Efficient Idle Virtual Machine Management for Heterogeneous Cloud using Common Deployment Model

C.Saravanakumar[1], C.Arun[2]
[1] Research scholar, Sathyabama University, Faculty of Computer Science and Engineering, St. Joseph's Institute of Technology, OMR, Chennai, Tamil Nadu, India
[e-mail: mailofcsk@gmail.com]
[2] Professor, Department of ECE, R.M.K College of Engineering and Technology, Chennai, Tamil Nadu, India
[e-mail: arun.c@rmkcet.ac.in]
*Corresponding author: C.Saravanakumar

Received September 30, 2015; revised December 15, 2015; revised January 20, 2016; accepted February 9, 2016; published April 30, 2016

Abstract

This paper presents an effective management of VM (Virtual Machine) for heterogeneous cloud using Common Deployment Model (CDM) brokering mechanism. The effective utilization of VM is achieved by means of task scheduling with VM placement technique. The placements of VM for the physical machine are analyzed with respect to execution time of the task. The idle time of the VM is utilized productively in order to improve the performance. The VMs are also scheduled to maintain the state of the current VM after the task completion. CDM based algorithm maintains two directories namely Active Directory (AD) and Passive Directory (PD). These directories maintain VM with proper configuration mapping of the physical machines to perform two operations namely VM migration and VM roll back. VM migration operation is performed from AD to PD whereas VM roll back operation is performed from PD to AD. The main objectives of the proposed algorithm is to manage the VM's idle time effectively and to maximize the utilization of resources at the data center. The VM placement and VM scheduling algorithms are analyzed in various dimensions of the cloud and the results are compared with iCanCloud model.

Keywords: Common Deployment Model, CloudSched, VM Placement, VM Scheduling, iCanCloud

1. Introduction

Virtualization is a technique used to create virtual resources such as CPU, storage, network, OS, file, memory which are consumed by the end user as a service through VM. VM is an emulation of a physical computer which runs OS and applications based on architecture and functionalities [1]. It is a software computer that runs the physical resources in a host with virtual devices to provide functionalities like portability, security, efficiency and reliability. The VM Monitor supports the execution environment and manages the VM resources such as policy-based automation, virtual hard disk, life cycle management, live migration and real-time resource allocation [2]. There are two types of architectures in VM management namely type I and type II. Type I architecture is used to establish the VM communication and hardware whereas type II architecture is used to run the VM on host operating system.

Heterogeneous transportation data are collected for processing of various traffic sensors and to minimize the high cost computation processing of massive transportation data using parallelized fusion on multisensor transportation data [3]. Generic methodological framework for cyber-ITS is used to transform the data analytics requirement in Intelligent Transportation Systems (ITS) by performing the functions such as tasks scheduling, data centric and operation centric transformation by using High Performance Computing (HPC) architecture [4]. The computing tasks are created based on the data pieces collected from Global Position System (GPS). It performs the parallel map-matching function for measuring the projected points and selecting the shortest distance with high efficiency and accuracy [5].

2. Related Work

An overshadow technique has been introduced in order to protect the page in the memory, but it is not suitable to protect the virtual CPU state [6]. VM measuring framework detects and supports VMM's rollback mechanism over a single physical machine which is not suitable for multiple physical machines in hybrid cloud environment [7]. Traditional snapshot techniques are applied directly to the virtuafl cloud storage, so they are inefficient to provide the trustworthiness. This will be achieved through SNPdisk (snapshot disk) and it is a para-virtualization snapshot mechanism with sparse tree and a snapshot data file [8].

Virtual machine consolidation approach is used to collect inactive physical servers to make an active physical server with a minimum number. An Energy-Efficient VM consolidation supports the VM live migration across various VM monitors with the parameters like VM's CPU state, main memory, and network connections etc [9]. CoTuner is a coordinated auto-configuration framework with two agents namely VM-Agent and App-Agent. These are used to configure the VM and its applications are maintained in the VM clusters [10]. In this technique, the VMs from VM clusters are configured coordinately which leads to the performance and synchronization problem. The host overhead problem has fixed by using detection algorithm. It also improves the quality of VM consolidations and maximizes the time interval between VM migrations and overloaded hosts [11]. It takes large searching and allocation time during migration of VM to overloaded host.

Usher is a framework with set of plugins used to collect the usage statistics of resources such as CPU, memory and Network within VM. The adjustment of memory allocation is handled by Working Set Prober (WS Prober) on each hypervisor [12]. CloudSched model has been used for allocating the VM with physical machines on cloud. This algorithm lacks support in simulation environment with multi dimensional resource utilization. This process can be addressed by extending the support to multiple federated data centers in various dimensions with the help of interoperability. A user priority is needed for an assignment of VM in order to achieve effective utilization [13].

The existing VM placement and VM scheduling algorithms are suitable only for placing and scheduling the physical machine to the VM without any consideration of the performance with respect to task types, but it is related to the task execution. The existing iCanCloud model manages the idle VM in the VM repository but endure with a performance problem because of the latency in the communication network. The proposed algorithm mainly focuses on the reduction of communication network latency by applying VM migration and VM rollback operation in AD and PD in the hypervisor level; in turn performance will get improved.

3. Brokering Mechanisms

broker plays vital role in the cloud interaction between cloud provider and customer. It reduces the complexity during the service access and retains the customer. A broker can get requests from the customer and forwards them to the cloud for task execution and send service response to the customer. A simple model for the cloud customer and cloud provider interaction via broker is shown in **Fig. 1**.

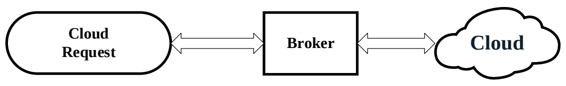


Fig. 1. Cloud interaction via brokering mechanism

In this model the broker plays a simple forward operation between the interacting entities. The interaction between homogeneous clouds and a broker that performs various service task is shown in **Fig. 2**.

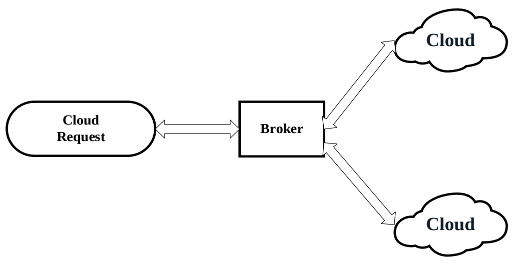


Fig. 2. Homogeneous cloud interaction via brokering mechanism

The heterogeneous cloud can interact and share the services by improving quality. There are two types of interaction namely federation and multi-cloud. Cloud has a broker with different capabilities which are coordinated with the help of CDM. It manages various cloud brokers by applying various rules to regularize the services among cloud customers is given in **Fig. 3**.

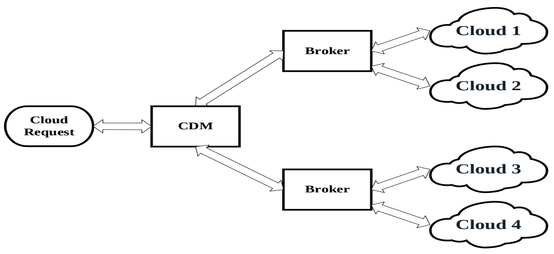


Fig. 3. Heterogeneous cloud interaction via CDM

4. Architectural Model

4.1 System Model

CDM based model gives a complete description of VM management and its architectural model is shown in **Fig. 4**. This model is derived from the CloudSim and iCanCloud for an effective management of VM in heterogeneous cloud [14], [15]**.** The CDM gets the cloud request from the customer, selects the suitable broker and submits the job to the cloud. It also maintains the log, which holds the current status about various brokers during interaction. The broker collects the information from cloud information service and they are maintained in the database for selecting a suitable VM in the data center. The proposed work focuses on VM allocation in the data center during the task execution. The requesting tasks are categorized based on the size and execution time. If the requirement of the task is less than the resources which currently exist in the VM then the VM is reallocated after it completes its execution.

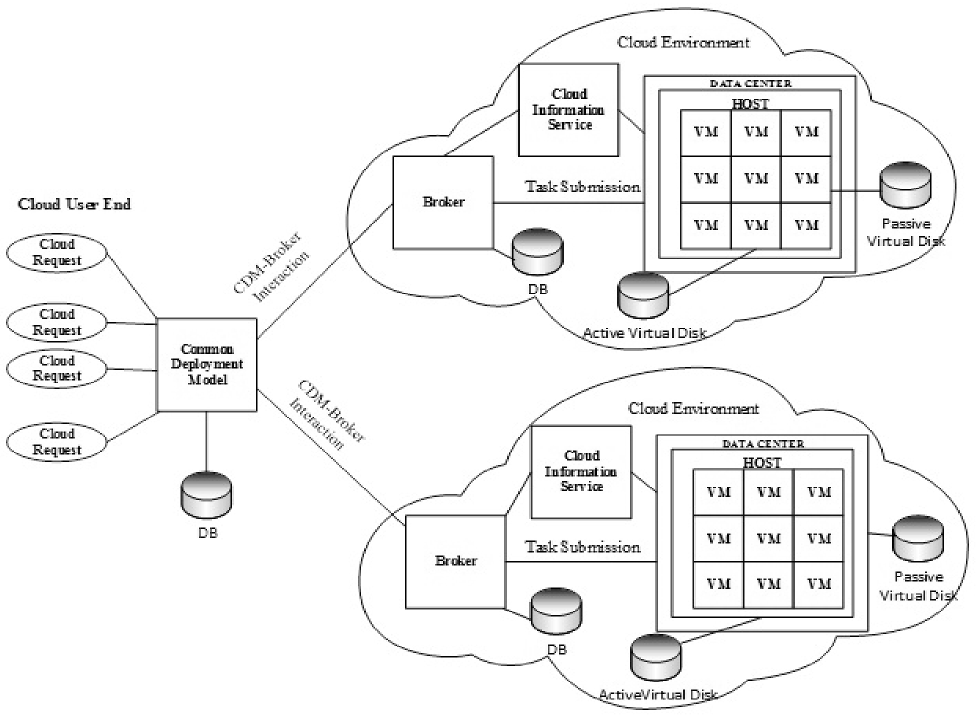


Fig. 3. Architecture of proposed model using CDM

4.2 VM Directories

The proposed work is used to allocate the VM to the newly arrived task with maximum allocated time. For example, VM life time is 10 time units and actual time taken for execution is 5 time unit. The VM will complete its execution before the stipulated time; this VM remains in idle state until the time gets elapsed. This time can be used effectively for executing another requesting task, so this VM is migrated to the PD until suitable task arrived for execution. The idea of the proposed work is to keep the idle VM to execute the new requesting task. The task size is compared with idle VM size in PD. If it is matched then the task will be executed otherwise idle VM waits for suitable task. The proposed model uses two directories namely AD and PD. AD maintains the VM with placement of VM for executing the task, whereas PD maintains the idle VM which is migrated from AD. The main objective of the proposed work is to achieve full utilization of the VM by considering idle time. The flow chart of proposed VM directories to execute the task in idle VM is shown in **Fig. 5**.

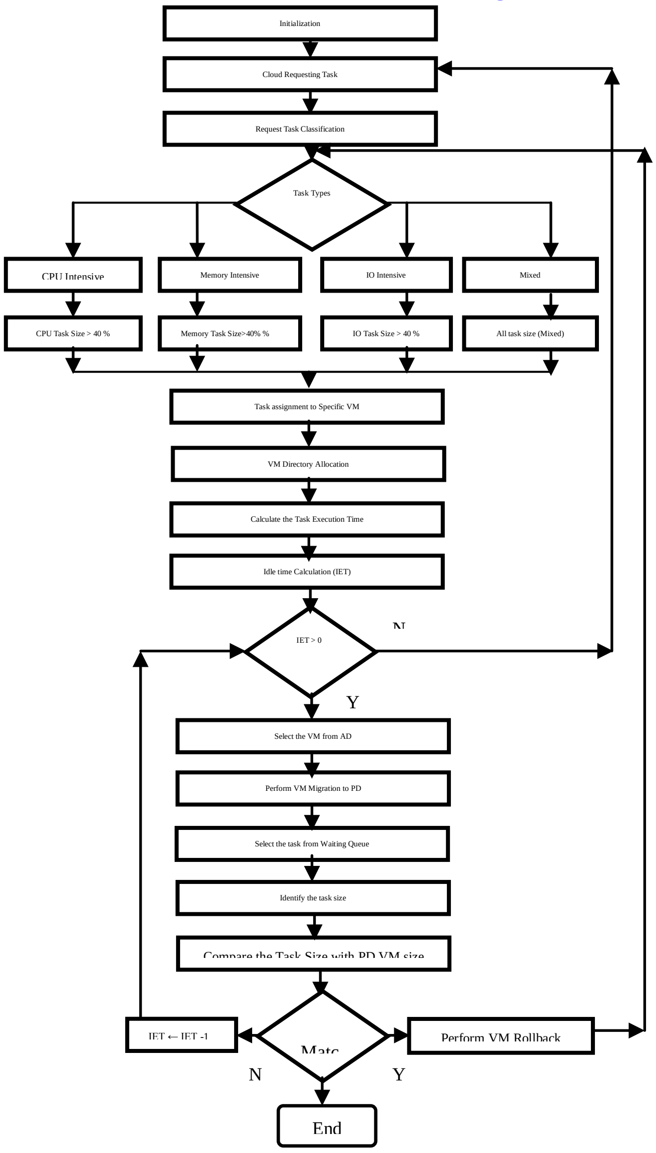


Fig. 5. Flowchart for proposed VM management

4.3 Significance of Idle VM

The mapping of Physical Machine (PM) with the VM is done using the parameters like CPU in GHZ, Memory in GB, Bandwidth in MB and IO. The existing algorithms are analyzed and compared with these parameters which give an absolute analysis related to task execution time. Suppose the VM completes its task before the stipulated time leads the task to idle state i.e. idle VM. The idle VMs will consume the cloud resources. The idle time of the VM does not measured in VM placement algorithms. The proposed work focuses on the VM placement technique with the consideration of idle time during the task execution over VM. Three dimensions have been analyzed in the existing algorithms are CPU, Memory and Bandwidth. The proposed algorithm has added another dimension called idle time. The requesting tasks are classified into CPU intensive, memory intensive, I/O intensive, or mixed type, so that the idle time is measured in an accurate manner. **Table 1** shows the task classification in the VM management [16]. The idle VMs are remains running although no task is allocated. These idle VMs consume CPU, Memory and storage resources like active machines. The idle VMs are analyzed based on the above parameters with utilization. Veenam VMware have taken a idle time report for a week with 1000 maximum CPU usage (MHz), 1000 maximum active memory (MB) and idle radio is 50%. This analysis report for idle VM is shown in **Table 2**. The idle VM consumes the usage of CPU, Memory, Network, Disk and storage consumption in high rate (%), so this can be reused to some other task for execution. The proposed model is used to allocate the task to idle VM based on the task types in order to execute the task.

Table 1. Task types

Task Type	**Conditions**
CPU Intensive Task	$PM_{Task} > 40\%$ and $PM_{Memory} < PM_{Task}$ and $PM_{IO} < PM_{Task}$
Memory Intensive Task	$PM_{Memory} > 40\%$ and $PM_{Task} < PM_{Memory}$ and $PM_{IO} < PM_{Memory}$
IO Intensive Task	$PM_{IO} > 40\%$ and $PM_{Memory} < PM_{IO}$ and $PM_{Task} < PM_{IO}$
Mixed	Otherwise

Table 2. Veenam VMWare analysis report for Idle machines

Type of VM (Number of VM)	**Idle VM (6)**	**Normal VM (15)**	**Switched Off (28)**
CPU Usage (MHz)	613.92	12413.38	-
Memory Usage (MB)	620.5	16372.33	-
Network Usage (KB/sec)	1354.53	2494.55	-
Disk Usage (KB/sec)	77	3145.72	-
Storage Consumption (GB)	52.83	553.02	561.92

4.4 VM Migration and Rollback

The proposed VM management performs two operations namely VM migration and VM rollback. VM migration is a process of moving the VM from AD to the PD whereas VM rollback is a process of moving the VM from PD to AD. The components which are used in the proposed technique are task list, AD and PD. All the requested tasks are placed in the task list which will be mapped to respective VM from AD for execution. If the task has been completed earlier than the stipulated time then the idle VM are migrated to PD for allocating the requesting task. The expected lifetime of a VM is defined as the time taken to execute the task in AD and the time which spend in PD. Complete process flow of the proposed VM management is given in **Fig. 6**.

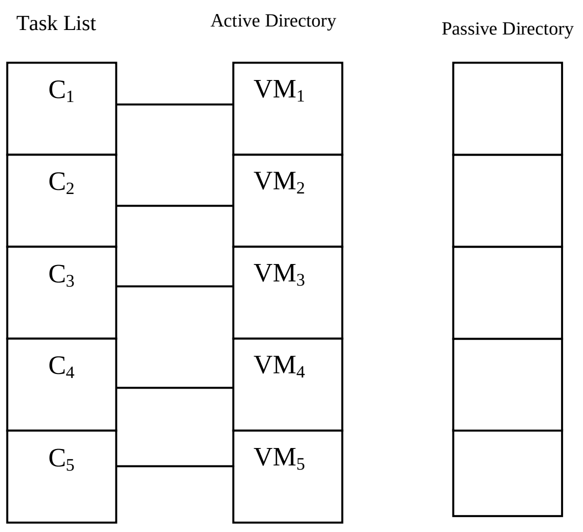


Fig. 6(a). Initial configuration

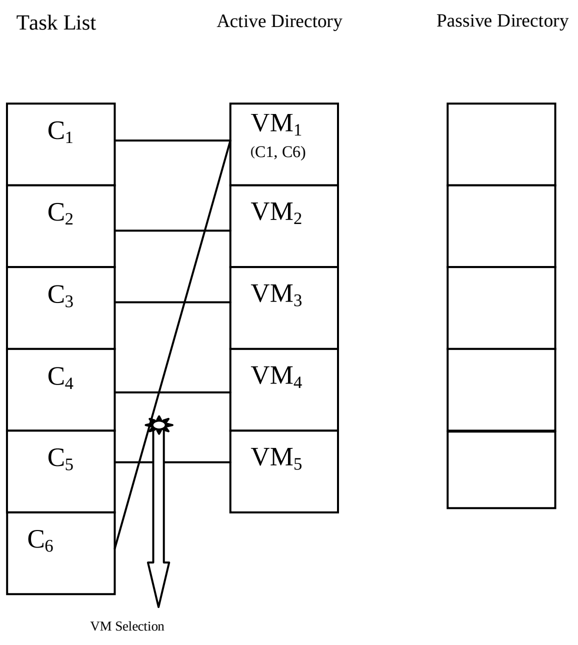


Fig. 6(b). VM Selection and submission of cloud requesting task

Initially, PD is kept empty and 5 tasks are allocated to the VM for executing in AD which are shown in **Fig. 6(a)**. The requesting tasks are placed in the task list and the respective VMs are selected in AD. The PD is still empty because none of the VM completes its execution. If task 6 arrived for execution it need to identify respective VM with the required ability to execute the another task. VM_1 has the ability to execute two tasks namely C_1 and C_6, so this VM is used to execute the newly arriving task is shown in **Fig. 6(b)**.

The task C_2 is executed in VM_2 which completes earlier than the VM_2's life time, so it performs VM migration. The dotted line in **Fig. 6(c)** represents the migration of VM_2 from

AD to PD. VM_2 is considered as an idle VM by retaining its current state in PD until suitable task arrived for execution or time reaches to zero.

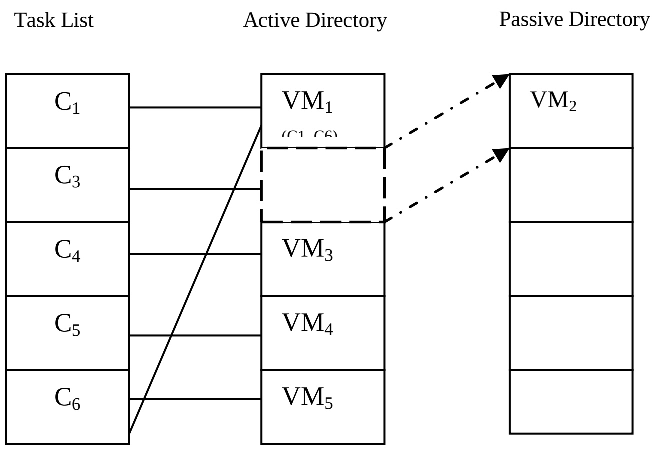


Fig. 6(c). Cloud requesting task C_2 completes its job and VM_2 moves to PD

Normally the VM is selected for execution either from AD or PD when the new task is arrived. When task C_7 arrives for execution it needs to identify the suitable VM in AD. If no other VM is suitable for execution then it goes to PD for identifying the suitable VM. If it is found then it will be moved to AD for execution otherwise idle VM will remain in PD only. The dotted line in **Fig. 6(d)** represents the VM rollback operation and **Fig. 7** presents the initial configuration. The graph is plotted by considering execution time in x-axis. Initially, all the tasks are assigned in AD.

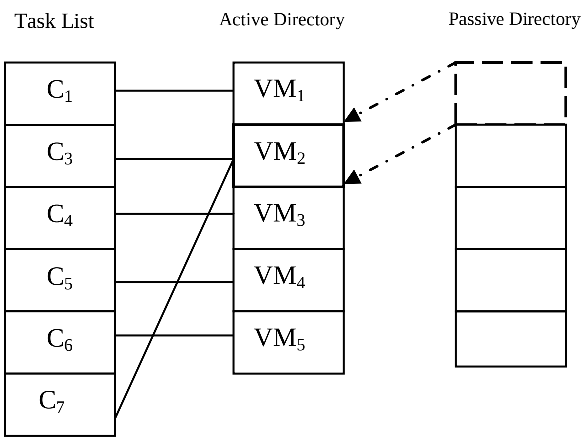


Fig. 6(d). New requesting task C_7 needs VM then VM_2 moves from PD

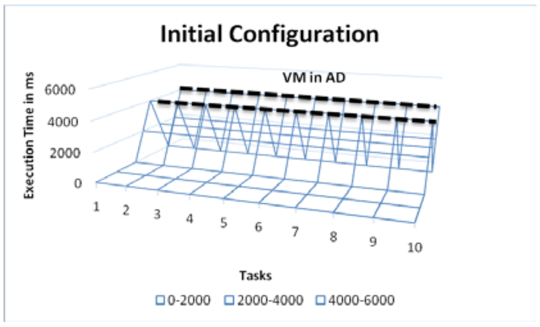


Fig. 7. Initial configuration

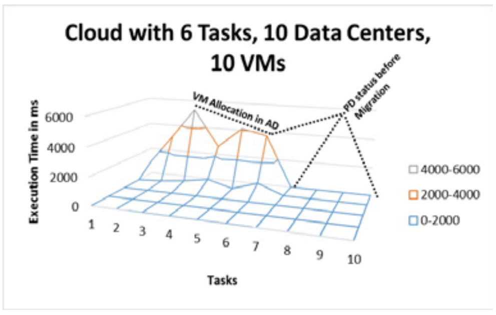


Fig. 8. VM allocation

If 6 tasks are arrived for execution from 10 Data centers and 10 VMs then the 6 tasks are allocated to the specific VM in AD. No other task will be assigned to the VM in AD i.e. task from data center 7 to data center 10. The current execution status of task in AD is shown by dotted line in **Fig. 8**.

The VM migration operation is presented in **Fig. 9**. The idle time and VM time are calculated based on the VM in PD and the current execution status of the AD and PD are shown by dotted line. The task 5 to task 10 are in idle time, so it will be migrated to PD.

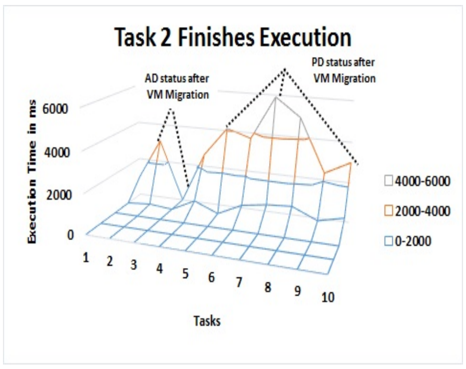


Fig. 9. VM migration

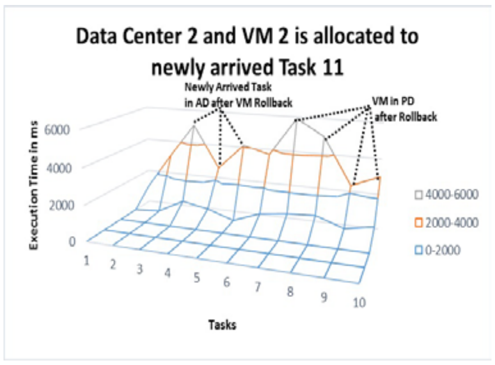


Fig. 10. VM roll back

The new task is executed and analyzed based on the capacity of VM. If the task size is suitable with VM it will be moved from PD to AD. The newly arrived task is allocated to the respective VM for performing task execution. The current status of PD and AD are shown in **Fig. 10** by dotted line.

4.5 Idle Time Calculation

The cloud consists of various VM types such as small, micro,medium, large, xlarge, 2xlarge, 4xlarge, 8xlarge and the number of CPUs 1, 2, 4, 8, 16, 48 for executing task. **Table 3** depicts that the attribute of VM with VM Types, Task Types and idle time. For example VM_1 type is small and its CPU, memory, task size are 1, 256, 156 respectively. The idle time of VM_1 is 100 (μs), so the VM_1 is migrated to the PD until the new task is allocated to VM_1 or time gets elapsed. If a new task arrives with size 90 MB then it is allocated to idle VM in PD by performing VM rollback operation.

Table 3. Characteristic of VM types, task types and idle time

VM Name	VM type	Cores	Memory(MB)	Task Size (KB)	Idle Time (μs)	Task type
VM1	m1.small	1	256	156	100	CPU Intensive
VM2	t1.micro	1	256	156	100	CPU Intensive
VM3	c1.medium	2	512	312	200	CPU Intensive
VM4	m1.large	2	512	376	136	CPU Intensive
VM5	m2.xlarge	2	2048	1689	359	Memory Intensive
VM6	m3.2xlarge	4	4096	3879	217	IO intensive
VM7	m2.4xlarge	8	4096	3451	645	Memory Intensive
VM8	cc2.8xlarge	16	6144	5000	1144	Mixed

The VM_1 migrates to PD for remaining idle time and waits for a task. The idle time of various tasks in AD is consolidated and allocated to the newly arrived tasks which are described in the following pseudo code.

```
If idle time of the task < CPU intensive task then
    Idle time is negligible
Else if idle time of the task >= Memory intensive task then
    Consolidate the idle time for new task allocation
Else if idle time of the task >= IO intensive task
    Consolidate the idle time for new task allocation
Else    idle time is unpredictable
```

5. Analysis of VM Directories

5.1 Initial configuration

The following equations are used to analyze the initial configuration of the proposed model

$$\text{Task List}=\bigcup_{i=0}^{n} C_i,\ n>0 \tag{1}$$

$$\text{AD List}=\bigcup_{i=0}^{n} VM_i,\ n>0 \tag{2}$$

$$PD\ List=\bigcup_{i=0}^{n} VM_i, n=0 \tag{3}$$

Where, C is a cloud requesting task

5.2 VM migration

The equation used to analyze the VM migration operation of the proposed model is described as

$$\text{PD List} \leftarrow \text{AD} - \bigcup_{j=0}^{vc} VM_j,\ vc>0 \tag{4}$$

Where vc is a VM count

5.3 VM rollback

The equation used to analyze the VM roll back operation of the proposed model is given by

$$\text{AD List} \leftarrow \text{PD-}\bigcup_{j=0}^{vc} \text{VM}_j, vc > 0 \tag{5}$$

5.4 Dynamic allocation of newly arrived task

The equation used to analyze the allocation of newly arrived task of the proposed model is written as

$$\text{AD Dynamic} \leftarrow \bigcup_{i=0}^{m} \text{VM}_i + \bigcup_{j=0}^{n} \{\text{VM}_{new}\}, \{m, n\} > 0 \tag{6}$$

5.5 AD parallel allocation

The equation used to analyze the parallel allocation of the proposed model is expressed as

$$\text{AD Parallel} \leftarrow \bigcup_{i=0}^{m} \{\text{VM}_i + \text{Vm}_j\}, m > 0 \tag{7}$$

6. Analysis of Task Classification

The proposed VM management algorithm is mainly based on AD and PD which supports the maximum utilization of VM for arriving task. Let DC = {host_1, host_2… host_n} denote the total number of hosts in the data center and H = {PM_1, PM_2…PM_n} denotes the total number of resources in the physical machines. Let PM = {CPU, BandWidth (BW), RAM, storage, I/O Rate} denotes the common characteristics of physical machine. The task submitted by the customer is denoted as T = {task_1, task_2 … task_n} and the list of VM is specified as VM list = {VM_1, VM_2…VM_n}. The parameters like size, execution time are used to improve the efficiency in VM management. The task is allocated to the VM is calculated based on the time taken for selecting the VM and allocating the task to specific VM. This time difference is analyzed and the task execution is performed based on backoff time and it is expressed as

$$\text{Backoff Time} = \text{Difference}\left\{\sum_{i=1}^{n}(\text{RTA}_i), \sum_{i=1}^{n}(\text{VMS}_i)\right\} \tag{8}$$

Where, RTA and VMS are the size of Requested Task and size of VM respectively.

$$\text{Task Category } \tau = \begin{cases} \text{CPU Intensive,} & \text{Level}_1 \\ \text{Memory Intensive,} & \text{Level}_2 \\ \text{IO Intensive,} & \text{Level}_3 \\ \text{Mixed,} & \text{Otherwise} \end{cases} \tag{9}$$

The requesting tasks are categorized as Level_1, Level_2 and Level_3 is shown in equation 9. Level_1 task is allocated to the VM in AD and executes the task by VM in a stipulated time. Level_2 tasks never allocated to a single VM because the capacity of the VM is not fit for current execution, so it performs the VM consolidation process. If the task belongs to Level_3 category then it is allocated to the VM in AD. Once it completes the execution, it will be migrated to the PD until the time gets elapsed. During this period any new task arrives it will be allocated to the VM by performing rollback operation. The main objective of the proposed algorithm is to exploit the VM execution time very effectively. The specification of task categories described in **Table 4**.

Table 4. Specification of various Task Types

VM Type	CPU Intensive			Memory Intensive			IO Intensive			Mixed Type		
	CPU/ GHz	Memory /GB	BW /MB	CPU /GHz	Memory /GB	BW /MB	CPU / GHz	Memory / GB	BW /MB	CPU /GHz	Memory /GB	BW /MB
Type1	2	1	1	2	40	2	2	1	2000	2	1	2
Type2	7	4	4	10	120	8	10	4	2400	10	120	8
Type3	10	8	7	16	13	15	16	12	3200	16	12	3200

7. Statistical Model for VM placement

The success of VM placement is evaluated using the sampling theory for categorizing the successful and unsuccessful placement of the VM [17].

$$E(X_m)=N_{vm}.P_{cpu} \tag{10}$$

Where,

X_m is the number of success placement

N_{vm} is the constant placement probability

P_{cpu} is the CPU intensive task for each placement

$$V(X_m)=N_{vm}.P_{cpu}.Q_{cpu} \tag{11}$$

$$Q_{cpu}=1-P_{cpu} \tag{12}$$

$$\text{Success of VM Placement } = \frac{X_m}{\alpha}=\alpha \tag{13}$$

$$E(\alpha)=E(\frac{X_m}{N_{vm}})=\frac{1}{N_{vm}}E(X_m)=\frac{N_{vm}.P_{cpu}}{N_{vm}}=P_{cpu} \tag{14}$$

$$V(P_{cpu})=V(\frac{X_m}{N_{vm}})=\frac{1}{n^2}V(X_m)=\frac{N_{vm}.P_{cpu}.Q_{cpu}}{(N_{vm})^2}=\frac{P_{cpu}.Q_{cpu}}{N_{vm}} \tag{15}$$

$$\text{Standard Error for Placement } \alpha=\sqrt{\frac{P_{cpu}.Q_{cpu}}{N_{vm}}} \tag{16}$$

$$Z_{vm}=\frac{\alpha-E(\alpha)}{\alpha} \tag{17}$$

$$\text{Successful VM Placement}=\frac{\alpha\text{-}P_{cpu}}{\alpha}\sim N(0,1) \tag{18}$$

CPU Intensive based VM Placement=

$$P_{vm}\text{-}Z_{\gamma}\frac{\sqrt{P_{cpu}\cdot Q_{cpu}}}{N_{vm}}<\alpha<P_{vm}+Z_{\gamma}\frac{\sqrt{P_{cpu}\cdot Q_{cpu}}}{N_{vm}} \tag{19}$$

The same analysis is applicable for other tasks such as memory intensive and IO intensive task for VM placement which is represented in the equations (20) and (21) respectively. The mixed type tasks are analyzed with different parameter.

Memory Intensive based VM Placement =

$$P_{vm}\text{-}Z_{\gamma}\frac{\sqrt{P_{memory}\cdot Q_{memory}}}{N_{vm}}<\alpha<P_{vm}+Z_{\gamma}\frac{\sqrt{P_{memory}\cdot Q_{memory}}}{N_{vm}} \tag{20}$$

$$\text{IO Intensive based VM Placement} = P_{vm}\text{-}Z_{\gamma}\frac{\sqrt{P_{io}\cdot Q_{io}}}{N_{vm}}<\alpha<P_{vm}+Z_{\gamma}\frac{\sqrt{P_{io}\cdot Q_{io}}}{N_{vm}} \tag{21}$$

The task allocation in AD is based on the category and PD is based on the idle time of the task which are expressed in the equation (22) and (23) respectively.

$$\text{Task Allocation in AD} = (vm_i \in VM_{AD}) \leftarrow (t_i \in \tau),\forall_i \tag{22}$$

$$\text{Task Allocation in PD} = (vm_i \in VM_{PD}) \leftarrow \{(t_i \in \tau) \leftarrow (vm_i \in VM_{AD})\},\forall_i \tag{23}$$

Where , t_i is execution time for VM and τ is a threshold level

8. Comparison of VM Placement Algorithms

VM placement is a technique used to map the VM to the physical machines. There are various placement algorithms are used to manage the VM in cloud. These algorithms are broadly classified into power based and application QoS based approaches [18]. The first fit algorithm places the physical machines to the VM in First Come First Serve (FCFS) basis but it is difficult to find VM for placement [19]. The single dimensional best fit algorithm places the VM by considering the capacity of physical machines, so it leads the performance problem [20],[21]. The volume based best fit algorithm focuses on all dimensions of physical machine and maps to the respective VM [22]. The characteristics of physical machines are represented as a dot product of the vectors for selecting and mapping a best physical machine to VM [23]. The load balanced fit algorithm balances the load among various physical machines with highest capacity [24]. In next fit algorithm all VMs are allocated based on the searching criteria of physical machine. If the first selected machine is unfit, then it selects the next machine until it finds the proper VM [25]. The random fit algorithm maps the physical machine to VM randomly who meets the requirement during the placement. The comparisons of various algorithms are analyzed based on the time taken to complete the VM placement and resource utilization is shown in **Fig. 11**.

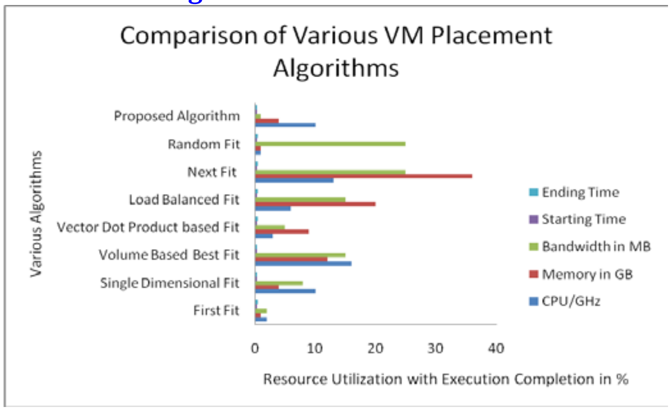
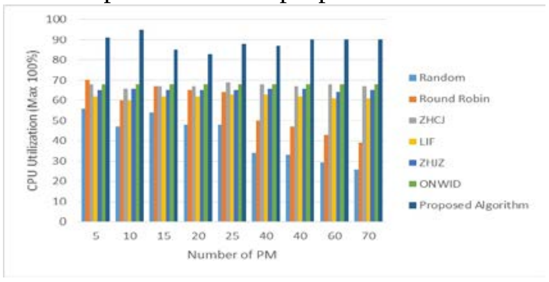


Fig. 11. Comparison of VM placement algorithms

9. Comparison of VM Scheduling algorithms

The CPU utilization with number of PM that achieves maximum CPU utilization and it is compared with the proposed CDM based algorithm is shown in **Fig. 12**.



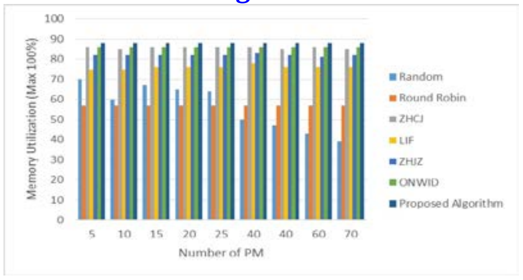
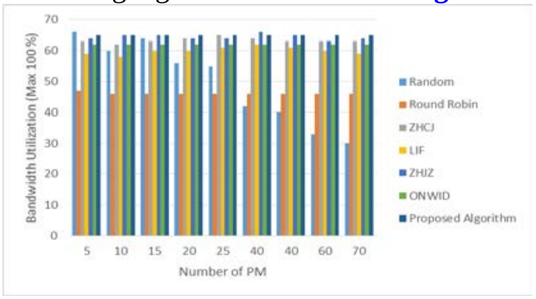


Fig. 12. Comparison of CPU utilization with various algorithms

Fig. 13. Comparison of memory utilization with various algorithms

The memory utilization of the proposed algorithm is given in **Fig. 13**. This algorithm gives a better result when compared to all existing algorithm and it is very close to ONWID algorithm. Bandwidth utilization of proposed algorithm is very close to the existing algorithms are shown in **Fig. 14**. The proposed algorithm focuses on VM Management, so no need to improve the bandwidth. The data center utilization which gives better result when compared to the existing algorithms is shown in **Fig. 15**.



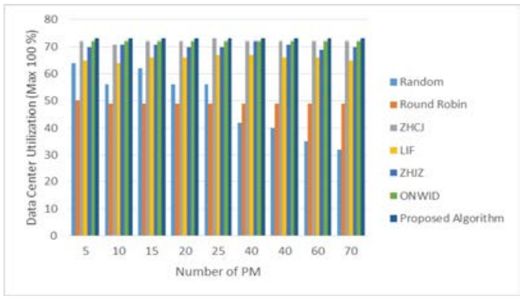


Fig. 14. Comparison of bandwidth utilization with various algorithms

Fig. 15. Comparison of data center utilization with various algorithms

Utilization is based on the number of physical machines compared with various algorithms such as Random, Round Robin, ZHCI, LIF, ZHJZ, ONWID and also with the proposed algorithm. Round robin algorithm utilizes all resources in balance manner but it consumes excess power. Random algorithm is used to schedule the task to the available VM in random manner but it suffers long waiting time before being served. ZHCJ algorithm allocates the VM with highest utilization of resources, so it leads overhead problem due to the consumption of high volume resources. ZHJZ algorithm chooses the physical machine with low cost resources and it suffers performance problem due to larger task. LIF algorithm allocates the VM based on demand characteristics of resources such as CPU, memory and network etc, so it consumes higher bandwidth and higher memory. ONWID algorithm utilizes the resources for task execution based on the increasing order but it faces starvation problem.

The objective of the proposed algorithm is to achieve maximum utilization of the CPU in the physical machines. The analysis depends on various physical machine scheduling but the proposed algorithm adds the VM idle time which is reused to execute other task. It also provides better CPU utilization related to the VM task execution. The memory utilization is achieved in maximum level because the task execution depends on memory. The bandwidth utilization is equivalent to the random and ZHJZ algorithm. Data center utilization is achieved at the maximum level by exploiting the resources with mapping from physical machine to VM in AD and PD.

10. Comparison of iCanCloud and Proposed CDM model

VM management latency of iCanCloud model is higher than the CDM based model because of the latency in communication network during VM Migration and VM rollback operation. The CDM based model maintains an idle VM in hypervisor itself without considering communication network latency, so it takes minimum time for VM mapping is shown in **Fig. 16**. The comparison between migration and rollback time is shown in **Fig. 17** and **Fig. 18** respectively. The idle time of iCanCloud and proposed model is shown in **Fig. 19** and **Fig. 20**. The comparison of task execution over idle VM is shown in **Fig. 21**. The CDM based model executes more requesting task when compared to iCanCloud model because it takes minimum latency time during VM migration and VM rollback operation.

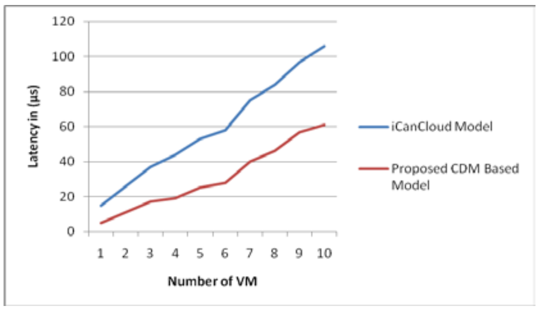


Fig. 16. Comparison of VM Latency

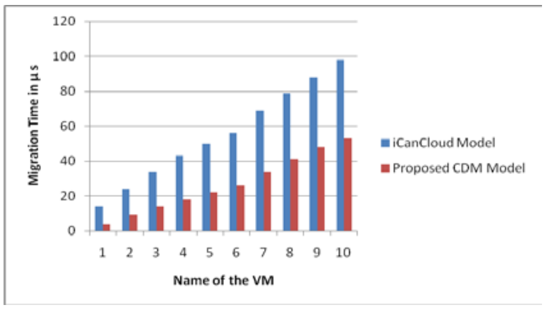


Fig. 17. Comparison of Migration Time

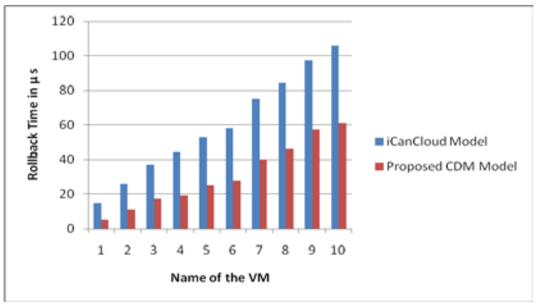


Fig. 18. Comparison of Rollback Time

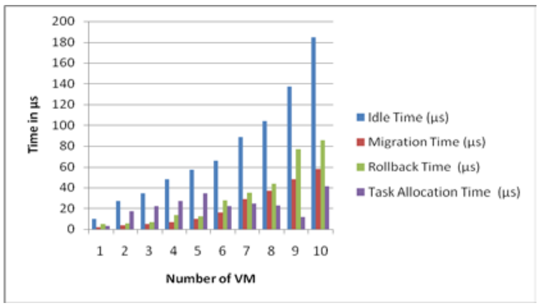


Fig. 20. Idle Time of CDM based Model

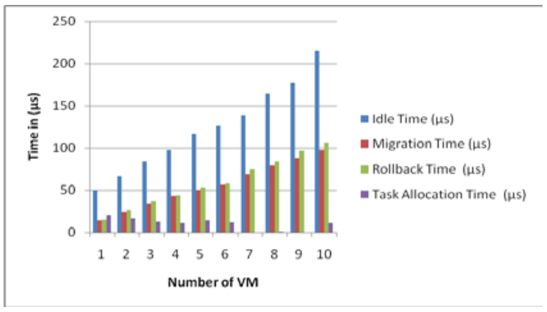


Fig. 19. Idle Time of iCanCloud Model

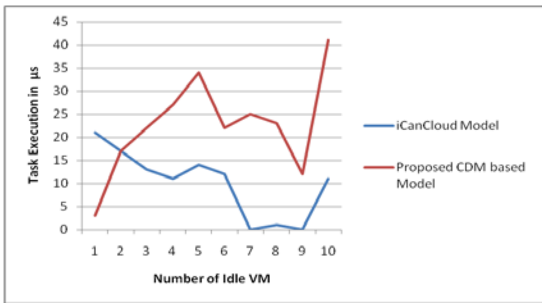


Fig. 21. Comparison of Idle VM Task Execution

11. Conclusion

The proposed model is focused on the idle time of the VM with various task types. Various methods and algorithms have been studied and analyzed with the operation like VM management, VM placement and VM scheduling. The VM placement algorithm is compared with resource utilization based on task execution. The different brokering mechanisms are identified and also implemented in CDM based multi-cloud interaction. The idle time of the VM which has already completed its execution is utilized productively by using two directories such as AD and PD. Various scheduling algorithms were compared because without VM scheduling and VM placement it is difficult to manage VM. The utilization of CPU, Bandwidth, Memory, IO and data centers are analyzed and compared with the proposed model. The iCanCloud model has been tested for idle VM management in VM repository. The latency problem occurs in VM mapping between hypervisor and VM repository is solved by retaining the idle VM in hypervisor itself. From the results it is clear that the CDM based model takes reduced latency in VM Management.

References

[1] Hossein Ghiasi and Mostafa Ghobaei Arani, "Smart Virtual Machine Placement Using Learning Automata to Reduce Power Consumption in Cloud Data Centers," *Smart Computing Review*, vol.5, no.6, pp.532-562, Dec. 2015. Article (CrossRef Link)

[2] Fei Xu, Fangming Liu, Linghui Liu, Hai Jin, Bo Li and Baochun Li, "iAware: Making Live Migration of Virtual Machines Interference-Aware in the Cloud," *IEEE Transactions on Computers*, vol. 63, no. 12, pp. 3012-3025, December, 2014. Article (CrossRef Link)

[3] Yingjie Xia, Xiumei Li and Zhenyu Shan, “Parallelized Fusion on Multi-sensor Transportation Data: A Case Study in CyberITS,” *International Journal of Intelligent Systems*, vol. 28, no. 6, pp. 540-564, March, 2013. Article (CrossRef Link)

[4] Yingjie Xia, Ting Zhang and Shengbao Wang, “A Generic Methodological Framework for Cyber-ITS: Using Cyber-infrastructure in ITS Data Analysis Cases,” *Fundamenta Informaticae*, vol. 133, no.1, pp. 35-53, January, 2014. Article (CrossRef Link)

[5] Yingjie Xia,Yuncai Liu, Zhoumin Ye,Wei Wu and Mingzhe Zhu, “Quadtree-based Domain Decomposition for Parallel Map-Matching on GPS Data,” in *Proc. of 15th IEEE Conference on Intelligent Transportation Systems Conference*, pp.808-813, September 16-19, 2012. Article (CrossRef Link)

[6] K.T.Raghavendra, “Virtual CPU Scheduling Techniques for Kernel Based Virtual Machine (Kvm),” in *Proc. of IEEE International Conference on Cloud Computing in Emerging Markets*, pp. 1-6, October 16-18, 2013. Article (CrossRef Link)

[7] Qian Liu, Chuliang Weng, Minglu Li and Yuan Luo, ”An In-VM Measuring Framework for Increasing Virtual Machine Security in Clouds,” *IEEE Computer and Reliability Societies*, vol. 8, no. 6, pp. 56-62, December, 2010. Article (CrossRef Link)

[8] Lei Yu, Chuliang Weng, Minglu Li and Yuan Luo Shanghai, ”SNPdisk: An Efficient Para-Virtualization Snapshot Mechanism for Virtual Disks in Private Clouds,” *IEEE NETWORK* vol. 25, no. 4, pp. 20-26, August, 2011. Article (CrossRef Link)

[9] Pablo Graubner, Matthias Schmidt and Bernd Freisleben, “Energy-Efficient Virtual Machine Consolidation,” *IEEE Computer Society*, vol. 15, no.2, pp. 28-34, March, 2013. Article (CrossRef Link)

[10] Xiangping Bu and Cheng-Zhong Xu, “Coordinated Self-Configuration of Virtual Machines and Appliances Using a Model-Free Learning Approach,” *IEEE Transaction on Parallel and Distributed Systems*, vol. 24, no. 4, pp. 681-690, April, 2013. Article (CrossRef Link)

[11] Anton Beloglazov and Rajkumar Buyya, “Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers under Quality of Service Constraints,” *IEEE Transaction on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1366-1379, July, 2013. Article (CrossRef Link)

[12] Zhen Xiao, Weijia Song and Qi Chen, “Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment,” *IEEE Transaction on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107-1117, April, 2013. Article (CrossRef Link)

[13] Wenhong Tian, Yong Zhao, Minxian Xu, Yuanliang Zhong and Xiashuang Sun, “Optimized Cloud Resource Management and Scheduling,” *A Toolkit for Modeling and Simulation of Real-time Virtual Machine Allocation in a Cloud Data Center*, vol. 12 ,no. 1, pp. 217-243, January, 2015. Article (CrossRef Link)

[14] N.Rodrigo, Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose and Rajkumar Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Software Practice and Experience*, vol. 41, no.1, pp. 23-50, August, 2011. Article (CrossRef Link)

[15] Castane G G, Nunez A and Carretero J, “iCanCloud: A Brief Architecture Overview,” in *Proc. of IEEE 2012 International Symposium on Parallel and Distributed Processing with Applications*, pp. 853 – 854, July 10-13 , 2012. Article (CrossRef Link)

[16] Kalim Qureshi, Babar Majeed, Jawad Haider Kazmi and Sajjad Ahmed Madani, “Task partitioning, scheduling and load balancing strategy for mixed nature of tasks,” *Journal of Super Computing*, vol. 59, no, 3, pp. 1348-1359, March, 2012. Article (CrossRef Link)

[17] P.Kandasamy,K.Thilagavathi and K.Gunavathy, *Engineering Mathematics*. 2nd Edition, Tamilnadu, India, 2000.

[18] Sameer Kumar Mandal, ”On-Demand VM Placement on Cloud Infrastructure,” *M.Tech Thesis*, National Institute of Technology, Rourkela, India, December, 2013. Article (CrossRef Link)

[19] Ankit Anand, “Adaptive Virtual Machine Placement supporting performance SLAs,” *M.Tech Thesis*, Indian Institute of Science, Bangalore ,India, pp. 1-15, June, 2013. Article (CrossRef Link)

[20] Dhaval Bonde, “Techniques for Virtual Machine Placement in Clouds,” *M.Tech Thesis*, Indian Institute of Technology, Mumbai, India, 2010. Article (CrossRef Link)

[21] A.K.Bhatia ,M. Hazra and S.K.Basu, ”Better-Fit Heuristic for One-Dimensional Bin-Packing Problem,” in *Proc. of IEEE Advance Computing Conference*, pp. 193 – 196, March 6-7, 2009. Article (CrossRef Link)

[22] N.Sprunk and A. Knoll, “Learning a Fuzzy System from Training data using the Munsteraner Optimisation System,” in *Proc. of IEEE International Conference on Fuzzy Systems*, pp. 1-7, June 10-15, 2012. Article (CrossRef Link)

[23] M.Batty and P. Kyaw, “Vector-Based Location Finding for Context-Aware Campus,” in *Proc. of IEEE International Conference on Wireless and Mobile Communications*, pp. 116 – 121, August 23-29, 2009. Article (CrossRef Link)

[24] T.Cheocherngngarn, J Andrian and Deng Pan, “Depth-First Worst-Fit Search based multipath routing for data center networks,” in *Proc. of IEEE Global Communications Conference* , pp. 2821 - 2826, December 3-7 , 2012. Article (CrossRef Link)

[25] Zhilin Zhu, Jinxue Sui and Li Yang, “Bin-packing Algorithms for Periodic Task Scheduling,” in *Proc. of IEEE WASE international conference on Information Engineering*, pp. 207-210, August 14-15 , 2010. Article (CrossRef Link)

Saravanakumar Chelliah received B.E degree in Information Technology from Bharathidhasan University, Trichy, India, in 2003, and the M.E degree in Computer Science and Engineering from, Vinayaga Mission’s University, Salem, India, in 2007. Currently he is working as an Associate Professor, Department of IT, St.Joseph’s Institute of Technology, Chennai, Tamilnadu, India. He is a life member of Computer Society of India (CSI) and published many national and international research articles. His research interests are Cloud Computing, Software reliability and Distributed Systems.

Arun Chokkalingam received the B.E degree in Electronics and Communication engineering from Bharathidasan University, Trichy, India in 2002 and the M.E degree from Anna University, Chennai, India 2004. He acquired his PhD (VLSI design) in 2009 from College of Engineering Anna University, Chennai. Presently he is working as Professor in the Department of ECE, RMK College of Engineering and Technology, Chennai, Tamilnadu, India. His research in error correcting codes addresses effectively decoding algorithm and VLSI Architecture. His research interests including digital communication, Cloud Computing and Distributed Systems.