Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 7 – No. 39, August 2023 – www.caeaccess.org

Development of a High Performance Toolkit for Modelling and Simulating Cloud Computing Environment and Application

Mary T. Kinga
Computer Science Department,
Federal University of Technology

Sunday O. Adewale
Computer Science Department,
Federal University of Technology

Folasade M. Dahunsi
Computer Engineering Department,
Federal University of Technology,
Akure, Nigeria

ABSTRACT

Presently, none of the current distributed (including grid and network) system simulators can offer the environment that can be directly used for modelling cloud computing environments and applications with high-performance rate, and maximum resource utilization. To overcome this challenge this research presents HiCloud: a new simulation framework that allows seamless modelling, simulation, and experimentation of emerging cloud computing infrastructures and application services. The developed system is a Cloudsim-based simulator that models cloud networks with minimum processing time and maximum resource utilization ratio. This research focuses on the resource utilization by using optimize execution time algorithm for service broker policy. It also takes into consideration the task migration approach for the load balancing algorithm that is used in the execution of tasks. The system was able to model cloud networks and application with high-performance metrics. The experimental results showed that the developed system has a better performance in terms of response time, execution time, makespan time and resource utilization ratio compared to existing systems.

Keywords

Simulation, Cloud Computing, Modeling, High Performance, Toolkit.

1. INTRODUCTION

The world is connected and more than two billion people connect to the Internet, the community of information technology has come up with a new service delivery mechanism called "Cloud Computing". Cloud computing is a new model that provides ubiquitous, convenient and on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) which can be rapidly provisioned and re-leased with little management effort or little service provider interaction [1]. The features of Cloud Computing include self-demand, comprehensive network access, resource pooling and scalability of resources. Cloud Computing provides services such as Software as a Service (SaaS), Platform as Service (PaaS) and Infrastructure as a Service (IaaS) and there are four types of Cloud models; Public Cloud, Private Cloud, Community Cloud and Hybrid Cloud [2][21]. Cloud infrastructure includes data centers that host servers and uses different levels of virtualisation techniques to offer cloud services [3][22].

The design of a Data Center of the next generation is important because clouds aim to power the next-generation data centers as an enabling platform for dynamic and flexible application provisioning. This is done by exposing the data center's capabilities as a network of virtual services (e.g. hardware, database, user-interface, and application logic) so that users can access and deploy applications from anywhere on the Internet-driven by the demand and Quality of Service (QoS) requirements [4][5]. By using the clouds platform which holds application, IT companies are freed from the common task of setting up hardware and software infrastructures. Therefore, they can focus more on innovation and the creation of business values for their application services [5][23]. The Cloud Network design is carried out to evaluate and know the requirement of the cloud network to be implemented [24][25]. This design is represented as a network diagram to be implemented physically. A well-designed network minimises cost and optimises network bandwidth [6][7]. It is not possible to carryout benchmarking experiments in repeatable, reliable, and scalable environments using real-world Cloud environments. A better alternative is the use of simulation tools. These tools give room for the possibility of evaluating the hypothesis (application benchmarking study) in a well-controlled environment whereby one can easily regenerate results. To evaluate the performance of Cloud provisioning policies, the software workload models and the resources performance models; in a repeatable manner under different system and user configurations and requirements is very difficult to obtain [26].

To overcome this challenge, this research work presents HiCloud: A Cloudsim-based toolkit that enables modelling and simulation of the cloud computing system and behavioural modelling of Cloud components like data centers, virtual machines (VMs) and resource provisioning policies. The usefulness of HiCloud simulator is shown in a scenario involving the dynamic scheduling of tasks in the cloud environment. The result shows that the Cloud computing model reasonably improves the application QoS requirements under fluctuating resource and service demand patterns with minimum makespan time and maximum resource utilization. This paper is structured in the following order: In section 2, the previous works that are related to the proposed work are discussed. In section 3, HiCloud architecture is discussed. The proposed cloud system implementation and the result are analysed in section 4. Section 5 concludes the aim of the proposed work and its future work.

2. RELATED WORKS

There is a need for the development of a cloud computing simulator to model cloud environments and application testing with better performance features. [8] presented CloudSim: a toolkit that models and simulates cloud computing environments and evaluates resource provisioning algorithms.

The study was based on the need for tools to test the newly developed methods, policies and mechanisms to efficiently manage cloud infrastructures, and the need to evaluate the hypothesis before a real deployment in an environment, where one can reproduce tests. [9] presented CloudAnalyst: A cloudsim-based visual modeller that analyzes cloud computing environments and their applications. The study was carried out due to the lack of tools that enable developers to evaluate the requirements of large-scale cloud applications in terms of the geographic distribution of both computing servers and user tasks. [10] presented iCanCloud: A flexible and scalable cloud infrastructure simulator. The study was premised on the need for the tool to decide the best starting conditions on pay-as-you-go scenarios. Based on this need iCanCloud was developed, a new simulator of cloud infrastructures with remarkable features such as flexibility, scalability, performance and usability.

[11] presented a design of the network infrastructure of a cloud data center for use in the health sector of Arequipa city. The study was based on the need for a powerful and hardy data center to facilitate the provision of healthcare products and services to patients in remote areas and patients who have limited access to quality medical services. [12] presented designing and building a data center network: an alternative approach with OpenFlow. The research was based on the need for a new approach in which the network can be treated more dynamically, where VMs and other resources can be quickly and flexibly introduced, moved, or modified as needs change, without the need for manual intervention to reconfigure the network. [13] presented building a private HPC cloud for computing and data-intensive applications. They aimed at amplifying the improved resource utilization of cloud computing for HPC applications due to their insatiable resource needs. Cluster virtualization was used to achieve optimal compute and data-intensive applications in private HPC cloud developed.

[14] presented the design and implementation of a peer-to-peer cloud system. The research is premised on maintaining a consistent structure over a set of unsecured computing resources. [6] presented dimensioning resilient optical grid / cloud networks.

[3] worked on cloud computing workload and capacity management using domain-specific modelling. Domain-specific modelling, model transformation, and time series analysis techniques with estimation algorithms were used to manage cloud computing workload and capacity. [7] presented the design of a government cloud network: case study Ondo state. A mathematical model was used to design the network, and specification from cloud management software was used to deploy AMD-Virtualised server. An established simulator (CloudSim and CloudAnalyst) was used to simulate the proposed G-Cloud that was designed.

[15] presented an enhanced deadline constraints based tasks scheduling mechanism for the cloud environment. A scheduling system for deadline sensitive workload or lease was developed. [16] presented an extended intelligent water drop algorithm for workflow scheduling in the cloud computing environment. The new algorithm extends the natural-based Intelligent Water Drops (IWD) algorithm to optimize the scheduling of workflows over the cloud. [17] presented a new cost-efficient approach for deadline-constrained workflow scheduling by dynamic provisioning of resources. In this work a new approach for dynamic provisioning of resources and a workflow scheduling algorithm that is cost-efficient and deadline-constrained was invented. [18] presented genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds.

[19] presented simulation modelling of cloud computing for the smart grid using cloudsim. A smart grid cloud was simulated using CloudSim. [20] worked on efficient task scheduling for a-budget constraint parallel applications on heterogeneous cloud computing systems. The research work shows that the preassignment of tasks with the minimum cost does not necessarily lead to the minimization of the schedule length. We have proposed the development of HiCloud, a high-performance toolkit for modelling and simulating cloud computing environment and applications.

3. METHODOLOGY

3.1 System Architecture

HiCloud is a CloudSim-based simulator that models and simulates cloud environments and application. It has a graphical user interface with some extensions of models for better performance. It was developed in java programming language and run through JCreator IDE. HiCloud can define simulation with a high degree of configuration and it is very flexible. The architecture of HiCloud simulator is shown in Fig. 1.

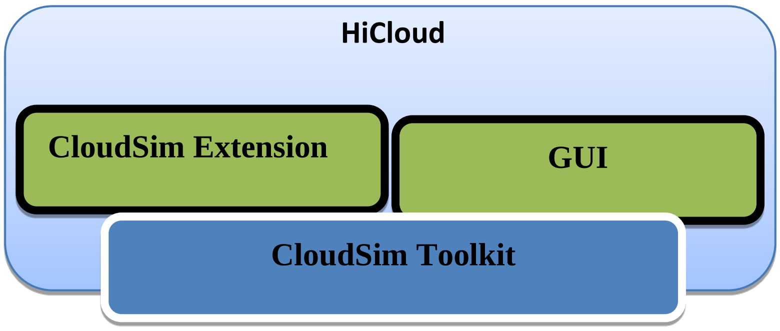


Fig 1: HiCloud Architecture

3.1.1 CloudSim Toolkit

CloudSim toolkit supports both system modelling and behaviour modelling of cloud infrastructures such as data centers, virtual machines and as well as resource provisioning policies. The data center entity controls several host entities. The host is a CloudSim component that represents a physical computing server in a cloud network. CloudSim supports simulation scenarios that assign specific CPU cores to specific VMs (a space-shared policy), dynamically distribute the capacity of a core among VMs (time-shared policy), or assigns cores to VMs on demand. HiCloud components are modelled into three main layers in the input phase. Layer 1 is the main configuration layer, layer 2 is the data center configuration layer and layer 3 is the advance layer. Values inputted in models in these three layers were simulated and then generated metrics such as average execution time, average response time, average makespan time and average utilization ratio. The overview of HiCloud layered architecture is shown in Fig. 2.

3.1.2 Layer 1 of HiCloud

Layer 1 is the main configuration class. This class contains the simulation duration class, user bases class, service broker policy class and the application deployment configuration class. In the simulation class, the simulation time was modelled as 60 seconds. The user bases class modelled a group of users as an entity UB1, UB2, UB3 and UB4. Regions of different user bases were modelled as region 0, region 1, region 2 and region 3. The request of users per hour was modelled as 60 requests per user base, data size per request was modelled as 100, the peak hour start time was modelled as 8am, and the peak hour end was modelled as 6pm as shown in Figure 6.

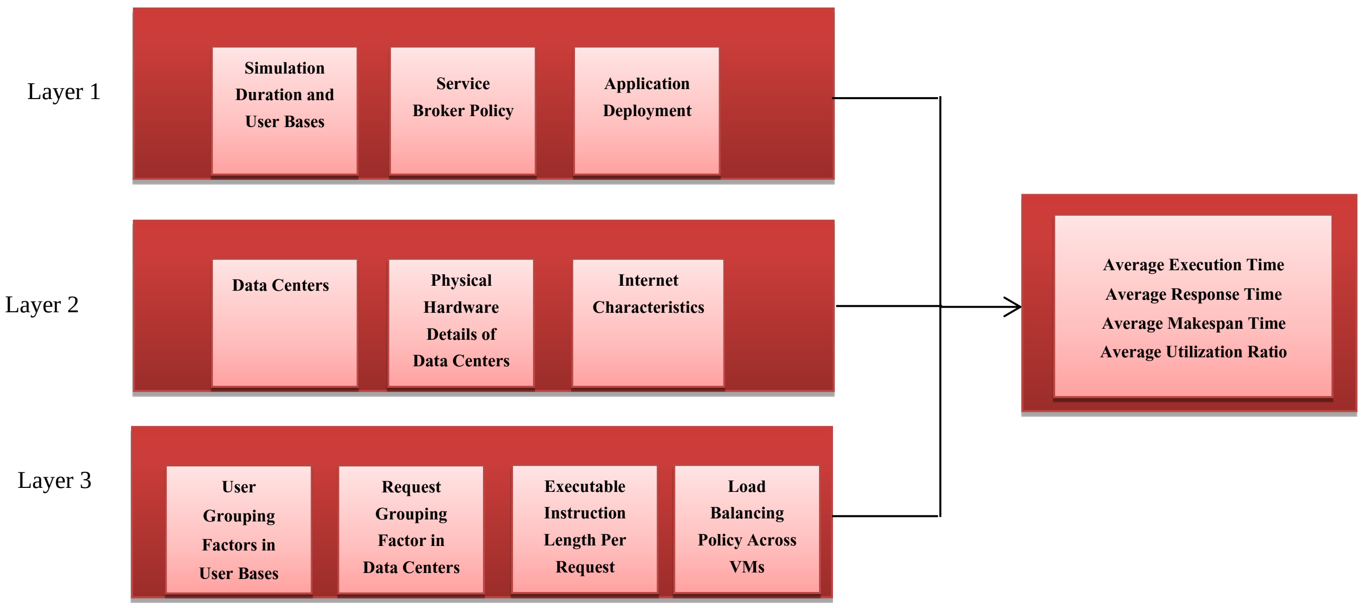

Layer 1
Simulation Duration and User Bases
Service Broker Policy
Application Deployment
Layer 2
Data Centers
Physical Hardware Details of Data Centers
Internet Characteristics
Layer 3
User Grouping Factors in User Bases
Request Grouping Factor in Data Centers
Executable Instruction Length Per Request
Load Balancing Policy Across VMs
Average Execution Time
Average Response Time
Average Makespan Time
Average Utilization Ratio

Fig 2: HiCloud Layer Architecture

The service broker policy class modelled an optimize execution time algorithm where tasks are allocated to the data center that has minimum execution time for fast response time. This execution time is calculated ahead. Fig. 4 shows the optimize execution time algorithm.

The application deployment configuration class modelled the parameters of virtual machines for executing task or services requested by users.

The service broker algorithm in Fig. 3 shows the detailed steps needed to perform optimize execution time algorithm. Finish time cannot be calculated before the task is executed so the expected finish time is calculated depending on the executable length (million instructions) and the server processor speed. The value gotten is used to calculate the expected finish time as follows:

1. The task arrives in the form of an Internet cloudlet and the size of the task is defined by its Executable instruction length (MI) and size (MB).

2. The resources (DCs) are defined by their processing speed (MIPS) and bandwidth (Mbps).

3. To schedule n tasks (T1, T2, T3…. Tn) onto m available resources (R1, R2, R3…. Rm), the expected time to process tasks on each of the resources is calculated using Equation 1.

$$T_{ij} = e_{ij} + r_j \qquad (1)$$

where T_{ij} denotes the expected running time of task i on resource j, e_{ij} denotes the execution time of the task i on resource j, r_j denotes the ready time of the resource j.

4. So each entity of the ETM matrix is computed as that equation then the algorithm chooses the entity with the min value, and according to that value assigns tasks to the right data center.

Response time is the processing time plus the cost of the request or the task transmission time, queued through the network nodes.

Given that the expected response time,

$$time = F - A + T_{delay} \qquad (2)$$

where F denotes the time to complete the task, A denotes the arrival time of the task. The algorithm only affects the processing time in a local environment of the data center. Therefore, the communication delay parameter can be omitted, hence T_{delay}= 0.

```
Input: a set of tasks, m data centers, ETM matrix.
Output: the schedule plan
Initiate the task set B.
While there are tasks not assigned do
    Update task set B.
        For i:task $v_i$ do
            Pull all Data centers status.
            Get the earliest resource available time.
            Find the Datacenter $D_{min}(v_i)$ giving the earliest finish time of $v_i$.
        End For
        Find the task-Data center pair $(v_k, D_{min}(v_k))$ with the earliest finish time.
        Assign the task $v_k$ to the cloud $D_{min}(v_k)$.
        Remove $v_k$ from B.
    Update the task set B
End While
```

Fig. 3: Service Broker Algorithm

3.1.3 Layer 2 of HiCloud

Layer 2 consists of the data center, the physical hardware details of the data center and the internet characteristics.

In the data center class, four data centers were modelled within four regions. In each data center, 100 machines were modelled. Each machinehasamemoryof512GB,4 processor, 1TB storage and 1000MB available bandwidth per machine. The data center architecture is x86 architecture with Linux operating system. The data center processor speed was modelled as 10000MIPS. A summary of these models is presented in Table 4.2.

The HiCloud environment maintains an $m \times n$ size matrix for all entities that are currently active in the simulation context. An entry *kij* in the matrix represents the delay that a message would undergo when it is being transferred from entity i to entity j over the cloud network. It means that an event from entity i to j will only be forwarded by the service broker when the total simulation time reaches the $t + d$ value, where t is the simulation time when the message was originally sent, and d is the network latency between entities i and j.

3.1.4 Layer 3 of HiCloud

Layer 3 consists of user grouping factors in the user bases, request grouping factor in the data center, executable instruction length per request (bytes), and load balancing policy across VMs in a single data center.

User grouping factors class modelled users in to groups according to the number of bases within a time zone and it was assumed that most users use the application during working hours for about 2 hours within 8GMT to 6GMT. One tenth of simultaneous online users during peak hours was online during off peak hours and also each user makes a new request at every 5 minutes.

In the request grouping factor class, the request sent to the cloud network is grouped in the data center as average peak requests. Executable instruction length per request (bytes) class modelled 250 bytes of instructions per request.

The load balancing policy class modelled a task migration policy. The load balancer balances workload by migrating excess tasks from overloaded virtual machines to under loaded virtual machines. This load balancing algorithm was used purposely to reduce the makespan time and increases the average resource utilization ratio of tasks in the cloud environment as shown in Figure 3.4. The algorithm uses FCFS and SJF in scheduling workload to the virtual machine. If the utilization level of the virtual machine is more than 80% (over loaded virtual machine) of its initial capacity, this means it is overloaded, therefore excess load is taken to a virtual machine that its utilization is less than 25% (under loaded virtual machine) of its capacity, until under loaded virtual machine reach to threshold limit of overload virtual machine. After transferring tasks, the algorithm will check the load balancing condition and transfer the task if there is any virtual machine that is not satisfied with the overloading condition. The scheduler schedules all tasks to the virtual machine in such a way that cloud user can execute their task in minimum makespan time and average resource utilization been at maximum. All tasks are non-priority basis and independent, every task has task length TL_i that is expressed in million instructions. Every task requires p processing speed, q number of CPU, r amount of main memory and bandwidth B in MBPS. The cloud task scheduler contains information about M virtual machines $VM_1, VM_2, VM_3 \ldots VM_M$. The capacity of each virtual machine and the capacity of all virtual machines were calculated using equation (3) and equation (4)

$$c = p \times q \tag{3}$$

where c denotes the capacity of each virtual machine, C denotes the capacity of all virtual machines, p denotes processing speed of the processor (CPU) in million instructions per second,
q denotes the number of CPU that are busy to execute the task.
The capacity of all virtual machine is calculated using equation 4.

$$C = \sum_{j=1}^{m} C_{VM} \tag{4}$$

The Cloud task scheduler allocates the workload to all virtual machines, and each virtual machine has a queue to store the task. The total length of a queue in the virtual machine represents the load on that virtual machine. Load on a virtual machine at a particular time t can be calculated as the number of tasks on a particular virtual machine divided by the service rate of the virtual machine as shown in equation 5

$$L = k \times l/s \tag{5}$$

where L denotes load on the virtual machine at time t, k denotes the number of task 1, 2, 3…N tasks
s denotes service rate of the virtual machine at time t, l denotes the length of tasks.
Service rate of virtual machine s is expressed as processing power p and number of CPU q in equation 6

$$s = p \times x(t) \tag{6}$$

where x =1,2,3…….. q.
The Total load at all virtual machine can be calculated using equation 7

$$Q = \sum_{j=1}^{n} L_j \tag{7}$$

where Q denotes total tasks at all virtual machines, L denotes load on the virtual machine at time t
If the workload is less than the data center capacity then the load balancer balances the load over all the virtual machines. If any virtual machine is over loaded, the load balancer transfers the task of the overloaded virtual machine to the under loaded virtual machine so that all the tasks can be executed in minimum time. Therefore, the task transfer time is calculated using equation 8 Task transfer time (t) is

$$t = l/B \tag{8}$$

where t denotes the task transfer time, l denotes the length of the task, B denotes the bandwidth
The execution time of task T_i on virtual machine VM_j can be calculated using equation 9

$$E = \sum_{i,j}^{n} e_{ij} \times {}^{l}/_{p} \times q \tag{9}$$

where E denotes the execution time of tasks t_i on virtual machine vm_j, e_{ij} denotes the execution time of task i on resource j, l denotes the length of the task, p denotes the processing speed of the processor (CPU) in million instructions per second, q denotes number of CPU that are busy to execute the tasks.
The response time of task is calculated using equation 10

$$R = a + E \tag{10}$$

where R denotes the response time, also a denotes the arrival time of task t_i, and E denotes the execution time of tasks $t_i$on virtual machine vm_j.
The makespan time of tasks can be calculated using equation 11

$$M = \sum_{j=1}^{n} E_j + t \tag{11}$$

where M denotes the makespan time, t denotes the task transfer time, E_j denotes the execution time of tasks.
The average resource utilization ratio of resources can be calculated using equation 12 and equation 13. The limits of the average resource utilization ratio are from 0 to 1, maximum value for ARUR is 1(resource utilization is 100%) and the worst value is 0 (resource is in ideal condition).

$$U = \left({}^{m}/_{M}\right) \tag{12}$$

Given that $m = \sum_{j=1}^{n} {}^{f_j}/_{d}$ (13)

where U denotes the average resource utilization ratio, m denotes mean time, M denotes makespan time, f denotes the time taken by resource vm_j to finish all the jobs, d denotes the number of resources.
The throughput value is also calculated using equation 14

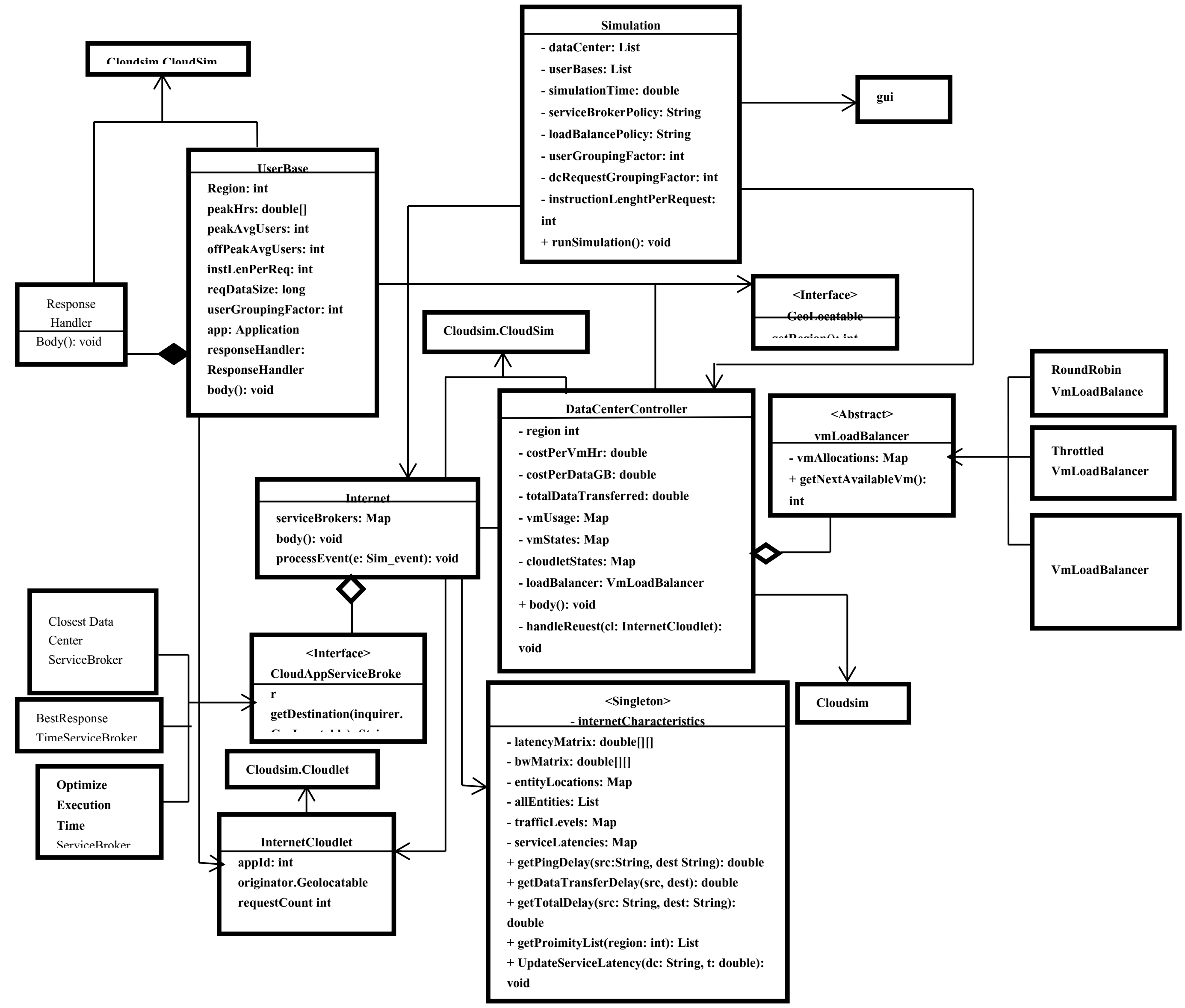
$$P = \sum_{0}^{i} J_i \tag{14}$$

where P denotes the throughput value, J denotes the job. The load balancing algorithm also calculates the task migration time and then adds it to the makespan time of the particular virtual machine that is executing the task. After submitting the entire task, the makespan time is now calculated.

```
Notation  of symbols
UVM[]= under loaded virtual machine array,
BVM[] = balanced virtual machine array,
OVM[] = overloaded virtual machine array,
UM = under loaded machine,
OM = overloaded machine.

Cloudlet Scheduling Operation
1. Get N number of task, and Sort it in decreasing order.
2. Get M number of the virtual machine, and Sort it in decreasing order of processing speed.
3. For ∀ $T_i \epsilon 0\ to\ N-1$ and virtual machine $R_j \in 0\ to\ M-1$.
4. Begin to assign task in a First come first serve order.
        End VM for loop, End task for loop

Load Balancing on Virtual Machine
5. Begin the for loop i=0 to VmSize-1 and cloudlet loop k=0 to CloudletSize-1
6. Get the load L and capacity of all virtual machine using the equations.
7. If $(L > C)$, load balancing on virtual machine cannot be possible, then use elasticity, otherwise start to check each VM.
8. Get the number of under loaded VM, balanced VM and overloaded VM.
 // UM, OM are variable UM=$C_{VM} * .25$,        OM=$C_{VM} * .8$,
9. Get the VM to transfer the task on to,
        Sort UM in increasing order and OM in decreasing order

The operation of transferring task
10 While OM!=∅ && UM!=∅,
                Begin For loop for OM
        If OM & UM exist, then transfer the tasks from OM to UM
        $T_i \rightarrow VM_j$ | Until load at $VM_j \leq OM || VM_j \geq UM$
        Also, calculate the transfer time of task ($TL_i$ / bandwidth).
11. Check the status of each VM, If there is any virtual machine that is still in overloaded condition, then repeat load balancing operation.
```

Fig. 4: Virtual Machine Load Balancing Algorithm

Figure 5 shows the class diagram of entities in HiCloud and their relationship



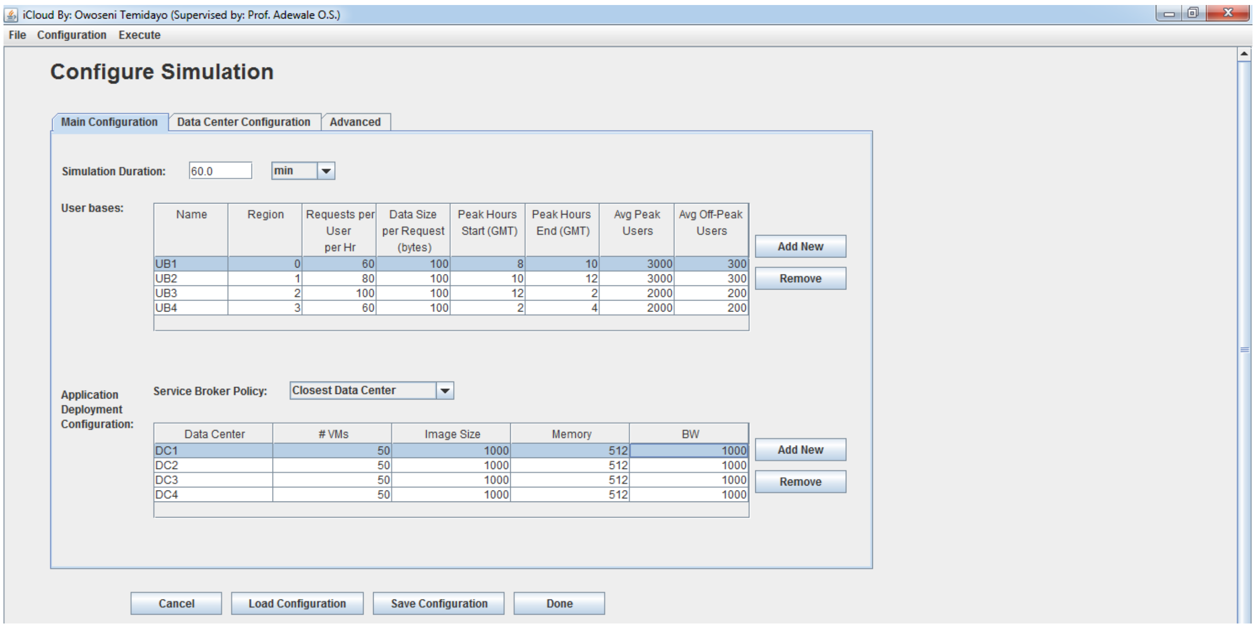
4. SYSTEM IMPLEMENTATION, RESULTS AND EVALUATION

4.1 System Implementation

The proposed system HiCloud was implemented using CloudSim toolkit 3.0 version which is written in java language. The software specification includes CloudSim toolkit 3.0 version, Java programming language, JCreator IDE. The hardware specification is Intel(R) core i3, CPU M370 @ 2.40GHz processor, 6GB RAM, 64-bit system, 500GB Hard disk and Windows 7 ultimate edition.

4.2 Simulation of the System

The simulation was carried out with different input values to show the efficiency of the system developed as shown in Fig. 6. A total of 4 userbase modelled a group of users in four geographical regions. Group of users were contained within a time zone and assumed that most users used the application during working hours for about 2 hours within the 8GMT to 6GMT. Assuming that one-tenth of simultaneous online users during the peak hours was online during the off-peak hours and each user made a new request every 5 response time, makespan time and utilization ratio after each simulation period. minutes when online. This implies that a total of 24 request/2hr was made and the data size of each request is 100byte/request. The hardware components of HiCloud were modelled into ‘main configuration’, ‘data center
configuration’ and ‘advanced’ as shown in Fig. 6, 7 and 8.
The metrics and parameters of hardware specification used in modelling cloud network in HiCloud simulator are shown in Table 1. The value for each component according to their configuration in Table 1 was inputted and simulated to generate the execution time, response time, makespan time and utilization ratio after each simulation period.



Configure Simulation

Simulation Duration: 60.0 min

User bases:

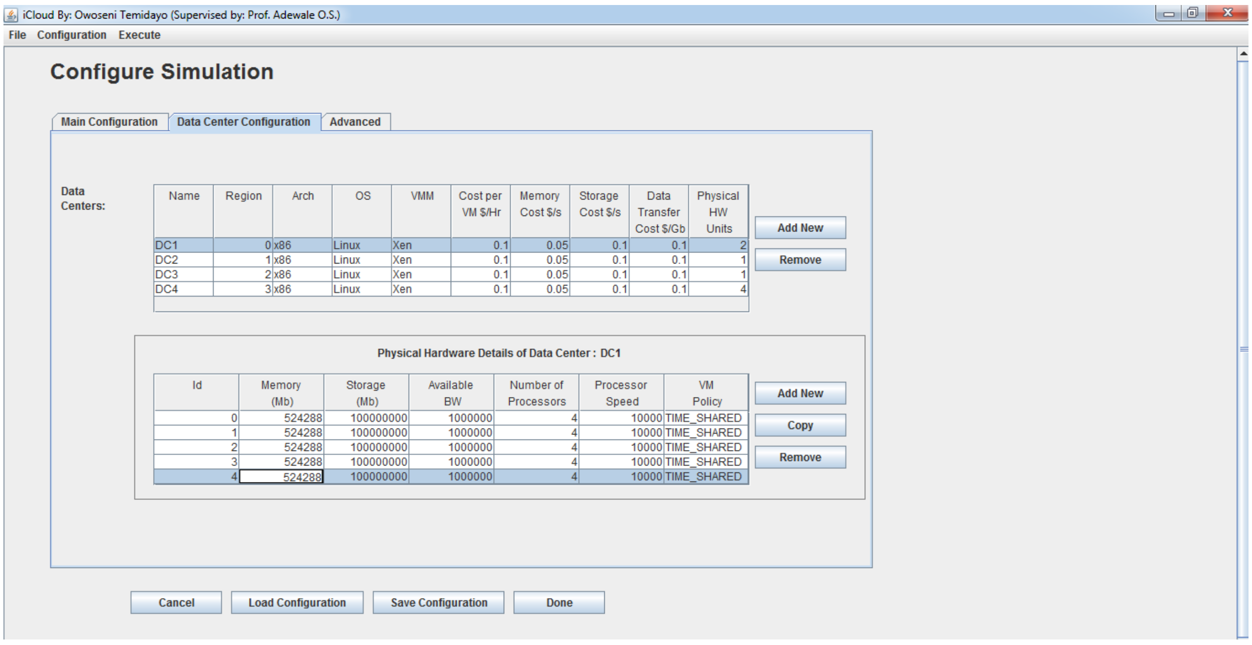
Name	Region	Requests per User per Hr	Data Size per Request (bytes)	Peak Hours Start (GMT)	Peak Hours End (GMT)	Avg Peak Users	Avg Off-Peak Users
UB1	0	60	100	8	10	3000	300
UB2	1	80	100	10	12	3000	300
UB3	2	100	100	12	2	2000	200
UB4	3	60	100	2	4	2000	200

Application Deployment Configuration:

Service Broker Policy: Closest Data Center

Data Center	# VMs	Image Size	Memory	BW
DC1	50	1000	512	1000
DC2	50	1000	512	1000
DC3	50	1000	512	1000
DC4	50	1000	512	1000

Fig. 6: Graphical User Interface of HiCloud



Configure Simulation

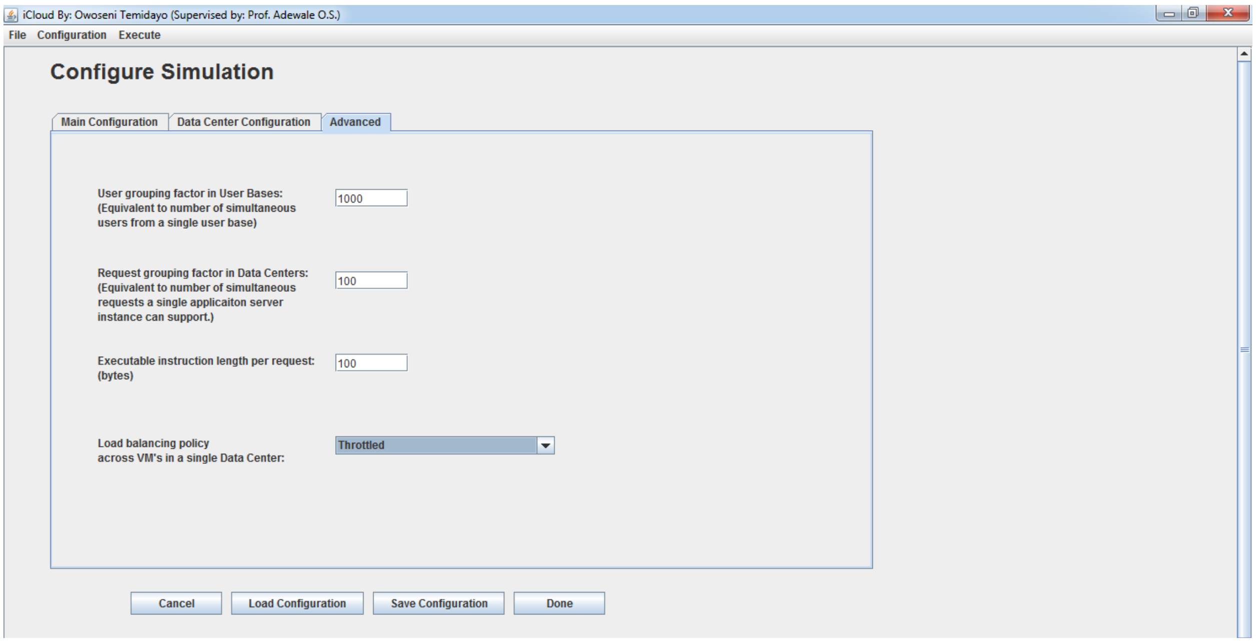
Data Centers:

Name	Region	Arch	OS	VMM	Cost per VM $/Hr	Memory Cost $/s	Storage Cost $/s	Data Transfer Cost $/Gb	Physical HW Units
DC1	0	x86	Linux	Xen	0.1	0.05	0.1	0.1	2
DC2	1	x86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC3	2	x86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC4	3	x86	Linux	Xen	0.1	0.05	0.1	0.1	4

Physical Hardware Details of Data Center : DC1

Id	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	524288	100000000	1000000	4	10000	TIME_SHARED
1	524288	100000000	1000000	4	10000	TIME_SHARED
2	524288	100000000	1000000	4	10000	TIME_SHARED
3	524288	100000000	1000000	4	10000	TIME_SHARED
4	524288	100000000	1000000	4	10000	TIME_SHARED

Fig. 7: Graphical User Interface of HiCloud


iCloud By: Owoseni Temidayo (Supervised by: Prof. Adewale O.S.)

File Configuration Execute

Configure Simulation

Main Configuration | Data Center Configuration | Advanced

User grouping factor in User Bases: (Equivalent to number of simultaneous users from a single user base)	1000
Request grouping factor in Data Centers: (Equivalent to number of simultaneous requests a single applicaiton server instance can support.)	100
Executable instruction length per request: (bytes)	100
Load balancing policy across VM's in a single Data Center:	Throttled

Fig. 8: Graphical User Interface of HiCloud

Table 1 Parameters and values used in HiCloud

Parameter	**Value**
Size of VM	1000MB
VM Memory	1024MB
VM Bandwidth	1000MB
DC – Architecture	x86
DC – OS	Linux
DC- VMM	Xen
DC–No of Machines	100
DC- Memory/ Machine	512GB
DC- Storage/ Machine	1TB
DC- Available Bandwidth/ Machine	1000MB
DC- No of Processors/ Machine	4
DC – Processor Speed	10000MIPS
DC – VM Policy	Time Shared
User Grouping Factor	1000
Request Grouping Factor	100
No of instruction per request	250
No of Userbase	4

No of data center	4
Established number of virtual machines	50
ServiceBroker	Optimize execution time
Vmloadbalancer	Task Migration

4.2 Experiment Result

In the experiment 1, ten scenarios were examined. In scenario one 10 virtual machines were used, in the scenario two 20 virtual machines were used, up till scenario 10 were 100 virtual machines were used. In experiment 1 an application was accessed by ten thousand (10,000) users with a different number of virtual machines ranging from 10, 20, …100; across 4 data centers with closest data center service broker and throttling load balancer. The performance of the data centers revealed that at 10VMs the response time was 100.31ms, at 20VMs the response time reduced to 95.60ms and when the VM got increased to 50VMs the response time of the cloud network became 83.67ms. Also at100VMs the response time got reduced to 60.95ms as shown in Table 2. The performance of experiment 1 shows a reduction in response time when the number of virtual machines are increased. This implies that for better performance the number of VMs per data center should be much to fasten the response time of the network.

Table 2 Experiment 1 Result using closest datacenter and throttling load balancer

Scenario	**No of Users**	**No of VM/4 data center with closest data center and throttling**	**Average response time (milliseconds)**	**Average execution time (milliseconds)**
1	10,000	10	100.31	39.7
2	10,000	20	95.60	31.58
3	10,000	30	91.57	30.47
4	10,000	40	87.25	29.75
5	10,000	50	83.67	28.46
6	10,000	60	80.59	27.30
7	10,000	70	76.33	26.47
8	10,000	80	72.55	25.89
9	10,000	90	70.11	24.62
10	10,000	100	60.95	23.74

In experiment 2 an application was accessed by ten thousand (10,000) users with a different number of VMs ranging from 10, 20,…100; across 4 data centers with optimize execution time service broker and throttling load balancer. The performance of the data centers shows that at 10VMs the response time was 59.68ms, at 20VMs the response time dropped to 56.37ms and when the VM got increased to 50VMs the response time of the cloud network became
45.50ms. Also at100VMs the response time was 29.90ms as shown in Table 3. The performance of experiment 2 shows a reduction in response time compared to experiment 1, this was due to allocating tasks to the data center with minimum execution time. This means that for better performance optimize execution time service broker should be employed in a cloud network. Hence the use of optimizing execution time as service broker policy in the proposed system will speed up the response time of the cloud network.

Table 3 Experiment 2 Result using optimize execution time service broker and throttling load balancer

Scenario	No of users	No of VM/4 data center with optimized execution time and throttling	Average response time (milliseconds)	Average execution time (milliseconds)
1	10,000	10	59.68	31.69
2	10,000	20	56.37	30.76
3	10,000	30	52.99	29.40
4	10,000	40	48.89	27.90
5	10,000	50	45.50	25.90
6	10,000	60	41.36	23.76
7	10,000	70	38.96	21.60
8	10,000	80	37.50	19.59
9	10,000	90	32.68	18.40
10	10,000	100	29.90	17.20

In experiment 3 an application was accessed by ten thousand (10,000) users with a different number of virtual machines ranging from 10, 20, …100; across 4 data centers with optimize execution time service broker and task migration load balancer. The performance of the data centers shows that at 10VMs the response time was 32.79ms, at 20VMs the response time dropped to 31.58ms and when increased to 50VMs the response time of the cloud network became 28.46ms. Also at100VMs the response time was 23.74ms as shown in Table 4. The performance of experiment 3 shows a high reduction in response time compared to experiment 1 and 2, this was due to allocating tasks to the data center with minimum execution time and dynamically migrates load from overloaded virtual machines to the under loaded virtual machines for faster processing. This means that; for better performance optimize execution time service broker and task migration load balancer should be used in a cloud network. Hence the use of optimizing execution time as service broker policy and task migration load balancer in the proposed system will speed up the response time.

Table 4. Experiment 3 Result using optimize execution time service broker and task migration load balancer

Scenario	No of users	No of VM/4 data center with optimized execution time and task migration (milliseconds)	Average response time (milliseconds)	Average execution time (milliseconds)	Average makespan time (milliseconds)	Average utilization ratio (percentage)
1	10,000	10	32.79	18.65	1.80	0.95
2	10,000	20	31.58	17.22	2.73	0.90
3	10,000	30	30.47	16.78	3.62	0.80
4	10,000	40	29.75	15.36	4.55	0.70
5	10,000	50	28.46	14.84	5.45	0.60
6	10,000	60	27.30	13.93	6.37	0.50
7	10,000	70	26.47	12.82	7.31	0.40

8	10,000	80	25.89	11.25	8.28	0.30
9	10,000	90	24.62	10.93	9.21	0.20
10	10,000	100	23.74	9.95	10.14	0.10

4.3 Comparison of Experiment 1, 2 and 3

In comparing the three experiments; with 10VMs and 10,000 users as shown in Table 5, the response time of the cloud network in experiment 1 reduces from 100.31 milliseconds to 59.68 milliseconds in experiment 2 and the execution time also reduces from 39.7 milliseconds to 31.69 milliseconds in experiment 2 due to the use of optimized execution time service broker policy. The better performance is now compared to the result of experiment 3. The response time of experiment 2 which was 59.68 milliseconds reduced to 32.79 milliseconds in experiment 3 and the execution time also reduces from 31.69 milliseconds to 18.65 milliseconds as a result of deploying the task migration load balancer policy. Hence the big distance between the curves in Fig. 9 and Fig. 10. When 50VMs and 10,000 users were experimented, the response time reduces from 83.67 milliseconds in experiment 1 to 45.50 milliseconds in experiment 2 and reduces to 28.46 milliseconds in experiment 3. The huge difference in the execution time of experiment 3 shows the efficiency of the task migration load balancer compared to experiment 1 and 2 when 50VMs were deployed. The performance of the cloud network with 90VMs in experiment 2 is what is being recorded when 10Vms were deployed in experiment 3. This is due to the use of optimizing execution time service broker and task migration load balancer. This means that the use of a larger number of VMs is of no importance once an optimize execution time service broker and task migration load balancer is deployed in the cloud network.

Table 5. Comparison of Experiments 1, 2 and 3 with 10,000 Users and 4 Data Centers

		Experiment 1		**Experiment 2**		**Experiment 3**			
		Closest Data Center and Throttling		**Optimize execution time and Throttling**		**Optimize execution time and Task Migration**			
No of VM	**No of Users**	**Avg. Resp. Time(ms)**	**Avg. Exe. Time(ms)**	**Avg. Resp. Time(ms)**	**Avg. Exe. Time(ms)**	**Avg. Resp. Time(ms)**	**Avg. Exe. Time(ms)**	**Avg. Mksp. Time(ms)**	**Util. Ratio**
10	10,000	100.31	39.7	59.68	31.69	32.79	18.65	1.80	0.95
20	10,000	95.60	31.58	56.37	30.76	31.58	17.22	2.73	0.90
30	10,000	91.57	30.47	52.99	29.40	30.47	16.78	3.62	0.80
40	10,000	87.25	29.75	48.89	27.90	29.75	15.36	4.55	0.70
50	10,000	83.67	28.46	45.50	25.90	28.46	14.84	5.45	0.60
60	10,000	80.59	27.30	41.36	23.76	27.30	13.93	6.37	0.50
70	10,000	76.33	26.47	38.96	21.60	26.47	12.82	7.31	0.40
80	10,000	72.55	25.89	37.50	19.59	25.89	11.25	8.28	0.30
90	10,000	70.11	24.62	32.68	18.40	24.62	10.93	9.21	0.20
100	10,000	60.95	23.74	29.90	17.20	23.74	9.95	10.14	0.10

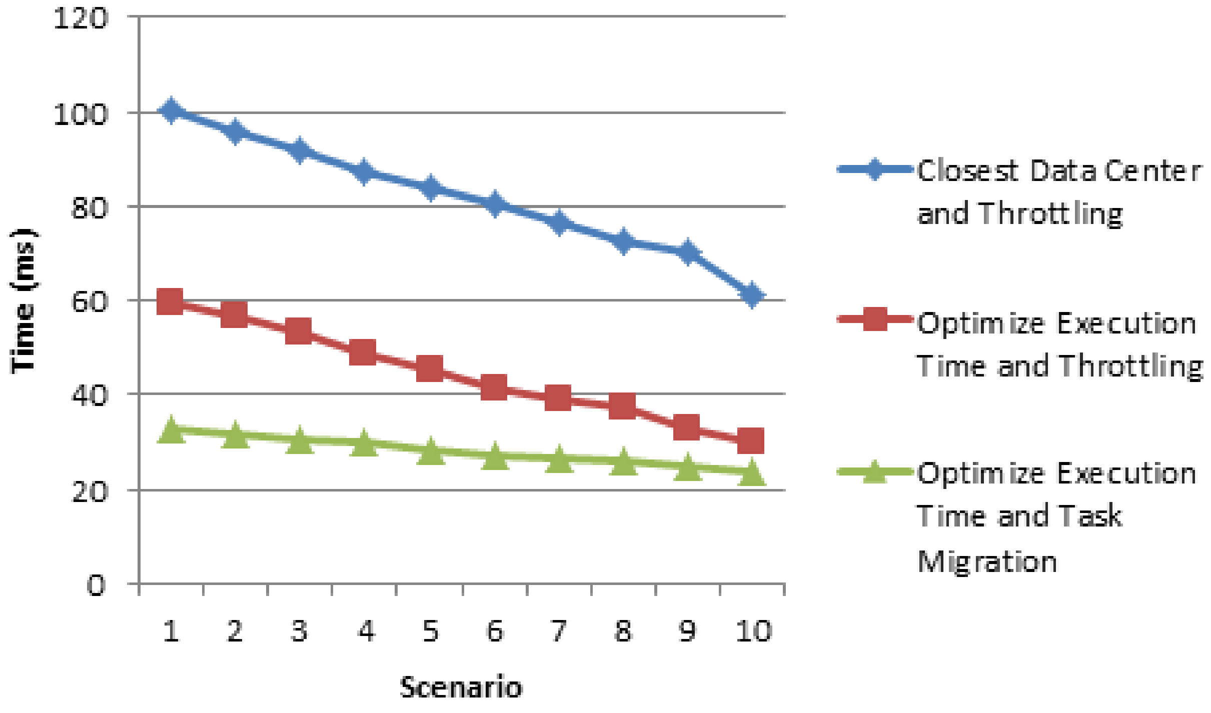


Fig. 9: Analyzing response time of experiment 1, experiment 2 and experiment 3

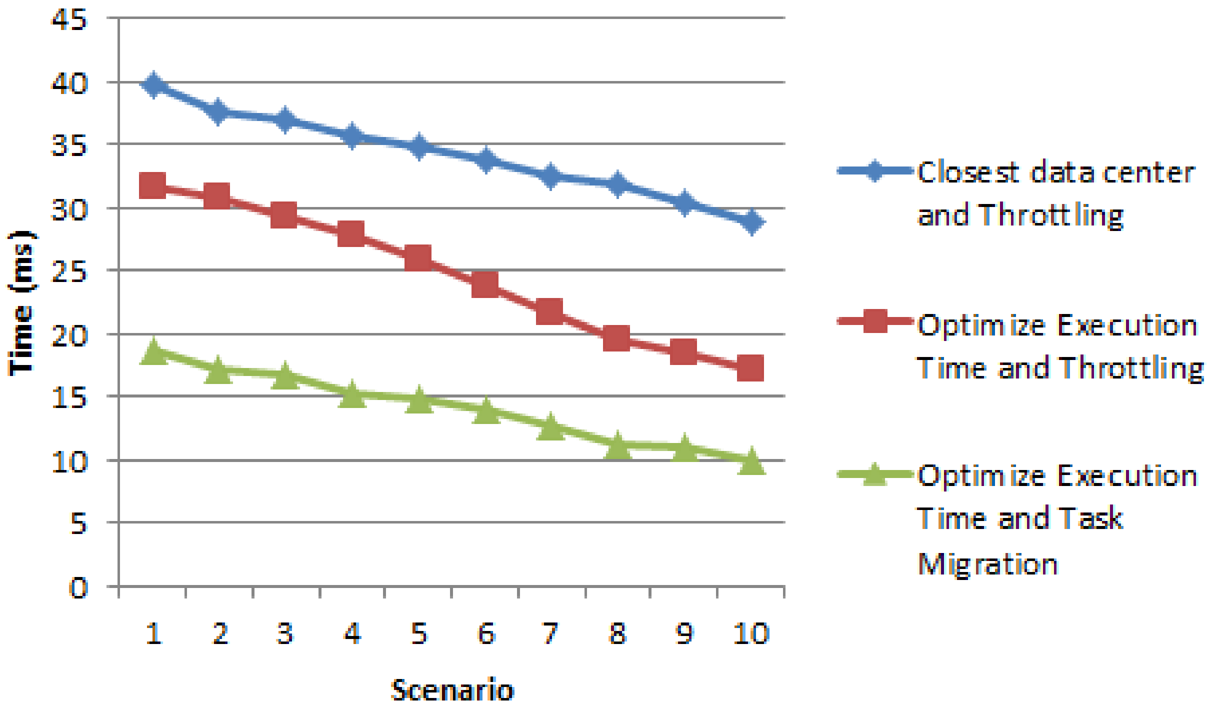


Fig. 10: Analyzing the execution time of experiment 1, experiment 2 and experiment 3

4.4 Critical Evaluation of Results

The result of experiment 1 shows that when 10,000 users use the cloud network with 10VMs using closest data center and throttling. The execution time was 39.7 milliseconds and the response time was 100.31 milliseconds, which is high compared to 31.69 milliseconds and 59.68 milliseconds when optimizing execution time and throttling was used in experiment 2. The result of optimizing execution time and task migration is the best as shown in experiment 3, with an execution time of 18.65 milliseconds and response time of 32.79 milliseconds. This result got better when the number of VMs got increased to 50VMs. The execution time dropped to 14.84ms and the response time reduces to 28.46 milliseconds. Both execution and response time of the proposed system decreases as several virtual machine increases to 100VMs but the makespan time increases because of the time taken to balance the load over virtual machines. Therefore to have an effective network a Cloud network of 50VMs with optimize execution time and task migration policy is established since at this point execution time, response time and the makespan time is minimized and also resource utilization ratio is optimized.

4.5 Comparative Analysis with Other Works

The metrics used in the existing systems were used in the developed system. With 10,000 users, optimize execution time service broker and task migration load balancer; the comparison generated is shown in Table 4.7. When the established configuration of [9] was inputted into the developed system the response time was 78.27 milliseconds instead of 125.07 milliseconds derived in CloudAnlyst. Also, when the established configuration of [10] was inputted into the developed system the response time was 31.14 milliseconds instead of 60.73 milliseconds for the iCanCloud. When the established configuration of [16] was inputted into the developed system the response time was 23.84 milliseconds instead of 45.6 milliseconds gotten in CloudSim. These results shows that iCloud has better performance value compared to existing systems.

5. CONCLUSION AND RECOMMENDATION

Cloud computing and its characteristics have been discussed in this thesis. This research work present HiCloud: a new simulation tool that allows seamless modelling, simulation, and experimenting with emerging cloud computing infrastructures and application services. The developed system allocates tasks to the data center with minimum execution time and schedules tasks to virtual machines using the task migration approach that dynamically balance the load. Experimental results revealed that the developed system has a better quality of system load balancing and the utilization of system resources. Researchers and industry-based developers can use HiCloud to test the performance of a newly developed application service in a controlled and easy to set-up environment. The performance testing of HiCloud simulator is time effective. Developers can easily model and test the performance of their application services in Cloud environments with minimum performance time and utilization ratio maximized. Security models can be introduced to the toolkit to strengthen the cloud computing paradigm.

6. ACKNOWLEDGEMENTS

This journal is an extract from my PhD thesis in the year 2021. Many thanks to my Supervisor Prof. O. S. Adewale for his effective supervision, care, patience and understanding at different stages of this research work. It has been God and you sir all the way. I also appreciate my Co-supervisor Dr. (Mrs.) F. M. Dahunsi for her contributions and encouragement. May God bless you; bless all your endeavors and your families.

7. REFERENCES

[1] Ye Z., Liu S., Yin Y., Jin Y., (2017). User-Oriented Many-Objective Cloud Workflow Scheduling Based on an Improved Knee Point Driven Evolutionary Algorithm. Journal of Knowledge Based Systems. 12(4), 16-24.

[2] Chou F., and Chou D., (2015). Cloud Computing from the Perspective of System Analysis.International Journal of Engineering Research and Applications (IJERA), 3(5), 100-115.

[3]Mohamad R. P., Kolovos D. S., and Paige R. F., (2014). Cloud Computing Workload and Capacity Management Using Domain Specific Modelling. 14th International Conference on Modelling and Simulation. IEEE DOI 10.1109/UkSim.2014.1

[4] Kashikolaei S. M. G., HosseinabadiA. A. R., Saemi B., Shareh M. B., Sangaiah G. B. (2019). An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm. The Journal of Supercomputing. https://doi.org/10.1007/s11227-019-02816-7

[5] Arabnejad V., Bubendorfer K., and Bryan N. (2017). Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources. Future Generation Computer Systems. 14(7), 56-63.

[6] Develder G., Kumar H., and Bhoi F., (2016). Dimensioning Resilient Optical Grid / Cloud Networks. Journal of Computers. 53(4), 50-58.

[7] Owoseni M. T., (2014). Design of Government Cloud Network: Case Study Ondo State. Nigerian Journal of Technology (NIJOTECH). 35(3),608-617.

[8] Calheiros R. N., Ranjan R., De Rose C. and Buyya R., (2009). CloudSim: A Novel Framework for modelling and Simulation of Cloud Computing Infrastructures and Services. International Journal of Advanced Research in Computer Science and Software Engineering, 20(7) 67-77.

[9] Wickremasinghe B., Virogho D., and Joane F., (2010). CloudAnalyst: A CloudSim-basedVisual Modeller for Analysing Cloud. Journal of IEEE Computer Society, 16(10), 1-12.

[10] Nunez A., Vazquez-Poletti J.L., Caminero AC and Castre G.G. (2016). iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator. Journal of Grid Computing, 35(4), 30-38.

[11] Talavera O. and Santisteban R. (2015). Design of Network Infrastructure of a Cloud Data Center for Use in Health Sector. International Journal of Engineering and Technology (IJET).15(8), 141-150.

[12] Mehra T. (2012). Designing and Building a Datacenter Network: An Alternative Approach with OpenFlow. IDC, Analyze the Future. Future Generation Computer Systems, 56(10), 339-347.

[13] Taifi, G., Lu, H., and Bou J. (2013). Building a rivate HPC Cloud for Compute and Data-Intensive Applications. International Journal on Cloud Computing: Services and Architecture, 13(2), 145-154.

[14] Babaoglu O., Moreno M., and Tamburini M., (2012). Design and Implementation of Peer-to-Peer Cloud System. Università di Bologna, Dipartimento di Scienze dell'Informazione Mura A. Zamboni 7(5), 52-59.

[15] Nayak S. C., Sasmita Parida S., Tripathy C., and Pattnaik P. K., (2018). An enhanced deadline constraint based task scheduling mechanism for cloud environment. Journal of King Saud University –Computer and Information

Sciences, 7(1), 33-42. https://doi.org/10.1016/j.jksuci.2018.10.009

[16] Elsherbiny S., Eldaydamony E., Alrahmawy M., and Reyad E. A., (2017). An extended Intelligent Water Drops algorithm for workflow scheduling in cloud computing environment. Egyptian Informatics Journal. 19(10), 21-29.

[17] Singh V., Gupta I., Prasanta K. Jana P. K., (2017). A Novel Cost-Efficient Approach for Deadline-Constrained Workflow Scheduling by Dynamic Provisioning of Resources. Future Generation Computer Systems 14 (4), 39-47.

[18] Shishido H. Y., Estrella J. C., Toledo C. F. M., Arantes, M. S., (2017). Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds. Journal of Computers and Electrical Engineering. 12(10), 15-24.

[19] Mehmi S., Harsh K., Vermab A.L., and Sangal T., (2017). Simulation modelling of cloud computing for smart grid usingCloudSim. Journal of Electrical Systems and Information Technology 4 (10), 159–172. http://dx.doi.org/10.1016/j.jesit.2016.10.004

[20] Chen W., Xie G., Li R., Bai Y., Fan C. and Li K., (2017), Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems. Future Generation Computer Systems 14(1), 1-8.

[21] Adhikari M., Amgoth T., and Srirama S. N., (2019). A Survey on Scheduling Strategies for Workflows in Cloud Environment and Emerging Trends. Future Generation Computer Systems 68(4), 1-36. https://doi.org/10.1145/3325097

[22] Choudhary A., Gupta I., Singh V., and Jana P. K., (2018). A GSA based Hybrid Algorithm for Bi-objective Workflow Scheduling in Cloud Computing. Future Generation Computer Systems 13 (17), 29-39.

[23] Igor L. S., Luci P., Flavia C., Delicato H., Gabriel M., O., Claudio M. F., Samee U. K., Albert Y. Z., (2019). Zeus: A resource allocation algorithm for the cloud of sensors. Future Generation Computer Systems. 92 (20), 564-581.

[24] Wang W., Zeng G., Tang D., and Yao J., (2017). Cloud – DLS: Dynamics Trusted Scheduling For Cloud Computing.Journal of Expert System with Applications. 40(16), 2310-2317.

[25] Seenuvasan P., Kannan A., and Varalakshmi P. (2017). Agent-Based Resource Management in a Cloud Environment. Journal of Applied Mathematics & Information Sciences, 11(3), 777-788

[26] Er-Raji N., Benabbou F. and Eddaoui A., (2016). Task Scheduling Algorithms in the Cloud Computing environment: Survey and Solutions. International Journal of Advanced Research in Computer Scienceand Software Engineering, 6(1), 604-608.