoo, R. Dash, M. Tiwary, and S. Sahoo, "Network Virtualization: Network Resource Management in Cloud." *Resource Management and Efficiency in Cloud Computing Environments*, pp. 239, 2016.

[6] Chandrashekhar P., and Rajnikant B., "Priority based dynamic resource allocation in cloud computing." Cloud and Services Computing (ISCOS), *2012 International Symposium on. IEEE*, 2012.

[7] Y. O. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, and Y. Coady, "Dynamic resource allocation based on distributed multiple criteria decisions in computing cloud." in *3rd International Conference on Cloud Computing*, pp. 91-98, 2010.

[8] S. Sahoo, B. Sahoo, A. K. Turuk, S. K. Mishra, "Real Time Task Execution in Cloud Using MapReduce Framework." *Resource Management and Efficiency in Cloud Computing Environments*, pp. 190, 2016.

[9] S. Sahoo, S. Nawaz, S. K. Mishra, and B. Sahoo, "Execution of real time task on cloud environment." *In 2015 Annual IEEE India Conference (INDICON)*, pp. 1-5, December 2015.

[10] D. Puthal, S. Nepal, R. Ranjan, and J. Chen, "Threats to Networking Cloud and Edge Datacenters in the Internet of Things." *IEEE Cloud Computing*, Vol. 3(3), pp. 64-71, 2016.

[11] D. Puthal, S. Nepal, R. Ranjan, and J. Chen, "A Secure Big Data Stream Analytics Framework for Disaster Management on the Cloud." *In 18th IEEE International Conferences on High Performance Computing and Communications*, pp. 1218-1225, 2016.

[12] D. Puthal, S. Nepal, R. Ranjan, and J. Chen, "DLSeF: A Dynamic Key Length based Efficient Real-Time Security Verification Model for Big Data Stream." *ACM Transactions on Embedded Computing Systems*, Vol, 16(2), pp.51, 2016.

[13] M. Tiwary, K. S. Sahoo, B. Sahoo, and R. Misra, "CPS: a dynamic and distributed pricing policy in cyber foraging systems for fixed state cloudlets." *Computing*, pp. 1-17, 2016.

[14] H. Goudarzi and M. Pedram, Multi-dimensional SLA-based Resource Allocation for Multi- tier Cloud Computing Systems, in *IEEE International Conference on Cloud Computing*, pp. 324- 331, September 2011.

[15] Z. Xiao, S. Weijia, and C. Qi, "Dynamic resource allocation using virtual machines for cloud computing environment." *IEEE Transactions on parallel and distributed systems*, pp. 1107-1117, 2013.

[16] X. Song,M. Yaofei, and T. Da, "A load balancing scheme using federate migration based on virtual machines for cloud simulations." *Mathematical Problems in Engineering*, 2015.

[17] J. Ni, Y. Huang, Z. Luan, J. Zhang, and D. Qian, "Virtual machine mapping policy based on load balancing in private cloud environment." Cloud and Services Computing (ISCOS), *International Conference on Cloud and Service Computing (CSC)*, pp. 292-295, 2011.

[18] K. Yang, J. Gu, T. Zhao, and G. Sun, "An optimized control strategy for load balancing based on live migration of virtual machine." Services Computing (SCC),*Sixth Annual ChinaGrid Conference. IEEE*, pp. 141-146, 2011.

[19] J. R. Cornabas, "A distributed and collaborative dynamic load balancer for virtual machine." *European Conference on Parallel Processing. Springer Berlin Heidelberg*, pp. 641-648, 2010.

[20] S. K. Mishra, R. Deswal, S. Sahoo, and B. Sahoo, "Improving energy consumption in cloud," *In 2015 Annual IEEE India Conference (INDICON)*, pp. 1-6, December 2015.

[21] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities." *In IEEE International Conference on High Performance Computing Simulation, 2009. HPCS'09*, pp. 1-11, 2009.

[22] A. Shoukat, H. J. Siegel, M. Maheswaran, and D. Hensgen, "Task execution time modeling for heterogeneous computing systems," *In Heterogeneous Computing Workshop, 2000.(HCW 2000)*, pp. 185-199, 2000.

[23] R. Buyya, R. Rajiv, and Rodrigo N., "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services." *International Conference on Algorithms and Architectures for Parallel Processing, Springer Berlin Heidelberg*, pp. 13-31, 2010.

[24] S. K. Mishra, P. P. Parida, S. Sahoo, B. Sahoo, and S. K. Jena, "Improving Energy Usage in Cloud Computing Using DVFS", *International Conference on Advance Computing and Intelligent Engineering (ICACIE 2016)*, Springer, 2016.

[25] I. Foster, Y. Zhao, and S. Lu, Cloud Computing and Grid Computing 360-degree compared, *in proc. Grid Computing Environments Workshop*, pp. 99-106, 2008.

[26] D. Grosu, A. Chronopoulos, and M. Leung, "Cooperative load balancing in distributed systems," *in Concurrency and Computation: Practice and Experience*, pp: 1953-1976, 2008.

[27] L. Heng, C. Haopeng, M. Sixiang, and D. Wenyun, "Dynamic Virtual Resource Management in Clouds Coping with Traffic Burst." Services Computing (SCC),*International Conference on. IEEE*, pp. 590-596, 2014.