Using Ant Colony System to Consolidate VMs for Green Cloud Computing

Report submitted in fulfillment of the requirements for the Exploratory Project of

Second Year B.Tech.

by

Shivansh Saini, Parv Jain

17075054, 17075063

Under the guidance of

Dr. Ravi Shankar Singh

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi 221005, India
May 2019

Dedicated to

My parents, teachers,.....

Declaration

I certify that

1. The work contained in this report is original and has been done by myself and the general supervision of my supervisor.

2. The work has not been submitted for any project.

3. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi
Date: May 3, 2019

Shivansh Saini, Parv Jain
B.Tech.
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Certificate

*This is to certify that the work contained in this report entitled "**Using Ant Colony System to Consolidate VMs for Green Cloud Computing**" being submitted by **Shivansh Saini, Parv Jain (Roll No. 17075054, 17075063)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Dr. Ravi Shankar Singh
Associate Professor
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Place: IIT (BHU) Varanasi
Date: May 3, 2019

Acknowledgments

I owe my deep gratitude to our project teacher Dr. Ravi Shankar Singh, who took keen interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system. I would also like to thank Mr. Medara Rambabau, Research scholar for spending his valuable time to guide us through the project.

Place: IIT (BHU) Varanasi

Date: May 3, 2019 **Shivansh Saini, Parv Jain**

Abstract

With the increasing demand for cloud services, the high energy consumption of cloud data centres is a significant problem that needs to be handled. One way to overcome this problem is to consolidate the Virtual Machines on the Data centres dynamically. A VM consolidation approach uses live migration of VMs so that some of the under-loaded Physical Machines (PMs) can be switched off or put into a low-power mode. If a Physical Machine (PM) has no consumption at a specified time, it still consumes some energy(static energy). In this project, we further extend this approach by putting the Physical Machine (PM) to sleep mode, if not in use. We try to implement the algorithm which will reduce the energy consumption of cloud datacenters while maintaining the desired Quality of Service(QoS). The VM consolidation is an NP-Hard problem, so we use a metaheuristic algorithm called Ant Colony System (ACS). The proposed ACS-based VM Consolidation (ACS-VMC) approach finds a near-optimal solution based on a specified objective function.

Contents

Chapter 1

Introduction

Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. The term is generally used to describe data centres available to many users over the Internet. Large clouds, predominant today, often have functions distributed over multiple locations from central servers. If the connection to the user is relatively close, it may be designated an edge server[1]. Cloud computing is a paradigm of distributive computing and is used in many aspects. In the past decade, Cloud Computing has gone from the obscure service to multi-billion dollar corporations to become an integral part of the internet ecosystem. The cloud has long operated at the juncture of the most meaningful technologies to dominate the data management landscape, facilitating everything from big data deployments to predictive and prescriptive analytics, applications of cognitive computing to those of edge computing.

1.1 Motivation of the Research Work

Due to the ever-increasing demand for cloud services, there has been a significant increase in energy consumption (EC) of the cloud data centres. High energy consumption not only translates to a high operating cost but also leads to higher carbon emissions[2]. With the decreasing sources of energy and the ill effects on the environment, research communities are being challenged to find efficient energy-aware resource management strategies. According to a recent Greenpeace report, Make IT Green: Cloud Computing and its Contribution to Climate Change, the electricity consumed by cloud computing globally will increase from 632 billion kilowatt hours in 2007 to 1,963 billion kWh by 2020, and the associated CO2 equivalent emissions would reach 1,034 megatonnes.[14] With the ever-increasing demand of cloud services, it becomes a need of an hour to optimise the use of cloud services in such a way that the requirements of users are well met and wastage of various resources is also reduced.

Chapter 2

Cloud Computing

2.1 Basics of Cloud Computing

Simply put, cloud computing is the delivery of computing servicesservers, storage, databases, networking, software, analytics, intelligence and moreover the Internet (the cloud) to offer faster innovation, flexible resources and economies of scale[3]. A person does not need to worry about the infrastructure, nor for the software or memory management or any other issue. Every service is available on demand, and we can scale in or scale out our usage at anytime we want. This helps to save our money as we use cloud services on a pay-as-you-go basis. Cloud computing is a significant shift from the traditional way companies think about IT resources. The six common reasons organisations are turning to cloud computing services are Cost, Speed, Global Scale, Productivity, Performance and Security.

2.1.1 Data Centres

The network of hardware resources (processing elements, memory and storage) offered by the cloud provider enabling the delivery of shared software applications and data.

2.1.2 Physical Machine(PM)

A Physical Machine(PM) or compute node provides the ephemeral storage, networking, memory, and processing resources that can be consumed by virtual machine instances.

2.1.3 Middleware

Middleware is a layer, which extends over multiple machines and offers each user the same interface, thus organizing the distributed system.

2.1.4 Broker

A broker is part of a Cloudsim datacenter and acts on behalf of a user. It hides VM management, VM creation, submission of cloudlets to these VMs and destruction of VMs.

2.1.5 Cloudlets

A cloudlet is a mobility-enhanced small-scale cloud datacenter that is located at the edge of the Internet. The main purpose of the cloudlet is supporting resource-intensive and interactive mobile applications by providing powerful computing resources to mobile devices with lower latency.

2.1.6 Virtual Machine(VM)

A virtual machine (VM) is an operating system (OS) or application environment that is installed on software, which imitates dedicated hardware.

A cloud data centre consists of m heterogeneous PMs that have different resource capacities. Each PM contains a CPU, which is often a multi-core. The CPU performance can be defined in terms of Millions of Instructions Per Second (MIPS). Also, a

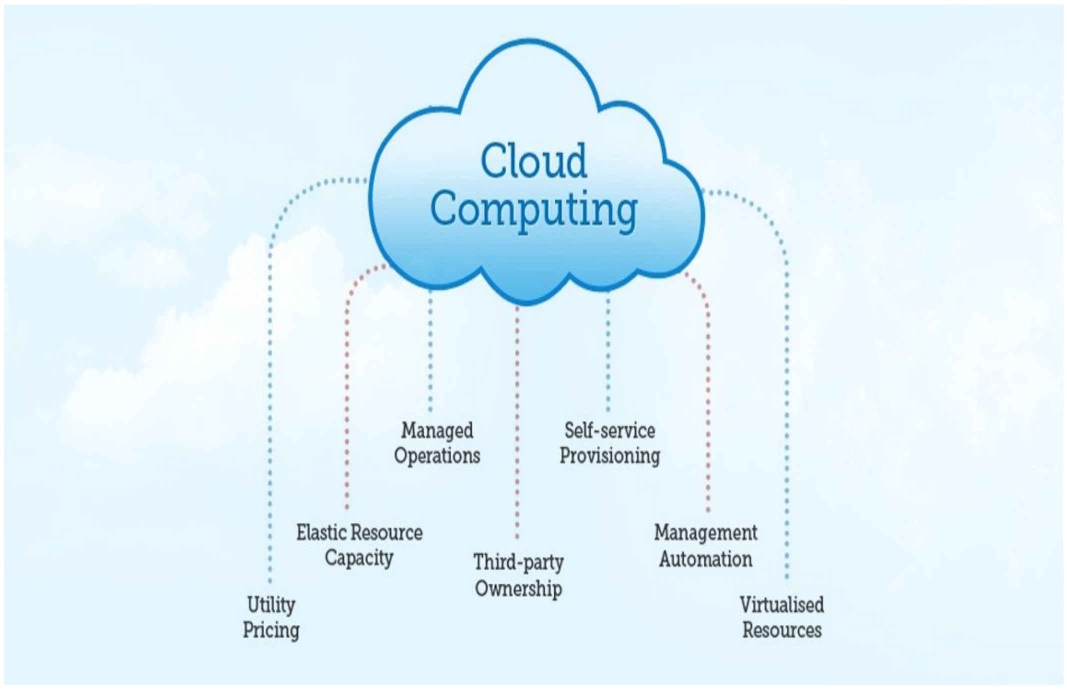


Figure 2.1 Cloud

PM is also characterized by the amount of memory, network I/O, and storage capacity. At any given time, a cloud data centre usually serves many simultaneous users. Users submit their requests for provisioning of n VMs, which are allocated to the PMs. The length of each request is specified in millions of instructions (MI).

2.2 Deployment models

2.2.1 Public cloud

Public Cloud allows the accessibility of systems and its services to its clients/users publicly. Examples of Public cloud providers: Google, AWS, Microsoft etc.

2.2.2 Private Cloud

Private Cloud allows the accessibility of systems and services within a specific boundary or organization. It provides enhanced level of security as compared to public cloud.

2.2.3 Community cloud

Community Cloud is the one in which the setup of the cloud is shared manually among different organizations that belong to the same community or area.

2.2.4 Hybrid cloud

Hybrid Cloud is one in which services are combination of other clouds.

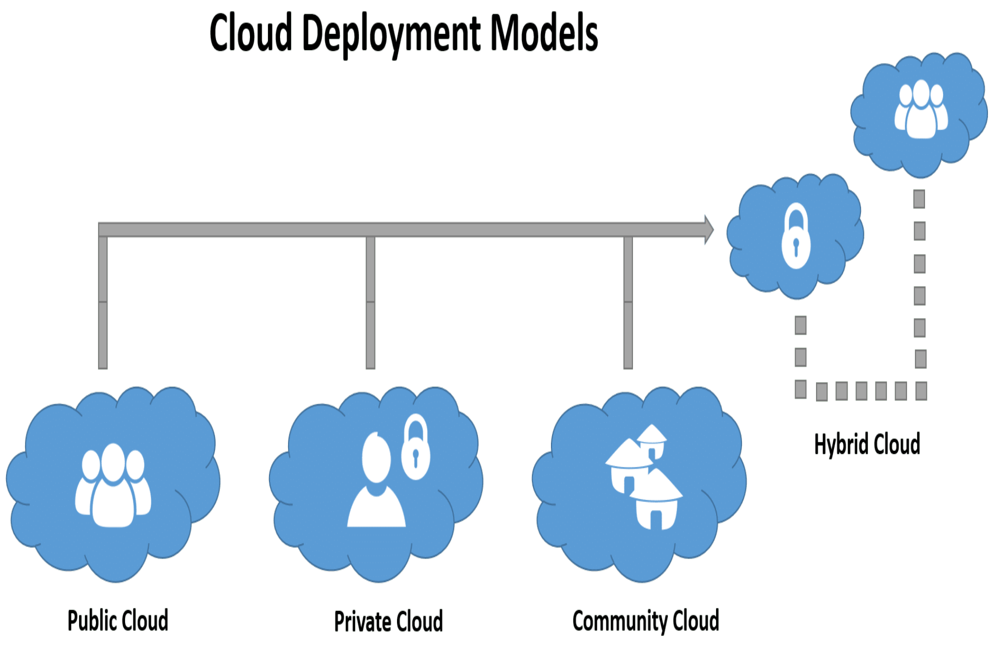


Figure 2.2 Various Deployement Models

2.3 Types of Cloud Services

2.3.1 SaaS (Software as a Service)

SaaS is the software distribution model that is deployed on the internet in which a cloud service provider provides applications.

2.3.2 PaaS (Platform as a Service)

PaaS is the software distribution model that is deployed on the internet in which a cloud service provider provides a framework on which a developer can build upon to develop or customize applications.

2.3.3 IaaS (Infrastructure as a Service)

IaaS is a way of providing Cloud computing infrastructure such as virtual machines, storage drives, servers, operating systems and networks.

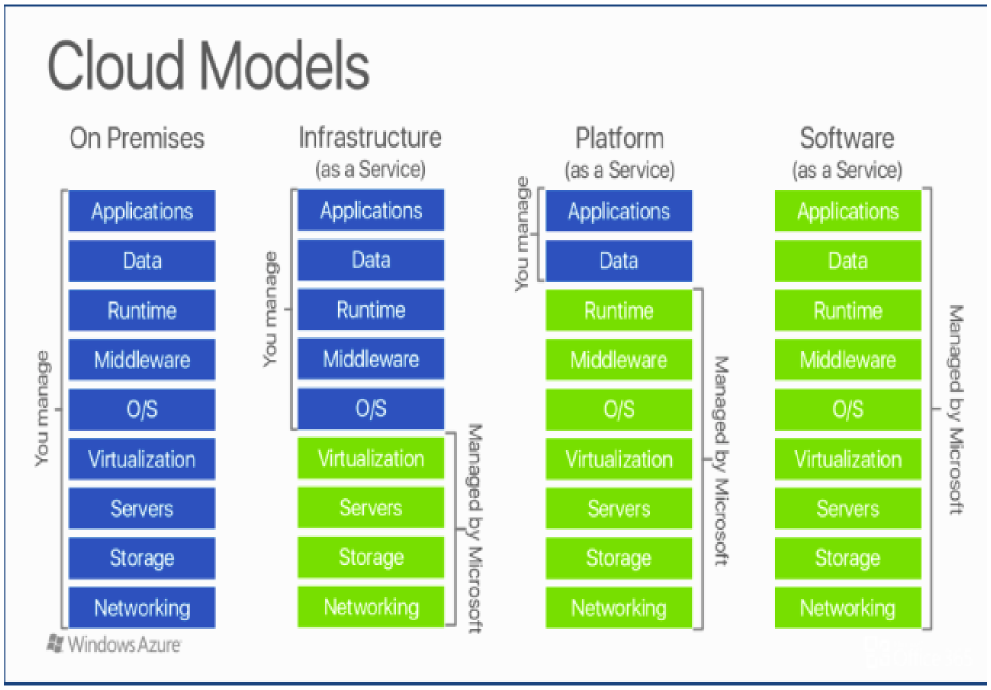


Figure 2.3 Services provided by various Cloud Models

2.4 Goals of Cloud Computing

2.4.1 Transparency

Transparency of a particular aspect of a distributed system means to hide away the component for that aspect from the users. Types - Access, Location, Migration, Relocation, Replication, Concurrency and Failure.

2.4.2 Openness

System should offer services according to standard rules that describe the syntax and semantics of those services. Services are generally specified through interfaces defined in Interface Definition Language.

2.4.3 Scalability

System should be scalable with respect to at least 3 dimensions - size, geography and administration.

2.4.4 QoS and SLA

QoS requirements are commonly formalised in the form of Service Level Agreements (SLAs), which specify enterprise service-level requirements for the data centre in terms of minimum latency or maximum response time. A SLA is a contract between a service provider and the end user that defines the level of service expected from the service provider.

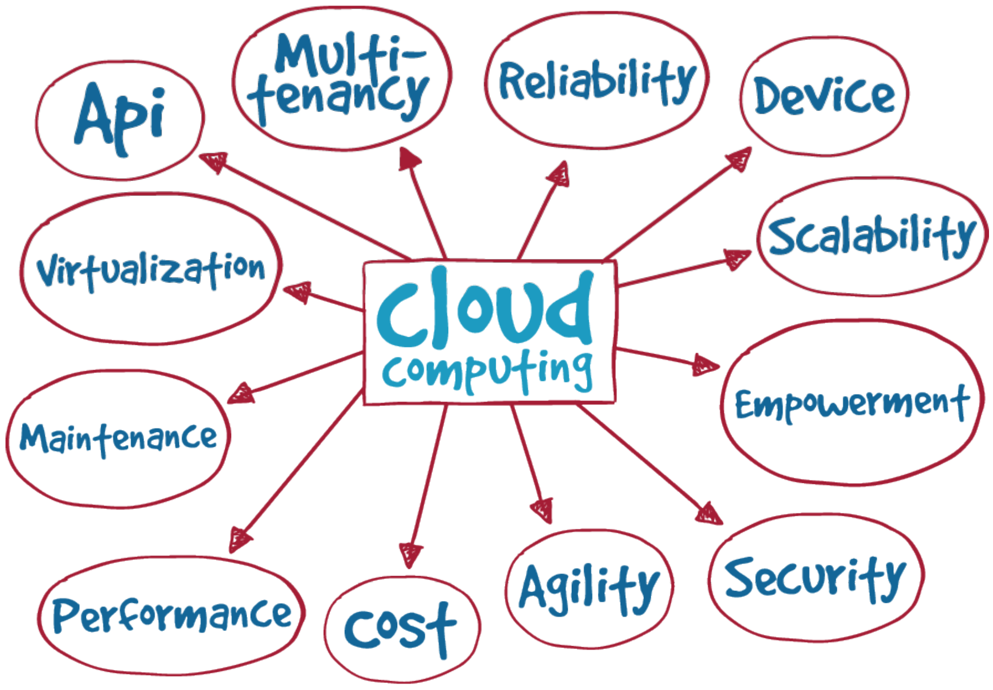


Figure 2.4 Goals

Chapter 3

Framework and Methodology

3.1 Technology stack

3.1.1 Cloudsim Plus

CloudSim Plus is a full-featured and a highly extensible simulation framework enabling modelling that is used for simulation, and experimentation of Cloud computing. It allows users to focus on specific system design issues to be investigated, without concerning the low-level details related to Cloud-based infrastructures and services.

Cloud computing is the leading technology for the delivery of reliable, secure, fault-tolerant, sustainable, and scalable computational services. For assurance of such characteristics in cloud systems under development, it is required timely, repeatable, and controllable methodologies for evaluation of new cloud applications and policies, before actual development of cloud products. Because the utilisation of real testbeds limits the experiments to the scale of the testbed and makes the reproduction of results cumbersome, the computer-based simulation may constitute an interesting tool. This project is suitable to quickly develop such simulation scenarios and run them quickly, in a typical PC.[4]

3.1.2 Java

Java is a popular programming language, created in 1995. Oracle owns it, and more than 3 billion devices run Java. It is used for Mobile applications (especially Android apps), Desktop applications, Web applications, Web servers and application servers, Games, Database connection and much more.[5]

The reason Java is used so extensively is: Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.), it is one of the most popular programming languages in the world; it is easy to learn and simple to use, it is open-source and free, it is secure, fast and powerful and it has huge community support (tens of millions of developers).

3.2 Ant Colony Optimization(ACO)

ACO is a multi-agent approach (Meta-Heuristic) to challenging combinatorial optimisation problems, such as travelling salesman problem (TSP) and network routing. This algorithm is inspired by the behaviour of real ants. While moving from their nest to a food source and back to their nest, ants deposit a chemical substance on their path called pheromone. Other ants can smell pheromone, and they tend to prefer routes with a higher pheromone concentration. The pheromone-based communication of biological ants is often the predominant paradigm used.
Thus, ants use a simple form of indirect communication called stigmergy to find better paths between their nest and the food source. It has been shown experimentally that this simple pheromone trail following behaviour of ants can give rise to the emergence of the shortest paths. The optimum solutions emerge only from the global cooperation among the members of the colony who concurrently build different solutions. Moreover, to find a high-quality solution, it is imperative to avoid stagnation, which is a premature convergence to a suboptimal solution or a situation where all ants end

up finding the same solution without sufficient exploration of the search space. In ACO metaheuristic, stagnation is avoided mainly by using pheromone evaporation and stochastic state transitions.[6]

This algorithm is a member of the ant colony algorithms family, in swarm intelligence methods, and it constitutes some metaheuristic optimisations. Initially proposed by Marco Dorigo in 1992 in his PhD thesis, the first algorithm was aiming to search for an optimal path in a graph, based on the behaviour of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behaviour of ants. Like most metaheuristics, it is challenging to estimate the theoretical speed of convergence. [7]

Ant colony optimisation algorithms have been applied to many combinatorial optimisation problems, like network routing. It has also been used to produce near-optimal solutions to the travelling salesman problem.

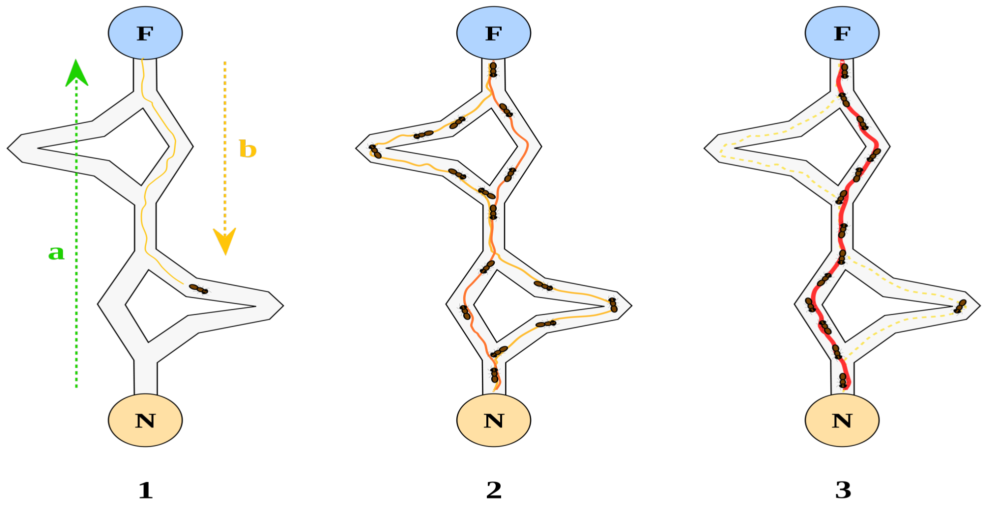


Figure 3.1 Ant Colony Optimization(ACO)

3.3 Using Ant Colony Optimization(ACO) algorithm for optimal VM consolidation

The Virtual Machines are initially allocated to the physical machines based on the Best Fit Decreasing Approach. This approach first sorts all the Virtual Machines in decreasing order of their utilisations and then starts allocating. It allocates the physical machines in such a way that unused capacity of destination PM is minimised.
Due to dynamic workloads, the resource utilisation of VMs continues to vary over time. Therefore, an initial efficient allocation approach needs to be augmented with a VM consolidation algorithm that can be applied periodically. In our proposed approach, the ACS-VMC algorithm is applied periodically to adapt and optimise the VM placement according to the workload.
A local agent (LA) detects PM status: normal, over- Loaded, predicted overloaded, or under-loaded. Moreover, a global agent (GA) dynamically consolidates VMs into a reduced number of PMs by using our proposed ACS-based VM Consolidation algorithm as shown in Fig. 3.2.
In this system, the GA collects the status of individual PMs from the LAs and builds a global best migration plan using our algorithm. The migration plan is then followed by VMMs to perform actual migrations of VMs. The PMs in the VM consolidation problem is analogous to the cities in the TSP, while the tuples are analogous to the edges that connect the cities.[8]

3.3.1 Algorithm

The pseudocode of the proposed ACS-based VM Consolidation algorithm is given as Fig. 3.3. For the sake of clarity, the concepts used in the proposed algorithm and their notations are tabulated in Fig. 3.11.

The ACS algorithm is described as:

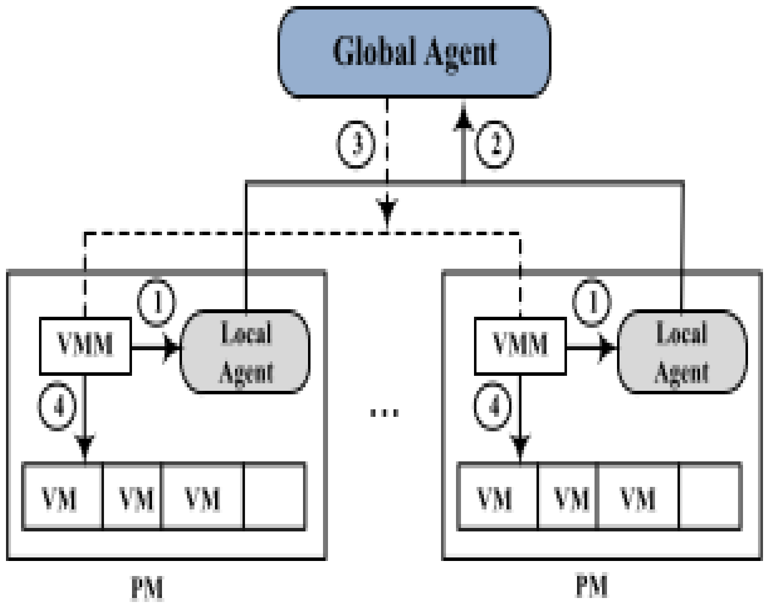
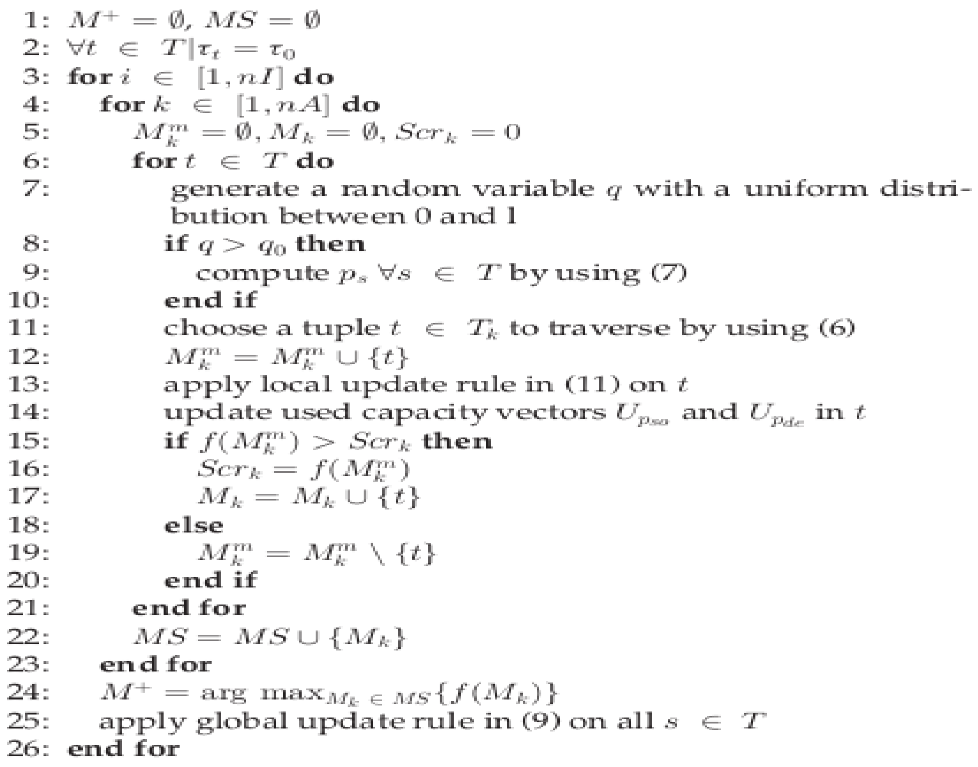


Figure 3.2 The System Architecture



```
1:  $M^+ = \emptyset$, $MS = \emptyset$
2:  $\forall t \in T | \tau_t = \tau_0$
3:  for $i \in [1, nI]$ do
4:     for $k \in [1, nA]$ do
5:        $M_k^m = \emptyset, M_k = \emptyset, Scr_k = 0$
6:        for $t \in T$ do
7:           generate a random variable $q$ with a uniform distribution between 0 and 1
8:           if $q > q_0$ then
9:              compute $p_s \forall s \in T$ by using (7)
10:          end if
11:          choose a tuple $t \in T_k$ to traverse by using (6)
12:          $M_k^m = M_k^m \cup \{t\}$
13:          apply local update rule in (11) on $t$
14:          update used capacity vectors $U_{p_{so}}$ and $U_{p_{de}}$ in $t$
15:          if $f(M_k^m) > Scr_k$ then
16:             $Scr_k = f(M_k^m)$
17:             $M_k = M_k \cup \{t\}$
18:          else
19:             $M_k^m = M_k^m \setminus \{t\}$
20:          end if
21:       end for
22:       $MS = MS \cup \{M_k\}$
23:    end for
24:    $M^+ = \arg\max_{M_k \in MS}\{f(M_k)\}$
25:    apply global update rule in (9) on all $s \in T$
26: end for
```

Figure 3.3 Algorithm for VM Consolidation using ACO

- Each PM hosts one or more VM's from the set of Virtual Machines. Moreover, in context to VM migration, each PM is a potential source PM for the VMs already residing on that PM.

- Thus, the proposed ACS-VMC algorithm creates a set of tuples T, where each tuple t consists of three elements: the source PM p_{so} , the VM to be migrated v, and the destination PM p_{de}.
  $t = (p_{so}, v, p_{de})$

- It is imperative to reduce the computation time of the consolidation algorithm, which is primarily based on the number of tuples |T|. Thus, when making the set of tuples T, the algorithm applies three constraints, which result in a reduced set of tuples by removing some least important and obsolete tuples.
  The first constraint ensures that only a predicted overloaded, an overloaded, or an under-loaded PM is used as a source PM p_{so}.
  The second constraint further restricts the size of the set of tuples |T| by ensuring that none of the overloaded P and predicted overloaded PMs become a destination PM p_{de}
  The third constraint ensures that no VM migration takes place to a sleeping host if the source host is not overloaded.

- We want a migration plan, which, when enforced, would result in a minimal set of active PMs needed to host all VMs without compromising their performance. Thus, the objective function of the proposed algorithm is given in Fig. 3.4.

$$f(M) = |P_s|^{\gamma} + \frac{1}{|M|},$$

Figure 3.4 Objective Function

- Unlike the TSP, there is no notion of a path in the VM consolidation problem. Therefore, in our approach, the pheromone is deposited on the tuples. Each of the nA ants uses a stochastic state transition rule to choose the next tuple to traverse. The state transition rule in ACS is called pseudo-random-proportional-rule [9]. According to this rule, an ant k chooses a tuple s to traverse next by applying the rule given in the Fig. 3.5. S is a random variable selected according to the probability distribution given in (7), where the probability p_s of choosing the tuple s is given by the formulae in Fig. 3.6 where Γ denotes the amount

$$s = \begin{cases} \arg\max_{u \in T_k} \{[\tau_u] \cdot [\eta_u]^{\beta}\}, & \text{if } q \leq q_0 \\ S, & \text{otherwise}, \end{cases} \tag{6}$$

Figure 3.5 Formulae to choose the next tuple to traverse

of pheromone, and η represents the heuristic value associated with a particular tuple. β is a parameter to determine the relative importance of the heuristic value concerning the pheromone value. The Heuristic value of a tuple is defined according to formulae given in Fig 3.7.

$$p_s = \begin{cases} \frac{[\tau_s] \cdot [\eta_s]^{\beta}}{\sum_{u \in T_k} [\tau_u] \cdot [\eta_u]^{\beta}}, & \text{if } s \in T_k, \\ 0, & \text{otherwise}. \end{cases} \tag{7}$$

Figure 3.6 Formulae to calculate the probability of choosing the tuples

- The exploitation helps the ants to quickly converge to a high-quality solution, while at the same time, the biased exploration helps them to avoid stagnation by allowing a wider exploration of the search space.

- In addition to the stochastic state transition rule, ACS also uses a global and a local pheromone trail evaporation rule. The global pheromone trail evaporation

$$\eta_s = \begin{cases} \left(\left|C_{p_{de}} - (U_{p_{de}} + U_v)\right|_1\right)^{-1}, & \text{if } U_{p_{de}} + U_v \leq C_{p_{de}} \\ 0, & \text{otherwise,} \end{cases}$$

Figure 3.7 Heuristic value of a tuple

rule is applied towards the end of an iteration after all ants complete their migration plans. It is defined according to Fig. 3.8 where $\Delta+$ is further explained in Fig. 3.9

$$\tau_s = (1 - \alpha) \cdot \tau_s + \alpha \cdot \Delta^+_{\tau_s}, \qquad (9)$$

Figure 3.8 Global Pheromone Trail Evaporation Rule

$$\Delta^+_{\tau_s} = \begin{cases} f(M^+), & \text{if } s \in M^+ \\ 0, & \text{otherwise.} \end{cases}$$

Figure 3.9 Additional pheromone amount to reward tuples belonging to global migration plan

- The local pheromone trail update rule is applied on a tuple when an ant traverses the tuple while making its migration plan. It is according to Fig. 3.10. Γ_0 is the initial pheromone level, which is an experimentally obtained value.

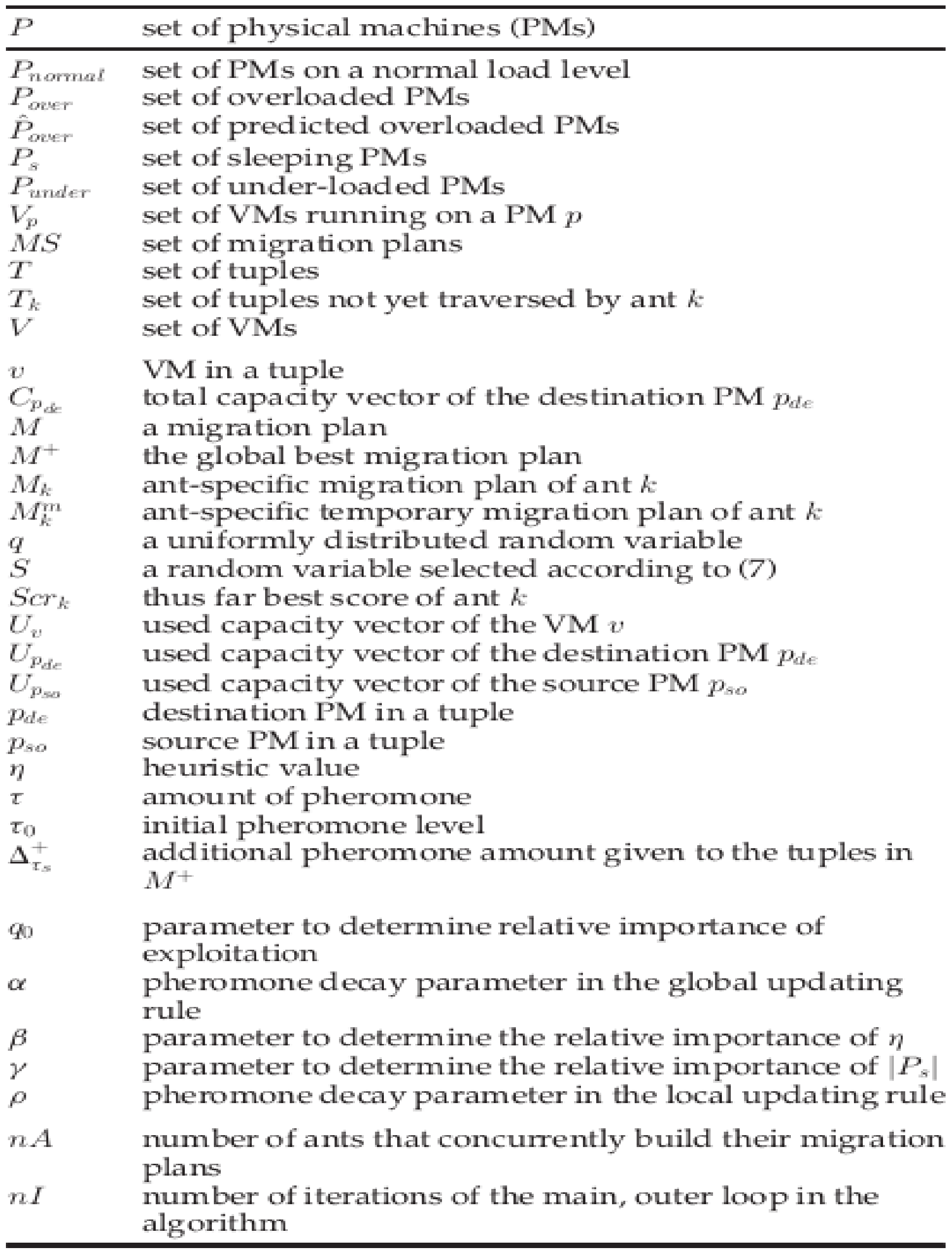
- The above algorithm creates the tuples and sets the pheromone value of each tuple to Γ_0. The algorithm iterates over nI iterations. In each iteration, all the nA ants concurrently build their migration plans. Each ant traverses over all the tuples. At a given time it chooses the next tuple according to formulae (6) and adds it to the local migration plan. The local pheromone trail update rule is applied, and the used capacity vectors of physical machines are updated. If the local migration plan of the ant yields the score better than its previous

$$\tau_s = (1 - \rho) \cdot \tau_s + \rho \cdot \tau_0,$$

Figure 3.10 Local Pheromone Trail Update Rule

best score, the migration plan and the global scores are changed. Finally, after all the ants have traversed their paths, the best global score and the respective migration plan is selected.

The pseudo-random-proportional-rule in ACS and the global pheromone trail update rule are intended to make the search more directed. The pseudo-random-proportional-rule prefers tuples with a higher pheromone level and a higher heuristic value. Therefore, the ants try to search other high quality solutions in a close proximity of the thus far global best solution. On the other hand, the local pheromone trail update rule complements exploration of other high quality solutions that may exist far from the best-so-far global solution. This is because whenever an ant traverses a tuple and applies the local pheromone trail update rule, the tuple looses some of its pheromone and becomes less attractive for other ants. Therefore, it helps in avoiding stagnation where all ants end up finding the same solution or where they prematurely converge to a suboptimal solution.



P	set of physical machines (PMs)
P_{normal}	set of PMs on a normal load level
P_{over}	set of overloaded PMs
$\hat{P}_{over}$	set of predicted overloaded PMs
P_s	set of sleeping PMs
P_{under}	set of under-loaded PMs
V_p	set of VMs running on a PM p
MS	set of migration plans
T	set of tuples
T_k	set of tuples not yet traversed by ant k
V	set of VMs
v	VM in a tuple
$C_{p_{de}}$	total capacity vector of the destination PM p_{de}
M	a migration plan
M^+	the global best migration plan
M_k	ant-specific migration plan of ant k
M_k^m	ant-specific temporary migration plan of ant k
q	a uniformly distributed random variable
S	a random variable selected according to (7)
Scr_k	thus far best score of ant k
U_v	used capacity vector of the VM v
$U_{p_{de}}$	used capacity vector of the destination PM p_{de}
$U_{p_{so}}$	used capacity vector of the source PM p_{so}
p_{de}	destination PM in a tuple
p_{so}	source PM in a tuple
η	heuristic value
τ	amount of pheromone
τ_0	initial pheromone level
$\Delta_{\tau_s}^+$	additional pheromone amount given to the tuples in M^+
q_0	parameter to determine relative importance of exploitation
α	pheromone decay parameter in the global updating rule
β	parameter to determine the relative importance of η
γ	parameter to determine the relative importance of $\|P_s\|$
ρ	pheromone decay parameter in the local updating rule
nA	number of ants that concurrently build their migration plans
nI	number of iterations of the main, outer loop in the algorithm

Figure 3.11 Summary of the concepts and their Notions

Chapter 4

Results

4.1 Experimental design and setup

For the benchmarks of our algorithm, we've used the open data, provided by **Numerical Aerodynamic Simulation (NAS) Systems Division** at NASA Ames Research Center. The workload was logged for three months from October 1993 in a **128-node iPSC/860 hypercube**[10]. We used the cleaned and converted log in Standard Workload format[11].

SwfWorkloadFileReader class in Cloudsim Plus was used to read and build cloudlets using this file. Each cloudlet was assigned a suitable VM, and correspondingly hosts were created according to requirements of VMs at initial simulation. For our hosts, we selected one server configuration of **HP ProLiant ML110 G3 (1 x [Pentium D930 3000 MHz, 2 cores], 4GB)**. CPUs are sufficient to evaluate resource management methods that are designed for multi-core CPU architectures.

Table 4.1 Amount of energy consumption of HP G3 server at different load levels.

Load Levels(%)	**Energy Consumption(Watts)**
0	105
10	112
20	118
30	125
40	131
50	137
60	147
70	153
80	157
90	164
100	169

4.2 Benchmarks

To evaluate the performance of our implemented algorithm, we considered four metrics - SLA violations, energy consumption, number of migrations, and energy-SLA violations.

4.2.1 SLA Violations

Maintaining the desired QoS is an important requirement for cloud data centres. In our simulations, we also simulated an SLA contract and enforced it in the datacenter. All the jobs submitted to broker should be completed. When at least some cloudlets couldn't complete their job, then it's in violation of SLA. Our algorithm should minimise such violations.

4.2.2 Energy Consumption

We consider the total energy consumption of the physical resources in a data centre that is required to handle the application workloads. The energy consumption of a PM depends on the utilisation of its CPU, memory, disk, and network card. Most studies have shown that the CPU consumes more power than memory, disk storage, and network interface. Therefore, the resource utilisation of a PM is usually represented by its CPU utilisation. Instead of using an analytical model of energy consumption, we used the real data in the SPECpower benchmark [12]. Table 4.1 illustrates the amount of energy consumption of HP G3 server at different load levels. Energy consumptions of the whole datacenter should be minimised for our goal of green cloud computing.

4.2.3 VM Migrations

Live VM migration is a costly operation that includes some amount of CPU processing on the source PM, the link bandwidth between the source and destination PMs, the downtime of the services on the migrating VM, and the total migration time [15]. Therefore, one of our objectives was to minimize the number of migrations. The length of a VM migration in CloudSim takes as long as it needs to migrate the memory assigned to the VM over the network bandwidth link between source and destination PMs.

4.3 Experimental Results

In this section, we compare the proposed ACS-VMC approach with four heuristic algorithms for dynamic reallocation of VMs in datacentres. [13].

- Local Regression - The main idea of the method of local regression is fitting simple models to localized subsets of data to build up a curve that approximates the original data. The proposed method is based on the Loess's method [16] proposed by Cleveland [17]

- Median Absolute Deviation - It uses median absolute deviation for a host to be marked as overloaded

- Best Fit Static Threshold - This VM placement algorithm assigns the VM in such way that power consumption is increased minimally. VM is placed in the host with decreasing utilization order.

- Inter Quartile Range - It decides the threshold of a host to be marked as overloaded using interquartile range,

Table 4.2 ACO parameters

α	β	γ	ρ	Q_0	nA	nI	Γ_0
0.1	0.9	5	0.1	0.9	3	2	10.0

Due to unavailability of high-end systems, we ran our simulations on a minimum amount of reliable data to get results faster. Moreover, in our simulations, we set 500 seconds as the Scheduling Interval, which is the the scheduling delay to process each event received by the datacenter. For our ACS-VMC algorithm we have used the parameters as mentioned in Table 4.2.

We ran simulations with 5, 10, 20 and 25 cloudlets to test our algorithm against all the cases. Keep in mind that each of the cloudlets had different start time and length to simulate a real workload. From the Fig. 4.1, with less number of cloudlets, our algorithm behaved as a simple best-fit approach with no migrations and hence consumed the same amount of power. With increasing jobs, our algorithm did some VM migrations and consumed less power than in the case of no migrations. Other heuristic algorithms failed to migrate any VM at all and even consumed a lot more power, besides BFST. Best Fit Static Threshold algorithm even failed to execute at least 5 VMs, which were supposed to start in the beginning. Because of its brazen SLA violations, BFST can't be considered better than our algorithm solely due to consuming less energy.

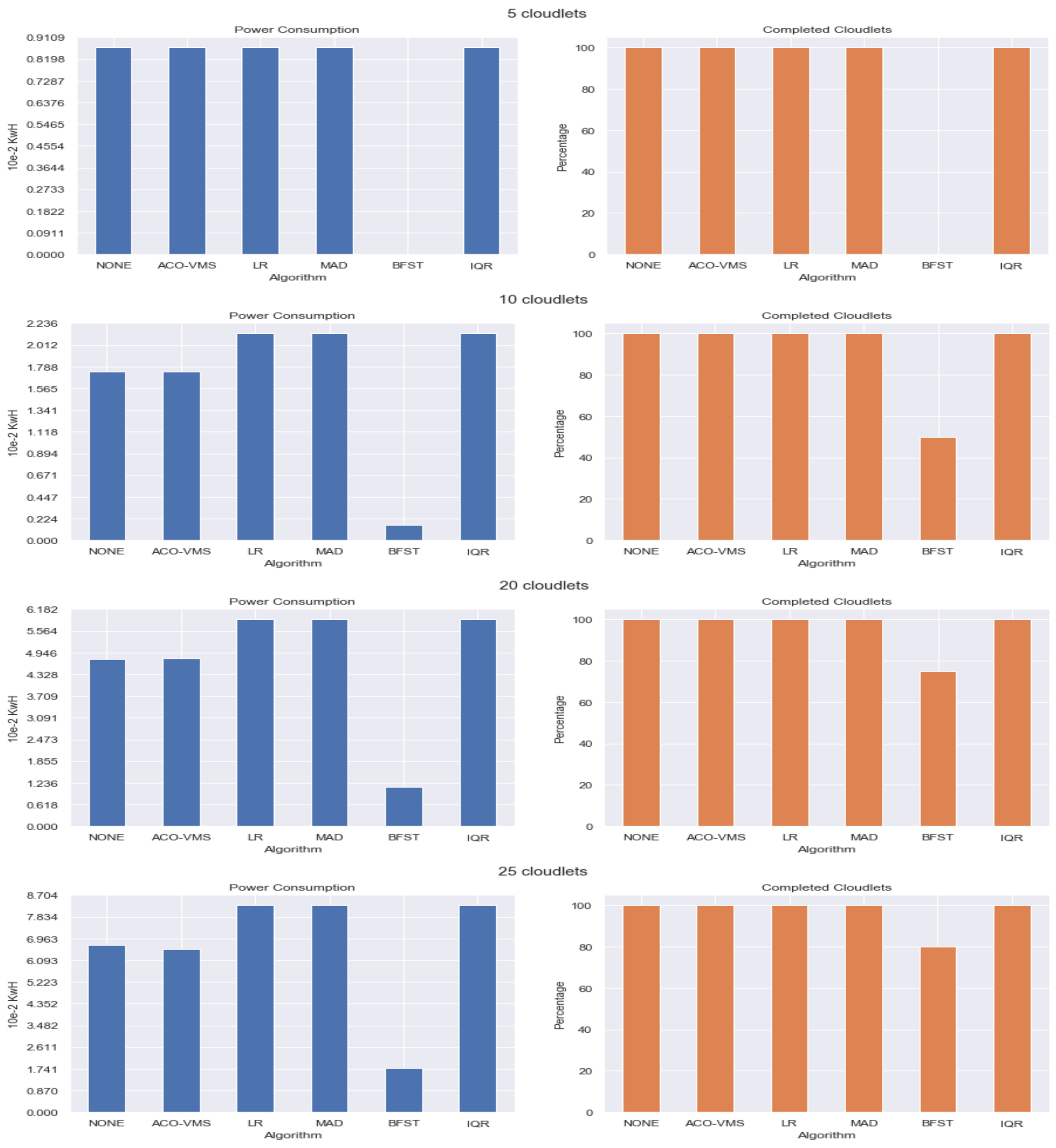


Figure 4.1 Comparison of Power consumption

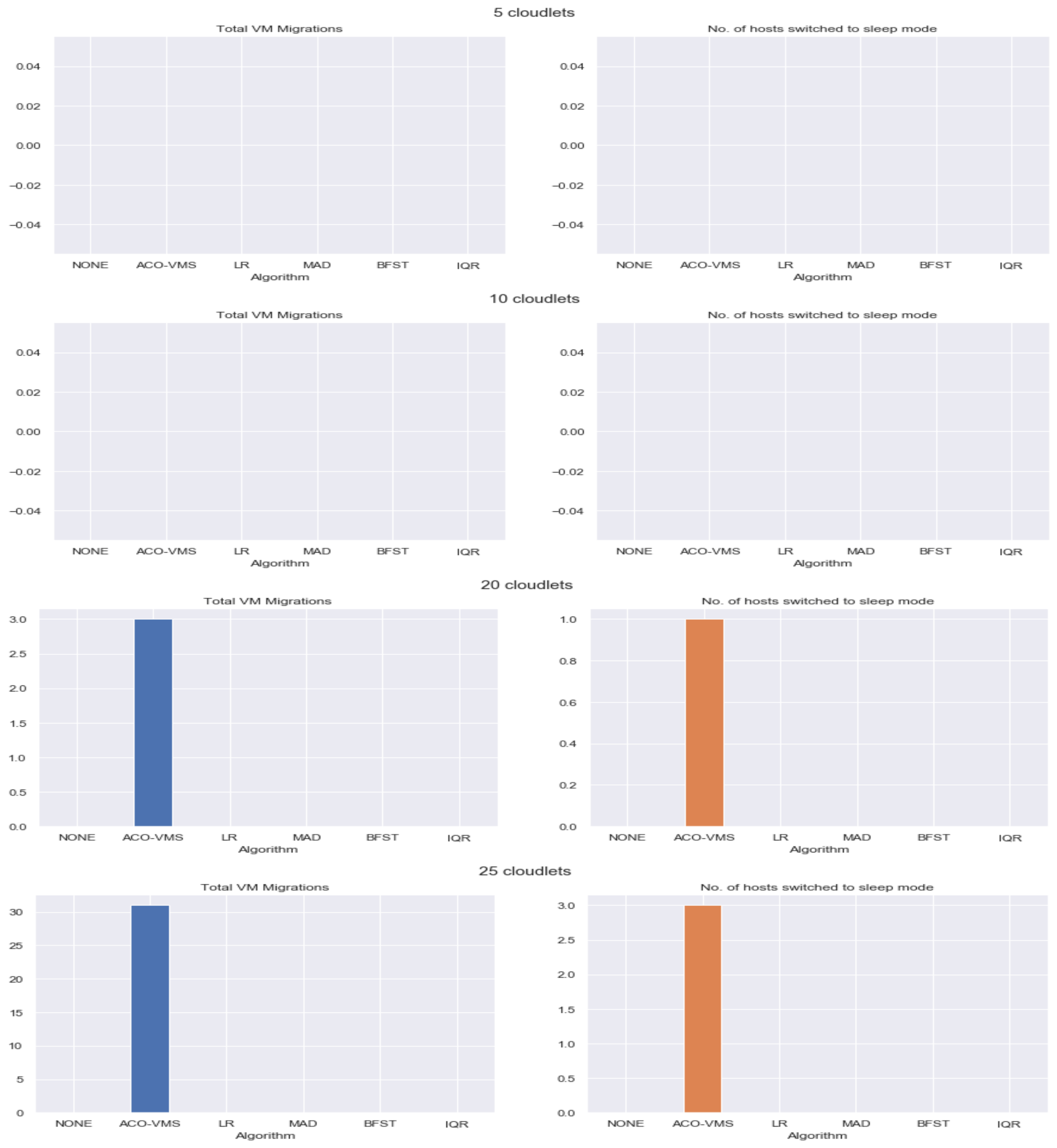


Figure 4.2 Comparison of number of VM Migrations

From the Fig. 4.2, which compared the number of VM migrations, our algorithm was the only one who was able to migrate at all, and that too with a larger number of cloudlets. With the number of cloudlets and lack of free hosts, other algorithms couldn't migrate any VM at all. However, our algorithm managed to find a good migration plan in some cases, which even had some hosts switching to sleep mode.

Chapter 5

Conclusion

In this report we presented a SLA-aware metaheuristic algorithm for dynamic VM consolidation. Since the problem of VM consolidation is NP-Hard, we used Ant colony Optimisation to get an efficient and near optimal migration plan. When compared to other VM migration algorithms, ACS-VMC also minimized SLA violation and number of migrations. Hence this approach can be used in real datacenters to promote Green cloud computing.

For the future work, we will further improve the complexity of our algorithm so that it can be used for larger workloads. Being inspired from the approach used by DeepMind [18], we plan to use the power of machine learning which could predict the approximate load on the datacenter in the near future and accordingly switch off some sleeping hosts to save more energy.

References

1. En.wikipedia.org. (2019). Cloud computing. [online]
Available at: https://en.wikipedia.org/wiki/Cloud_computing [Accessed 3 May 2019].

2. A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, A taxonomy and survey of energy-efficient data centers and cloud computing systems, Adv. Comput., vol. 82, pp. 47111, 2011.

3. Azure.microsoft.com. (2019). What is cloud computing? A beginners guide — Microsoft Azure. [online] Available at: https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/ [Accessed 3 May 2019].

4. CloudSim Plus. (2019). CloudSim Plus. [online] Available at: http://cloudsimplus.org/ [Accessed 3 May 2019].

5. W3schools.com. (2019). Introduction to Java. [online]
Available at: https://www.w3schools.com/java/java_intro.asp [Accessed 3 May 2019].

6. Monmarch Nicolas, Guinand Frdric and Siarry Patrick (2010). Artificial Ants. Wiley-ISTE. ISBN 978-1-84821-194-0.

7. M. Dorigo, Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italy, 1992.

8. Fahimeh Farahnakian, Adnan Ashraf, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Ivan Porres, and Hannu Tenhunen "Using Ant Colony System to Consolidate VMs for Green Cloud Computing" IEEE TRANSACTIONS ON SERVICES COMPUTING, Vol. 8, No. 2

9. C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, Live migration of virtual machines, in Proc. 2nd Conf. Symp. Netw. Syst. Design Implementation, 2005, vol. 2, pp. 273286.

10. D. G. Feitelson and B. Nitzberg, "Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860". In Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph (Eds.), Springer-Verlag, 1995, Lect. Notes Comput. Sci. vol. 949, pp. 337-360.

11. Cs.huji.ac.il. (2019). Parallel Workloads Archive: Standard Workload Format. [online] Available at: http://www.cs.huji.ac.il/labs/parallel/workload/swf.html [Accessed 3 May 2019].

12. Spec.org. (2019). SPECpower_ssj 2008. [online]
Available at: http://www.spec.org/power_ssj2008/ [Accessed 3 May 2019].

13. A. Beloglazov and R. Buyya, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centres, Concurrency Comput.: Prac. Exp., vol. 24, no. 13, pp. 13971420, 2012.

14. Greenpeace International. (2019). Make IT Green: Cloud computing and its contribution to climate change - Greenpeace International. [online] Available at: https://www.greenpeace.org/international/publication/7099/make-it-green-cloud-computing-and-its-contribution-to-climate-change/ [Accessed 3 May 2019].

15. A. Murtazaev and S. Oh, Sercon: Server consolidation algorithm using live migration of virtual machines for green computing, IETE Tech. Rev., vol. 28, no. 3, pp. 212231, 2011.

16. W. Cleveland, C. Loader Smoothing by local regression: principles and methods Stat Theory Comput Aspects Smoothing (1996)

17. W. Cleveland Robust locally weighted regression and smoothing scatterplots J Am Stat Assoc (1979), pp. 829-836

18. DeepMind. (2019). DeepMind AI Reduces Google Data Centre Cooling Bill by 40% — DeepMind. [online] Available at: https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/ [Accessed 3 May 2019].