Consolidating Virtual Machines by using Ant Colony System algorithm for Green Cloud Computing

Thesis · October 2019

CITATIONS
0

READS
273

1 author:

Md. Kaviul Hossain
BRAC University
6 PUBLICATIONS **10** CITATIONS

SEE PROFILE

Consolidating Virtual Machines by using Ant Colony System algorithm for Green Cloud Computing

by

Md. Kaviul Hossain
17101192
Mutasimur Rahman
17101445
Azrin Hossain
17101398
Samin Yeaser Rahman
17101075

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2020

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Md. Kaviul Hossain 17101192	Mutasimur Rahman 17101445
Azrin Hossain 17101398	Samin Yeaser Rahman 17101075

Approval

The thesis titled "Consolidating Virtual Machines by using Ant Colony System algorithm for Green Cloud Computing" submitted by

1. Md. Kaviul Hossain (17101192)
2. Mutasimur Rahman (17101445)
3. Azrin Hossain (17101398)
4. Samin Yeaser Rahman (17101075)

Of Summer, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 5, 2020.

Examining Committee:

Supervisor:
(Member)

Dr. Md. Motaharul Islam
Professor
Computer Science & Engineering
United International University

Head of Department:
(Chair)

Mahbubul Alam Majumdar
Professor and Dean
Department of Computer Science and Engineering
Brac University

Abstract

Cloud computing is an internet based computing system enabling convenient, flexible, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interactions. Although cloud is a virtual and non-physical network-based concept, its implementation includes physical and wired connectivity among servers, hard-drives and other network devices resulting in large number of data centers. As such power and energy consumptions increase total carbon footprint which is very hazardous for the environment. Green Cloud Computing is a concept of consuming less physical power and saving energy by making the infrastructure more virtual. With the help of the Ant Colony System (ACS) algorithm, it is possible to consolidate the Virtual Machines and make Cloud Computing more virtual than it already is. This algorithm describes a mechanism of indirect coordination, through the environment, between agents or actions e.g. communication about paths between ants with the help of pheromones. Using this technique, the Virtual Machines (VMs) that constitute Cloud Computing can be amalgamated and consolidated. Thus, Green Cloud Computing can be easily globalized.

Keywords: Virtual machine, hyper-visor, state-transition rule, physical machine, meta-heuristic, virtualization.

Dedication

Dedicated to every single soul who works hard to bring a change to the world of science.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, to our advisor Dr. Md. Motaharul Islam sir for his kind support and advice in our work. He helped us whenever we needed help.
Thirdly, to Marco Dorigo, researcher and inventor of the Ant Colony System algorithm, who helped us in our research by providing necessary resources.
And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ACS Ant Colony System

$AIVMM$ Active Idle Virtual Machine Migration

AS Ant System

BFD Best Fit Decreasing

CDN Content Delivery Network

CE Cloud Exchange

CO Carbon Monoxide

CO_2 Carbon-di-oxide

COP Combinatorial Optimization Problem

CPU Central Processing Unit

EC Evolutionary Computation

GA Global Agent

$HACOPSO$ Hybrid Ant Colony Optimization with Particle Swarm Optimization

$IaaS$ Infrastructure as a Service

IDE Integrated Development Environment

LA Local Agent

$MACO$ Multi-objective Ant Colony Optimization

$MIMPS$ Migration of Idle Machines via Parking Server

NIC Network Interface

OEM Order Exchange and Migration

$OEMACS$ Order Exchange Migration Ant Colony System

$PaaS$ Platform as a Service

PM Physical Machine

PSO Particle Swarm Optimization

QoS Quality of Service

RAM Random-access memory

$SaaS$ Software as a Service

SLA Service Level Agreement

$SLAV$ Service Level Agreement Violation

TSP Travelling Salesman Problem

VM Virtual Machine

VMC Virtual Machine Consolidation

VMM Virtual Machine Monitor

VMP Virtual Machine Placement

VMs Virtual Machines

XML eXtensible Markup Language

Chapter 1

Introduction

1.1 Motivation

Cloud Computing is the future of online storage and processing. The most important feature of today's storage system and remote accessibility is its ubiquitous availability and multi-tenancy, which grows more valiantly with the help of cloud computing. The salient features and gradual development of cloud computing was our main motivation to work in this field. Communication, storage and accessibility has become piece of cake because of the bulky characteristics of Cloud Computing. Big data management has been a notable success of this field. Although there remain some drawbacks in the system, the problem with high power and energy consumption by the idle virtual machines and emission of harmful gases like CO_2, CO, Methane etc. in a data center has motivated us to work on it and leave noticeable changes in that aspect. Our motive is to design an algorithm that will not only work perfectly with the current one but also solve the high-power consumption problem with remarkable changes. From our background study of the system, we learnt that many researchers have worked with the problem at different times and they came up with both single and multi-objective algorithms to fix the problem. Thus, we aimed higher and tried to introduce a self-explanatory system or algorithm called Active and Idle Virtual Machine Migration algorithm that not only solves the existing glitch but also makes the process more efficient and consolidated. We have discussed our contributions elaborately in the next section (1.2).

1.2 Major Contributions

Most researchers have introduced different algorithms to address the problem of excess energy consumption of idle virtual machines. They have used algorithms like multi-objective ant colony optimization (MACO), hybrid ant colony optimization with particle swarm optimization (HACOPSO) and much more to fix the problem. [1] But at a certain point, the algorithms fail to solve the problem with all the objectives intact. In order to sort that problem and address the existing drawbacks of the algorithms, we came up with such an algorithm that can consolidate the cloud architecture by making virtual machines consume energy in an efficient way and also keep the integrity of the system intact and unchanged. To do such, we have maintained integrity of the cloud architecture and consolidated the virtual machine placement problem in collaboration of a meta-heuristic algorithm called the Order

Exchange Migration Ant Colony System (OEMACS). We have also ensured Green Cloud Computing by reducing high thermal throttle in physical machines like servers that host the Virtual Machines in a data center. We successfully migrated idle or zombie VMs from one server to another containing maximum idle VMs, which has ensured no unwanted lag of power supply for the actively working VMs.

1.3 Thesis Overview

The subsequent sections of the thesis have been organized as follows. Chapter 2 is the literature review which contains related existing approaches along with the problem statement relevant to our proposed model. Chapter 3 includes all the background information related to our work and how we are using the resources to get the desired output, along with a review of all the existing similar models that researchers came up with at different times. Section 3.1 contains the introduction to Cloud Computing along with its salient features and architecture. Section 3.2 is the section discussing the process of virtualization and its collaboration with Clouds. Section 3.3 is the elaborated narration of the Ant Colony System Algorithm used to design our proposed model. In chapter 4, we have described the proposed model of our works along with relevant graphs and figures. It includes the overall models relevant to our solution algorithm. The predicted results and relevant discussions are showed in chapter 5. Lastly, the summary of the report and conclusion as well as some future work-plan is done in chapter 6. In future work part, we will talk about our future ambition and working plan to upgrade our proposed model for better performance.

Chapter 2

Literature Review

Cloud Computing has made our lives easy and simple as we can access our important files, documents, folders and all other on the go resources, anywhere and everywhere we go. But the system itself is not that simple as it seems. Cloud Computing is a complex computing technique with a sophisticated architecture and vast service domain. The whole architecture can be divided into two sub-domains: physical and virtual. A cloud computing system is comprised of both physical machines (servers) and virtual wares (VM-wares). With the help of an intermediate application named Hypervisor, the mainframe of a server is converted into 'N' virtual machines. A server has four basic components: CPU (Processor), RAM (Memory), Hard-Drive (Storage) and NIC (Network Interface). Hypervisor virtualizes these four components of a physical machine and divides them into 'N' virtual machines. The value of n for every physical machine depends on the configuration of the four basic components a server is composed of.

Cloud computing is a relatively new computing paradigm. It leverages several existing concepts and technologies, such as data centers and hardware virtualization, and gives them a new perspective. Cloud computing provides three service models and two deployment models. The three service models are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Similarly, the two deployment models are private cloud and public cloud. Furthermore, on the basis of combination of the major deployment models, two more models can be constructed. They are hybrid clouds and community clouds.

With its pay-per-use business model for the customers, cloud computing shifts the capital investment risk for under or over provisioning to the cloud providers. Therefore, several public IaaS, PaaS, and SaaS cloud providers, such as Amazon, Google, and Microsoft, operate large-scale cloud data centers around the world.Moreover, due to the ever-increasing cloud infrastructure demand, there has been a significant increase in the size and energy consumption of the cloud data centers. High energy consumption not only translates to a high operating cost, but also leads to higher carbon emissions. With the rapid growth in the number and size of cloud data centers, the energy consumption as well as equipment cooling costs has arisen to new heights. Studies have shown that data centers around the world consumed 201.8 KWh of electricity in 2010, enough to power 19 million average household [4]. This consumption accounted for 1. 1%-1.3% of the worldwide total and the rate was expected to increase to 8% by 2020[4]. At present, most cloud servers utilize between 11% and 50% of their total resources most of the time. The power consumed by an

active but idle server is at the ratio between 50% and 70% of a fully utilized server [4]. Therefore, placing the VMs of a lowly utilized server onto other servers and gracefully scheduled down the lowly utilized server will efficiently reduce the power consumption.

In recent years, some attempts have been made to reduce the energy consumption of data centers. One effective and commonly use method is consolidation. Server consolidation is a technique of reducing the energy cost of the data center by migrating active but idle virtual machines to fewer servers. All the active but not working virtual wares are migrated to a single server and put into low power mode while the actively working virtual machines are migrated to a single server. As a server with 'N' fully loaded virtual machines are taking the full load whereas the idle machines are hibernating in a completely different server. There are many existing methods of server consolidation. The main conception of these methods is to use active virtual machine migration to consolidate virtual machines periodically, so that some low-load physical machines can be released and then terminated or hibernated. The main research of dynamic server consolidation is to decide when to reallocate virtual machines from overloaded physical machines, as this directly effects resource utilization.

In our paper, we are focusing on executing the server consolidation method by combining Order Exchange and Migration (OEM) paradigm with a complex but efficient algorithm called the Ant Colony System (ACS) algorithm. Ant Colony System algorithm or the ACS algorithm is a versatile, robust, heuristic and population-based approach towards solving the Travelling Salesman Problem (TSP). In this algorithm, 'm' ants (algorithm agents) focus on determining the shortest path from their source i to destination j by undergoing s iterations in time t. In real life, ants use pheromones to mark their paths between their food source and homes. The level of pheromones on their paths act as their stigma of influencing more ants to follow that path. Using this very unique techniques of ants, Mark Dorigo (1961) developed an algorithm of shortest path trace in machines called the 'Ant Colony System' algorithm.

Chapter 3

Background Study

3.1 Cloud Computing

Cloud computing is an internet-based computing system enabling convenient, flexible, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interactions. It is the process of storing, sharing and controlling data and files in a remotely manner where a user can share their data with a wide pool of other users. Although cloud is a virtual and non-physical network-based concept, its implementation includes physical and wired connectivity among servers, hard-drives and other network devices resulting in large number of data centers.

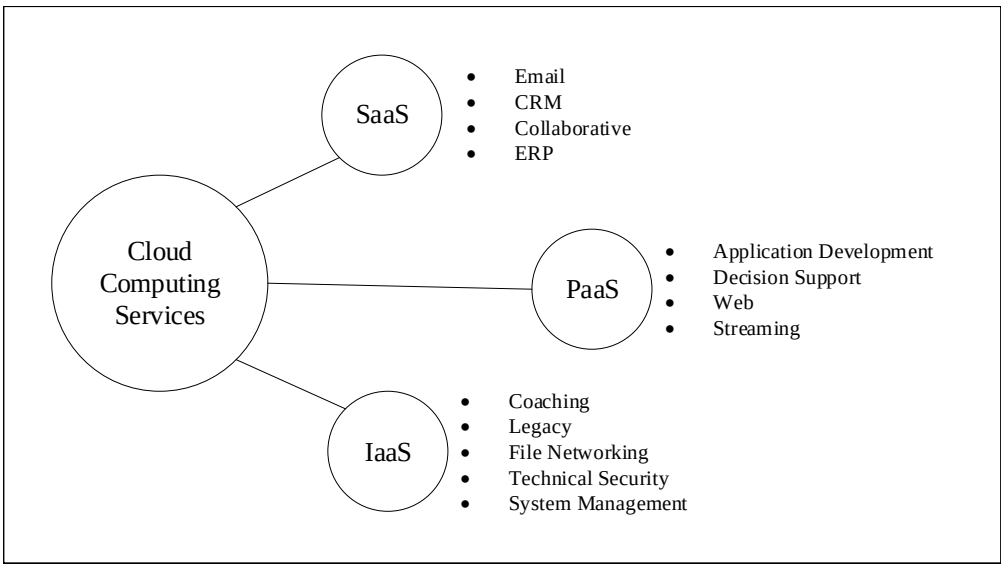

Figure 3.1: Cloud Computing

3.1.1 Salient Features of Cloud Computing

Salient features of cloud computing include on-demand service, resiliency, ubiquitous access, multi-tenancy, self-serviced and pay as you need properties. Cloud is an enormous entity hosting more than ever seen before data that can be accessed from

anywhere on earth through internet. Anyone anywhere at anytime can access the cloud and store or preserve any data needed. A user can not only store data in a cloud but also share the data with a pool of other users which makes cloud computing a multi-tenant concept.

Whenever a server faces power shutdown due to power failure or regular maintenance, the data is stored and migrated to another server. As such, a customer or user can access the data anytime they want even if a server is down. Cloud computing is resilient technique of data management. One can easily use this platform and become an expert in it. It is a ubiquitous computing model that allows a person to access and use it from any corner. Any major failure of power or connectivity does not harm the overall architecture and its functionality.

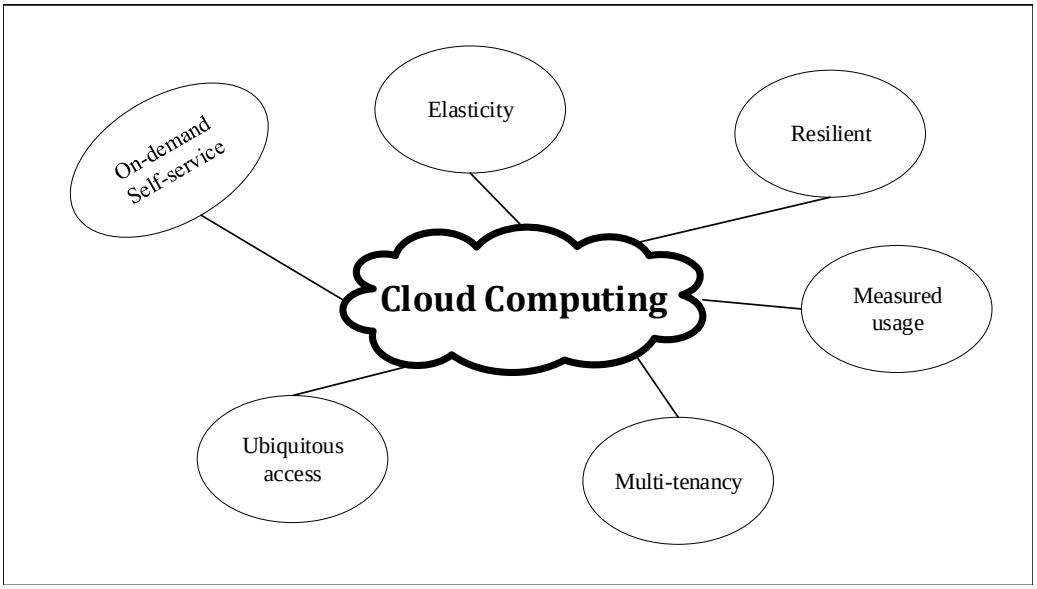

Figure 3.2: Features of Cloud Computing

3.1.2 Cloud Computing Framework Architecture

The resources of physical cloud and center middle-ware abilities together provide the reason for conveying IaaS and PaaS. SaaS capabilities are provided by the user-level middle-ware. The SaaS/PaaS service providers are different from the IaaS service providers, usually third-party service providers develop PaaS/SaaS services.

Cloud applications: Cloud layer consists of applications that can be accessed by the end-users. End-users can use the SaaS applications over the internet which is provided by the SaaS providers. In this layer, the users can send their possess application as well.

User-level middle-ware: Different software frameworks, for example Ruby on Rails, Asynchronous JavaScript and XML (AJAX) are incorporated in this layer. These frameworks help developers to build a project efficiently. This layer provides platforms for cloud software development, modeling, documentation and other composition tools as well. Additionally, frameworks from this layer help the application from the upper level.

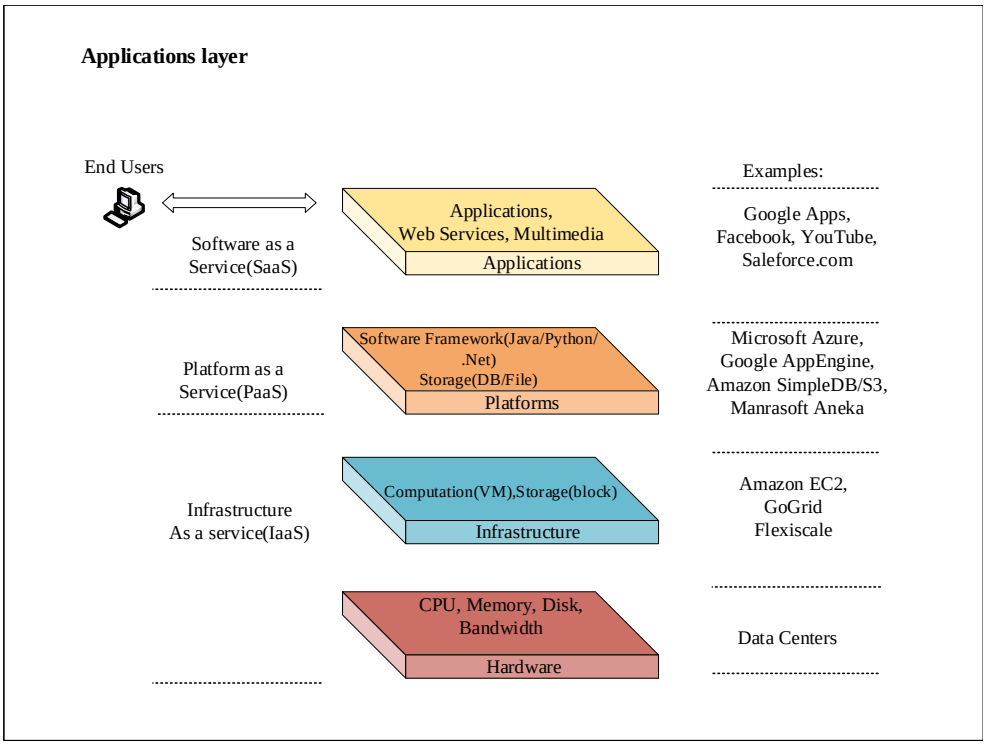

Figure 3.3: Application Layer of Cloud Computing

Core middle-ware: For facilitating the user-level services, the core middle-ware provides an appropriate run-time environment for applications. The core middle-ware services include remote management, execution management and monitoring, storage system, security services. Few of the references are Manjrasoft Aneka, GoGrid, Amazon EC2 and S3. Manjrasoft Aneka provides the customer with a platform for developing applications hosted in the cloud. GoGrid or Amazon EC2 and S3 offers the customer with virtual hardware and storage on top of which they can build their infrastructure (IaaS).

System level: This is the bottom layer of the cloud architecture that focuses on Infrastructure-as-a-Service (IaaS). It provides access to computing resources that includes storage servers and application servers. These servers are handled by top-level virtualization.

3.1.3 Cloud Modelling

CloudSim is a framework for modeling and simulation of cloud computing infrastructure. According to CloudSim, entity is an instance of a component. IaaS can be simulated by increasing data center entity. Data center entity controls host entities. The hosts are allocated to one or more VMs. The VM allocation policy assigns the hosts to VMs. Comparatively, one or more application services can be allocated within single VM instance. A data center can control few hosts. Host works as a physical server and a component of CloudSim.

VM allocation generates VM instances on hosts that match software environment, specifications and characteristics of the SaaS provider. CloudSim supports the improvement of custom application service models and the clients have to expand the

core cloud-let object for executing their application service[14]. CloudSim offers customizable policies for provisioning host and resources to virtual machines on a first-come-first-serve service, hardware requirements. Although, policies used by public cloud providers are not available publicly.

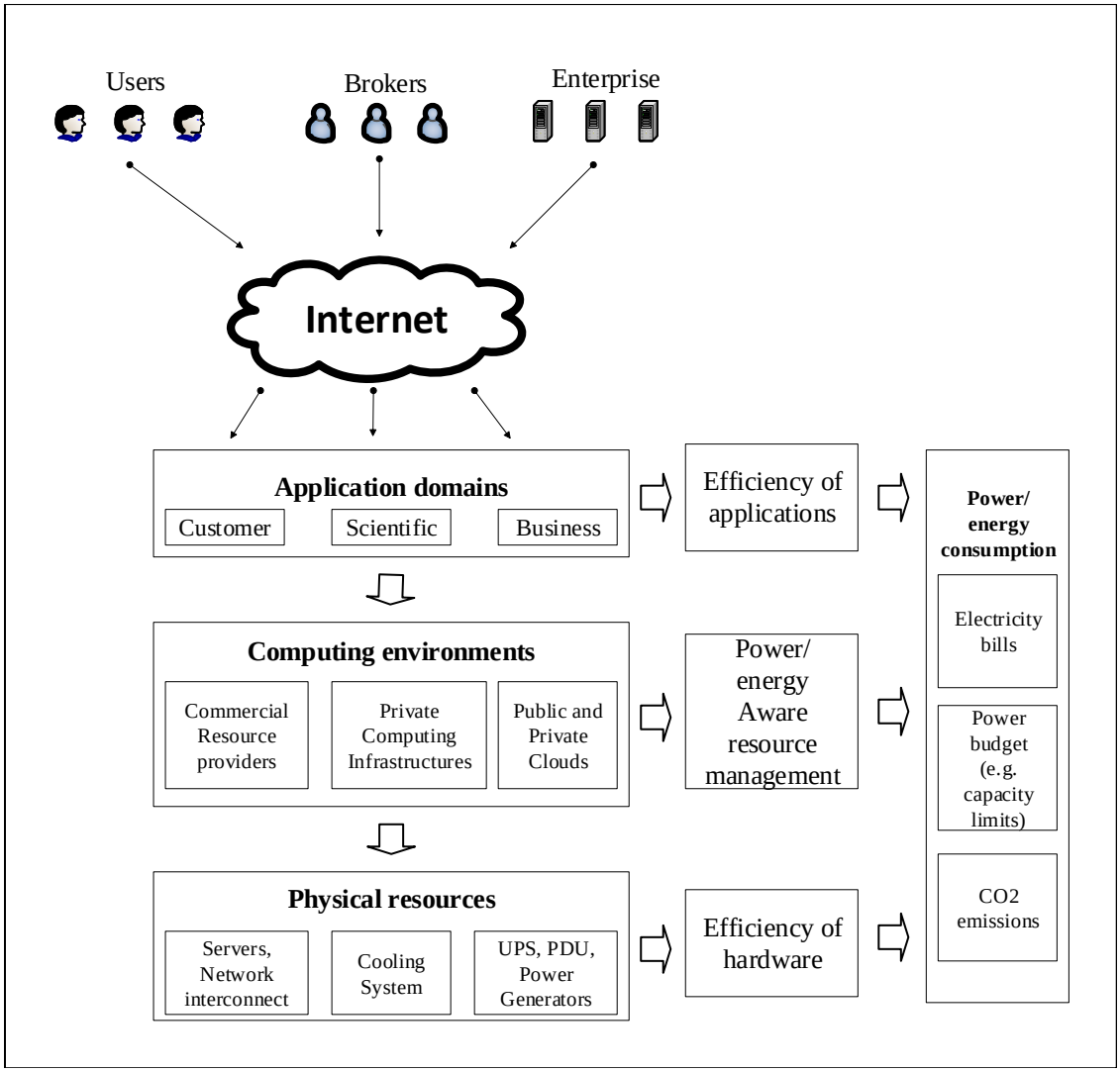

Figure 3.4: Cloud Modelling

Host allocation policy does the provision of processing cores to VMs for each host component. The policy comprehends several hardware requirements such as memory storage, CPU share, number of CPU cores. CloudSim supports the idea of allocating particular CPU cores to particular VMs, assigning the capacity of a core among VMs dynamically or on demand. These ideas are known as space-shared policy and time-shared policy. Each host component represents a VM scheduler component. The scheduler component can be executed either by the space-shared policy or the time-shared policy.

3.1.4 Inter-Networking of Clouds

To ideally serve client needs around the world, cloud service suppliers have a few centers at distinctive location over the internet. Federated cloud provides remarkable

execution and economic benefits by enhancing the capability of SaaS providers and present better service by optimizing the service. Cloud providers do not need to set up a new data center in every location since every member is allowed to obtain additional resources from federation dynamically.

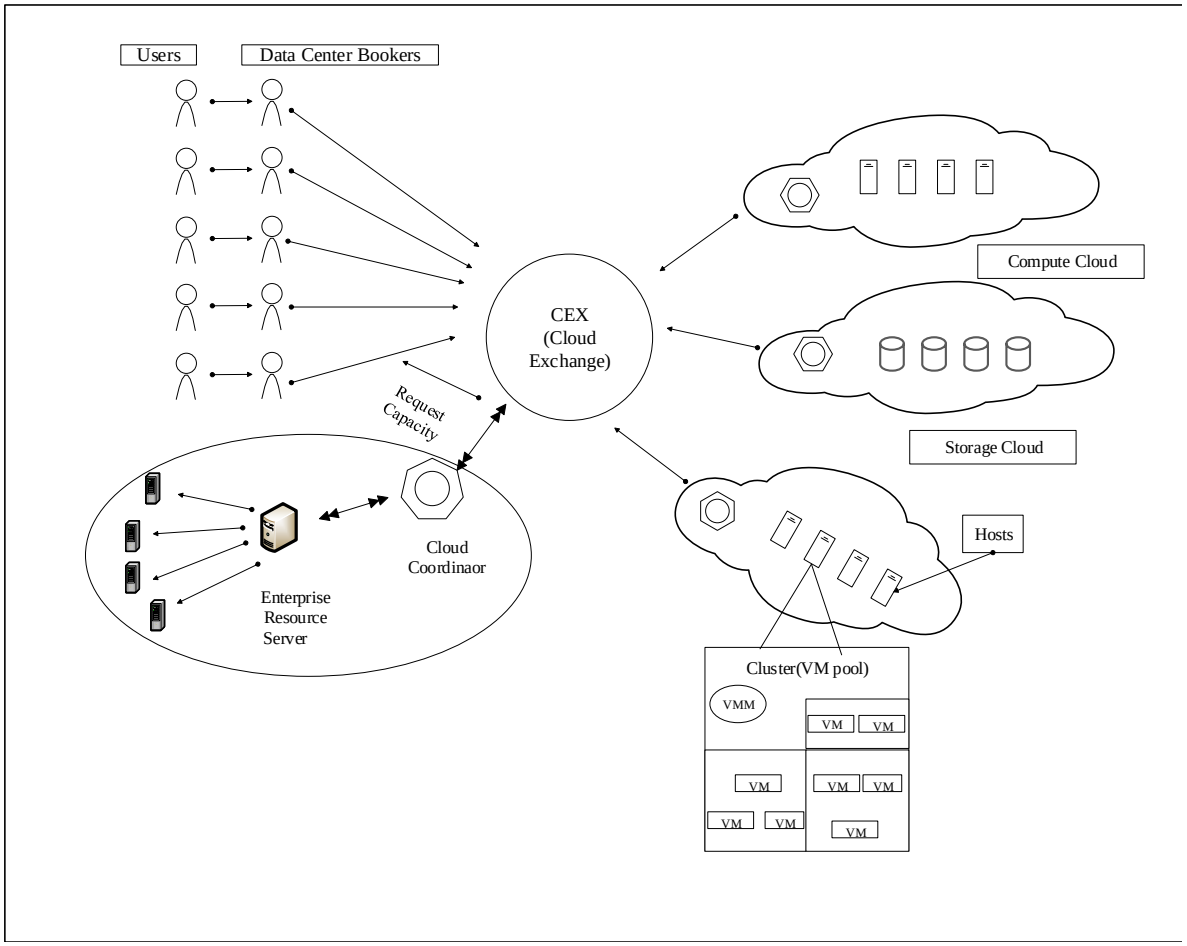

Figure 3.5: Inter-Networking of Cloud

Cloud coordinator is one of the major elements of clouds and their federated network. Cloud coordinator is responsible for exporting cloud services to the federation by implementing basic functionalities such as scheduling, virtualization, dynamic monitoring, application composition etc. Cloud Exchange (CEx) facilitates the cloud brokers acting on behalf of the SaaS providers by identifying appropriate cloud service[16]. Various social networks and content delivery networks (CDNs) may benefit from the federated cloud service architecture[19]. Social networking sites like Facebook, MySpace are using multi-tiered web application. These sites are providing dynamic services to lots of end-users. All component will run on a different virtual machine that can be hosted in data centers maintained by different cloud service providers. Every developer can choose their cloud service provider according to their plug-in. As a result, a social networking site is structured by a large number of diverse services that may be hosted by lots of data centers around the world.

3.1.5 Green Cloud Computing

The process of ensuring low power consumption and high resource utilization along with proper and regular outcome is basically green computing. The concept of green cloud computing is a part of green computing as servers and physical hardware used in the process emits CO2 and other carbon-based gases that are highly hazardous for the environment. According to Bioscience Biotechnology Research Communication (BBRC), consumption of power and energy by server in cloud data centers are increasing at an alarming rate. Fig.(3.6) shows the scenario in a graphical form.

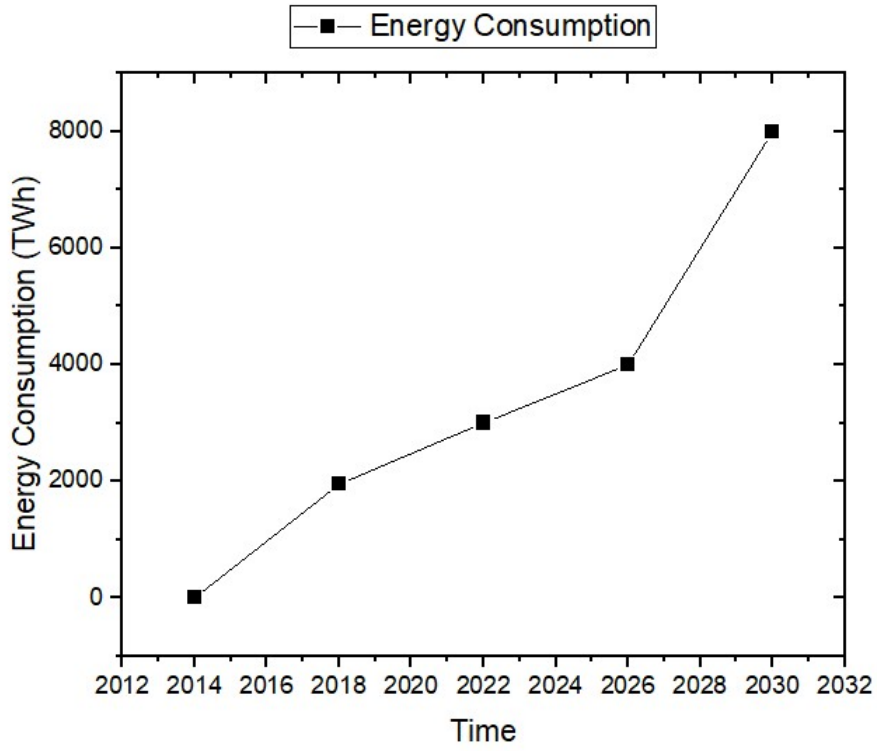

Figure 3.6: Energy Consumption by Cloud Data Centers

In order to ensure a green cloud computing environment, we have to stop the high-power consumption of idle virtual machines in the cloud servers. Whenever a VM is assigned a task, it takes memory and processing powers from the under-laying servers. After the task is completed, the VM becomes idle but it does not stop consuming that resources completely. Rather it keeps using the resources mostly memory to stay idle. As such, power is consumed which causes more greenhouse gas emission.

In order to stop excessive green-house gas emission and ensure green cloud computing, we introduced the Active & Idle Virtual Machine Migration algorithm that helps virtual machines to utilize resources to a great extent and makes idle virtual machines migrate to a server with highest number of idle VMs in it.

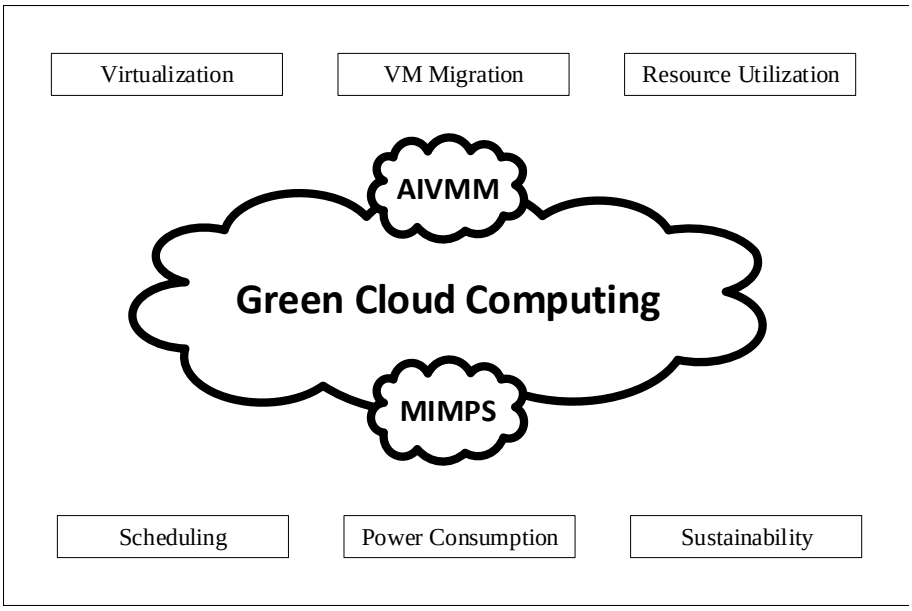

Figure 3.7: Green Cloud Computing

3.2 Virtualization

3.2.1 The Process of Virtualization

Virtualization is the process of virtualizing an operating system or an entity like software, hardware, kernel etc. with the help of an intermediary application known as 'hypervisor' so that the resources of a physical machine can be properly utilized. Virtualization is a modern technique which is very efficient and progressive. With the help of virtualization, physical resources like RAM, CPU and hard-drive storage can be properly utilized[22][23].

The process of virtualization is of two types- full virtualization and para-virtualization. The under-laying physical machine with the main operating system is called the host machine. With the help of a software or application called the hypervisor, the host operating system is virtualized and so are the physical resources. As such, more than one virtual machines are created on top of the physical host and each of the virtual machines are assigned processor, memory and storage.

There are four major components that a machine needs to operate properly. They are processor, memory, storage and network bandwidth. While installing a virtual machine over a physical machine, these four components are allocated to the VMs. A virtual machine can perform all the tasks that a physical machine can. The main difference between these two types of machine is the virtualized property itself.

Full virtualization uses an unmodified version of the guest operating system. The guest addresses the host's CPU via a channel created by the hypervisor. The guest then communicates directly with the CPU. This process is the fastest virtualization[21]. On the other hand, paravirtualization uses a modified guest operating system. The guest communicates with the hypervisor and the hypervisor passes the unmodified calls from the guest to the CPU and other interfaces, both real and virtual.

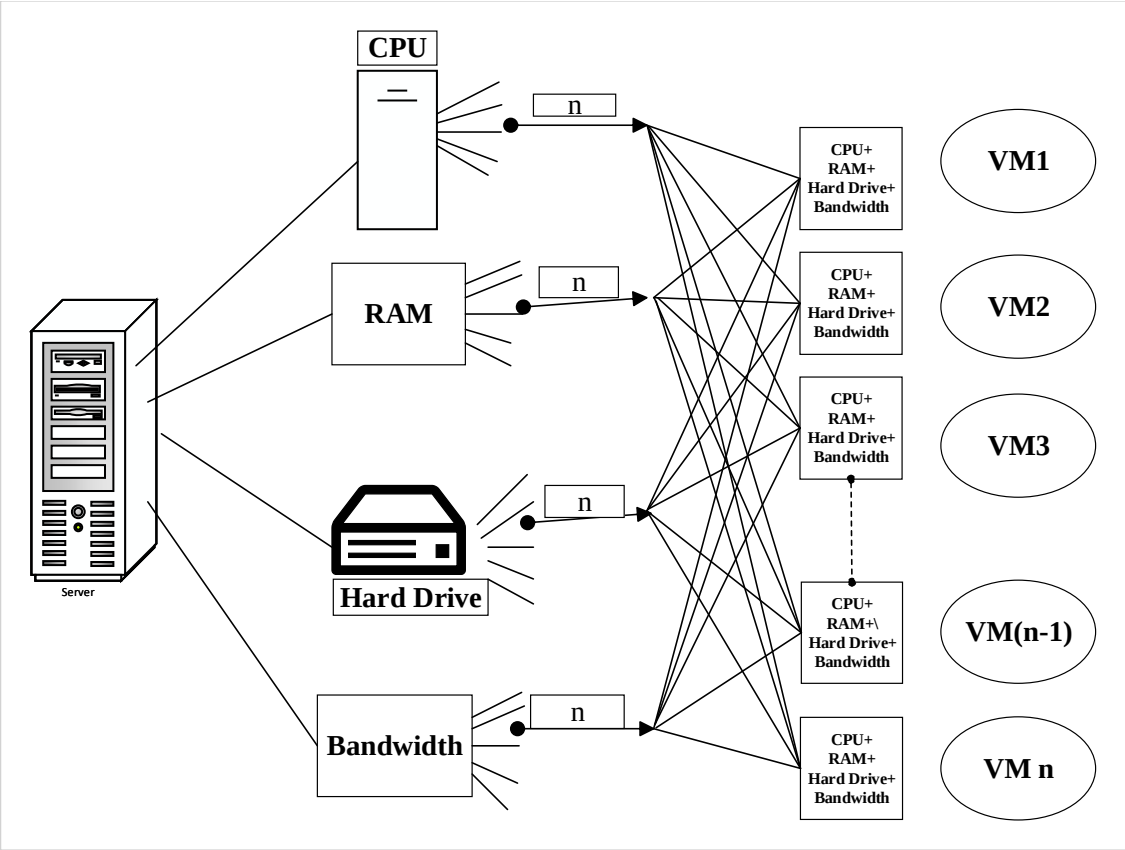

Figure 3.8: Virtualization

3.2.2 Virtual Machine allocation in Cloud Computing

The infrastructure of cloud computing and grid computing is different because of virtualization tools. Additionally, cloud has an extra virtualization layer. VMs are isolated yet they share processing cores and system bus. Usually, each VM use the processing power that is available within the host. However, VM can demand more processing power than the availability. To resolve this issue, CloudSim offer VM allocation in two different levels: host and VM level. At first level, the processing power of each core of each VM is determined. Then, at VM level, the VM distributes the available power to individual application services. Here, we will assume a task unit as an individual application service within the VM.

With the help of a software named CloudSim, it has become extremely easy to allocate virtual machines in the Cloud architecture. There are two provisioning policies maintained by CloudSim[16]. They are- time-shared and space-shared policies. CloudSim applies these policies in all the levels. In time-shared policy, a VM completes the tasks assigned to it with respect to time. It complete one task at a time and moves to the new one. While in space-shared policy, a VM can complete more than one cloudlet (task) simultaneously by sharing the resources. fig 1.8 depicts a provisioning scenario where the space-shared policy is applied to both virtual machines and cloudlets.

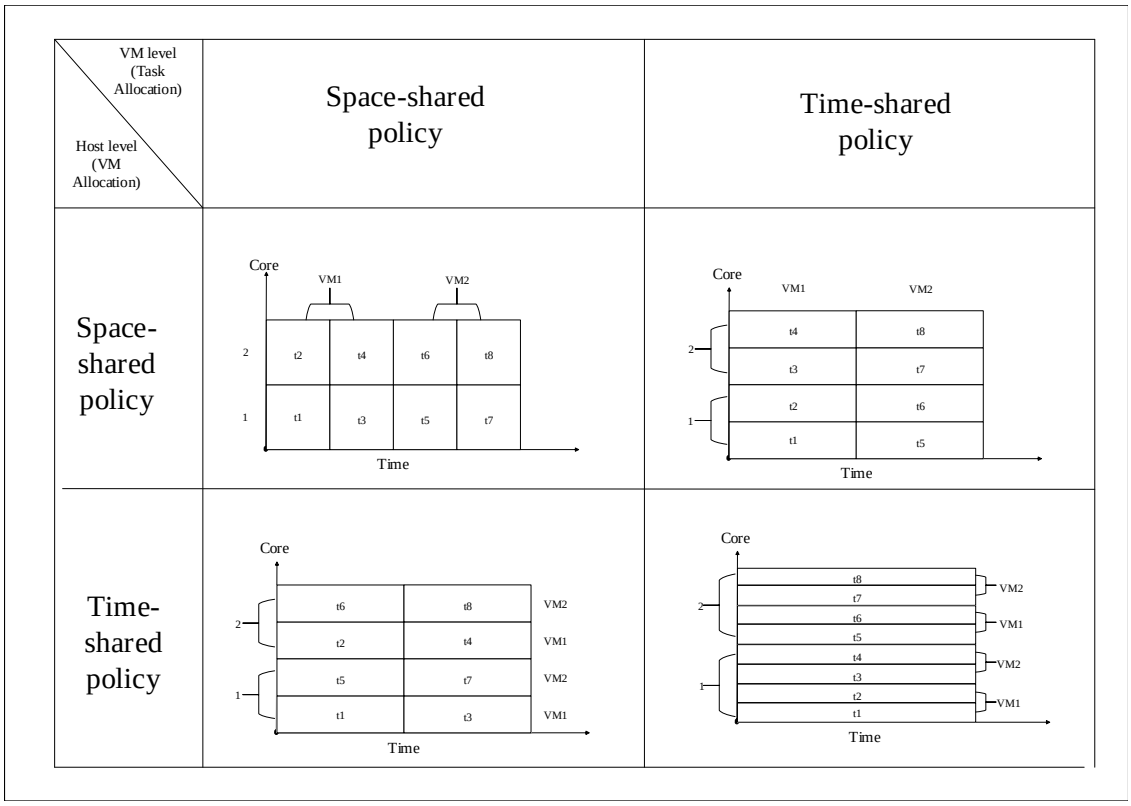

Figure 3.9: Scheduling Policy

3.2.3 Virtual Machine Migration

Virtualization permits dividing one physical machine into multiple virtual machines that shares the same physical resources and runs simultaneously as well. Due to avoid load balancing and make the physical machine fault tolerant, virtualization allows virtual machine (VM) to migrate from one physical machine to another physical machine. To reduce energy consumption of cloud data centers, virtual machine consolidation is an emerging solution. VM consolidation is a technique that combines various virtual servers to run on a physical server at the same time.

Live Virtual Migration:
Live virtual migration is the movement of a virtual machine from one physical server to another server without any downtime. It helps in load balancing and also ease VM installation across physical machine in data center. This is effective in several schemes. VMs can be migrated from a failed physical machine to a stable one[18]. Idle VMs can be moved to another physical machine to optimize resource utilization. In case of load balancing, VMs can be migrated across multiple machines.
Live migration can take place in two processes:

- switching the control from source host to destination host
- move the data storage including virtual disks to the destination host.

The two common approaches can be distinguished by the processes mentioned earlier:

1. **Pre-copy:** At first, the memory is shifted and then the execution is shifted. The memory is shifted by a course of iterations. When the migration proceeds, the memory pages starts to get checked and then transferred from the source host to the destination host. The VM runs on the source host while shifting the memory pages. The pages that have been dirtied (changed) during the iteration are re-copied to the destination until the number of dirty pages is lower than the re-copied pages. After that, the VM will stop on the source host and the last memory pages are shifted to the destination. The execution state like CPU state, registers are shifted to the target host. Then the VM will resume on the destination host. In pre-copy, higher number of dirty pages can slow down the performance.

2. **Post-copy:** In post-copy, the execution is shifted first, then the memory is shifted. The VM is stopped on the source host. The subset of the execution state is transferred to the destination first. The execution is resumed at the destination. At the destination host, if the VM tries to access any page that has not moved to the destination yet, then it will create a fault page known as network faults. Network pages fault can bring degrade in performance. In post-copy, the state of VM is assigned over both source and destination host.

Total migration time, down time, preparation, resume these factors are used to examine the performance of live migration.
There is another hybrid method that brings out the best from pre-copy and post-copy method. There are few state information like devices and frequently accessed pages of the VM, is added to the processor state. This approach decreases the number of network fault pages. This method also transfer a set pages in its locality when network fault page is found.

3.3 Ant Colony System Algorithm

Initially proposed by Marco Dorigo in 1992 in his PhD thesis, the Ant Colony System (ACS) algorithm was aiming to search for an optimal path in a graph, based on the behavior of artificial ants seeking a path between their colony and a source of food .In the field of operations research, the ACS algorithm is a probabilistic technique for solving computational optimization problems with the help of graphs. In this algorithm, artificial ants stand for multi-agent methods inspired by the behavior of real ants. The pheromone-based communication of biological ants is often the predominant paradigm used.

The main reason of selecting the ACS algorithm for our research is that our main problem i.e the Virtual Machine Placement problem is a NP-hard problem as well as a Combinatorial Optimization Problem (COP) which means that this problem has a finite set of solutions but in order to choose the best solution we need an EC Algorithm. An EC algorithm is an evolutionary computation algorithm that helps to improve the resource utilization and reduce energy consumption of our VMP problem. Out of all EC algorithms, the Ant Colony System algorithm addresses the problem more efficiently with strong logical inference.

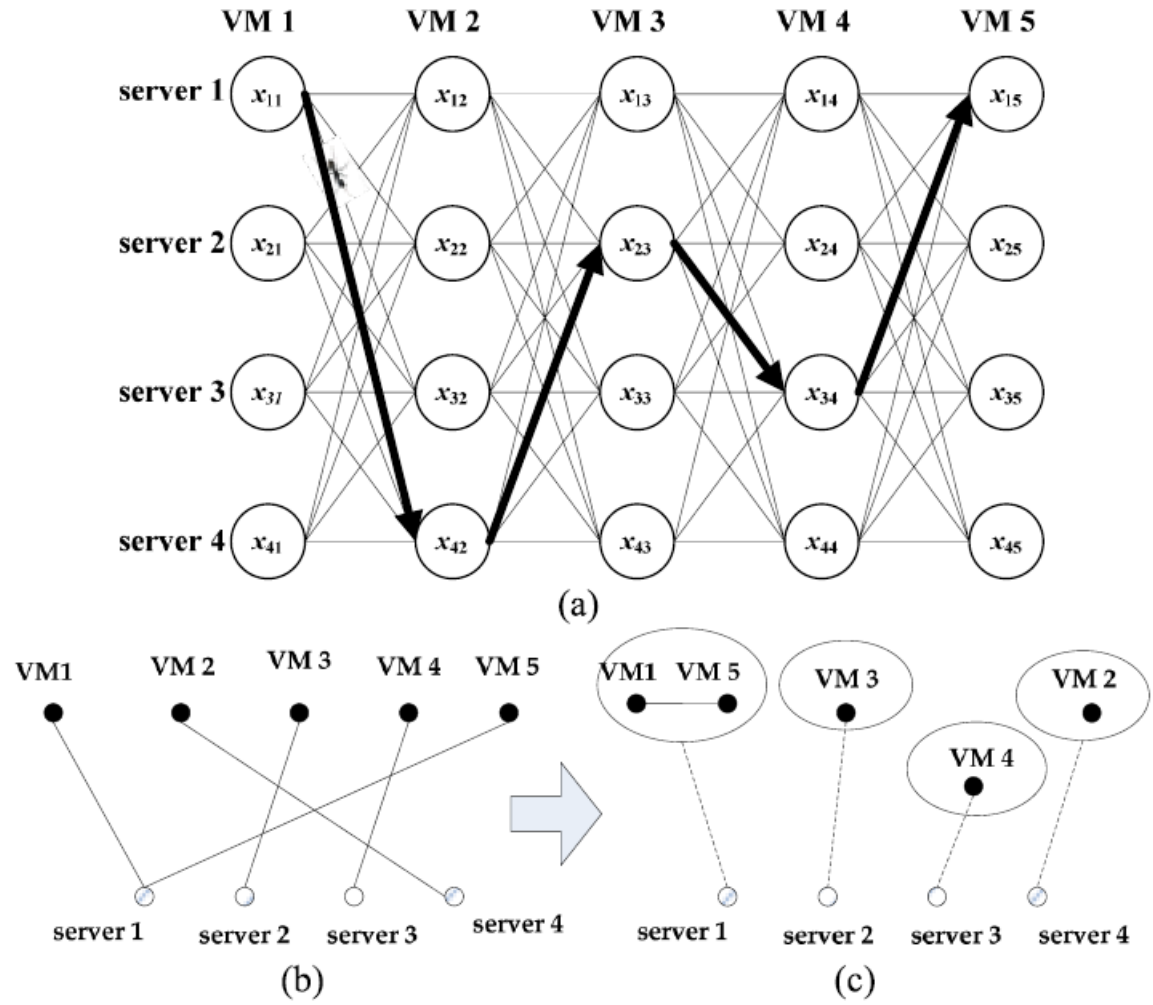

Figure 3.10: Virtual Machine Placement Problem

3.3.1 ACS Algorithm & its Essential Features

Ants are disciplined, industrious, blind insects that work very hard to gather food all summer so that they have enough food stored in the cold seasons. Of all insects and bugs, ants show the most systematic display of unity in performance. Ants are not only stigmergic but also catalytic. In fact, ants are auto catalytic, and they automatically influence their comrades (other ants) to get involved in any altruism thus helping themselves.

In our research, we are using such a system where placement problem of virtual machines in servers will be solved using stigmergic algorithm. The system that we are following in order to consolidate virtual machines and migrate them to active but idle servers, is called Ant System (AS) and the algorithms we introduce in collaboration to that system is called the Ant Colony System Algorithm (ACS). Although there are some differences between a real ant and the artificial ants (agents of the algorithm). The advantages that the artificial ants have are:

- They have memory
- They are not completely blind unlike real ant
- They live in an environment where time is a discrete factor.

In an Ant Colony System algorithm, an artificial ant acts almost the way a real ant behaves when it moves between two points, its home and remote food source. In order to choose the shortest path between these two points, the ant uses trails of pheromones to compare all the possible paths between them.

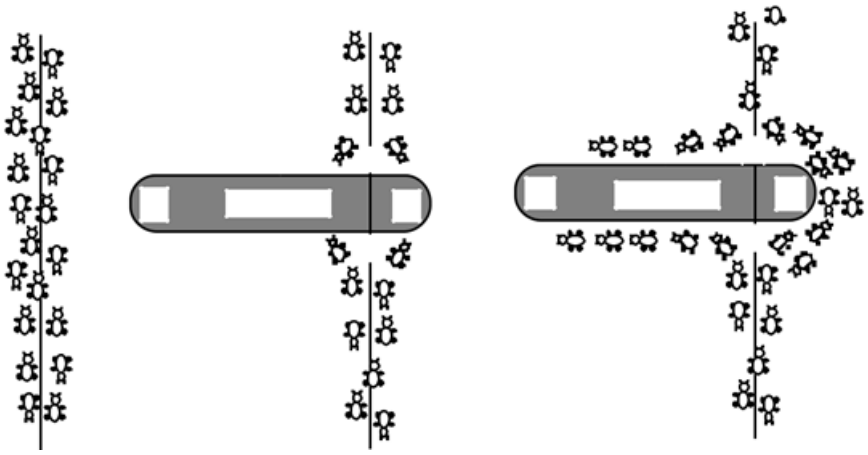

Figure 3.11: Propagation of real ants

- Ants follow a path between points A and E.
- An obstacle is interposed; ants can choose to go around it following one of the two different paths with equal probability.
- On the shorter path more pheromone is laid.

In order to understand the concept of ant system and how ants communicate, travel and collect food, let us take the above fig.(3.11). Let us suppose that the distances between point D and H is equal to the distance between point B and H and between point B and D via C which is equal to 1. Let C be the middle point between D and B fig.(3.12a). Now since time is discrete in our algorithm, we can measure the events happening at different fraction of time by taking t= 1, 2, 3....t. In the beginning, let us assume that 30 ants are coming to point B from A and 30 to point D from E at each time unit, such that each ant walks at a speed of 1 per time unit, and while walking an ant lays down a pheromone trail of intensity 1 (at time t), which evaporates completely and instantaneously in the middle of the successive time interval (t+1, t+2).

When t=0 there is no trail, as no ant has traveled through that path before but 30 ants are in B and 30 in D. Their choice about which way to go is completely random. Therefore, on an average the ants divide into two groups of 15 and one group moves towards point H while the other moves towards point C. fig.(3.12b). When t=1, 30 new ants starts to come from E and A. After they reach point B and D they find a trail of intensity 15 on the path that leads to H, laid by the 15 ants

that went that way from B, and a trail of intensity 30 on the path From B to C, obtained as the sum of the trail laid by the 15 ants that went that way from B and by the 15 ants that reached B coming from D via C fig.(3.12c). The reason for trail being high on path BC and DC is that, the distance between these three points is less compared to the paths between D, H and B. Since we considered all the ants moving at a same speed, the only variable in this scenario is the distance between the points. As such, by the time ants move from B to H then to point D (and vice versa), ants on the path BC and CD take half the time to complete the route BCD and DCB.

The probability of choosing a path is therefore biased, so that the expected number of ants going toward C will be the double of those going toward H, 20 against 10 respectively. In every simulation, the same is true for all the new 30 ants coming towards D from point E and towards B from A.

The process continues until all of the ants eventually choose the shortest path.

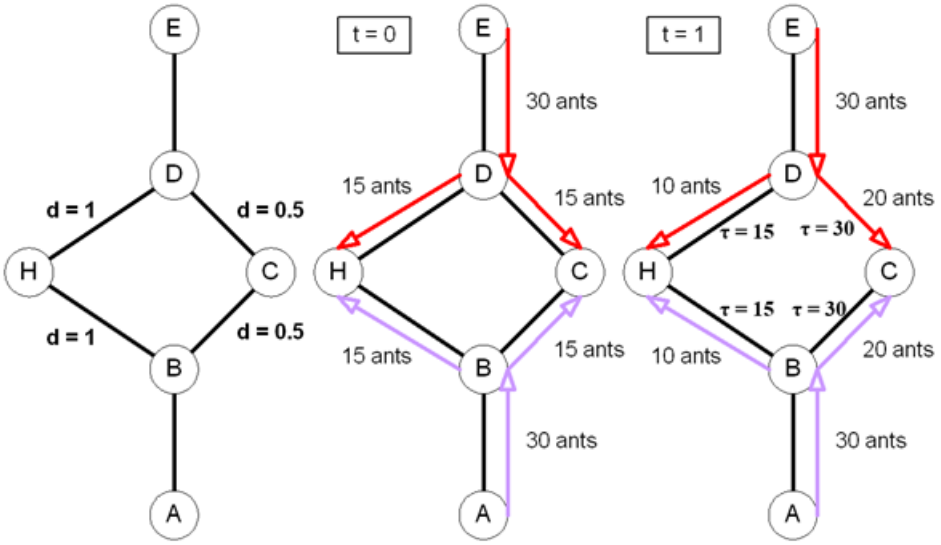

Figure 3.12: Propagation of artificial ants

- The initial graph with distances.
- At time t=0 there is no trail on the graph edges; therefore, ants choose whether to turn right or left with equal probability.
- At time t=1 trail is stronger on shorter edges, which are therefore, in the average, preferred by ants.

The idea is that if at a given point an ant has to choose among different paths, it will always choose the paths with high pheromone trails in it with higher probability. Furthermore, high trail levels are synonymous with short paths. This very trait of following ancestors or activities of preceding companions is called Stigmergy.

Ants as agents: The most common problem solved by using the ACS algorithm is the Travelling Salesman Problem (TSP). In this problem, each ant acts as a simple agent with the following characteristics:

- It chooses the town to go to with a probability that is a function of the town distance and of the amount of trail present on the connecting edge.
- In order to make legal tours, transitions to already visited towns are disallowed until a tour is complete (this is controlled by a tabu list[7]).
- When an ant completes a tour, it lays a substance called pheromone trail on each edge (i, j) visited.

The symmetric TSP has a Euclidean based problem space. We use d_{ij} to denote the distance between any two cities in the problem. As such

$$d_{ij} = [(x_i - x_j)^2 + (y_i - y_j)^2]^1/2 \tag{3.1}$$

Let $\tau_{ij}(t)$ denote the intensity of pheromone trail on edge (i, j) at time t. At the end of each algorithm cycle, the trail intensity between two nodes or cities are updated. Each ant subsequently deposits trail of quantity Q/Lk on every edge (i, j) visited in its individual tour. The sum of all newly deposited trail is denoted by $\Delta\tau_{ij}$. Following trail deposition by all ants, the trail value is updated using the following formula:

$$\tau_{ij}(t + n) = p \times \tau_{ij}(t) + \Delta\tau_{ij} \tag{3.2}$$

Here, p is the rate of trail decay per time interval,
τ_{ij} denotes the change in trail intensity and

$$\Delta\tau_{ij} = \sum_{k-1}^{m} \tau_{ij} \tag{3.3}$$

In the TSP, two factors drive the probabilistic model:

- Visibility, denoted by η_{ij}, equals the quantity $1/d_{ij}$
- Pheromone Trail, denoted by $\tau_{ij}(t)$

These two factors play an essential role in the central probabilistic transition function of the Ant System. In return, the weight of either factor in the transition function is controlled by the variables α and β, respectively. The Probabilistic Transition function used by each ants to decide the next node or city to travel is:

$$P_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[n_{ij}]^{\beta}}{\sum_{k=allowed_k}[\tau_{ij}(t)]^{\alpha}[n_{ij}]^{\beta}} \tag{3.4}$$

A high value of means that trail is very important and therefore ants tend to choose edges chosen by other ants in the past. On the other hand, low values of make the algorithm very similar to a stochastic multi-greedy algorithm [8].

Ants choose the next city or node to travel to with the help of state transition rule. This rule is used to balance between **exploration** and **exploitation**.
if $q \leq q_0$ (exploitation)

$$S = arg_{u \in J_{(T)}} max[\tau(r,u)][\eta(r,u)]^{\beta} \tag{3.5}$$

Here q_0 is a constant parameter, q is a random variable, and S is the outcome of the probabilistic transition function.

Local updating rule: When ants move from one node to another in a single cycle, the pheromone trail between two nodes keep updating by different ants. This update of the pheromone trail is called local updating rule. The following formula denotes the rule:

$$\tau(r,s) \leftarrow (1-\rho)\tau(r,s) + \rho\Delta\tau(r,s) \tag{3.6}$$

Here τ is a predetermined constant or function. The edge (r,s) is updated following each iteration of an ant search.
ρ is the evaporation constant. Here $0 \leq \rho \leq 1$

Global updating rule: In ACS, only the globally best ants (i.e. the ant which constructed the shortest tour from the beginning of the trial) is allowed to deposit pheromone. This update is performed after all ants have completed their tours.

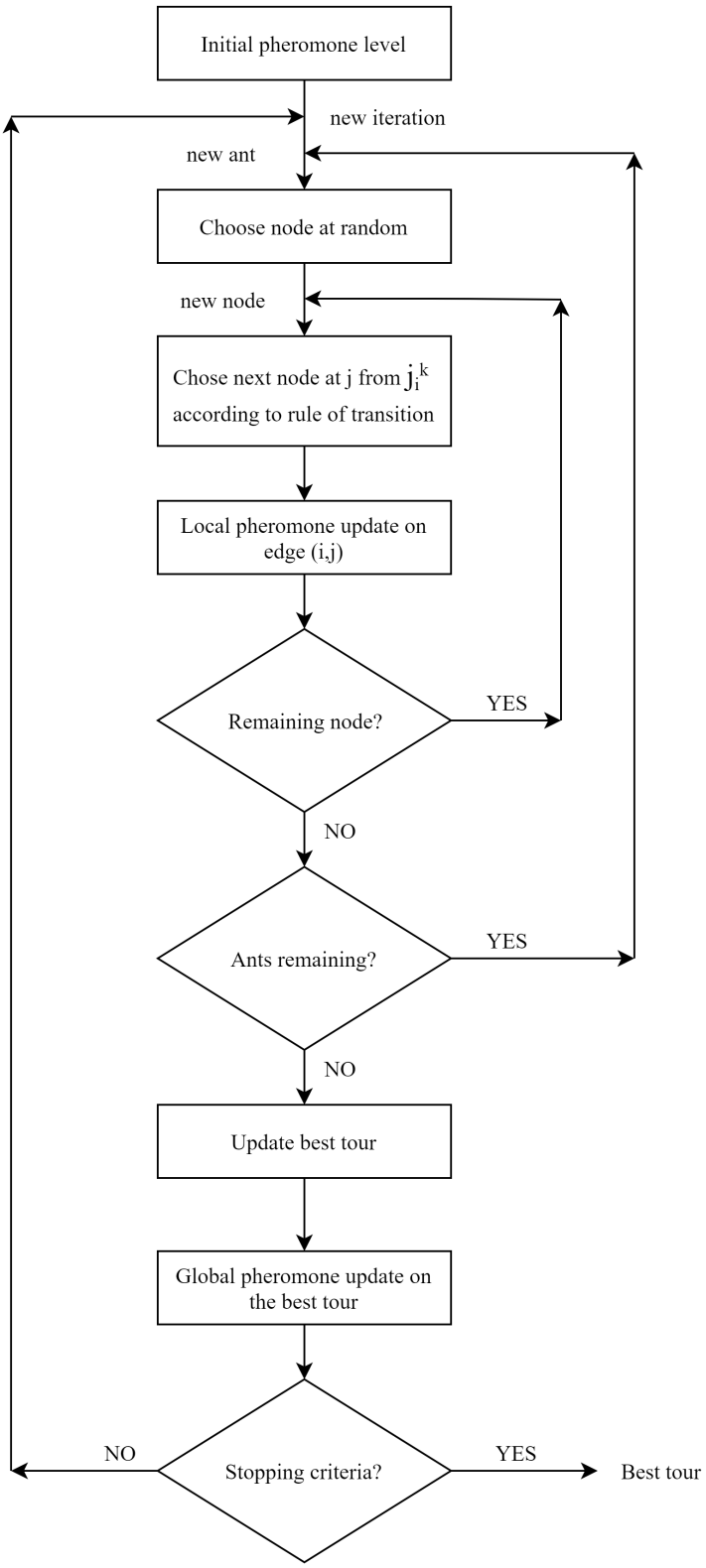

Figure 3.13: Flowchart of ACS Algorithm

Computational Complexity of the ACS:
The complexity of this ACS algorithm is $O(NC \times n2 \times m)$ if we stop the algorithm after NC cycles, where n is the number of cities and m is the number of ants. A linear relation between the number of towns and the best number of ants has been found through multiple researches, so the complexity of the algorithm is $O(NC \times n3)$.

Virtual Machine Consolidation using ACS Algorithm
The Service Level Agreement (SLAs) has formalized that the desired level of Quality of Service (QoS) has to be delivered to the customer by the server providers. There have been many attempts to reduce the energy consumption of data centers while satisfying QoS requirements over the past years. The two most current used techniques are dynamic server provisioning and Virtual Machine (VM) consolidation. Dynamic server provisioning switched-off or put into low-power mode the servers when the demand is low. Same when the demand is high, additional servers are switched-on or put back into high power mode. On the other hand, Dynamic VM consolidation uses hardware virtualization technology. It shares a Physical Machine (PM) among multiple performance-isolated platforms called VMs and the sharing of PM resources is handled by the Virtual Machine Monitor (VMM). It leads to lower energy consumption. But to maximize resource utilization, the PM resource should be utilized properly. Ant Colony System (ACS) algorithm helps to overcome the problem. The simulation result of ACS-based VM consolidation (ACS-VMC) shows that it maintains the desired QoS while it reduces energy consumption. It also outperforms other VM consolidation approaches.

The Best Fit Decreasing (BFD) algorithm is used to allocate the initial VMs to PMs. This algorithm helps an efficient start. But the resource utilization of VMs continues to vary due to dynamic workloads. To adapt and optimize this workload, the ACS-VMC algorithm is proposed. This model consists of two agents. Local Agent (LA) and Global Agent (GA). LA observes the current resource utilization of the PM to solve the PM detection sub-problem. GA supervise and optimize the VM Placement by using ACSS-VMC algorithm. As fig.(3.11) shows, LA monitors the CPU utilization and sets the PM into four different categories as normal, overload, predicted overload and under-loaded. Then the GA collects the data of every PMs from the LAs. By using the ACS-VMC algorithm it builds a global best migration plan. GA tells VMMs to perform the consolidation and VMMs perform the actual migration of VMs.

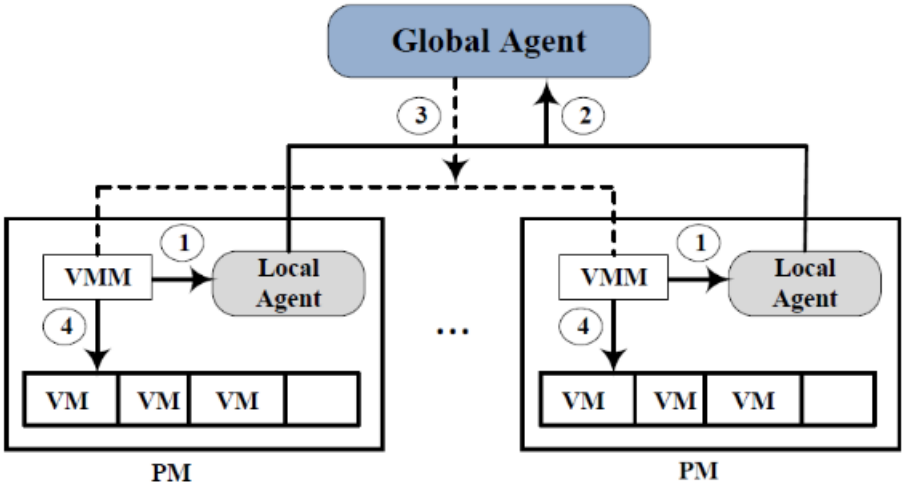

Figure 3.14: The system architecture of OEMACS

3.3.2 OEMACS Algorithm

We are designing an ACS-with OEM operations to decrease the number of active servers and then reduce energy usage for cloud computing. The resulting OEMACS algorithm looks for a result that has less numbers of servers to host all VMs. We identify the minimal number as M_{min} in the possible globally best solution S_{gb}. At the beginning M_{min} = n where N is the number of VMs. It implies that S_{gb}'s original globally best possible approach is to install the N VMs on N servers with one VM corresponding to the one server. We add a pheromone value $\tau(k, j)$ for every pair of VM_k and VM_j $(j \neq k)$ then assign it as $\tau_0 = (N^{-1})$. The pheromone value shows the choice of two VMs to be allocated to the same server.

Solution Construction:
After initiation, OEMACS can create solutions iteration by iteration such that less servers will find more possible solutions. In Every iteration $t(t \geq 1)$, OEMACS attempts to find an acceptable alternative with one server less than M_{min}. Consequently, the ant attempts to put the N VMs on the $M_t = M_{min}1$ servers.

Every ant continues a cycle of constructing by selecting vertices from a design scheme. The example of fig.(3.11) shows that N=5 VMs and $M_t = 4$ servers which are available. The vertices of the design scheme are set into a $(M_t \times N)$ matrix. Each vertex x_{ij} indicates a server assigned VM. Ants 'possible path is suggested by the undirected arc between two vertices in the corresponding columns. In fig.(3.11), $S = \{s_1, s_2, s_3, s_4\}$,where each s_i denotes a set of VMs assigned to server I,is designed by the path $(x_{11,}x_{42,}x_{23}x_{34}x_{15})$.This indicates that all four servers are active to the five VMs as $s_1 = 1, 5, s_2 = 3, s_3 = 4and, s_4 = 2$.

Each ant requires similar processes to create a solution as per the design scheme as seen in fig.(3.11). Notice that the VMs are shuffled arbitrarily before each cycle. Then the ant assigns VMs one by one to the servers to build a solution. We define the

solution building process depending on one ant based on a partial solution under construction. For an ant, there are N steps to build a solution, with each step choosing a suitable server for the corresponding VMs. At $lth(1 \leq l \leq N)$ step, a set of available servers I_m(for placing VM_j) is defined as

$$I_m = i | \sum_{n=1}^{N} x_i n \times c_n + c_j \leq c_i and \sum_{n=1}^{N} x_i n \cdot m_n + m_j \leq M_i, 1 \leq i \leq M_t \tag{3.7}$$

Here i stands for the remaining resources of both CPU and memory for server i to host the unassigned VM_j. By using a state transition rule, the ant selects a suitable server i from the set I_m. In that state transition rule, the pheromone and heuristic values are mentioned as described below.

OEMACS stores the pheromone between VMs. So, we develop a system for converting the stored pheromone between VM and server instead of VM pairs. The choice between VM_j and server i represents loading the VM_j with VMs (the set of s_i) which are already installed on server i. If the pheromone between VM_k and VM_j, is identified by $\tau(k, j)$. We calculated the preferred value $\tau(i, j)$ of VM_j to be assigned to server i as the pheromone average between VM_j and VM_s which are put on server i. If any VM is not applied on server i, the value of $\tau(i, j)$ is assigned as τ_0. So now, we have

$$T_(i, j) = \begin{cases} \frac{1}{|S_i|} \sum_{k \in S_i} \tau(k, j) & if, |s_i| \neq 0 \\ \tau_0 & otherwise \end{cases} \tag{3.8}$$

Where s_i= existing VM set,

$|s_i|$ = number of applied VM on server i.

The heuristic knowledge is to extract a better collection from a greedy approach for the current local situation. To use a limited number of active servers, each server requires to host more VMs, resulting in greater usage of the server's resource. It also helps to balance the utilization of highly utilized and lowly utilized resources, which is effective for the full use of the servers. The heuristic knowledge is set for utilization improvement. The heuristic knowledge measures improved utilization that VM_j can bring to server i, the value is calculated as

$$\eta(i, j) = \frac{1.0 - |\frac{C_i - UC_i - c_j}{C_j} - \frac{M_i - UM_i - m_j}{M_i}|}{|\frac{C_i - UC_i - c_j}{C_i}| + |\frac{M_i - UM_i - m_j}{M_i}| + 1.0} \tag{3.9}$$

here UC_i and UM_i represent the usage of CPU and memory. OEMACS holds two factors designing the heuristic information:

1. the resource utilization for both CPU and memory

2. the balance of available resources

Increasing the usage of resources and utilizing multiple services in a balanced manner, facilitates VM consolidation and decreases the number of active servers.

The probability for assigning an unassigned VM_j to server i with pheromone and heuristic information is calculated by

$$\rho(i,j) = \frac{T(i,j)\eta)(i,j)^{\beta}}{\sum_{kI_m} T(k,j)\eta(k,j)^{\beta}}, \forall i \epsilon I_m \tag{3.10}$$

Here, $\beta > 0$ and it denotes the importance of heuristic information.
For VM_j, it selects server i from the server set I_m by applying the given state transition rule:

$$i = \begin{cases} arg^{max}_{K \epsilon I_m} T(k,j)\eta(k,j)^{\beta} & if, q \leq q_0 \\ I & otherwise \end{cases} \tag{3.11}$$

Here, q is a number selected randomly and uniformly distributed in [0,1], i is also a random number chose from Im based on probability distribution. $q_0 (0 \leq q_0 \leq 1)$ controls the exploitation and exploration of the ant and the parameter is predefined. If $q \leq q_0$ then the ant selects the server according to the preferred value of T and maximum value of heuristic η,it is calculated by $T(i,j)\eta(i,j)^{\beta}$. If not, then the ant selects server i according to the probability for assigning VM_j.

To solve the issue of overloaded server after joining VM, we constructed a complementary rule to assign VM_j to server i as

$$i = \begin{cases} arg^{min}_{1 \leq K \leq M_t} over(K) & if, q \leq q_0 \\ Q & otherwise \end{cases} \tag{3.12}$$

Here, Q is an integer selected randomly in $[1, M_t]$ according to the probability distribution formula mentioned below. If qq_0 then the ant selects the server according to the minimum amount of overload rate after joining VM_j. The overload rate over(i) is calculated as

$$Over(i) = \frac{|C_i - UC_i - c_i|}{C_i} + \frac{|M_i - UM_i - m_j|}{M_i} \tag{3.13}$$

Else, VM_j will be assigned according to the probability distribution rule $q(i,j)$ to server i. Here $(1 \leq i \leq M_t)$. The probability distribution rule:

$$q(i,j) = 1 - \frac{over(i)}{\sum_{k=1}^{M_t} over(k)} \tag{3.14}$$

Fitness Function:
In every iteration, after an ant has completed constructing a solution the best way to evaluate the solution is to find its fitness value. In order to find the fitness of a constructed solution, let us assume that S is an obtained solution of the Virtual Machine Placement (VMP) problem. With the help of the following hierarchical models, we evaluate the obtained solution:

$$f_1(s) = \begin{cases} \sum_{i=1}^{M_t} y_i & if, \text{S satisfies the capacity constraint} \\ M_t + 1 & otherwise \end{cases} \tag{3.15}$$

$$f_2(s) = \sum_{i=1}^{M_t} [(\frac{|C_i - UC_i|}{C_i} + \frac{|M_i - UM_i|}{M_i}) \cdot y_i] \tag{3.16}$$

Where, M_t is the total number of provided servers in the current iteration t; y_i denotes whether server I is used in the solution S thus $y_i = [0, 1]$. In eq.(3.15) if the obtained solution is feasible then the function $f_1(S)$ shows the number of active servers, which is not larger than M_t. Else, $f_1(S)$ is set to $M_t + 1$ in order to differentiate from the feasible solutions, which denotes that the number of servers needed to use is the same as the best feasible solution in the previous or last iteration. And that number equals to M_{min} as $M_t = M_{min}$–1.

Again, Eq.(3.16) helps to determine the fitness value of the obtained solution S, by calculating the approximation of the placement in order to fill up the servers and to evaluate the resource utilization. For two solutions, in order to pick the more fitted one we compare the value of f_1 first and choose the solution with a smaller value of f_1. If the values of f_1 seem to be equal, we compare the values of f_2 of the same solutions and pick the smaller one.

Eq.(3.15) gives the value of servers that we need to consolidate the process and eq.(3.16) gives the fitness value of the solutions by calculating the resources used. As such, the solutions with fewer servers and higher utilization is chosen in every iteration.

Management of Pheromone:
Pheromone on a path records the historical preference information. In the Ant-Colony Optimization process, two different types of pheromone updating rules are applied in order to update the pheromone left on the paths travelled. After an ant constructs a solution in a single iteration, the local pheromone updating rule is performed on each VM-pair (r, s) existing on the same server i. The updating rule is:

$$\tau(r, s) \leftarrow (1 - \rho)\tau(r, s) + \rho\Delta\tau(r, s) \tag{3.17}$$

Here, $0 < \rho < 1$ is the pheromone decay parameter.
On the contrary, only the best solution of an iteration is allowed to perform the global pheromone update at the end of each iteration. After all the agents (ants) have completed constructing their solutions throughout the iterations, the best solution of the ongoing or current iteration can be found and denoted as S_b. This solution can either be feasible or infeasible.

S_b being feasible denotes that OEMACS has found a new feasible solution with servers no more than Mt. As such, the feasible globally best solution S_{gb} can be updated to S_b and M_{min} can be updated to f1(Sb).However, if S_b turns out to be infeasible then it denotes that OEMACS cannot find a feasible solution with M_t servers. In such a case, OEMACS uses OEM (Order Exchange and Migration) local

search on S_b. In both cases, the global pheromone updating is carried out on S_b in order to increase the pheromone layer on the VM-pair of the same host server of S_b as:

$$\tau(k,j) = (1- \in) \cdot \tau(k,j) + \in \cdot \Delta\tau_i, If(k,j) \in s_i, \forall s_i \in s_i, (0 <\in< 1) \tag{3.18}$$

$$\Delta\tau_i = \frac{1}{f_1(s_b)} + \frac{1}{NC_i + NM_i + 1} \tag{3.19}$$

$$NC_i = \frac{RC_i}{C_i} \tag{3.20}$$

$$NM_i = \frac{RM_i}{M_i} \tag{3.21}$$

Where, $\epsilon(0 < \epsilon < 1)$ is the pheromone enhance parameter, $f_1(S_b)$ is the number of physical machines i.e. servers used in S_b, NC_i NM_i represents the amount of normalized remaining CPU and Memory resource of server i respectively. Furthermore, RC_i and RM_i denotes the amount of remaining CPU and Memory resource of server i while C_i and M_i represents the total amount of CPU and memory of i. The main purpose for using the above equations is to record an almost 'full' VM group and a better solution. The more occupies the server is, the more pheromone of VM pairs increases. The local and global pheromone updating techniques play different roles in guiding the agents in their search act. The local updating rule decreases the appeal to group VM pair which has been found by the last agent and to help the other ants explore new assignment space. On the other hand, global pheromone updating is done to consolidate the tie between VM pairs in the good assignments and guide the agents to come up with better solutions.

The complete OEMACS Algorithm

The complete algorithm of OEMACS is composed of two separate algorithms. They are ACS (Ant Colony System) and OEM (Order Exchange Migration). The full combined algorithm is given below:

Step 1: Initialization. The parameter foe ACS is set to τ_0. The feasible globally best solution is set as S_{gb} for placing N VMs on N servers. Thus, the number of minimum servers is set to $M_{min} = N$. Set iteration $t = 1$ and maximum iteration as T.

Step 2: Set $M_t = M_{min} - 1$. In each iteration, m ants construct m solutions and perform local pheromone updating on each solution.

Step 3: The fitness function $f(S)$ is applied in order to evaluate the fitness value of the constructed solution.

Step 4: The best solution Sb of the current iteration is set after evaluating the fitness value of the constructed solution. If S_b is feasible, S_{gb} is updated as S_b and $M_{min} = f_1(S_b)$ is set. Otherwise OEM local search is performed on S_b. S_{gb} and M_{min} are updated respectively if local search succeeds.

Step 5: After S_{gb} and M_{min} are locally updated, global pheromone update is eventually done on S_b and S_{gb}.

Step 6: Check if t is less than or equal to T. If not equal, then set $t = t + 1$ and go to Step 3. Otherwise terminate the algorithm.

Throughout the procedure, Step 3 is a main process of the OEMACS Algorithm.

3.3.3 Collaboration of OEMACS & Cloud Computing

With the collaboration of OEM & ACS Algorithms, cloud computing has become surprisingly plausible and faster. Migrating idle machines from one server to another has never been this efficient. The combination of these two algorithms has given researchers the hope of something more useful and convenient.

In order to transform an infeasible solution into a feasible one, local search is inevitable. Whenever an infeasible solution is found, the OEM (Order Exchange Migration) local search is performed on the current solution Sb with the view to make it feasible. This local search is carried out before the global pheromone update. The search technique includes two procedures. First an order exchange operation and after that, a migration operation. Both the operations try to adjust the VM assignments to turn an overloaded server into a non-overloaded one.

i) **Order Exchange Operation:** The order exchange operation swaps VM between different servers. The main objective of this operation is to enhance the resource utilization and share the resources of a non-overloaded server at the same time to avoid unbalance resource utilization of over-loaded servers. To do so, every single VM of an overloaded server is compared with every single VM of the non-overloaded server with the view to make both the servers non-overloaded and balanced. In order to reduce the computational burden of this process, the following exchange strategy is followed:

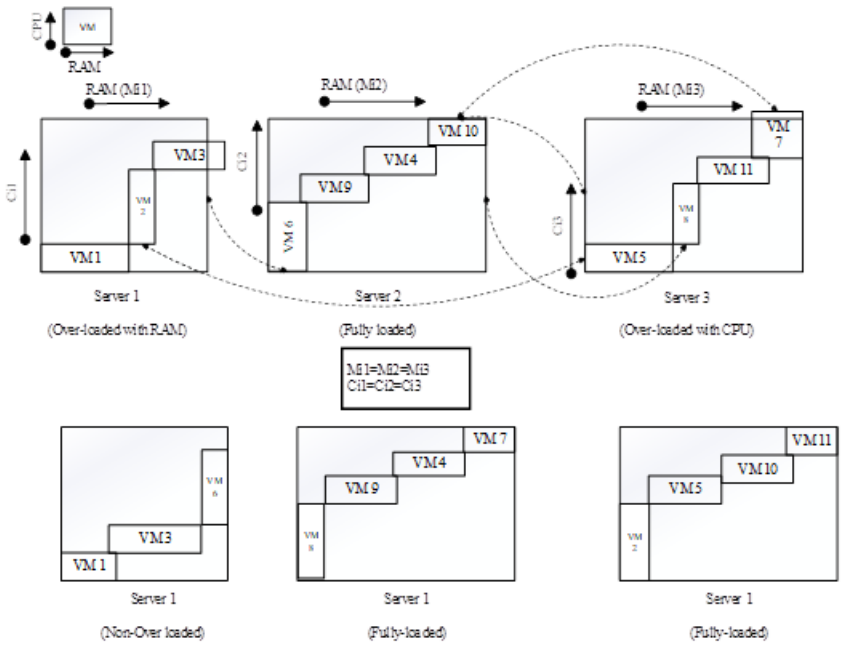

Figure 3.15: Order Exchange & Migration

For every overloaded server i, the VMs on it are sorted out on the basis of the absolute difference between CPU RAM requirements and preferences are given to the VMs with higher differences, in the process of exchange. After sorting is done in the overloaded server i, a non-overloaded server k is selected and the VMs on that server are sorted out on the basis of the absolute difference between CPU RAM requirements and in this case the contrast method is followed while giving preferences to the VMs i.e. the VMs with smaller resource difference are sorted to the front. After that, every single VM in the queue of server i are compared with those of server k one by one to determine whether any two VMs can be exchanged. The process ends until both the servers i k becomes non-overloaded due to the VM exchange; or all VMs on server k have been checked for exchange but no suitable VM was found. As such if server i cannot be turned into a non-overloaded server, a new non-overloaded server is selected, and the same process is continued again.

Fig.(3.15) presents possible transition directions of the exchange operation in an example illustration. The rectangles of different shapes denote the VMs inside servers. The dotted arrows show the possible exchange scheme of VMs to make the overloaded servers balanced and non-overloaded. In the example, 3 servers contain 11 VMs. The horizontal line of the VM rectangles represent the RAM size and the vertical line represents the CPU size of all the Virtual Machines (VM). Server 1 and Server 3 are overloaded with RAM and CPU respectively. In order to make the two servers balanced and non-overloaded, VM 2,6,7,8 and 10 swapped their places with the other VMs in the non-overloaded servers.

ii) **Migration Operation:** The migration operation schedules the migration of a VM from an overloaded server to a non-overloaded server that can offer enough RAM and CPU to this VM. Before migration, the VMs in the overloaded server are checked one by one and compared with every single VMs on the non-overloaded server to determine which VM can be migrated to the destination server i.e the non-overloaded server. The operation terminates when all the overloaded servers becomes balanced and non-overloaded or there is no significant VM movement possible. A possible migration scenario is illustrated in fig.(3.12), where VM 2,6,7,8, and 10 are migrated from their originating overloaded servers to destination non-overloaded servers.
The Migration operation is performed after the Order Exchange Operation. In an early stage, the number of servers is large enough in order to find a feasible solution. As such, local search does not work then. In a later stage, the local searching technique can convert the infeasible solution into a feasible one as the technique approaches the optima at this stage.

3.4 Similar Algorithms

Power consumption by physical machines or servers is an important issue in cloud architecture. Recent studies have shown that the power consumed by servers can be assumed to be linear with CPU utilization. An active but idle server can burn about 50% to 70% of the power consumed by the server working at full load [2]. This scenario can be sketched using the following power model:

$$P(u) = k.P_{max} + (1 - k).P_{max}.u \tag{3.22}$$

Here, P_{max} is the maximum power consumed by a fully loaded server

k is the fraction of power consumption in idle state
u is the CPU utilization of the server.

Since power consumed by the idle virtual machines in an active server is the main drawback of this system and major cause of energy wastage, reducing the number of active servers as well as migrating idle virtual machines to a completely different server will be a logical solution to ensure green cloud computing.

The main objective of our research is to consolidate virtual machine and globalize green cloud computing. Virtualization is a powerful technology that facilitates better use of the available data center resources using a technique called Virtual Machine (VM) consolidation which involves gathering of several virtual machines into a single physical server. To address the problem of high energy usage, it is necessary to eliminate inefficiencies and waste in the way electricity is delivered to computing resources, and in the way these resources are utilized to serve application workloads. This can be done by improving the physical infrastructure of data centers as well as resource allocation and management algorithms.

VM consolidation involves live migration, which is the capability of transferring a VM between physical servers with a close to zero down time is an effective way to improve the utilization of resources and energy efficiency in cloud data centers. The computer's ability to store, retrieve and manipulate large amounts of data rapidly and cheaply has led to its wide spread use. But, at each stage of computer's life throughout its use, and into its disposal, it exhibits some kind of environmental problems. So in this paper, we are focusing on globalizing green cloud computing. Green computing is the environmentally responsible use of computers and related resources. Such practices include the implementation of energy-efficient central processing units (CPUs), servers and peripherals as well as reduced resource consumption and proper disposal of electronic waste (e-waste).

So, in order to make cloud computing safer, more efficient and environment friendlier, use of stigmergic algorithm is inevitable. With the use of an autocatalytic, multi-greedy heuristic algorithm, it is possible to keep the energy emission rate of servers bounded within a considerable range and thus we can ensure green computing.
After the very introduction of the Ant Colony System Algorithm in 1992, researchers and programmers all around the globe showed interest in solving the COP and NP-hard problems that could be solved using such algorithms only. They have used many similar EC-algorithms like Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Multi-objective Ant Colony Optimization (MACO), Hybrid ACO and PSO and much more to address the VMP problem.

In all these algorithms, there remained some drawbacks and deficiencies which in return did not help in solving the problem in the most effective way possible. For such reasons, we have come up with a more efficient and rigorous EC-algorithm that will help address the VMP problem in a better way.

Chapter 4

Proposed Model

The model that we came up with is an algorithm to solve the Virtual Machine Placement problem. Our proposed model is called Active Idle Virtual Machine Migration Algorithm. With the help of this algorithm, we can easily migrate VMs from an overloaded server to a non-overloaded server and at the same time ensure the presence of maximum idle VMs in a destination server. With the help of this algorithm, the high-power consumption problem can be fixed and thus the carbon footprint of a server will come down to a great extent. Based on the objectives presented above, we designed a model that tends to optimize the utilization of resources used and energy consumed. These models try to minimize the resource wastage, SLA violations and excess energy consumption in the cloud infrastructure. The models are as follows:

$$Min \sum_{i=1}^{n} S_i \cdot SLA \tag{4.1}$$

$$Min \sum_{i=1}^{n} W_R = \sum_{i=1}^{n} [S_i \times \frac{|UC - UM|}{\sqrt{(UC)^2 + (UM)^2}}] \tag{4.2}$$

$$Min \sum_{i=1}^{n} P(u) = \sum_{i=1}^{n} [S_i \times (k \times P_{max} + (1-k) \times P_{max} \times u)] \tag{4.3}$$

$$u = UC_i = \frac{\sum_{i=1}^{n} V_i \cdot UC_j}{\sum_{i=1}^{n} V_i}; 0 \leq V_i \leq 1; 0 \leq S_i \leq 1 \tag{4.4}$$

For Si, possible constraints are

$$\sum_{i=1}^{n} S_i \cdot UC \leq R_{cpu} \tag{4.5}$$

$$\sum_{i=1}^{n} S_i \cdot UM \leq R_{Mem} \tag{4.6}$$

4.1 Active & Idle Virtual Machine Migration Algorithm

The algorithm with the help of which actively working virtual machines and idle VMs can be migrated from one overloaded or non-overloaded server to another non-overloaded server with the view to make power utilization efficient can be called Active Idle Virtual Machine Migration (AIVMM). Since our main problem regarding the Cloud Infrastructure and Virtual Machine Architecture is the massive energy consumption of the idle virtual machines which is around 50% - 70% of the total server input power, we plan to solve this problem by separating the idle VMs from the actively working VMs in a server. In order to do so, we need to design a model that can detect the number of actively working and idle VMs in a server at a given time. Virtual Machines act as independent, enclosed machines that use the resources they are allocated with during the process of virtualization. A VM is comprised of Processor (CPU), Memory (RAM), Space (Hard drive) and Network Bandwidth of its own. Based on the usage of CPU and RAM, we can easily detect which VM is actively busy and which VM is sitting idle. On using that information, it becomes quite simple to migrate all idle or active VMs from one server to another.

Our idea is to exchange all idle virtual machines of a single server with the actively working fully loaded ones of a non-overloaded server in order to ensure maximum number of active VMs in a single server at most of the times. As such, the actively working VMs will face no interruption in their power consumption since idle VMs tend to consume 50% - 70% of the total power of host server.

Judging from the CPU and RAM utilization of a VM, for an instantaneous time t, the following scenarios can co-exist in a cloud environment:

- $U_{m_j} > U_{c_j}[RAM > CPU]$
- $U_{m_j} < U_{c_j}[RAM < CPU]$
- $U_{m_j} \approx U_{c_j}[RAM \approx CPU]$

Here, U_{m_j} represents the memory utilization of a VM, U_{c_j} represents the processor or CPU utilization of a VM and $0 \leq U_{m_j} \leq 1, 0 \leq U_{c_j} \leq 1$ i.e U_{m_j} and U_{c_j} represents the percentage of RAM & CPU utilization.

4.2 Working Model

In the process of calculating total number of active and idle VMs of a single server at an instant t, we will use the resource utilization percentage of every single virtual machine. As a result, it will become easier to decide whether to migrate active VMs or idle ones. The possible scenarios of our applied method would be:
a) When $RAM > CPU$,

1. $U_{m_j} >> U_{c_j}$
2. $U_{m_j} \approx U_{c_j}$

In (1), the RAM utilization is much greater than the CPU utilization which denotes that the VM is idle as it is using RAM or memory to keep itself on but not using much CPU compared to its RAM usage.

In (2), the RAM utilization is almost or exactly equal to the CPU utilization which denotes that the virtual machine is actively working with full load. Although the scenario of RAM usage being exactly equal to the CPU usage can be very rare.
b) When $RAM < CPU$,

1. $U_{m_j} << U_{c_j}$
2. $U_{m_j} \approx U_{c_j}$

When the CPU of a VM is working with full load but it is saving the data in cache or temporary memory rather than the RAM, then the CPU usage of the machine seems to be greater than the volatile memory i.e. RAM usage.

Here, in (1) the CPU utilization is much greater than the RAM utilization which denotes that the VM is active, but it is saving its data in the temporary memory or cache memory. A such the virtual machine is online and working with full load.

Furthermore, in (2) the RAM utilization is almost or exactly equal to the CPU utilization which denotes that the virtual machine is actively working with full load. Although the scenario of RAM usage being exactly equal to the CPU usage can be very rare.

Constraints of the functioning models (1), (2) and (3) are as follows:

For model (1), the possible constraints are:

- $|U_{m_j}–U_{c_j}| > 0$
- $0 \leq (U_{m_j} - U_{c_j}) \leq 1$
- $0 \leq (U_{m_j} - U_{c_j})/(U_{m_j} + U_{c_j}) \leq 1, [U_{m_j} + U_{c_j} \neq 0]$

For model (2), the possible constraints are:

- $|U_{c_j}–U_{m_j}| > 0$
- $0 \leq |U_{c_j} - U_{m_j}| \leq 1$
- $0 \leq (U_{c_j} - U_{m_j})/(U_{c_j} + U_{m_j}) \leq 1, [U_{m_j} + U_{c_j} \neq 0]$

For model (3), possible constraints are:

- $U_{m_j}–U_{c_j} = 0$
- $U_{m_j}–U_{c_j} = -(U_{c_j}–U_{m_j})$
- $|U_{m_j}–U_{c_j}| = |U_{c_j}–U_{m_j}|$

In order to calculate the total number of idle virtual machines in a server for an instant t, the following equations can be used:

$$V_0 = U_{m_j} - U_{c_j} \tag{4.7}$$

$$V_0\text{`} = \begin{cases} 0, & 0 \leq V_0 \leq 0.3 \\ 1, & 0.4 \leq V_0 \leq (0.9 \approx 1) \end{cases} \tag{4.8}$$

$$V_{idle} = \begin{cases} \sum_{0=1}^{N1} V_0\text{`}, & [V_0\text{`}\epsilon N1] \\ 0, & otherwise \end{cases} \tag{4.9}$$

Here, N1 is a set of virtual machines where $U_{m_j} > U_{c_j}$, Vo represents the difference of RAM and CPU utilization while V_0‘ represents a piecewise function of V_0. V_{idle} denotes the number of idle virtual machines in a server.

Again, in order to calculate the total number of working virtual machines in a server for an instant t, the following equations can be used:

$$V_a = U_{c_j} - U_{m_j} \tag{4.10}$$

$$V_a\text{`} = 1, if 0 \leq V_a \leq (0.9 \approx 1) \tag{4.11}$$

$$V_{active} = \begin{cases} \sum_{a=1}^{N2+N3} V_a\text{`} + c, & [V_a\text{`}\epsilon(N2+N3)] \\ c, & otherwise \end{cases} \tag{4.12}$$

$$c = \sum V_a \tag{4.13}$$

Here, N2 is a set of virtual machines where $U_{c_j} > U_{m_j}$, V_a represents the difference of CPU and RAM utilization of VM while V_a‘ represents a function of V_a. V_{active} denotes the number of actively working virtual machines in a server, c represents the number of active VMs from eq.(4.7)

When the number of actively working VMs and idle VMs at any time t, in any server i turns out to be equal, then the following equations can be used to calculate the total CPU and memory utilizations of the VMs:

$$UC_{V_a} = \sum_{j=1}^{V_{active}} U_{c_j} \times c_j \tag{4.14}$$

$$UM_{V_a} = \sum_{j=1}^{V_{active}} U_{m_j} \times m_j \tag{4.15}$$

Here, UC_{v_a} and UM_{v_a} denotes the CPU and Memory utilization of actively working virtual machines respectively, while U_{c_j} and U_{m_j} represents that of a VM. Also, c_j and m_j represents the total CPU and memory capacity of a VM.

$$UC_{Vi} = \sum_{j=1}^{V_{idle}} Uc_j \times c_j \tag{4.16}$$

$$UM_{V_i} = \sum_{j=1}^{V_{idle}} U_{m_j} \times m_j \tag{4.17}$$

The equations to calculate the total CPU and RAM usage of both an actively working and an idle virtual machine are respectively:

$$TU_{V_a} = UC_{V_a} + UM_{V_a} \tag{4.18}$$

$$TU_{V_i} = UC_{V_i} + UM_{V_i} \tag{4.19}$$

Here TU_{v_a} and TU_{v_i} represent the total utilization of both CPU and RAM by the active and idle VMs respectively, UC_{v_a} and UM_{v_a} represents the CPU and memory utilization of the active VMs while UC_{v_i} and UM_{v_i} represents those of the idle VMs.

The algorithm AIVMM is composed of ACS (Ant Colony System Algorithm) and OEM (Order and Exchange Migration) along with VMM (Virtual Machine Migration). The complete algorithm is sketched below:

Step 1: Initialization. Compute the number of active and idle VMs in a single host server. Take the number of active VMs as V_{active} and the number of idle VMs as V_{idle}. Compare V_{active} and V_{idle}. [Three possible scenarios can occur. Such as- i) $V_{idle} > V_{active}$, ii) $V_{idle} < V_{active}$ and iii) $V_{idle} = V_{active}$]

Step 2: The parameter for ACS is set to τ_0. The feasible globally best solution is set as S_{gb} for placing N VMs on N servers. Thus, the number of minimum servers is set to $M_{min} = N$. Set iteration $t = 1$ and maximum iteration as T.

Step 3: Set $M_t = M_{min} - 1$. In each iteration, m ants construct m solutions and perform local pheromone updating on each solution.

Step 4: The fitness function $f(S)$ is applied in order to evaluate the fitness value of the constructed solution.

Step 5: The best solution S_b of the current iteration is set after evaluating the fitness value of the constructed solution. If S_b is feasible, S_gb is updated as S_b and $M_{min} = f1(S_b)$ is set. Otherwise OEM local search is performed on S_b. S_gb and M_{min} are updated respectively if local search succeeds.

Step 6: After S_gb and M_{min} are locally updated, global pheromone update is eventually done on S_b and S_gb.

Step 7: Check if t is less than or equal to T. If not equal, then set $t = t + 1$ and go to Step 3. Otherwise terminate the algorithm.

Step 8: After t terminates, calculate $V`_{active}$ and $V`_{idle}$ of the host server. If $V`_{active} = V_{active}$ and $V`_{idle} = V_{idle}$, then move forward to step 9. Otherwise update $V`_{active}$ and $V`_{idle}$ and then move to step 9.

Step 9: (a) If $V_{idle} > V_{active}$, then migrate all the actively working VMs from host server to a nearby (ACS) non-overloaded (OEM) server with the opposite scenario i.e. the server in which $V_{idle} < V_{active}$. The idle VMs of the destination server will be exchanged with the actively working ones from the host.

(b) If $V_{idle} < V_{active}$, then migrate all the idle VMs from host server to a nearby (ACS) non-overloaded (OEM) server with the opposite scenario i.e. the server in which $V_{idle} > V_{active}$. The actively working VMs of the destination server will be exchanged with the actively working ones from the host.

(c) If $V_{idle} = V_{active}$, calculate the CPU utilization ($UC_{v_a}\&UM_{v_a}$) and memory utilization ($UC_{v_i}\&UM_{v_i}$) of the VMs (both actively working and idle) and compare the total utilization of both type VMs using Eq.(4.18) & (4.19).

4.3 Salient features of AIVMM

4.3.1 A multi-objective approach towards VMM

Virtual Machine Migration (VMM) is a common phenomenon in cloud architecture. For the assurance of continuous connectivity and better performance, migration of virtual machine from one server to another is a must. In order to migrate a VM from one server to another and to accommodate the VM with required resource and enough power in the destination server, some parameters are needed to be satisfied. These parameters are:

- SLA (Service Legal Agreement)
- Resource Utilization
- Power or energy utilization

4.3.2 Service Level Agreement (SLA)

Maintaining the QoS (Quality of Service) is an important requirement in cloud computing. QoS requirements are commonly formalized as SLAs, which specify enterprise service-level requirements for data center in terms of minimum latency or maximum response time[28]. A workload independent metric called the SLA Violations (SLAV) is used to measure the SLA delivered in an IaaS cloud. It measures both violations due to Over-utilization (SLAV(O)) and violations due to Migrations (SLAV(M))[10]. Both these violations characterize the level of SLA Violations by the infrastructure independently and with equal importance.

$$SLA = \frac{\sum Req - AReq}{\sum Req} \tag{4.20}$$

Here, AReq denotes allocated requests

$$SLAV = SlAV(O) \times SLAV(M) \tag{4.21}$$

SLAV(O) represents the percentage of time during which active servers have experienced the resource utilization of 100%. It is defined as:

$$SLAV(O) = (M^{-1}) \sum_{i=1}^{M} \frac{T_{u_i}}{T_{a_i}} \tag{4.22}$$

Here, M is the number of servers, T_{u_i} represents total time server i has experienced the utilization of 100% leading to a SLAV; T_{a_i} is the total duration of server i being in the active state.

SLAV(M) represents the overall performance degradation by VMs due to migrations. It is computed as:

$$SLAV(M) = (N^{-1}) \sum_{j=1}^{N} \frac{C_{d_j}}{C_{r_j}} \tag{4.23}$$

Here, N is the total number of virtual machines. C_{d_j} is the performance degradation estimation of VM_j by migration, C_{r_j} is the total CPU capacity requested by VM_j in its lifetime.

4.3.3 Resource Utilization

To make full use of the resources, we have considered the CPU and the RAM to define the resource wastage in eq.(4.24) Here, W_R represents resource wastage, UC and UM denotes CPU and RAM utilization respectively. When W_R is large, more energy gets wasted.

$$W_R = \frac{|UC - UM|}{\sqrt{(UC)^2 + (UM)^2}} \tag{4.24}$$

4.3.4 Energy Utilization

Our focus is to minimize and utilize proper energy consumption as high use of energy creates high level radiation and increases pollution. To ensure green cloud computing our foremost target is to prevent idle virtual machines from disrupting the necessary power consumption for the actively working machines. Past researches have shown that idle virtual machines consume nearly 50% to 70% of the total server power and as such the neighboring virtual machines working with full load cannot get the salient power proportionately in time[11]. Thus, there remains a lag in service. The scenario can be depicted from the following model:

$$P = K \times P_{max} + (1 - k) \times P_{max} \times u \tag{4.25}$$

$$u = UC_i \tag{4.26}$$

Here, UC_i represents CPU Utilization of the server, P_{max} represents the peak power, k denotes the idle power consumption co-efficient.

4.4 Limitations of AIVMM

Although AIVMM algorithm helps to exchange active and idle virtual machines among servers and decrease unnecessary power consumption, this model in under development and in its testing phase. There are some limitations viable in the current model. They are:

- Time complexity of the algorithm is an important factor. The run-time complexity of AIVMM is quadratic and greater than that of OEMACS.
- Since this algorithm helps to migrate idle and actively working VMs from one server to other, it is a continuous process. An idle VM can start working anytime in the process, even when its being migrated while an actively working VM can cease to work and become idle any time in the process[29].
- In some cases, the differentiation between the resource utilization becomes quite complex. So, determining the number of active and idle VMs in a server can get complicated and problematic.
- Eq.(4.7) and (4.10) have some logical limitations. The equations are used to calculate the usage differences of RAM & CPU and CPU & RAM respectively. The equations do not clarify the scenario when $CPU = RAM \approx 0$.

Chapter 5

Result Analysis

5.1 Result

We have successfully evaluated our algorithm- AIVMM in an advanced simulator CloudSim that helps to simulate cloud environments with necessary resources and infrastructure. The hardware device that we used in this purpose had the following configurations:

- Hardware - Intel Core i5-7200U 2.5 GHz Processor, 64-bit CPU, 1TB Hard Drive, 4GB DDR4 Memory
- Software - Eclipse IDE
- Environment - CloudSim 3.0.3 Language - Java
- OS - Windows 10 Education
- Network Bandwidth - 40 Mbps

We first formulized the AIVMM algorithm using the Eclipse IDE in Java. Then we collaborated the algorithm with OEMACS as AIVMM is a dependent algorithm. We conducted our experiments in CloudSim 3.0.3 that contain all the necessary classes required to simulate a cloud environment

5.2 Analysis

5.2.1 Data Analysis of OEMACS & ACS Algorithm

Iteration	Max City Count	Max Ant Count	Highest Avg. Time (s)
150	4000	3200	2354.88
500	4000	3200	2641.52
1000	4000	3200	2932.66
2000	4000	3200	3905.34

Table 5.1: Dataset for ACS algorithm Simulation

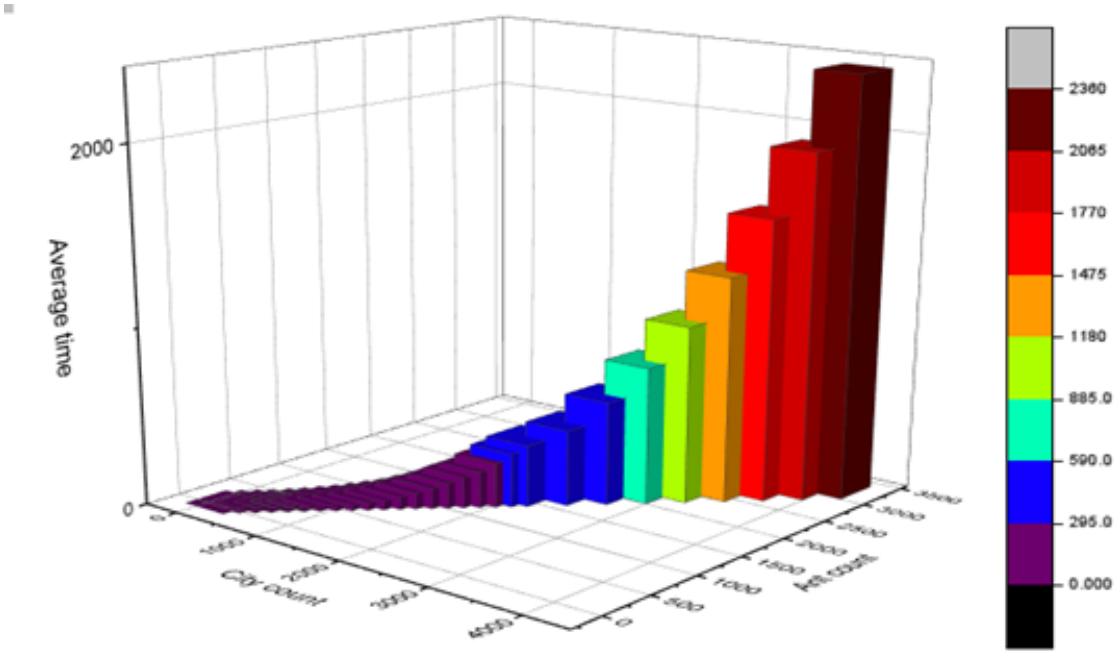

Figure 5.1: Average Time Complexity of OEMACS with Maximum Iteration 150

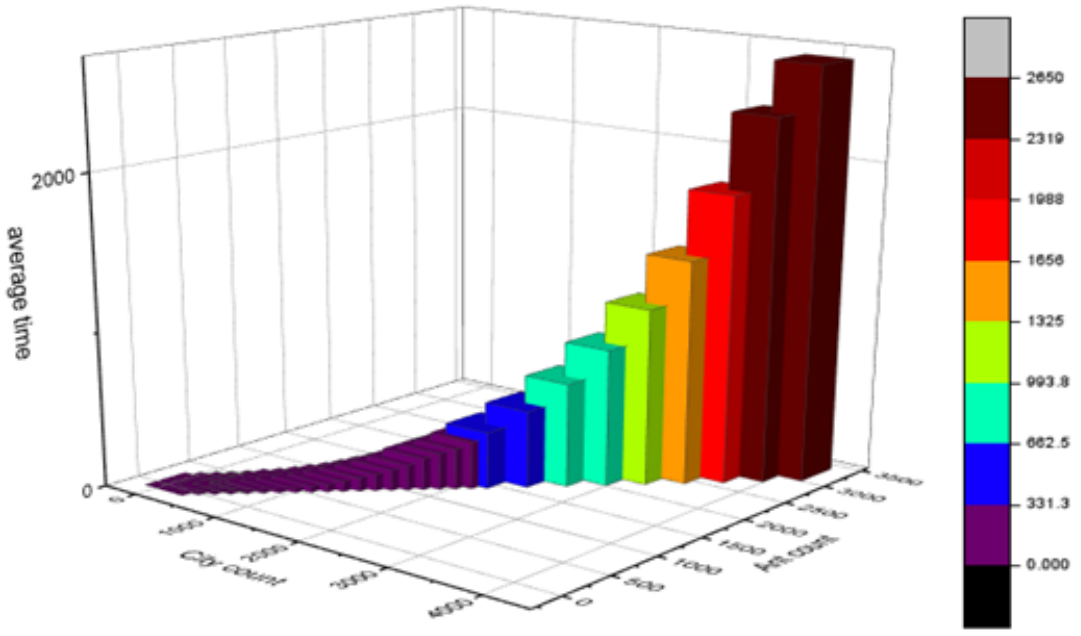

Figure 5.2: Average Time Complexity of OEMACS with Maximum Iteration 500

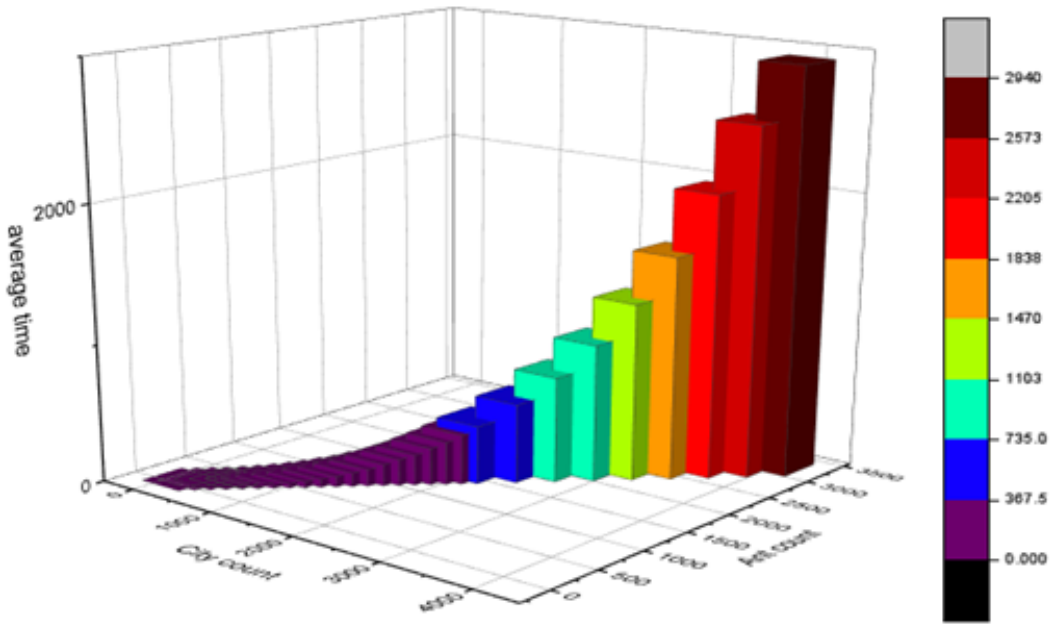

Figure 5.3: Average Time Complexity of OEMACS with Maximum Iteration 1000

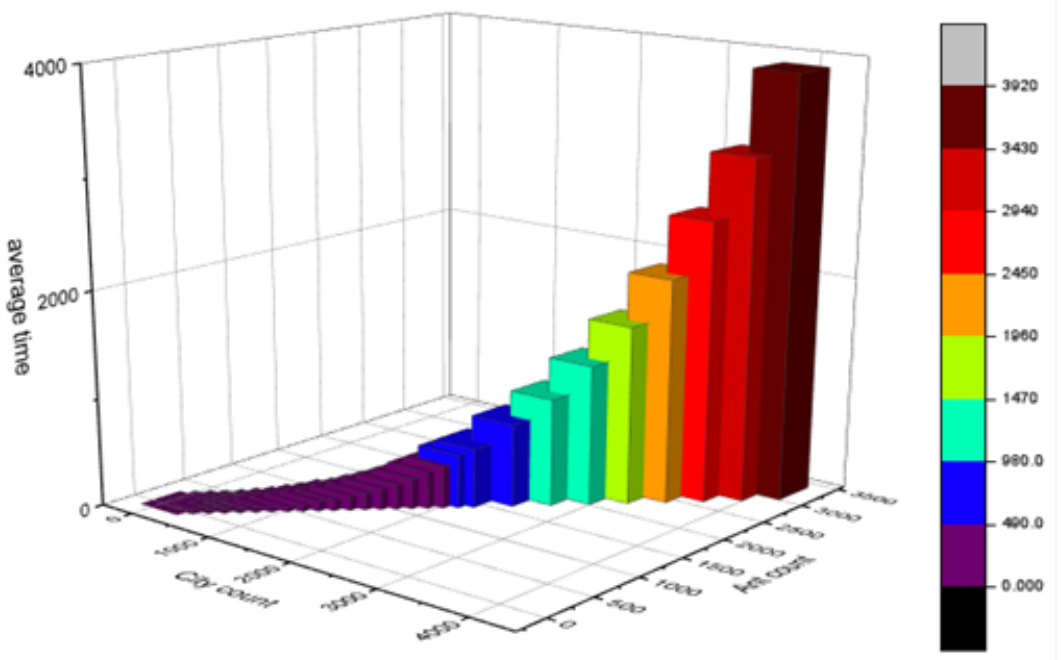

Figure 5.4: Average Time Complexity of OEMACS with Maximum Iteration 2000

5.2.2 Data Analysis of AIVMM Algorithm

We used the following datasets to represent the output of our simulation in which we tried to determine the relation between time and number of VMs added to the system after each time.

Time to instantiate (sec)	Number of VMs
1.1	1000
1.4	1200
1.6	1400
1.9	1600
2.2	1800
2.5	2000
2.8	2200
3	2400
3.4	2600
3.9	2800
4.1	3000

Table 5.2: Time to Simulation Instantiation

Amount of memory (kb)	Number of VMs
1000	1000
1100	1200
1210	1400
1311	1600
1412	1800
1513	2000
1615	2200
1716	2400
1817	2600
1918	2800
2019	3000

Table 5.3: Memory to Simulation Instantiation

Fig.(5.5) and (5.6) represents the amount of time and memory required respectively for the simulation instantiation when the number of virtual machines increases in a host server. The amount of time required is exponential to the number of VMs while the amount of memory required has a linear relationship.

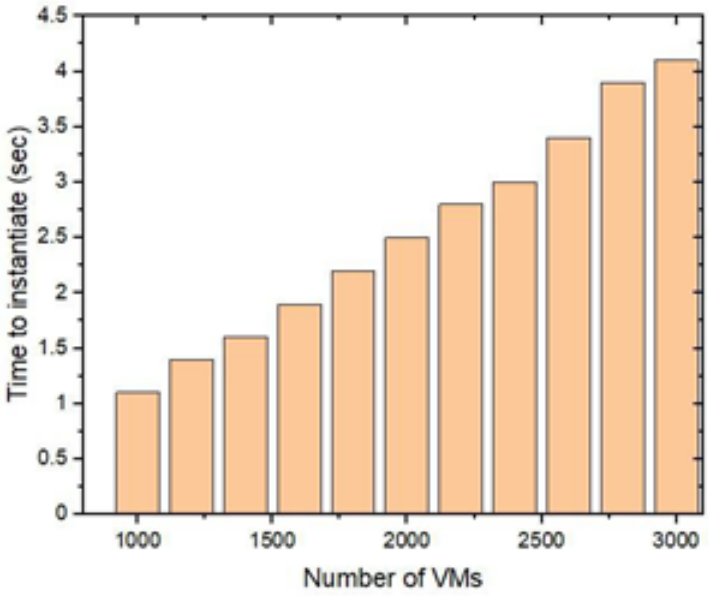

Figure 5.5: Time to Simulation instantiation

Regarding time overhead related to simulation instantiation, the growth in terms of time increases exponentially with the number of virtual machines. Here, the time required to 3000 virtual machines is approximately 4 seconds, which is reasonable and standard for an experiment.

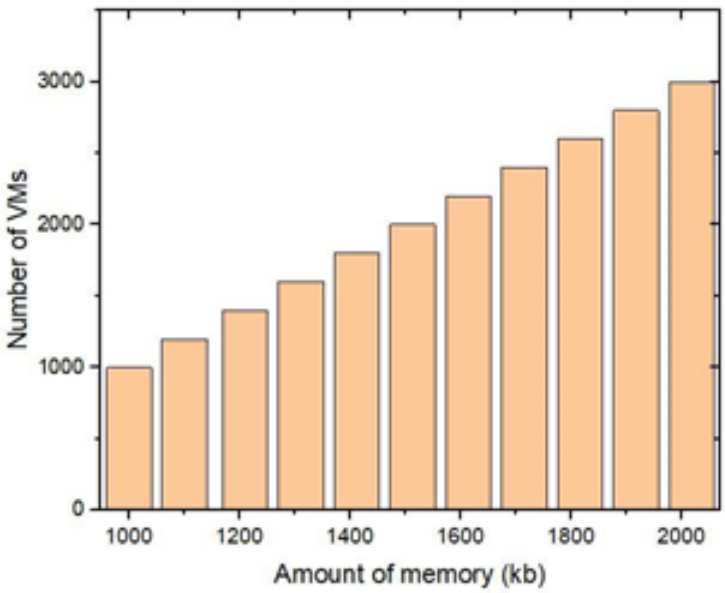

Figure 5.6: Memory to Simulation instantiation

The growth in memory allocation is linear, with an experiment with 3000 virtual machines demanding approximately 2.1 MB RAM. It makes our simulation moderately suitable to run in desktop computers with standard processing powers.

We used the following dataset to represent the comparison between our parent algorithm- OEMACS and collaboration of OEMACS and AIVMM from the perspective of run-time.

VM ID	Time to migrate (sec) OEMACS	Time to migrate (sec) OEMACS+AIVMM
VM1	3.2	5.6
VM2	4.5	6.2
VM3	2.3	4.5
VM4	4.1	6.3
VM5	2.4	4.5
VM6	2.9	7.1
VM7	3.4	5.4
VM8	3.2	5.8
VM9	2.1	4.1
VM10	4.2	6.5
VM11	3.2	5.7
VM12	3.6	6.01
VM13	2.5	4.3
VM14	2.8	4.9
VM15	4.3	5.1

Table 5.4: OEMACS Migration

Fig.(5.7) denotes the comparison between our parent algorithm (OEMACS) and collaboration of all three algorithms mentioned in this paper (OEMACS and AIVMM). The graph is sketched with time to migrate VMs against 15 virtual machines.

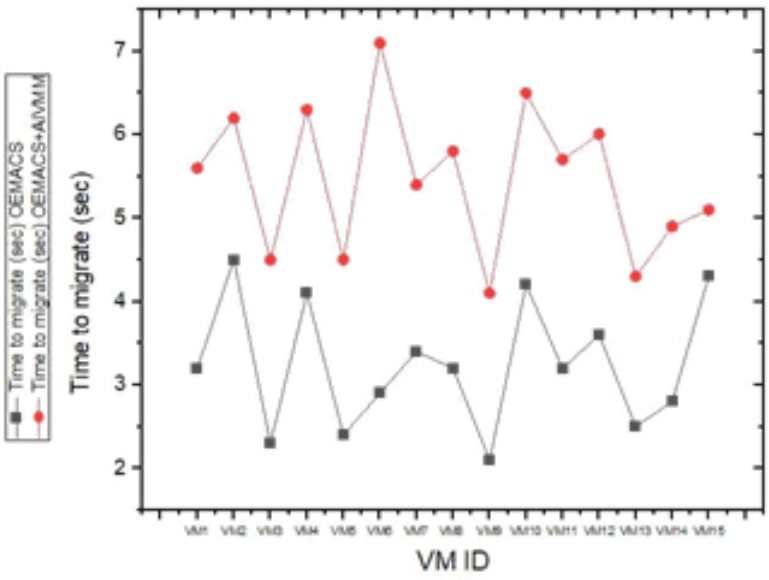

Figure 5.7: OEMACS Vs (OEMACS+AIVMM)

Chapter 6

Conclusion & Future Work

6.1 Conclusion

Virtual Machine Consolidation is a very salient feature of today's cloud infrastructure. With the help of continuous VM migration, cloud has become more resilient and lenient. But because of thermal throttle and excessive heat emission, the process of virtual machine migration has become a major issue. We have designed Active Idle Virtual Machine Migration algorithm which has not only solved the problem but has also consolidated the process to a great extent keeping the collateral parameters intact. Although there are limitations regarding run-time and migration threshold, in near future we plan to solve these issues by collaborating Migration of Idle Machine via Parking Server algorithm along with Active Idle Virtual Machine Migration algorithm.

6.2 Future Work

Our future plan for the AIVMM algorithm is to upgrade it by collaborating with another EC algorithm known as Migration of Idle Machines via Parking Servers (MIMPS) algorithm. In this algorithm, the idle virtual machines of a data center will be initially parked in a parking server, waiting to be assigned a task. After a VM gets assigned, it will switch its mode from idle and active and thus the system will migrate it back to its initial position. If the initial position does not remain vacant, the system will initialize OEMACS algorithm to migrate the VM to its nearest non-overloaded server. Thus, a balance will be created and the idle VMs will not consume unnecessary power or energy in their idle state.

Bibliography

[1] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," Evolutionary Computation, IEEE Transactions on, vol. 1, no. 1, pp. 53–66, 1997

[2] A. Ashraf and I. Porres, "Using ant colony system to consolidate multiple web applications in a cloud environment," in 22nd Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 2014, pp. 482–489.

[3] M. Dorigo, V. Maniezzo, and A.Colorni, "The ant system: optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man, and Cybernetics–Part B, vol. 26, No. 2, pp. 29–41, 1996.

[4] X. Liu, J. Zhang, "An Energy Efficient Ant Colony System for Virtual Machine Placement in Cloud Computing," Evolutionary Computation, IEEE Transactions on, vol. 22, no. 1, pp.30-65, 2018.

[5] C. Yuan, and X. Sun, "Server Consolidation Based on Culture Multiple-Ant-Colony Algorithm in Cloud Computing," Sensors 2019

[6] L.M. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," Proceedings of IEEE International Conference on Evolutionary Computation, IEEE-EC 96, IEEE Press, 1996, pp. 622–627.

[7] D. Fogel, "Applying evolutionary programming to selected traveling salesman problems," Cybernetics and Systems: An International Journal, vol. 24, pp. 27–36, 1993.

[8] R. Michel, M. Middendorf, An ACO algorithm for the shortest super sequence problem, in: D. Come, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, UK, 1999, pp. 51–61.

[9] D. Feng, Z. Wu, Z. Zhang, "A multi-objective migration algorithm as a resource consolidation strategy in cloud computing," Plos One 2019

[10] Y. Gao, H. Guan, L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," Journal of Computer and System Sciences 79 (2013) 1230-1242

[11] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in Proc. IEEE/ACM Int. Conf. Grid Comput., Lyon, France, 2011, pp. 26–33.

[12] Y. Sahu, R. K. Pateriya, and R. K. Gupta, "Cloud server optimization with load balancing and green computing techniques using dynamic compare and balance algorithm," in Proc. IEEE 5th Comput. Intell. Commun. Netw. Conf., Mathura, India, 2013, pp. 527–531.

[13] B. B. J. Suseela and V. Jeyakrishnan, "A multi-objective hybrid ACOPSO optimization algorithm for virtual machine placement in cloud computing," Int. J. Res. Eng. Technol., vol. 3, no. 4, pp. 474–476, 2014.

[14] R. N. Calheiros, R. Ranjan and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," in Wiley Online Library (wileyonlinelibrary.com), 24th August 2010

[15] Buyya R, Ranjan R, Calheiros RN. InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services. Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010), Busan, South Korea. Springer: Germany, 21–23 May 2010; 328–336.

[16] Buyya R, Ranjan R, Calheiros RN. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. Proceedings of the Conference on High Performance Computing and Simulation (HPCS 2009), Leipzig, Germany. IEEE Press: New York, U.S.A., 21–24 June 2009; 1–11.

[17] Y. Tohidirad, S. Abdezadeh, Z. S. Aliabadi, "Virtual Machine Scheduling in Cloud Computing Environment", in International Journal of Managing Public Sector Information and Communication Technologies (IJMPICT), Vol. 6, No. 4, December 2015

[18] R. M. Chawda, O. Kale, "Virtual Machine Migration Techniques in Cloud Environment: A Survey", in International Journal for Scientific Research Development, vol. 1, issue 8, 2013

[19] R. Ranjan and R. Buyya. Decentralized Overlay for Federation of Enterprise Clouds. Handbook of Research on Scalable Computing Technologies, K. Li et. al. (ed), IGI Global, USA, 2009 (in press).

[20] J. Kommeri, T. Niemi, J. K. Nurminen, "Energy efficiency of dynamic management of virtual cluster with heterogeneous hardware", in International Journal for Scientific Research Development, vol. 5, issue 9, 2015

[21] J. Kommeri, T. Niemi, O. Helin, "Energy efficiency of server virtualization", in International Journal On Advances in Intelligent Systems, vol. 5 (2012)

[22] M. Dorigo, M. Birattari and T. Stutzle, "Ant Colony Optimization Artificial Ants as a Computational Intelligence Technique", Iridia – Technical Report Series: TR/IRIDIA/2006-023.

[23] F. Salfner, P. Troger, A. Polze, "Downtime Analysis of Virtual Machine Live Migration", in The Fourth International Conference on Dependability, vol. 12, 2011

[24] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum, "Optimizing the migration of virtual computers," SIGOPS Oper. Syst. Rev., vol. 36, no. SI, pp. 377– 390, 2002.

[25] S. Akoush, R. Sohan, A. Rice, A. W. Moore, and A. Hopper, "Predicting the Performance of Virtual Machine Migration," Modeling, Analysis, and Simulation of Computer Systems, International Symposium on, vol. 0, pp. 37–46, 2010.

[26] I. K. Kim, S. Zeng, M. Humphrey, "A Supervised Learning Model for Identifying Inactive VMs in Private Cloud Data Centers", in Middleware Industry '16, December 12-16, 2016, Trento, Italy

[27] S. N. Wang, H. X. Gu, and G. Wu, "A new approach to multi-objective virtual machine placement in virtualized data center," in Proc. IEEE 8th Int. Conf. Netw. Archit. Stor., Xi'an, China, 2013, pp. 331–335.

[28] P. Patel, A. Ranabahu, and A. Sheth, "Service level agreement in cloud computing," in Proceedings of the Cloud Workshops at OOPSLA, pp. 1–10, 2009.

[29] H. Liu, C.-Z. Xu, H. Jin, J. Gong, and X. Liao, "Performance and energy modeling for live migration of virtual machines," in Proceedings of the 20th ACMInternational SymposiumonHigh- Performance Parallel andDistributed Computing (HPDC '11), pp. 171–181, June 2011.

[30] F. Farahnakian, T. Pahikkala, P. Liljeberg, and J. Plosila, "Energy aware consolidation algorithm based on K-nearest neighbor regression for cloud data centers," in Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on, Dec 2013, pp. 256–259.