ENERGY EFFICIENT VIRTUAL MACHINE CONSOLIDATION FOR CLOUD ENVIRONMENT

A Thesis

Submitted in fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

By

OSHIN SHARMA
Enrollment No. 146202

IN

COMPUTER SCIENCE & ENGINEERING

Under the Supervision of

Dr. HEMRAJ SAINI

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

AND

INFORMATION & TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT, SOLAN-173234, HIMACHAL PRADESH, INDIA

March, 2018

Dedicated To

My family

TABLE OF CONTENTS

DECLARATION……………………………………………………………….. V

SUPERVISOR'S CERTIFICATE…………………………………………… VI

ACKNOWLEDGEMENT…………………………………………………… VII

ABSTRACT……………………………………………………………….. IX

LIST OF FIGURES……………………………………………………….. XI

LIST OF TABLES…………………………………………………………… XIV

CHAPTER 1 1

INTRODUCTION 1

1.1 RESEARCH PROBLEM AND OBJECTIVES 5

1.2 CONTRIBUTIONS 6

1.3 THESIS ORGANIZATION 8

CHAPTER 2 10

A TAXONOMY ON VM CONSOLIDATION FOR ENERGY EFFICIENT CLOUD COMPUTING 10

2.1 INTRODUCTION 10

2.2 ENERGY EFFICIENT CLOUD COMPUTING 12

2.2.1 Power and Energy Modeling in Cloud Environment 12

2.2.1.1 Techniques for the measurement of Power consumption 13

2.2.1.2 Modeling of power consumption 13

2.2.1.3 Problems related to Energy and Power consumption 13

2.2.1.4 Modeling of power consumption for VM migration 14

2.2.2 Power Saving Techniques for Cloud Environment 15

2.3 STATE OF ART FOR ENERGY EFFICIENT VM CONSOLIDATION 16

2.3.1 Virtualization- backbone of cloud computing 16

2.3.2 Virtual Machine Consolidation 18

2.3.2.1 Detection of Hot-spots and excess capacity servers 20

2.3.2.2 Allocation policies 21

2.3.2.3 Virtual Machine migration 32

2.4 RESEARCH AND CHALLENGES FOR ENERGY EFFICIENT VM CONSOLIDATION 35

CHAPTER 3 37

MEDIAN BASED THRESHOLD (MEDTH) AND ANALYTIC HIERARCHY PROCESS (AHP) FOR HOST AND VM SELECTION 37

3.1 INTRODUCTION....37
3.2 RELATED WORK....38
3.3 PROPOSED SYSTEM MODEL....39
3.3.1 Data center's power model....40
3.3.2 Energy model....41
3.3.3 Performance Metrics....41
3.4 PROPOSED VM CONSOLIDATION METHOD....42
3.5 PROPOSED APPROACH FOR DETECTION OF HOT SPOTS AND EXCESS CAPACITY SERVERS....44
3.5.1 Median based threshold approach for finding over utilized and underutilized host machines (MEDTH)....44
3.5.2 Results and discussions....45
3.6 PROPOSED APPROACH FOR VM SELECTION USING AHP-VM METHOD....47
3.6.1 VM selection policy using RAM i.e. memory occupied by VM, CPU utilization and migration time taken by VM.....48
3.6.2 Maximum CPU utilization and Memory (RAM) occupancy....49
3.6.3 Minimum time for migration....49
3.6.4 AHP-VM Analytic hierarchy process for VM selection....49
3.6.5 Results and Discussions....51
3.6.5.1 Performance evaluation....52
3.7 SUMMARY....58
CHAPTER 4....60
SLA AND PERFORMANCE EFFICIENT HEURISTICS FOR VM PLACEMENT....60
4.1 INTRODUCTION....60
4.2 RELATED WORK....61
4.3 PROPOSED SOLUTION FOR VM PLACEMENT....62
4.3.1 VM placement as a bin packing....62
4.3.1.1 Proposed Heuristics for VM Placement ARBF- Available Resource Best Fit....63
4.4 RESULTS AND DISCUSSION....66
4.4.1 Data center architecture and performance metrics....66
4.4.2 Results and discussions....68
4.4.3 Statistical analysis....72
4.5 SUMMARY....73
CHAPTER 5....75

META HEAURISTICS APPROACH FOR VM PLACEMENT 75
5.1 INTRODUCTION 75
5.2 RELATED WORK 76
5.3 PROPOSED DECISION-MAKING MODEL FOR VM PLACEMENT 77
    5.3.1 Modeling of GA-VMP (Genetic Algorithm Virtual Machine Placement) 77
    5.3.2 Performance evaluation and results 80
5.4 NSGA-VMP (NON-DOMINATED SORTING GENETIC ALGORITHM FOR VIRTUAL MACHINE PLACEMENT) 84
    5.4.1 Problem formulation for NSGA-VMP 84
    5.4.2 Description of NSGA 87
        5.4.2.1 Creation of Initial population 88
        5.4.2.2 Infeasible solutions 89
        5.4.2.3 Selection operation 89
        5.4.2.4 Genetic operators 89
    5.4.3 VM placement process using NSGA 91
5.5 EXPERIMENTAL SETUP & RESULTS 92
5.6 SUMMARY 95
CHAPTER 6 97
BPGA: A NOVEL APPROACH FOR ENERGY EFFICIENT VM PLACEMENT 97
6.1 INTRODUCTION 97
6.2 RELATED WORK 98
6.3 SYSTEM MODEL FOR ENERGY AND POWER CONSUMPTION 99
6.4 VM PLACEMENT AS A MULTI-OBJECTIVE OPTIMIZATION 101
    6.4.1 Problem formulation 101
    6.4.2 VM Placement optimization 102
6.5 PROPOSED BPGA-VMP MODEL FOR VM PLACEMENT 104
6.6 PERFORMANCE EVALUATION 112
    6.6.1 Experimental Set up 113
    6.6.2 Results and Discussion 114
6.7 SUMMARY 117
CHAPTER 7 119
CONCLUSION AND FUTURE DIRECTIONS 119
7.1 CONCLUSION 119
7.2 FUTURE RESEARCH DIRECTION 120

APPENDIX 123
REFERENCES 151
LIST OF PUBLICATIONS 163

DECLARATION BY THE SCHOLAR

I hereby declare that the work reported in the Ph.D. thesis entitled **“Energy Efficient Virtual machine consolidation for data centers in cloud environment”** submitted at **Jaypee University of Information Technology, Waknaghat, India,** is an authentic record of my work carried out under the supervision of **Dr. Hemraj Saini**. I have not submitted this work elsewhere for any other degree or diploma. I am fully responsible for the contents of my Ph.D. Thesis.

(Signature of the Scholar)
(Oshin Sharma)
Department of Computer Science and Engineering
Jaypee University of Information Technology, Waknaghat, India
Date ( )

SUPERVISOR’S CERTIFICATE

This is to certify that the work reported in the Ph.D. thesis entitled “**Energy Efficient Virtual Machine Consolidation for data centers in Cloud Environment**”, submitted by **Oshin Sharma** at **Jaypee University of Information Technology, Waknaghat**, **India,** is a bonafide record of his / her original work carried out under my supervision. This work has not been submitted elsewhere for any other degree or diploma.

(Signature of Supervisor)
(Dr. Hemraj Saini)
Department of computer Science and Engineering
Jaypee University of Information and Technology
Waknaghat, India (173234)
Date (   )

ACKNOWLEDGEMENT

I am glad to have the opportunity of thanking people, whose support made my journey possible.
I would first like to thank Dr. Hemraj Saini, my supervisor for giving me this opportunity to conduct my Ph.D. research work under his supervision at the Department of Computer Science and Information Technology at Jaypee University of Information Technology, Waknaghat, Solan, Himachal Pradesh. He has constantly encouraged and supported me throughout my stay in JUIT. His vision towards science and technology has motivated me to do high-quality work. It is because of his active support, I could pursue research of my own interest and able to conclude my work in time. I have got many useful pieces of advice about research, career and life from him, which will be more valuable in my future professional career. Because of his availability, I never felt lack of guidance. It has been a wonderful experience to work under his supervision.

Besides my advisor, I would like to thank Professors Satya Prakash Ghrera for helping me in various academic issues. I would like to mention special thanks to Professors Vivek Sehgal for teaching wonderful courses on Internet of Things architecture and Design and for supporting me during my research.

I would like to thank Mr. Amit Shrivastva for helping me in various office related issues. I am also thankful to JUIT library staff that allowed me to use their facilities during my work. I am thankful to my lab-mate Geetanjali Rathee, Sukhnandan kaur, Vandna Mohindru, Gautam Kumar and Dinesh Kumar for helping me on and off the field.

I would like to mention special thanks to my friend Aastha Modgil for her valuable suggestions during my initial days at JUIT. Her support and encouragement was worth for me. I have many good memories with her.

I would like to thank my friends Reema Aswani and Ruhi Mahajan for making my stay enjoyable at JUIT. I will always remember those cherishable moments spend with them.

I would like to mention special thanks to my friend Nishtha Ahuja. I had lots of valuable discussions related to my work with her and she always motivated me during my tough times in JUIT. I have wonderful memories with her.

I would like to thank my cousin Neha Sharma. She has been always like an elder sister to me throughout my life. Her valuable suggestions regarding my career and in general, helped me a lot both professionally and personally. She was always there in JUIT for helping me in various academic and non-academic issues.

I would also like to convey my warm thanks to my friend Abhinav Sharma for listening to me always and providing me fun filled environment.

Finally, I would like to acknowledge my family for supporting me throughout my life. I am especially grateful to my mother Smt. Shashi Sharma and my father Mr. Navneet Sharma for teaching me the most valuable lesson of life that my job in life is to remain happy, to learn and to understand myself. This journey would not have been possible without their inspiration, love and prayers. I would like to thank my younger brother Onam Sharma for his love and care. He always remained by my side during my happy and hard times to motivate me.

Last but not the least; I would like to thank my husband Deepak Sharma who has been a constant source of encouragement during this journey. Thank you for always be there whenever I need you to just listen. I am truly thankful for having you in my life

Finally, I thank anyone else whose contribution I could have forgotten.

ABSTRACT

Cloud computing offers an on-demand computing model which has transformed the working environment of IT companies and thus, the trend of cloud computing has increased with the passage of time. Out of many different services of the cloud environment, IaaS (Infrastructure as a service) is one of the leading services which has given the birth to pay per use model and on the basis of this model, cloud service providers offer these services to the users. To satisfy the requirements of cloud users, it requires a large number of physical devices and, it requires a massive amount of electricity to power and cools down the several electronic components present inside these devices. It has been observed that the electricity consumption of data centers is 1.1% to 1.5% of the total electricity consumed all around the world and it is rising by 12% every year. This high consumption of electricity results in the emissions of high carbon dioxide and therefore, it is very necessary to reduce the consumption of energy for the betterment of environmental sustainability. Energy consumption of data center can be reduced by using minimum number of resources and improvement of their utilization. Moreover, the consumption of energy can be minimized by deactivating and reactivating the physical hosts present inside data center.

This thesis presents a new model, algorithms and heuristics for all different steps of dynamic virtual machine consolidation in Infrastructure as a service architecture that meets the requirements of SLA agreement and deals with energy-performance trade-off. The process of virtual machine (VM) consolidation has been selected for the improvement of energy consumption along with the use of live migration of virtual machines to minimize the number of active physical hosts. Thus, for the process of energy efficient VM consolidation, we have proposed a median based threshold approach (MEDTH) for the selection of underutilized and overutilized host present inside the data center. This proposed approach has been validated and showed better SLA performance than other policies by consuming less energy. Since the objective of energy consumption has not been achieved in this step, therefore, in the second step we proposed an Analytic Hierarchy Process for the selection of VMs for migration. It is a multi-criterion decision making approach for the selection. This algorithm selects the VMs on the basis of their memory occupancy, CPU utilization and migration time to achieve the objective of energy minimization. Results obtained from the evaluation of proposed algorithm showed better energy savings in comparison to conventional techniques.

We have formulated the problem of dynamic VM consolidation as a bin packing problem for the placement of selected VMs over most appropriate host i.e. the problem of VM Placement. For which we have proposed three different heuristics based on best-fit algorithms and they considered the present and historical behavior of the cloud resources. As, the main objective of this thesis is to minimize the energy consumed by data center but simultaneously, the excessive VM consolidation and migration may degrade the performance of data center. Therefore, our solution considers the overall performance of the system instead of only energy consumption to maintain a level of trust between cloud service providers and users. Furthermore, we have also solved the problem of VM placement using nature inspired Genetic Algorithm (GA) and analyzed that how the use of GA brings the better results for VM placement than classical bin packing algorithms. The only disadvantage of these genetic algorithms is that they have more execution time for solving the problems because of their large search space. For the further improvements of energy savings by using the idea of genetic algorithms we have again framed the problem of VM placement as optimization of multi objectives using Non-Dominated Sorting Genetic Algorithm (NSGA) where, our overall objective is to minimize the energy and SLA performance along with the migration count for the improvement of data center's performance. Finally, to solve the problem of VM placement we have designed a BPGA (Back Propagation-Genetic Algorithm) model which made use of both NSGA and back propagation neural network (BPNN). This BPGA model has been validated and it shows better results for energy consumption as well as other performance parameters of data centers. The proposed algorithms, heuristics and models for dynamic VM consolidation has been evaluated using an open source software framework called CloudSim using PlanetLab dataset for different workload conditions. Our solutions are robust, flexible and generic. Moreover, the experimental result shows that the overall VM consolidation process using the proposed BPGA VM Placement provides 19.8%, 9.7% and 4.5% of energy savings in comparison to GA, ACO and NSGA VM placement techniques.

LIST OF FIGURES

1.1 Schematic view of VM consolidation 3

1.2 Data Center Framework 4

1.3 VM consolidation method. Utilization of host 3 is low and all the VMs of host 3 are migrated to host 4, and host 3 is switched to sleep mode. 5

2.1 Major Cause of Energy Wastage inside Data Center 11

2.2 Worldwide problems of data center with high energy and power consumption 14

2.3 Virtualization inside cloud data center 17

2.4 Process of VM Consolidation 19

2.5 VM migration from under-utilized and over utilized hosts 21

2.6 VM Placement using bin packing 24

2.7 Live migrations of VM 34

3.1 System model used for proposed work 38

3.2 Flow chart for VM consolidation process 42

3.3 (a) Percentage of SLATAH, (b) Percentage of SLAV and (c) Percentage of PDM for random workload 46

3.4 (a) Percentage of SLATAH, (b) Percentage of SLAV and (c) Percentage of PDM for real workload 47

3.5 (a) Energy consumed by all VM selection policies. (b) Migration count during MAD. (c) Level of SLA violation caused by several VM selection policies. (d) Overall ESV metric and (e) total time to Execute the VM consolidation by using all VM selection policies with MAD 54

3.6 (a) Energy consumed by all VM selection policies. (b) Migration count during IQR. (c) Level of SLA violation caused by several VM selection policies. (d) Overall ESV metric and (e) total time to Execute the VM consolidation by using all VM selection policies with IQR. 55

3.7 (a) Energy consumed by all VM selection policies. (b) Migration count during THR. (c) Level of SLA violation caused by several VM selection policies. (d) Overall ESV metric and (e) total time to Execute the VM consolidation by using all VM selection policies with THR 56

3.8 (a) Energy consumed by all VM selection policies. (b) Migration count during LR. (c) Level of SLA violation caused by several VM selection policies. (d) Overall ESV metric and (e) total time to Execute the VM consolidation by using all VM selection policies with LR 57

4.1 Available Resources Best Fit (ARBF) model for VM Placement 63

4.2 SLA violations occur using several VM Placement policies 68

4.3 Migration count using several policies VM placement policies 69

4.4 Performance degradation during migration using different policies 69

4.5 Energy consumed by data center during VM placement 70

4.6 Execution time taken by different policies during VM placement 70

4.7 Results of ESM for different policies of VM placement 71

5.1 Decision-making model for VM Placement 77

5.2 Representation of initial population using Tree structure 79

5.3 Energy consumption for real environment 82

5.4 SLAV for real environment 82

5.5 Flow chart for NSGA 87

5.6 Initial solution representations 87

5.7 Example of cross over operation 89

5.8 Example of Mutation operation 89

5.9 Energy consumption using GA, ACO and NSGA VM Placement 93

5.10 SLA Violations during GA, ACO and NSGA VM Placement 93

5.11 Migration Count during GA, ACO and NSGA VM Placement 94

5.12 ESM_c during GA, ACO and NSGA VM Placement 94

6.1 BPGA model for VM Placement 105

6.2 Number of host V/s number of VMs 113

6.3 Migration count V/s Number of VMs 114

6.4 Energy Consumption during different VM Placement Policies 115

6.5 Level of SLA Violation during different VM Placement Policies 115

6.6 (A) Comparative values for ESM metric taken by different VM Placement Policies. (B) Cost of data center using various VM Placement Techniques 116

LIST OF TABLES

3.1	Configuration of servers	39
3.2	Power consumption of servers	40
3.3	Number of VMs for real workload environment	44
3.4	Percentage of Performance metrics for random workload	45
3.5	Percentage of Performance metrics for real workload	45
3.6	Pseudo code for AHP VM selection method	50
3.7	Workload characteristics	50
3.8	Comparative results using several VM selection policies with MAD	52
3.9	Comparative results using several VM selection policies with IQR	53
3.10	Comparative results using several VM selection policies with THR	54
3.11	Comparative results using several VM selection policies with LR	56
4.1	Pseudo code for ARBF H1	64
4.2	Pseudo code for ARBF H2	65
4.3	Pseudo code for ARBF H3	66
4.4	Comparative results for different VM placement policies	67
4.5	Comparative results for improvement of proposed heuristics	71
4.6	Comparison of all scenarios using paired t –tests	72
4.7	Comparison of benchmark algorithms with proposed heuristics regarding ESM metric	72
5.1	Number of VMs used on four different dates	80
5.2	Energy consumption and SLAV for dataset of 9th April 2011)	81
5.3	Energy consumption and SLAV for dataset of (11th April 2011)	81
5.4	Energy consumption and SLAV for dataset of (12th April 2011)	81
5.5	Energy consumption and SLAV for dataset of (20th April 2011)	81
5.6	Pseudo-code for VM Placement using NSGA	90
5.7	Test cases used for performance evaluation	91
5.8	Performance of data center using GA, ACO and NSGA	91
5.9	ESM_c consumption of GA, ACO and NSGA	92
6.1	Algorithm for pass 1 of BPGA model	106
6.2	Algorithm for Pass 2 of BPGA model	108

6.3	Simulate function for pass 2	109
6.4	Instances of VMs taken from Amazon EC2	112

CHAPTER 1

INTRODUCTION

Cloud computing is an on-demand model for the provisioning of resources that are provided by service providers on the basis of pay per usage [1]. Cloud computing has become very important in both academic, IT sector as well as in our day today lives due to its numerous characteristics like no upfront cost, on demand service, reliability, elasticity, and ease of access. various service providers such as Microsoft, Amazon, Google, IBM and many others provide various services for hardware, applications, platforms, and software and they are referred as Infrastructure as a service (IAAS), Platform as a service (PAAS), Software as a service (SAAS). The cloud service users can use these services from any location irrespective to any geographical, device, time restriction. Thus, this model cab be viewed as a "cloud" from where everyone can use its services or applications depending upon their requirements.

The different definition of cloud computing has been given by different associations. The National Institute of Standards and Technology (NIST) [2] defines the term cloud computing as: "a pay per use model for enabling available, on-demand network access to a shared pool of computing resources such as servers, applications, storage, networks" that can be rapidly provisioned.

University of California Berkeley [3] defined the cloud computing as: "(a) it is the illusion of infinite computing resources; (b) it has the ability to pay for what we have used (on demand); (c) the exclusion of an up-front promises by cloud users. The increasing trend of cloud computing has led to the deployment of a large number or data centers around the world [4] and thus, it is capable of supporting various computing services such as storage, servers, networks and applications for both e-sciences, e-business and much more over the network. This new paradigm of cloud computing is a big pool of easily accessible and readily usable virtualized resources such as platform, hardware and services (Example: CPU, memory, Java, .Net, email etc.). Data centers are the fundamental component of this new paradigm. These data centers are the collection of several electronic devices such as: servers, racks and therefore, they data centers are big source of energy consumption. Data centers are one of the world's main consumer of power and energy [5]. It has been analyzed that the cost of energy consumption will be increased by 50% after few years where as in last ten years the power consumption of servers has also increased by a factor of 10 [6], [7]. The energy consumption of these data centers is not only because of several computing devices [8], but also because of inefficient usage of cloud resources. Thus, there should be some green cloud computing solutions for better environment condition and to make data centers more efficient. Instead of maintaining the hardware efficiencies, the energy consumption can be lessened down by efficient utilization of resources.

Two most important techniques that are used for the improvement of resource utilization and to solve the problem of resource over provisioning are: 1) virtualization and 2) VM-consolidation. Virtualization is a technique that splits the single PM into several VMs due to which multiple physical resources can be used as logical or virtual resources [9]. Rest of the significance of virtualization are: improvement in resource utilization, server management and minimization of the cost of data center's infrastructure. Energy and powers of servers are wasted when they are not utilized and thus, the consumption of idle power becomes the reason for inefficiency of data centers. Consolidation of VMs and servers is a beneficial method for the improvement of power as well as energy consumption of data center. VMs are consolidated over PMs in order to avoid the usage of extra PMs. Moreover, for the reduction of active physical machines, the idle nodes should be turn off or kept in to low power or hibernate state or [10]. This method helps to decrease the idle power consumption. VM or server consolidation involves a key enabler technology known as VM Migration [11] because of which VMs are migrated from one to another PM. This technology also helps to improve the performance of systems by maintaining or balancing the load of the system by performing the migrations of VMs from overloaded or under loaded to other servers. VM Migration can be performed as: if a host H1 is under loaded i.e. it's host usage is low, then all the VMs from this host will be migrated to another host machine to avoid the unnecessary usage of H1, similarly if a host H1 is overloaded i.e. it's host usage is high, then some of VMs can be migrated to another host to reduce the load of H1. Figure 1 shows the schematic diagram of VM consolidation using virtualization. Where, we have taken 8 physical machines whose CPU utilization varies from 15% to 50% and thus after VM consolidation 3 physical machines are used and rest of others are in hibernate mode to minimize the energy consumption.

Moreover, this commercial success of cloud computing technology provides better QoS- Quality of Services that are documented in the Service level agreement (SLA) between cloud service providers and users [12]. Virtualization is the one of the best topic in cloud computing, which provides better QoS and deals with auto scaling, server/VM consolidation, energy conservation, load balancing and much more [13] - [16] because of its ability to run many OS- Operating Systems on single physical machine simultaneously by sharing the hardware resources.

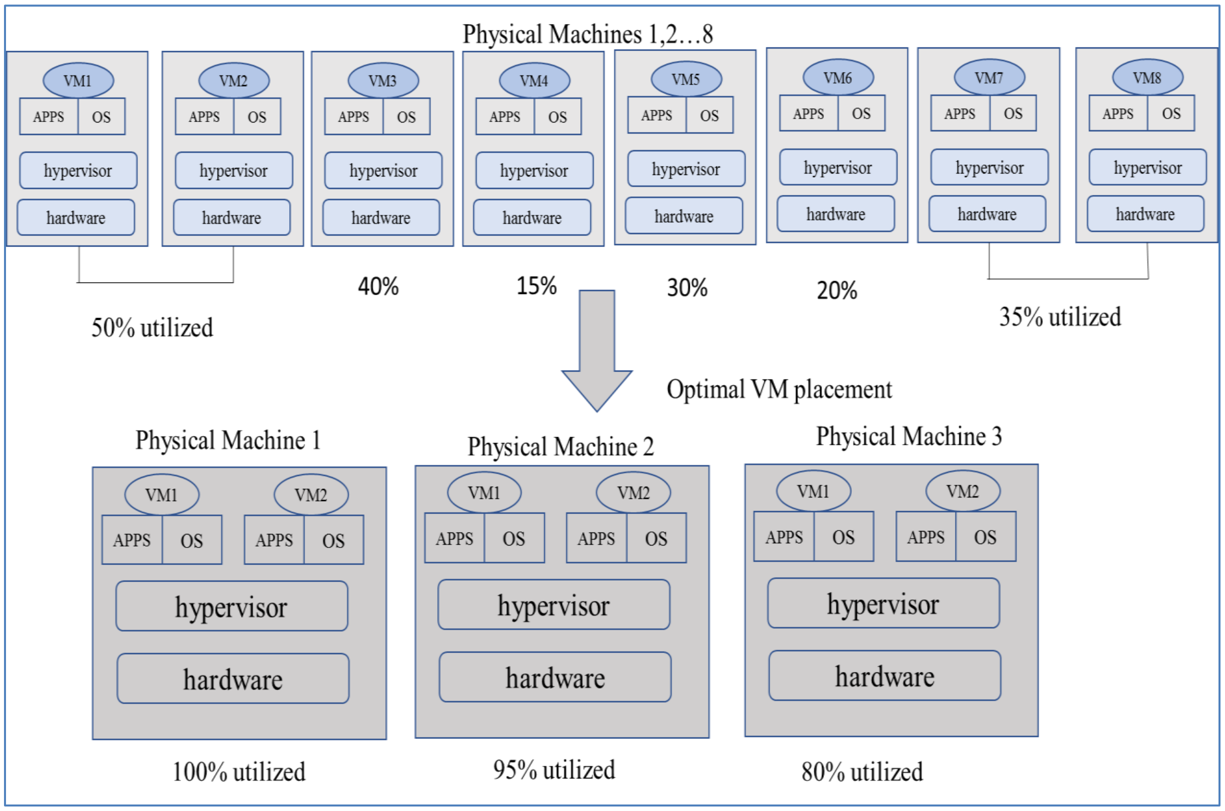

Physical Machines 1,2…8
VM1 APPS OS hypervisor hardware
VM2 APPS OS hypervisor hardware
VM3 APPS OS hypervisor hardware
VM4 APPS OS hypervisor hardware
VM5 APPS OS hypervisor hardware
VM6 APPS OS hypervisor hardware
VM7 APPS OS hypervisor hardware
VM8 APPS OS hypervisor hardware
50% utilized
40%
15%
30%
20%
35% utilized
Optimal VM placement
Physical Machine 1
Physical Machine 2
Physical Machine 3
VM1 VM2 APPS OS APPS OS hypervisor hardware
100% utilized
95% utilized
80% utilized

Figure 1.1: Schematic view of VM consolidation

The improper allocation of VMs on unsuitable host affects the interference of the different applications on same physical machines and this leads to the performance degradation with decreased level of Quality of services (QoS) for the applications. Therefore, certain issues should be resolved during VM consolidation process which can improve energy consumption, resource utilization and performance of data center. As, the excess of VM consolidation leads to the performance degradation; thus, it is the responsibility of a cloud provider to deal with tradeoff values of energy and performance. Energy consumption can be lessened down while meeting the SLA such that the QoS- quality of services should be maintained for the reliability of cloud environment. For the optimal VM consolidation, there is a need for data center framework which would be energy and SLA efficient. Figure 2 shows the framework of data center that we have used in our work.

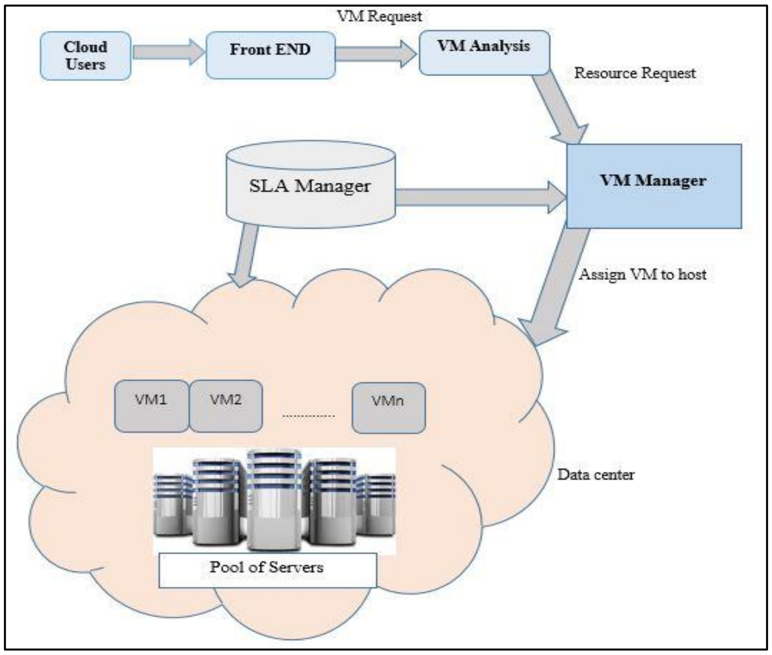


Figure 1.2: Data center Framework

This framework has two main components: a) SLA manager, b) VM analysis. The requested resources are determined by VM analyzer depending upon the number of requests made by cloud users which are calculated by front end servers and then VM analyzer send these requested resources for VM Placement and then it will check the current status of available resources on servers and allocate the VMs on physical hosts. It tries to allocate the VMs such that minimum number of servers are used to fulfil the resources that are requested by VMs. SLA manager checks the current status of SLA for each placement inside the data center. It will report back to VM placement analyzer if SLA violations increase after the placement so that appropriate placement of VMs on the host should be there which can provide trade-offs values for SLA violation and energy consumption. Thus, in this way, the objective of energy and performance efficient trade-offs can be fulfilled to provide better QoS. Solving the problem of VM consolidation in a cloud environment by dealing with energy performance trade-offs is a very challenging issue. This thesis presents the novel and complete solution for VM consolidation in IaaS (Infrastructure as a Service) cloud environment while dealing with QoS.

1.1 RESEARCH PROBLEM AND OBJECTIVES

This thesis deals with the research issues that occur during dynamic VM consolidation in IaaS cloud environment. Even if the problem of VM consolidation has been solved by many researchers, but still there is a lack of optimal algorithms to ensure energy efficient VM consolidation. The various problems are investigated to solve the problem of VM consolidation is as follow:

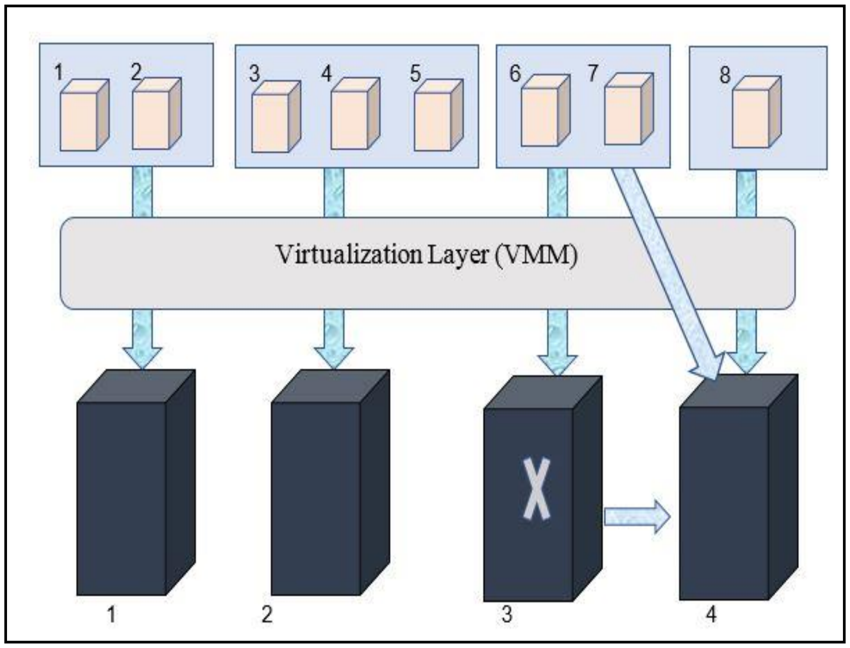


Figure 1.3: VM consolidation method. Utilization of host 3 is low and all the VMs of host 3 are migrated to host 4, and host 3 is switched to sleep mode.

When to migrate VMs?

The process of VM consolidation starts in two conditions: (a) hotspot, (b) excess spare condition. Hotspot means over utilized hosts from which VMs need to be migrated to some another host in order to minimize the performance degradation caused by overutilization of host, and second is to migrate the VMs from underutilized servers in order to reduce the energy consumption and improvement of resource utilization. Moreover, it is difficult to decide when to start the process of migration in both the cases to meet the QoS constraints.

Which VMs should migrate?

To select any one of the virtual machine or the complete set of VMs from the server is very important and difficult step in the process of VM consolidation. These selected VMs need to be allocated to some other hosts. Therefore, a beneficial decision must be made to provide suitable system configuration.

Where to migrate the selected VMs?

Another aspect that affects the quality of consolidation is VM placement, i.e. the placement of selected or new VMs to the new hosts or the most appropriate host. VM placement also affects the QoS while minimizing the energy consumption. Therefore, it is essential to find the most optimal VM placement method.

The process of VM consolidation is the combination of dynamic switching off/on the power states of host machines and it is necessary to determine which machine should be activated or deactivated to improve the resource utilization or to minimize the energy consumption. Thus, the first objective to provide the better and optimal solutions for above-mentioned problems and to explore the area of energy efficient cloud environment to understand the existing approaches and techniques. Second, is to develop energy efficient dynamic VM consolidation method. The third objective is to propose an algorithm for VM selection as well as VM placement during dynamic consolidation having workload independent conditions. Finally, the main aim of this thesis is the minimization of energy consumption and the cost of the data center while satisfying the parameters of QoS.

1.2 CONTRIBUTIONS

On the basis of the research problem and previously defined objectives, the contributions of this thesis are:

Literature Survey

Here, we have provided the most recent and efficient work related to the VM consolidation that is helpful for energy efficient computing or data centers.

Novel approaches for VM migration and selection

- We have solved the above mentioned three problems of VM consolidation step by step. First, we solved the problem of identifying the hotspot and excess spare condition by using the dynamic MEDTH algorithm, which is median based threshold algorithm for finding the host machine to start the migration.
- Second, we have formulated the problem of VM selection using multi-criteria decision-making algorithm i.e. AHP (Analytic Hierarchy Approach). It selects the VMs or set of VMs by taking the advantage of different alternatives using different criteria.
- Evaluation and performance analysis of proposed algorithms.

Heuristic Approach for VM Placement

- We have formulated the VM Placement problem as bin packing model.
- We have proposed three heuristics for the performance efficient VM Placement named as ARBFH (Available Resource Best Fit Heuristic) which is based on a best fit algorithm.
- Evaluation and comparison of proposed heuristics and other heuristic solution in terms of ESV i.e. Energy consumption, SLA violation and number of migrations.

Meta Heuristics Approach for VM Placement

- We have implemented the method of evolutionary algorithms for solving the problem of VM Placement and provide the mapping of VMs over host machine using GA (Genetic Algorithms).
- Next, we have formulated the problem of VM placement as multi-objective optimization and thus, we solved this multi-objective problem using another improved version of GA i.e. NSGA (Non-dominated Sorting Genetic Algorithm) which provides the pare to or optimal set of solutions.

- We have proposed a new BPGA model (Back Propagation neural network along with Genetic Algorithm) for energy efficient VM Placement. This model works according to both NSGA and BPNN i.e. Back Propagation Neural Network training algorithm.

Software implementation of VM consolidation

- Implementation of several algorithms proposed in every step of VM consolidation is conducted using an open source cloud computing platform i.e. CloudSim.
- This implementation uses the real-world workload traces collected from Planet Lab dataset [17]. Thus, the results obtained from simulations show that proposed algorithms for the dynamic VM consolidation provide a tradeoff values for energy and SLAV and minimizes the energy consumption of data center.

1.3 THESIS ORGANIZATION

This thesis is arranged into seven different chapters. Chapter 2 provides the existing literature of VM consolidation and energy efficient data centers. Also, we discussed about challenges, related issues and problems. This chapter is derived from [18]

- Oshin Sharma, Hemraj Saini, "State of Art for Energy Efficient Resource Allocation for Green Cloud Data centers", in International Journal of Control Theory and Application. Vol.9, No.11, pp. 5271-5280, 2016.

Chapter 3 presents the proposed algorithms for dynamic VM consolidation along with their competitive analysis. Here, we discuss the solution for the first and second step of VM consolidation by considering the level of SLA Violation and performance degradation as its two main objectives. This chapter is derived from [19], [20]

- Oshin Sharma, Hemraj Saini, "VM Consolidation for Cloud Data Centers using Median Based Threshold Approach", in the proceedings of 12th International Multi-Conference on Information Processing (IMCIP-2016). Vol. 89, pp.27-33.
- Oshin Sharma, Hemraj Saini, "Energy Efficient Virtual Machine Consolidation for Cloud Data Centers Using Analytic Hierarchy Process", In International Journal of Advanced Intelligence and Paradigms. (In Press).

Chapter 4 proposes novel heuristics based upon the bin packing algorithms for VM Placement. Here we have defined three different heuristics using current and previous used resources of data centers. Simulations have been conducted to show their performance and their ability to reduce the energy consumption of data center. This chapter is derived from [21].

- Oshin Sharma, Hemraj Saini, “SLA and Performance Efficient Heuristics for Virtual Machine Placement inside Cloud Data Centers”, In International Journal of Grid and High-Performance Computing. Vol. 9, No.3, 2017.

Chapter 5 compares the traditional bin packing algorithm with evolutionary approach for VM placement and discusses their performance evaluation. Also, this chapter presents a decision-making system and a novel VM placement algorithm using meta-heuristics approaches. Experiments are conducted to show the efficiency of our solution. This chapter is derived from [22], [23].

- Oshin Sharma, Hemraj Saini, “Performance Evaluation of VM Placement Using Classical Bin Packing and Genetic Algorithm for Cloud Environment”, in International Journal of Business Data and Communication Network. Vol. 13, No.1, pp.45-57, 2016.
- Oshin Sharma, Hemraj Saini, “Energy & SLA Efficient Virtual machine placement in Cloud Environment using NSGA (Non-dominated Sorting Genetic Algorithm)”, in International Journal of Information Security and Privacy. 2017 (in review)

Chapter 6 proposes a new model for the VM Placement using nature inspire algorithm i.e. GA along with help of well-known training algorithm i.e. BPNN (Back Propagation Neural Network). Simulations are conducted to show that how this model minimizes the energy consumption as well as the cost of the data center. This chapter also describes the architecture and implementation of CloudSim, an open framework for the cloud environment. This chapter is derived from [24].

- Oshin Sharma, Hemraj Saini, “BPGA: A Novel Approach for Energy Efficient Virtual Machine Placement in Cloud Data Centers”, In Journal of Computing. (in review)

Chapter 7 draws the conclusion and summarizes the major discussions, findings, challenges and future research directions.

CHAPTER 2

A TAXONOMY ON VM CONSOLIDATION FOR ENERGY EFFICIENT CLOUD COMPUTING

This chapter provides the recent literature on energy efficient cloud computing and dynamic VM consolidation. Also, this chapter discusses the future research challenges for VM consolidation.

2.1 INTRODUCTION

The trend of IT companies has been transformed from traditional to new on demand service as well as provisioning of the resources from resource pool due to one of the popular technology known as cloud computing. An organization can either build their private cloud for the management of resources or they can outsource the resources from some public cloud in order to avoid the high investment for the infrastructure of private cloud.

The increasing trend of cloud computing leads to the establishment of datacenters all around the world which consists of thousands of computing devices such as servers, racks, switches and much more. As these devices consume vast amount of electricity and results in the carbon dioxide (CO_2) emission to the environment. Thus, datacenters became the big source of carbon dioxide emission. For the environment sustainability, it is necessary to find some solutions to minimize the consumption of power inside data center. This can be performed by minimizing the wastage of electricity and by improving the infrastructure of the data center. The management of resources inside data centers such as their allocation and utilization is also responsible for data center's inefficiency. Therefore, recent advancement in the resource management results significant improvement in the efficiency of data centers.

For the improvement of energy inefficient data centers, it is very essential to understand how the power is distributed among the various components present inside data center. According to the report of EPA on data center's energy [25], the power consumption of a server is 40% of the data center's power and 80% of the total IT load. Similarly, the survey from open compute project [26] reported that 91% of energy consumption with in data center is only due to its computing resources. Thus, the source of high emission of CO_2 is also due to inefficient usage of computing resources. Data provided in [27] shows that most of the times, server's works for more than 15-50% of their overall capacity which results over provisioning of the resources and leads to the higher Total Cost of Acquisition (TCA) [28]. According to National Resource Defense Council's report [29], [30], mostly the data centers are unused and the underutilization of servers is also inefficient for energy aware data centers. Solution of this problem is the consolidation of servers, by which fewer number of hosts will run the same application with lesser power consumption and it will further reduce the overall energy. Along with this, during the low server utilization, idle servers consume 70% of the power. Therefore, these idle servers should be turn off to reduce the energy consumption. Another important reason for energy wastage inside the data center is the lack of standard metrics. So, there should be some energy efficient metrics for servers so that servers can be arranged according to their energy efficiency. Figure 2 shows the main cause of energy wastage inside the data center.

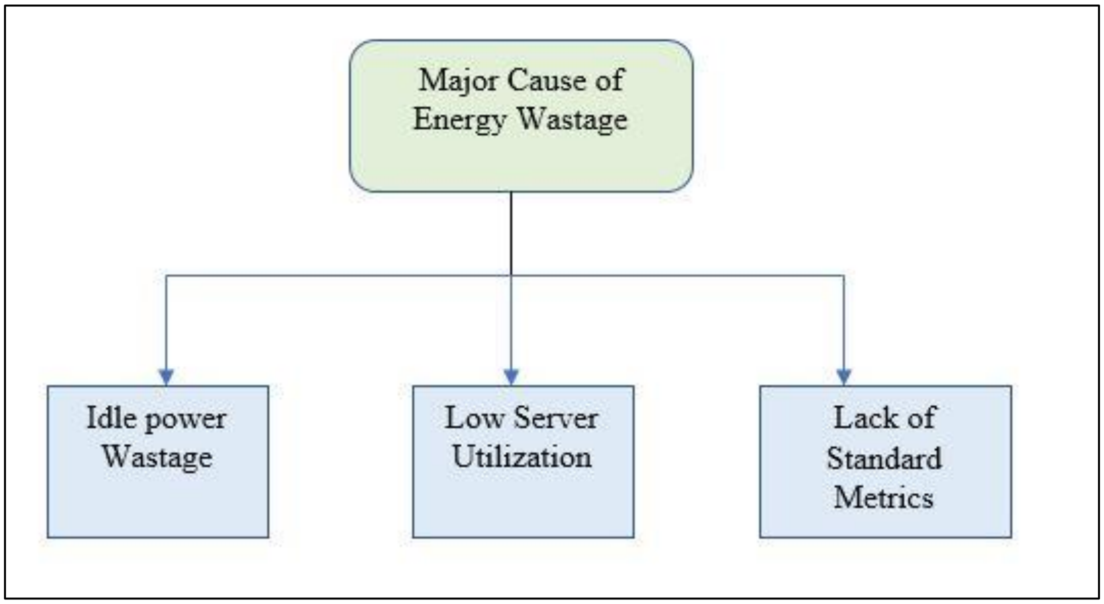


Figure 2.1: Major Cause of Energy Wastage inside Data Center

The concept of energy efficient cloud computing deals with the energy and power consumption at the hardware level, software level, operating system level and data center level. This chapter deals with the concept of energy efficient data center. This chapter provides the current research on VM consolidation for energy efficient cloud computing and discuss some research challenges within the research area.

2.2 ENERGY EFFICIENT CLOUD COMPUTING

As we discussed earlier, the minimization of energy and power consumption is the first objective for energy efficient cloud environment. The origin for the concept of energy efficient computing or we can say green computing is a program launched by U.S. Environmental Protection Agency [31] i.e. energy star. It was the volunteer program to identify the energy efficient products to minimize CO_2 emission. Monitors and computers were the first products they labeled. Later on, TCO certification program was developed by the employees of Swedish Confederation which includes the environmental requirements of the IT equipment's such as keyboards, computers, monitors, peripheral devices and mobile phones. Now a day, there are many industries that have their standard methods to minimize the carbon dioxide emission and consumption of energy with in data centers such as VMware, Intel, Microsoft, IBM, Dell, HP and many more.

2.2.1 Power and Energy Modeling in Cloud Environment

Before dealing with the measurement of power and energy consumption, it is very helpful to understand the relationship between them and their units of measurement. Power is the rate of the system while performing its work whereas, energy s the total amount of work done over a period of time. The measurement unit of power is watt (W) and for energy, it is Watt-hour (Wh). Power and energy are defined in equations, 2.1, 2.2. Where P is the power, W is the work done during the period of time T and E is the energy consumed.

$$P = \frac{W}{T} \qquad 2.1$$

$$E = PT \qquad 2.2$$

As, the energy and power consumption both are directly related to each other, but still the minimization of power does not always minimize the energy consumption.

2.2.1.1 Techniques for the measurement of Power consumption

From the data of Intel Labs [32], the main part of power consumption inside servers are CPU and after that memory. The best way to find the accurate energy consumption of servers is by directly measuring it. It can be possible only by installing extra hardware in the hosts or an intelligent monitoring system inside the data center. GOC (Green Open Cloud) [33] is the best example of energy monitoring system which has sensors to compute the electricity consumed by cloud resources. It provides the dynamic measurement of energy consumption. But in the case of power consumption of virtual machines, it cannot be calculated by any such kind of sensors. Some solutions have been proposed in [34], [35] by including power monitoring adapter between hypervisor and server driver modules, but it was also not able to provide power consumption per VM and provide the power consumption of virtualization layer.

2.2.1.2 Modeling of power consumption

To find the power consumption of the system, it is required to design the model for dynamic power consumption. The modern computer servers have built in power monitoring capabilities and by utilizing these capabilities; the power consumption can be calculated. From the literature [36] it has been concluded that there is a linear relationship between the CPU utilization of the machine and power consumption. From [37], [38], [39], [40] power models of the servers based on the simple utilization and assumed that CPU utilization is only responsible for power consumption shown in (2.3), where P is the total power consumption, P_{idle} is the idle power consumption of server, P_{busy} is the power consumption of fully utilized host or server and U is the CPU utilization.

$$P = P_{idle} + U * (P_{busy} - P_{idle}) \tag{2.3}$$

Later on, more complex power models came into existence which considered parameters like memory access rate, network access rate and hard disk access rate. Their examples are provided in [41], [42], [43], [44].

2.2.1.3 Problems related to Energy and Power consumption

Both the terms energy and power are interchangeable, but there exists a difference between them. If we consider and focus both of them, we can get better efficiencies and savings. In simple terms, energy efficiency deals with the total amount of electricity consumed by the system whereas power efficiency deals with the work done by the CPU for the amount of electricity consumed. In terms of data center efficiency, power consumption deals with the cost of infrastructure which is essential to maintain the energy consumption and system operation. The first major problem of high power consumption inside the data center is heat dissipation. More the electrical power consumed by the computing resources, more the power gets converted into heat which further required more power for the cooling systems. Moreover, the overheating of the components will reduce their lifetime and provide more error proneness of the components. Similarly, high emission of CO_2 or carbon footprints is the main problem of data centers which is caused by the high energy consumption and it contributes to the global warming. Above figure 2.2 shows the problems from high power and energy consumption inside the data center. According to the survey reports of [45], the increase in the trend of annual carbon emission (CO_2) was 42.8 million metric tons to 62.7 million metric tons. Thus, the reduction of carbon emission became an important problem and needs further advancement.

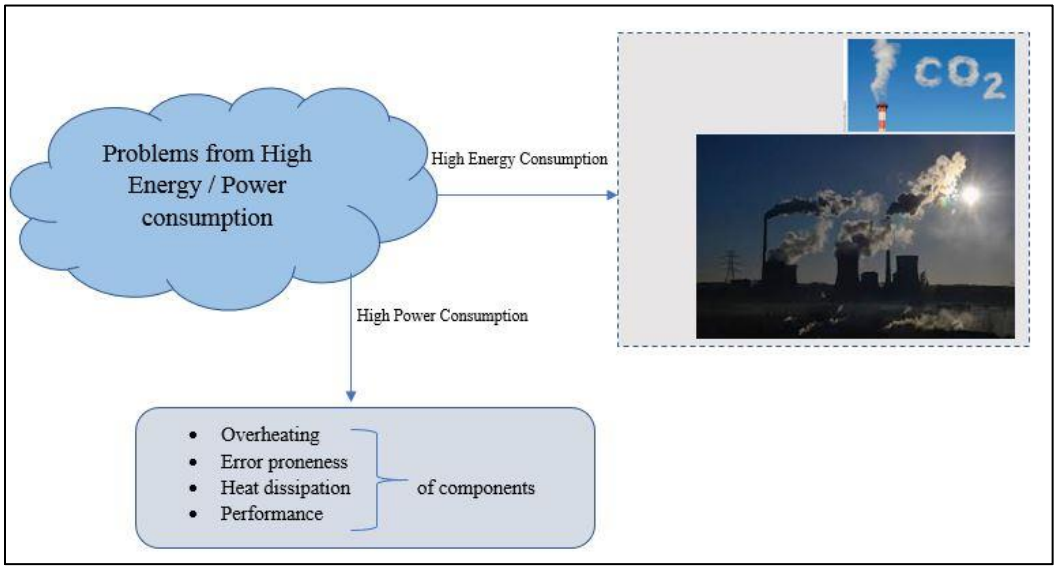


Figure 2.2: Worldwide problems of data center with high energy and power consumption

2.2.1.4 Modeling of power consumption for VM migration

To minimize the consumption of energy within the data centers, it is very important to estimate the power consumption by VM. Thus, the CPU utilization can also be used for the calculation of the power consumption of CPU by VM as similar to servers. In [43] author presents a VM Power monitoring method and Joule meter (a software for VM power estimation) on the basis of CPU utilization and Performance Monitoring Counters (PMC) [46]. The process of VM consolidation for energy efficient data centers also brings the power consumption which costs in terms of energy. Thus, the estimation of energy consumption for each VM migration became the key point for energy efficient VM consolidation. Studies from [47], [48], [49] investigate the model for the cost of energy during VM migration. They framed that the energy cost depends upon the available bandwidth and memory used by VM.

2.2.2 Power Saving Techniques for Cloud Environment

Three main powers saving techniques for energy efficient cloud environment are: VM Consolidation, powering down the servers and Dynamic Voltage Frequency Scaling (DVFS). Automatic switching off or powering down the idle servers which are not in use is a very interesting method to reduce the total energy consumed. Most of the times, many servers remain in the idle position inside data centers. Thus, the dynamic provisioning of the selection of these types of servers and put them to sleep mode became a very challenging task. Different approached for dynamically turn off and turn on the servers inside data centers have been proposed to minimize the energy consumption [50], [51], [52], [53], and [54]. Later on, the process of dynamic VM consolidation came into existence and became useful for the selection of servers that should be power down or power up. As the process of VM consolidation works in different steps, therefore, it is the key technique for the selection of most efficient servers. This process makes the use of fewer numbers of servers because its VM live migration technique simply migrates the VMs from one host to another and power down the underutilized servers. Details of the VM consolidations will be provided in the next section.

DVFS is also a tool that is used for the power management or to minimize the power consumption of servers. DVFS is an example of DPS (Dynamic Performance Scaling) which can be applied to the components of the computer which supports the dynamic adjustment of their performance. As the increase in frequency or voltage may increase the power consumption of system or vice a versa. Thus, the DVFS minimizes the number of instructions the processor executes to minimize the performance by which program take more time to execute [55]. The approach of DVFS can provide the energy savings but it is hardware dependent and therefore, for the idle server, the scope of power minimization will be very less. Mainly the technology of DVFS is used for the achievement of energy efficiency in multicore, multiprocessors and embedded systems. They are adopted for processors and more efficient computation-intensive VMs and are not suitable for input/output intensive VMs [56]. DVFS is hardly adopted for virtualized cloud systems. These reasons contribute to the use of fewer hosts and put the other idle hosts into sleep mode using VM consolidation which means deactivation of idle servers minimizes the energy and power consumption as well as improves the resource utilization.

2.3 STATE OF ART FOR ENERGY EFFICIENT VM CONSOLIDATION

The main focus of this thesis is to design the models for energy efficient VM consolidation for cloud data centers. A huge amount of research has been done in this particular area of research. To achieve this objective and to provide energy efficient solutions and to solve various problems, we have discussed existing state of art techniques and models. In this section, we provide the details of energy efficient VM consolidation.

2.3.1 Virtualization- backbone of cloud computing

Virtualization is the key technology of a cloud environment which makes the cloud resources available to cloud users by separating one physical machine into several virtual machines. With the help of virtualization, the cloud resources are available in the form of logical or virtual resources. Virtual machine monitor (VMM) is used during virtualization, which is also known as the hypervisor. The responsibility of this hypervisor is to virtualize the hardware of host machine into virtual resources due to which virtual machines can exclusively use them and maintain the isolation between the VMs. All the physical resources are virtualized and therefore, VMs containing their own operating systems can be executed on the physical machines [57].

The concept of the virtual machine was provided by popek [58] that it is an efficient copy of real machine which allows the multiplexing of the original physical machine. Nowadays many open source project and several companies offer some software packages that make the use of virtual computing. Figure 2.3 shows the process of virtualization thus; the use of virtualization is to reduce the amount of hardware used and to improve the resource utilization by creating multiple VMs over single hosts [59].

Improvement in the performance and ease of migration of VMs from one host to another using the concept of live migration are the few benefits of virtualization. The ability of run time migration of the VMs is known as dynamic VM consolidation which will be discussed in next subsection. There are three important solutions for virtualization technologies that support power management such as: Xen hypervisor, VMware solutions and Kernel based virtual machine (KVM).

VMware ESXi and VMware ESX Server are two virtualization solutions offered by VMware. It provides power management at host level via DVFS. Two services such as: VMware Distributed scheduler and VMware VMotion operates in combination with ESXi and ESX Server [60]. VMotion enables the live migration of VMs among physical machines whereas Distributed scheduler monitors the usage of the resources and maintain the balance among the VMs according to the current load. Moreover, distributed schedulers have Distributed Power Management (DPM) as a subsystem used for the reduction of power consumption of machines (servers) by dynamically switching off and on the extra spare servers [60].

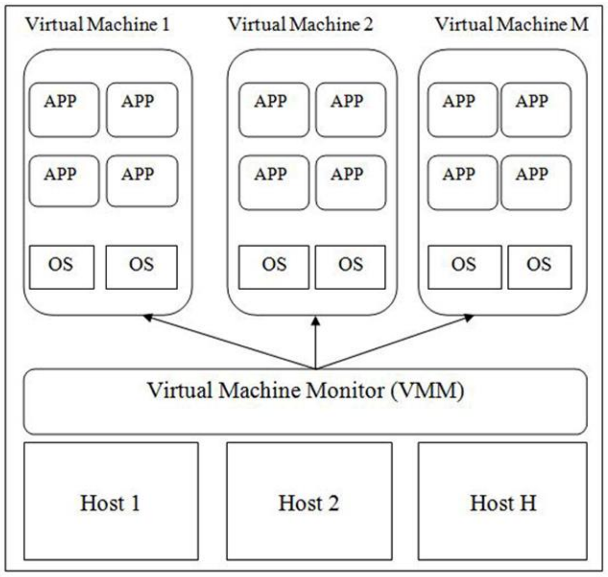


Figure 2.3: Virtualization inside cloud data center

Xen Hypervisor is open source technology for virtualization licensed under General Public License (GPL) and developed by Xen community. As similar to Linux power management, Xen also contains four commands for making changes in the power state of hardware. 1) Power save used to set the lowest clock frequency, 2) Userspace used to set the specific CPU frequency by the user, 3) On-demand used to choose the p-state according to the resources, 4) performance is used to set the highest available clock frequency. Xen enables the systems for its transformation from one state to another i.e. P-state to C-state (CPU active state to CPU sleep state). Apart from this, Xen also supports live and offline VM migration. For VM migration, it is very important that both the source and destination machines must be Xen running on them. Also, the destination host should have appropriate resources that can accommodate VM. Similarly, the Kernel based Virtual Machine (KVM) is also an open source software model for virtualization implemented as Linux Kernel. In this model, the role of hypervisor is played by Linux and the complexity of hypervisor implementation will be reduced to some extent.

2.3.2 Virtual Machine Consolidation

VM Consolidation is the most efficient procedure for the reduction of power as well as energy consumption of data center. VMs are consolidated over PMs in order to reduce the usage of number of physical machines. Moreover, it would be beneficial to keep the idle machines into sleep and hibernate state or turn them off during consolidation. By doing this, we can decrease the idle consumption of power with in data center. Sometimes the excessive consolidation delivers poor QoS- Quality of Service and may violate the SLA- Service Level Agreements between the service provider and user. Thus, VM consolidation should maintain an optimal balance between energy consumption and QoS [61] and deal with energy performance trade-offs.

Migration technology is the backbone for server or VM consolidation [11] which performs the migrations of VMs from one host to another and improves the performance of systems as well as maintain the load of the system. The problem of dynamic VM consolidation can be processed into four different steps [10] as mentioned below:

1. Selection of the hot spots i.e. over utilized machines within the data center which needs to migrate of some of its VMs.
2. Selection of Virtual Machines from above selected hot spots for migration.

3. Selection of excess capacity servers i.e. under-utilized host machine for the migration of all its VMs.
4. Design of the new placement policy for the selected VMs from host spots and excess capacity servers to some new host.

Thus, the overall process of VM consolidation is shown in figure 2.4. Here, the process of VM consolidation starts by finding any over utilized or under-utilized host machine present inside data center and followed by two most important processes: VM migration and VM placement. All these steps of VM consolidation can efficiently manage the energy issues.

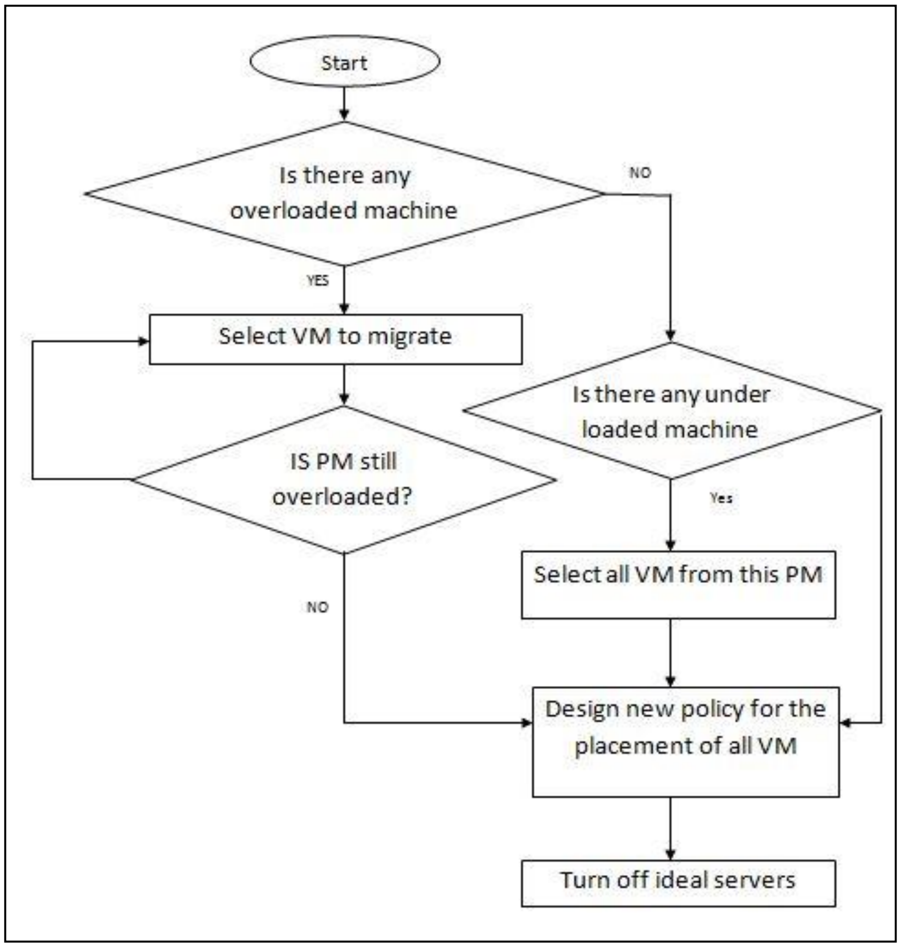

Start
Is there any overloaded machine
NO
YES
Select VM to migrate
Is there any under loaded machine
IS PM still overloaded?
Yes
Select all VM from this PM
NO
Design new policy for the placement of all VM
Turn off ideal servers

Figure 2.4: Process of VM Consolidation

There is a pool of physical machines present inside cloud environment where different applications are running over them. The problem of VM consolidation across these machines is associated with multidimensional vector packing problem and in our current work, we have considered CPU and memory utilization as two different dimensions. Suppose two virtual machines are working on the same physical machine, then the resource utilization of the physical machine is equal to the sum of the resources of these two virtual machines running on it. For example, X1% and Y1% are the percentage utilization of CPU and memory for VM1 and X2% and Y2% are for VM2. Thus, the utilization of physical node accommodating these two VMs are the sum of the vectors: (X1% + X2%, Y1% +Y2%). The detail and a recent survey of every step involved in the process will be discussed in next subsections.

2.3.2.1 Detection of Hot-spots and excess capacity servers

To start the process of VM consolidation, it is very important to detect the hot-spots and excess capacity servers inside the data centers. Hot spots are those servers which are over utilized. We can also say that we have to find the time for the migration of VMs from the host (Physical machine) based upon the utilization rate of the host. Such that if the utilization of the physical machine will be greater than the value of upper threshold and lower than the value of lower threshold then only it would be beneficial to start the migration. In this contrast author in [62] set 25% and 75% of utilization as the lower and upper threshold values for utilization. If the utilization of physical machine is less than a lower threshold value; all of the VMs will be migrated from the physical machine and then, it should be put into an idle mode in order to save energy. Similarly, if the utilization of host is greater than an upper threshold value; some of the VMs will be migrated from that host. For dynamic workload environment, this static method for setting upper and lower threshold is not suitable.

Several authors provided their own dynamic methods for determining upper and lower threshold. Anton et al. [10] presented four statistical methods for determining threshold values such as: LR (Local Regression), MAD (Median Absolute Deviation), IQR (Inter Quartile Range) and LRR (Local Robust Regression) these methods based on robust methods rather than classical methods, as they are more effective than classical methods [63]. Author in [64] proposed a robust estimator which is alternate to MAD and more efficient. Horri et al. in [6] used a novel technique VM-based Dynamic Threshold i.e. VDT for detecting under-utilized physical machines. In their method, they have used VMs on the host and CPU utilization of host for optimization along with hill claiming method. Similarly, Ehsan et al. [65] also proposed three policies for finding under-utilized host and they are: Migration Delay (MDL), Available Capacity (AC), TOPSIS available capacity, the number of VMs and Migration Delay (TACND) where TACND works on the principal of multi-criteria decision-making process. MDL is same as the MMT proposed by Anton et al. in [10] and AC considers resource capacity rather than resource utilization. We have also proposed median based auto-adjustment or dynamic method for determining the threshold which will be discussed in next chapter. Figure 2.5 shows the scenario where host 1 and host 4 are underutilized and over utilized respectively since 45% and 85% are lower and upper threshold values for utilization. Accordingly, the VMs are migrated to another host and host 1 and 4 will be turned off in order to save energy.

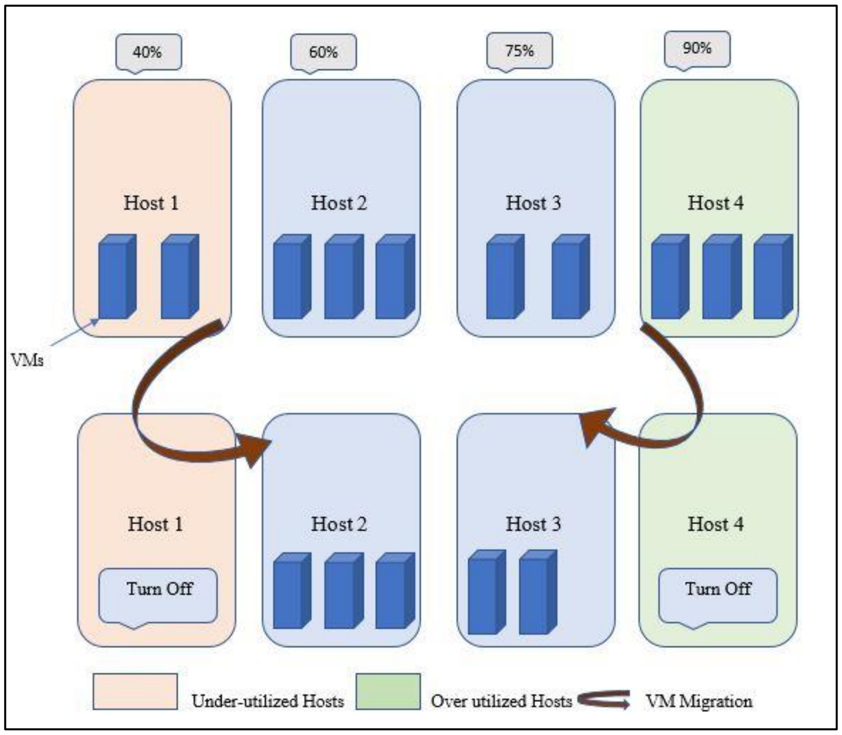


Figure 2.5: VM migration from under-utilized and over utilized hosts

2.3.2.2 Allocation policies

Selection of virtual machines for migration and their placement over appropriate hosts plays an important role to optimize the allocation. These two policies are the two different steps for VM consolidation process and are discussed as follows:

2.3.2.2.1 VM Selection

After the detection of over and under-utilized hosts, next task is to select the virtual machine for migration from these hosts. This process of VM selection works iteratively, the host needs to check again and again. If the host is still over or under-utilized; the VM selection policy again selects some more VMs for migration.

Anton et al. [10] proposed 3 different policies for the selection of VM migration and they are: Random choice policy (RC), minimum migration time policy (MMT) and Maximum correlation policy (MC). The names of these selection policies clarify their selection process. MMT selects those VM which requires minimum time to migrate VMs to another host or physical machine. Random choice policy randomly chooses the VM for migration and similarly, the last one MC policy migrate the VMs with highest degree of correlation of the CPU utilization with other VMs. Later on, in 2013 Wang et al. [66] also presented a method for VM selection from an over utilized host. Their selection strategy depends upon the CPU utilization of VMs known as double-threshold VM migration strategy. In [64] Hassan et al. presented the policy which considered both the CPU utilization as well as migration time of VMs when the virtual machines are migrating from one physical machine to another. This migration time can be considered as total memory used by VMs divided by the available bandwidth of that host. Sometimes VMs are selected depending upon their resource utilization and there are different types of resources such as CPU utilization, Memory, bandwidth and I/O. Due to different types of resources, it is not so easy to compare resource requirements of all the machines. Therefore, many functions are proposed in the literature for their comparison and one of them is “volume”. The volume function selects the VMs on the basis of the product of utilization of each resource individually [57]. In [67] another approach for selection of VMs has been proposed in which they try to select that VM which has more contribution towards the load change of their host. Their method performs three steps for selection: first is to evaluate the load of each VM and second is to sort the VM according to their load and finally to select the subset of VMs that are on the top of the list. On the basis of above-mentioned literature, we have also solved the problem of VM selection using multi-criteria decision-making process i.e. AHP Analytic Hierarchy Process. The details of the process will be discussed in chapter 3.

2.3.2.2.2 VM Placement

During the VM consolidation process, the most difficult step is to select the most appropriate host for the placement of selected VMs from the previous step. Thus, it is the most important step of the consolidation process. Several researchers have presented their own methods for the placement of VMs such that they can able to decrease the energy consumption of data center. In most of the existing work, the VM placement has been considered as the problem of bin packing, but it is the not only solution for this problem. The existing research work can provide us with the details of every possible solution for VM placement is shown below:

1) VM placement as Bin Packing

Anton et al. [5] discussed the VM placement as a bin packing where, the physical machines represent bins and virtual machines represent items. Moreover, they have considered this problem as NP-hard and therefore, modified the (BFD) Best Fit Decreasing algorithm in such a manner that not more than 11/9.OPT +1 bins can be used for packing. OPT means the optimal solution provides for a number of bins [68]. Their modified algorithm is known as (MBFD) Modified Best Fit Decreasing. The process of PABFD sorts all the VMs according to their CPU utilization and accordingly, the first VM will be placed over that particular host which will provide minimum increase in power consumption and so on. This process will choose the most efficient host for placement. Later on, Anton et al. [10] named it as Power Aware Best Fit Decreasing (PABFD). Figure 2.6 shows the scenario of VM placement using bin packing.

Weijia et al. [69] proposed VISBP i.e. Variable Item Size Bin Packing algorithm. According to their survey, they found that the packing of items within the online bin packing algorithms is without the knowledge of item's size. As, the resource demands of virtual machines may change with the time and thus, the size of items cannot be fixed for bin packing problems for which several authors presented a different technique for repacking. Some have handled this problem by removing the item from the bin and pack it again. Since the removal of items leads to deletion of items. Therefore, in [69] author provides the strategy of repacking some other items to the bin rather than changing the items. Thus, VISBP deals with the problem of packing of items when resource demands of VM (items) change.

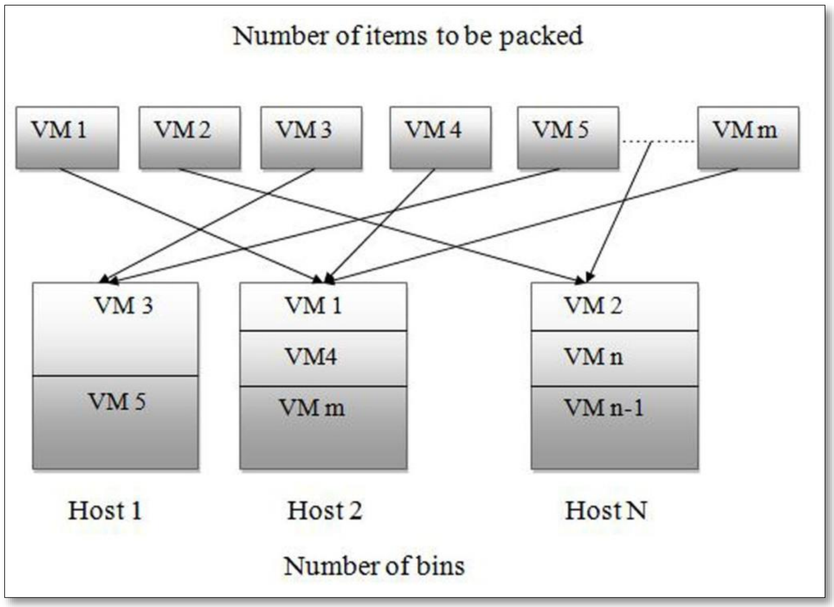


Figure 2.6: VM Placement using Bin Packing

Lovasz et al. [70] presented three different heuristics for the optimal solution of Energy and performance aware VM placement. They also related the problem with NP-hard algorithms. Their heuristics are: 1) BestFromRandom Heuristic with a complexity of O(n.m) where n represents VM and m represents PM. In this heuristic, the algorithm considers several random/valid VM-PM mapping and power consumption will be calculated for every mapping. The VM-PM mapping causing the minimum power consumption will be considered as the solution. 2) Greedy Heuristic- the approach of this heuristic is to sort the VMs according to their size i.e. sorting factor which is the combination of resource usage as well as total power consumption. VM list will be sorted in decreasing order and then, first VM with the biggest size will be removed from the list and mapped to the server with minimum power consumption. This heuristic also provides the VM-PM mapping with the complexity of O(n.m). 3) Modified First Fit Heuristic- it is similar to the previous one heuristic. In this, both the VMs and PMs will be sorted. VMs will be sorted in decreasing order and PMs will be in increasing order of their sorting factor. Similarly, the first VM will be removed from the VM list and mapped over the first PM of the PM list. The overall complexity of this heuristic is also O(n.m)

Consolidation of servers is widely adopted the technique for energy minimization since it minimizes the idle power within the infrastructure of the data center. There process of consolidation may be of two types: static consolidation and dynamic consolidation. During the static consolidation, the mapping of VMs to the PMs cannot be changed at runtime. As the resources are statically assigned, therefore, it cannot solve the problem of overprovisioning. During dynamic consolidation, the resources are allocated at run time i.e. if a load of the VMs changes then, PMs will be remapped. Thus, dynamic consolidation helps to minimize the overprovisioning of the resources. Multiple virtual servers on the same physical machine also strengthen the argument between the resources. By consolidation, multiple servers on the same physical server allow the non-exclusive usage of the resources which will lead to a certain level of performance degradation. The amount of acceptable performance degradation is always mentioned in SLAs.

Chaima et al. [71] presented their work on the allocation algorithm by making the use of bin packing problem. Their objective is to pack the virtual machines i.e. VMs into a number of physical machines i.e. PMs which are characterized by power consumption. The mapping of VMs over PMs depends upon some resource constraints such that: 1) VM can be mapped to only one server. 2) Each severs must has some maximum power limit which should not be beaten during the allocation of VMs. Their proposed algorithm is also an adoption of the Best-Fit algorithm. Here, the process of VM placement also starts with the sorting of virtual machines in the decreasing order of power consumption and constructs an ordered stack of VMs from which the first VM will be most power consuming VM and it will be placed on the server which has minimum power consumption. This process will be repeated until all the VMs are packed in the servers and remaining servers will be switch off to save the energy consumption

Mayank et al. [57] presented that consideration of available resources of physical host or destination is not enough. Some other important points should also be considered such as: after the mapping of some new VMs over the PMs, how will be the performance of VMs that already hosted over the PMs get affected. Their proposed algorithm also makes the use of bin packing algorithm along with vector packing algorithm where VMs and PMs are arranged in the order of their resources and then Best Fit or First Fit heuristics are applied on them to select the most appropriate PM. They also described the important concept of memory aware migration. According to which the VM can be remapped if the other PM host better memory sharing partner. For example, if two VMs are mapped over different PM and they have strong communication between them, then one of the VM will be remapped to the PM which hosts its communicating partner.

Horri et al. [6] presented their work on VM consolidation. They proposed VM placement algorithm provides the mapping of VM-PM according to the utilization of host and minimum correlation (UMC). The idea of UMC was presented by Verma et al. [72] that the probability of server to become over utilized will be more if the correlation between the applications that use same resources over an over provisioned host is more. According to this UMC, virtual machines will be placed over that host, whose value of CPU utilization has the lowest correlation with the CPU utilization of all other virtual machines on that host. The correlation of CPU utilization among VMs can be calculated using different coefficients [73]. Hill optimization method has been used by them for setting the values of the threshold.

Sina Esfandiarpoor et al. [74] presented a new direction for the energy efficiency of cloud data centers. Along with the efficient consolidation, they have also worked for the efficient structure of data center which means that fewer switches, racks and routers should be used without negotiating the SLA and idle cooling and routing equipments should be turned off for the reduction of energy consumption. For the efficient VM consolidation, they have used a different metric for the ranking of VMs i.e. Millions of Instructions Per Second (MIPS) instead of CPU utilization. They proposed VM placement algorithm by making some improvement on the MBFD algorithm presented in [10] and able to find the best physical machine for each virtual machine which causes the smallest increase of power consumption. For structure aware efficient VM consolidation, they considered network topology, cooling equipment, utilization of racks and utilization of the individual server. Rack utilization is the ratio of the Millions of Instructions Per Second requested by the virtual machine to the total Millions of Instructions Per Second capacity of physical machines. The VM placement by considering the rack utilization also used the method of threshold values. If the value of rack utilization is lesser than the threshold, then all the virtual machines are migrated from this rack to another rack such that the switches and cooling equipment of this racks can be turn off for energy savings. Three racks aware VM placement policies have been proposed by them. 1) RBR- virtual machine placement using Rack by Rack. 2) NUR- virtual machine placement in Non-Underutilized Racks. 3) HSRC- Hybrid Server and Rack Consolidation. The first rack aware policy sorts the racks according to their utilization and VMs will be sorted according to the MIPS, then first rack from the sorted list will be selected for the first VM from the sorted VM list and finally, the previous algorithm will be used for the selection of best host machine for selected rack. In the second policy of rack aware VM placement i.e. NUR, during the placement of each VM, first the best host will be found from all non-underutilized racks and if no host can be found then only the algorithm will try to find the host in the under-utilized racks. This algorithm provides more energy savings than RBR by minimizing the number of extra servers to be turned on. Third and last policy of HSRC combines both servers and racks for consolidation. In this policy, the rack consolidation will be used when the similar number of machines or servers are obtained from server consolidation otherwise HSRC will assign the VMs to the ON servers in a minimum number of racks.

Ehsan et al. [65] provided a multi-criteria algorithm for making the decision of choosing the most appropriate host known as TPSA policy. Selection of host depends upon the five different criteria taken by the author. TPSA simply calculates the score of all the hosts which are suitable for hosting the VM and select the host with the highest score. Author has divided the criteria into benefit and cost type. If the value of criteria with benefits type is more and value for criteria with cost type is low, the solution will be the optimum one and vice a versa. TPSA considered following conditions for calculation the score of host: 1) the selected hosts should have least increase in power consumption, 2) the selected hosts should have more available resources, 3) the selected host should have minimum number of VMs, 4) there should be minimum value of correlation of the VMs that are to be hosted on the selected machine, 5) the value of migration delay should be minimum for the VMs that are to be hosted on the selected machine. All these considerations reduce the percentage of SLATAH, PDM, level of SLA violation, the number of migrations and provide better energy savings.

Chowdhury et al. [75] proposed some algorithms rather than using best fit decreasing algorithms and their objective is also the reduction of power consumption while dealing with QoS

-Quality of Services. First, they proposed MWFDVP- Modified Worst Fit Decreasing VM Placement which is modified version of WFD-Worst Fit Decreasing. WFD is just opposite to that BFD which means, it chooses the physical machines with a maximum increase of power consumption. After that, they slightly modified the first one by placing the VMs over the host that has second minimum available power and known as second Worst Fit Decreasing technique (SWFDVP). The third one is the Modified First Fit Decreasing VM placement-MFFDVP, it will select the host from the first host presents in the host list and checks whether it is suitable or not for VM. If it is suitable; VM will be allocated to that host otherwise next host will be checked and similarly the process will go on. MFFDVP has been modified further and known as FFHDVP- First Fit Decreasing with Decreasing Host VM Placement. In this, hosts are sorted in decreasing order of their available power. Thus, the first virtual machine will be mapped over that physical machine which has maximum available power. After VM mapping, the host list will be again sorted in decreasing order and this will continue until all the VMs from VM list are not mapped over the hosts.

Moreover, the clustering technique has been also provided by the author. This clustering technique creates the clusters of virtual machines on the basis of its RAM and current CPU utilization. After the creation of clusters, first, their policy will find the host for those VMs which are the part of highly dense clusters followed by the VMs of second dense clusters. VM clusters are nothing but VM lists and these clusters are a group of objects with same attribute values. Those groups reside together and form clusters. In their clustering technique, they adopted centroid based clustering along with its most popular technique i.e. K-means algorithm. It will create VM clusters by assigning VM to its closest centroid which is calculated using CPU utilization and RAM. The process will compute the centroid of every cluster until the centroids do not change. From this clustering, they have modified all previous MWFDVP, SWFVP and FFHDVP algorithms and used their concept along with clustering technique. This modified algorithm were known as MWFVP_C (modified worst fit VM placement for clustering), SWFVP_C(second worst fit VM placement for clustering) and FFHDVP_C (first fit with decreasing host VM placement). Their novel technique provided better energy savings along with better quality of services.

2) VM Placement using Evolutionary techniques

Evolutionary techniques are inspired by the behaviour of a living organism and natural evolution. An evolutionary technique includes several algorithms and they are: Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and many other nature inspired algorithms. Recent work using these evolutionary algorithms shows the effectiveness and efficiency in the complex cloud environment.

These evolutionary algorithms follow the three steps for their entire process to take place: evaluation of fitness function, candidate selection and trial variation. The advantages of these evolutionary algorithms are that their computational complexity is polynomially related to the scale of the problem rather than exponential. For Genetic Algorithms, the computational complexity is proportional to a maximum number of generations and the size of the population and it is O(uG). whereas, for PSO the complexity is: O(MuG) and M represents the number of tasks. For ACO, the computational complexity has been found as O(MNuG) N represents the resources for M tasks. Thus, the complexity of ACO is greater than PSO which is further greater than GA [76]. Some of the recent work regarding the VM placement using evolutionary techniques discussed here.

Paolo et al. [77] has given the concept of genetic algorithm for the allocation of the virtual machines in distributed systems. Later on, this concept of GA for VM placement has been used by many researchers, during which the set representation plays an important role [78] where set represents the number of machines and set items signifies the virtual machines that need to be packed. However, the previous algorithms like bin packing and many others were not so good thus, various algorithms have been presented using GA to find the optimal solution of VM placement. A new concept was developed by Bandi et al. [79] which also made the use of genetic algorithm for VM placement by considering the current demand of VMs as well as the usage history of PMs for energy minimization. Shi Chen et al. [80] proposed the combination of hybrid genetic algorithm and knapsack problem for VM Placement by using multiple fitness in order to validate the efficacy of their algorithm. They have considered the function by distributing the complete load of system into 3 dimensions such as: CPU utilization, throughput and I/O rate for performing live migration of VMs and due to which they achieved the goal of lowering down the energy consumption and increase in resource utilization. The concept of hybrid genetic algorithm was also used by T. Thiruvenkadam et al. and Maolin Tang et al. [81-82]. They used this concept for load optimization during VM placement using 2 ways: a) packing of VMs after checking the load of every PM, b) use of hybrid GA for the optimization of VMs. Moreover, they have also attained the target of improvement in resource utilization with reduction in the energy consumption and SLA violation. Maolin Tang et al. considered the energy consumed by both the PMs and their communication network using the concept of basic GA for VM placement and after that, they enhanced the performance of cloud environment by using the concepts of hybrid GA.

Novel form of GA was presented by researchers [83-84] and named as Grouping Genetic Algorithm (GGA) to provide more efficient and optimal results for VM placement. These types of algorithms provide better results than non-grouping algorithms but relation between VMs, data centers and servers has not considered by them. Later on, to solve this problem Fereydon et al. [85] have chosen the GGA as reference algorithm and propose novel multi-level grouping algorithm (MLGGA). These novel algorithms were able to consider the relationship of these individuals as well as group of each individual in the search of optimal solutions. Another important work presented by Maolin et al. [83], which introduced the Reordering Grouping Genetic Algorithm (RGGA) for resolving the problem of bin packing problem which deals with multi-capacity bins during consolidation. It deals with various severs with varying capacities (storage, network, CPU, memory etc.) and VMs with variable weights. Yu-Shuang. D et al. [86] proposed another approach for VM placement using GA. Their proposed distributed parallel genetic algorithm (DPGA) performs in 2 stages. First stage picks the initial set of population from the solution space to obtain several solutions by implementing GA parallely on various hosts. Second stage takes the solutions obtained from the first stage as the initial set of population thus, the optimal solutions can be found. A novel family genetic algorithm (FGA) was proposed by Christina et al. [87]. This method has been used to improve the GA execution time by separating the population into different families. The group of families obtained from previous step will be processed parallely and thus, the FGA method reduce the execution time of VM placement algorithm.VM placement problem also has been solved by Gao et al. [88] by using the concept of Ant Colony Optimization (ACO) which is inspired from the collective foraging behaviour of real ant and their colonies. They have solved the problem of VM placement as a multi-objective optimization which uses the concept of dominance during the selection process. The approach of multi-objective optimization methods based on finding the set of paring to optimal solutions. Details for the concept of pareto dominance will be discussed in chapter 6. Similarly, [89], [90], [91] also used ACO based VM placement for efficient power consumption and resource wastage. Another approach which is the variant of ACO has been used by researchers for minimizing the resource wastage and power consumption during the VM placement and known as firefly colony algorithm. Layeb et al. [92] described the reasons for choosing this firefly colony algorithm for VM placement. Boominathan et al. [93] considered the problem of server consolidation as vector packing and solved it with firefly colony algorithm. The procedure of firefly colony algorithm based upon the behaviour of the flashing patterns of fireflies and accordingly, the process of VM allocation starts with the random allocation of VMs to the servers and in the second step, for the allocation of next VM over the same server will depend upon the attractiveness value. Higher the attractiveness value greater will be the probability of choosing. Similarly, the process will go on by updating the value of attractiveness for more optimal allocation. Moreover, the author also proposed fuzzy firefly colony approach for VM placement and used fuzzy sets for choosing VMs for allocation. Once the VM placement using ACO became new perspective of research, several authors started proposing their ideas by modifying the different function of ACO process for the achievement of multi-objective optimization.

3) VM Placement using Constraint and Stochastic Integer Programming

Constraint programming (CSP) can also be expressed as logic programming. It uses mathematical approaches along with a set of constraints, set of variables and domains to solve the complex problems of VM placement and provide optimal solutions. This approach of CSP is the variable assignment approach for maximizing or minimize the constraints while satisfying all the mentioned constraints.

Dupont et al. [94] presented a framework for the energy efficient resource allocation inside the data center to perform VM placement. Their approach was based upon the VM Repacking Scheduling (VRSP). Also, they have used SLA constraints for performing VM placement. Zhang et al. [95] also proposed a novel algorithm based on the constraint programming i.e. Virtual Cloud Resource Allocation (VCRA-CP). Their focus was to minimize the cost of resource usage by achieving the quality of services. Dong et al. [96] used different constraints for VM allocation such as size of the physical machine and network link capacity. They proposed two stage VM allocation algorithms in which, they used Best Fit Decreasing heuristic with min cut hierarchical clustering for bin packing. It will decrease the number of active hosts inside the data center and used maximum link utilization (MLU) to avoids network congestion. The second stage of VM allocation is the re-optimization of the allocation. In contrast to constraint programming, stochastic integer programming is one of the mathematical optimization technique where future demands are very uncertain [97]. These techniques use the estimation model with the probability distribution of data. As the future demands of VMs are not known, therefore, this approach can be used for optimal VM Placement. Bobroff et al. [98] proposed Measure Forecast Remap algorithm (MFR) for dynamic server consolidation and migration of VMs in order to minimize the level of Sla violation and demand of servers. Their algorithm includes three main steps: Measuring of historical data, forecasting the future requirements, and remapping of VM-PM. Speitkamp et al. [99] also formulated the problem of server consolidation using NP-hard optimization model. They analyzed the historical data and use it with LP relaxation based heuristic for server consolidation. Also, they have used capacity planning approach for the optimal placement of VMs.

2.3.2.3 Virtual Machine migration

VM migration is also the backbone for VM placement or VM consolidation process. virtual machines are the instances of operating systems running on the computers. Sometime, several VMs are running on the same computer and make it overburdened. During this time, it may be required to transfer some of the VMs to another machine. Thus, the VM migration is the process of transferring the virtual machines from one to another physical machine and allows the improvement in performance and fault tolerance. The concept of VM migration as provided by Clark et al. [100]. According to which the migration of VMs simply transfers its memory images from source machine to destination machine. In the past years, it was required to shut down the VMs first and allocate the resource to new physical machine during migration. After that, the VM files are moved to start the VM over the new machine. VM migration procedures are of two types: online and offline VM migration. During the VM memory transfer, the process of live migration guarantees the continuity of service provisioning to the hosted application where as non-live migration suspends the execution of application before the transfer of memory image.

The process of live migration includes two different techniques: Pre-copy and Post-copy VM migration. The details of these methods are discussed as below:

Pre-copy VM migration

This method of VM migration copies the memory pages from the source host to destination host without suspending the execution of the virtual machine. Pre-copy VM migration starts with the selection of destination host follows by the reservation of the resources on the destination host which guarantees to provide the requirements of VM. After that, all the pages will be transferred to the destination machine and successive iterations will be performed on memory pages until the final round has not been achieved by transferring the remaining dirty pages [101]. This step will increase the migration time with the increase in the rate of page updating. In the third step, the VM will be suspended from source machine and resume at the destination. All of the remaining states are also transferred to the destination. The advantage of this process is that both the source and destination machine has the copy of VM at this stage and thus, the copy of source machine can be useful in the case of failure. Now, in the fourth step, destination host acknowledges the source host that it has successfully received all of the VM images. After this acknowledgement source host rejects the original virtual machine and destination host acts as the primary host. Finally, the VM starts on the new host and resume its normal operations.

Post-copy VM migration

The process of post-copy VM migration starts by capturing all the states of VM such as I/O state, CPU state at the source machine. All these states are transferred to the destination machine and resumed there. After this, all the memory pages will be fetched from source servers until both the source and destination machines are synchronized. Figure 2.7 shows the pictorial view of live migration of VMs. Hines et al. [102] implemented the post-copy approach of live migration for migrating VMs from one host to another. According to them, there are four main components of post-copy approach live migrations: active pushing, demand paging, dynamic self-ballooning and prepaging. According to their performed evaluations on Linux and Xen, they showed that post-copy approach minimizes the total migration time. Michael et al. [103] also implemented post-copy approach and compared it with pre-copy live migration. Their evaluation showed some improvement in the migration time and a number of pages transferred. They used DSB- Dynamic self-ballooning for the elimination of free memory pages and thus, speed up the migration process.

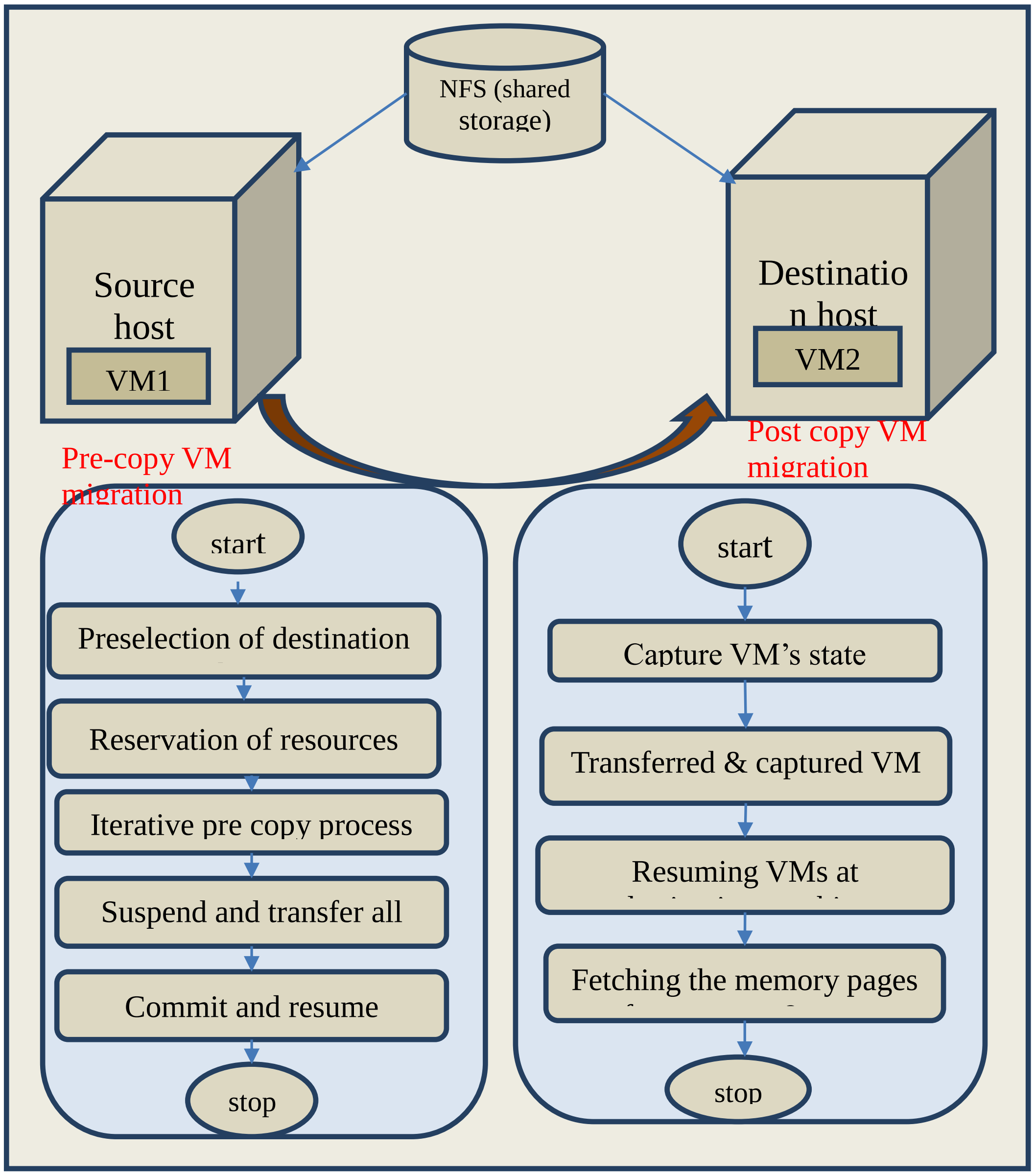


Figure 2.7: Live migrations of VMs

2.4 RESEARCH AND CHALLENGES FOR ENERGY EFFICIENT VM CONSOLIDATION

The advancement of virtualization has revolutionized the IT companies as well as academics by providing new possibilities and opportunities. By providing it's most interesting feature of VM migration to data centers, where VMs are created, resized, migrated and terminated according to the requirement provides a dynamic environment to the data centers. Utilization of virtualization and VM consolidation has widely adopted by IT infrastructures to increase the resource utilization of data centers as well as to reduce the operating costs. There are some benefits as well as challenges for adopting the VM consolidation. As, we have discussed the benefits of adopting VM consolidation thus, here we are discussing some challenges for the adoption of VM consolidation inside data centers.

- The process of VM consolidation puts several VMs on the single physical machine for hosting multiple applications. At some point, this may cause a single point of failure (SPOF). Also, the unavailability of several applications may occur dur to the up-gradation and maintenance of single server.
- The sharing of physical resources provides the contention of resources during the consolidation process; thus, it may affect the performance of applications.
- Applications which are delay sensitive such as online video and audio conferencing, VoIP –Voice over IP requires special consideration during the allocation of the resources.
- The migration step of VM consolidation provides the overhead on the CPU cycles of the servers as well as on the network links to data centers [104]. From the literature, we have found that the migration of application provides the degradation of the performance. Therefore, the design of VM consolidation process needs to be there to minimize the live migration of VMs.

Even with all these disadvantages of VM consolidation, the adoption of this process is increasing day by day due to its several benefits such as reduction in operational cost, minimization of energy consumption, improvement in the resource utilization of data center and much more. Therefore, several features, characteristics of the resources and applications that are hosted inside the data centers such as storage devices, deployment platform, physical devices, system software, types of applications and workloads and many more need to be considered during the implementation and designing the process of virtual machine consolidation. Moreover, there should be realistic power models for VM placement, allocation policies and for network devices for the optimization of data centers. The focus of current research is the energy awareness during the resource allocation and VM placement. The consideration of network overhead is another research direction that is not explored much yet. As we have discussed above that the sharing of the resources provides the contention of the resources which lead to profit minimization and SLA violation. Therefore, to understand the behaviour of resource usage pattern of application [104] for the efficient placements of VMs is an important point of consideration. It needs some research points for minimizing the contention of the resources for the efficiency of the data centers.

CHAPTER 3

MEDIAN BASED THRESHOLD (MEDTH) AND ANALYTIC HIERARCHY PROCESS (AHP) FOR HOST AND VM SELECTION

The purpose of this chapter is to propose an automatic detection method for over and underutilized host for the dynamic cloud environment. Along with this, this chapter also provides multi –criteria decision-making approach for VM selection.

3.1 INTRODUCTION

The demand for a cloud computing enormously increases the consumption of power and energy due to the occurrence of several electronic components inside data centres such as, switches, servers, racks and many others. Thus, these components also need a large quantity of electricity to cools down which also result in the emission of high carbon dioxide. From existing literature, it has been found that the consumption of data centers is 1.1% to 1.5% of overall electricity consumed all around the world and which is growing with the rate of 12% per year. To Minimize the energy consumption of the data centers plays a significant role for environmental sustainability. This energy consumption can be minimized by reducing the usage as well as improving the utilization of cloud resources. Therefore, dynamic VM consolidation plays a very important role and it is also an effective method for the reduction of energy consumption by switching off the idle machines which minimize the number of active hosts.

Dynamic VM consolidation constitutes four different steps and here in this chapter, we provide the solution for first two steps of VM consolidation. First, we have proposed a novel and automatic method for the selection of hot spots and excess capacity servers. The detection of these servers is useful for starting the process of migration. This median based threshold approach is used for finding the lower and upper threshold values for the selection of such servers. The minimization of performance degradation and SLA violation are the two main objectives that we have considered.

Secondly, we have provided the solution for the second step of VM consolidation process i.e. VM selection method. Analytic Hierarchy process has been used for the selection of VM for migration. In this approach, we have used the resources such as CPU utilization, RAM, migration time all together instead of using one at a time like previous research methods. Minimization of the total energy consumption and SLAV – Service Level Agreement Violation are the two main objectives that we have considered during the implementation of this method.

3.2 RELATED WORK

In the contrast of selecting hot spots and excess capacity servers, Jeffrey et al. [62] set 25% and 75% of utilization as the lower and upper threshold values for utilization. If the utilization of machine is less than the value of lower threshold; all of the VMs will be migrated from that machine and it would be turn off or kept into an idle mode to save energy consumption. Similarly, if the CPU utilization of machine is larger than the value of upper threshold value; few of the virtual machines will be migrated from that machine. This method of setting upper and lower threshold is static and not suitable. Anton et al. [10] also proposed a static method for finding threshold value and analyzed that this static method for finding thresholds is not beneficial for dynamic environment, as they do not adapt the changes in workload and therefore, they presented four statistical methods for determining threshold values such as: MAD, LR, LRR and IQR. The approach of linear regression also has been implemented by Fahimeh et al. [105], to predict the CPU usage of the host machine and then live migration process is used to detect underutilized and over utilized machine. Horri et al. in [6] used a novel technique VDT i.e. VM-based Dynamic Threshold for the selection of under-utilized hosts. In their method, they have used VMs on the host and CPU utilization of host for optimization along with hill claiming method.

Along with this, Anton et al. [10] also proposed 3 policies for the selection of virtual machines and they are: MMT- requires minimum time to migrate VM to another host, RC- randomly chooses the VM for migration and MC- migrates only those VMs that have highest value of the correlation of their CPU utilization with other VMs. In [64] Hassan et al. presented the policy which considered both the CPU utilization of VMs as well as migration time for the selection of VMs to migrate from one machine to another. Sometimes VMs are selected depending upon their resource utilization and there are different resources such as CPU, Memory, bandwidth, I/O. Due to different types of resources, it is not so easy to compare resource requirements of all the machines. Therefore, on the basis of recent findings and literature, we have proposed median based auto-adjustment or dynamic method for determining the threshold values that will further help for the selection of over and under-utilized servers. Also, we have solved the problem of VM selection using multi-criteria decision-making process i.e. AHP Analytic Hierarchy Process in which we used three resources such as the CPU utilization, RAM and migration time were taken by VMs all together for VM selection.

3.3 PROPOSED SYSTEM MODEL

System model considered in this work consists of big data centres which contains P different heterogeneous physical machines (i.e. PM) and each PM is categorized by its CPU performance in terms of MIPS (Millions of instructions per seconds), its total network bandwidth and total RAM occupied. These PM are also consisting of several heterogeneous (VMs) virtual machines to run user applications and to satisfy the needs of customers and they are also characterized by their total MIPS, Network bandwidth and RAM, so that several user can requests for the provisioning of VMs along with their particular characteristics.

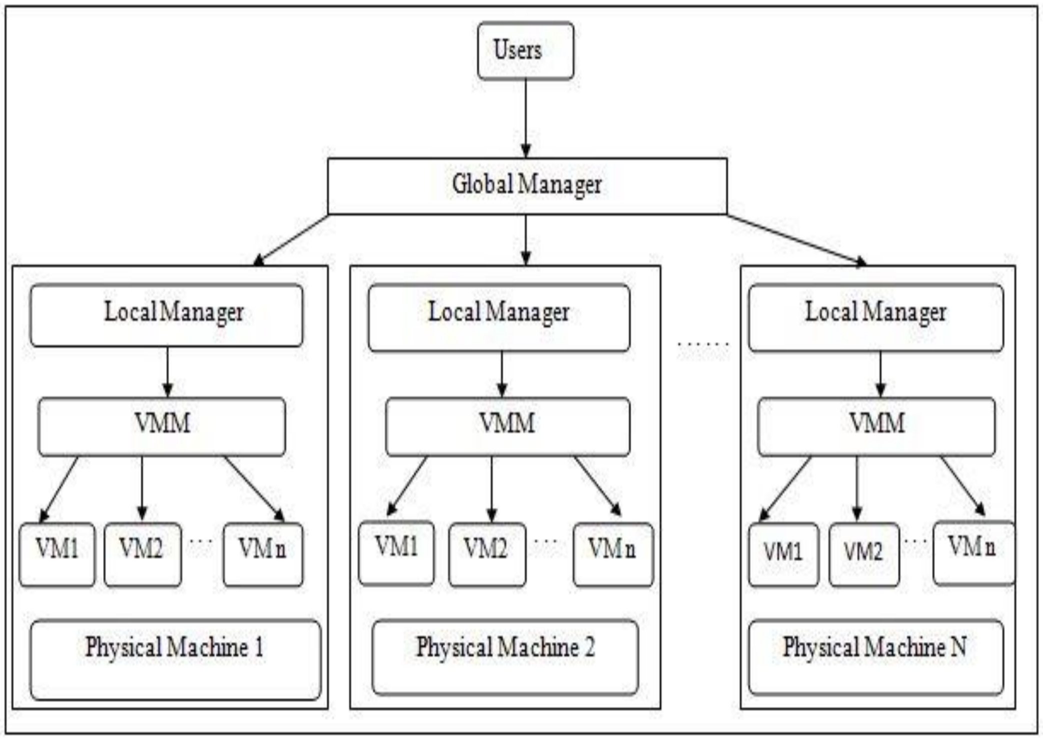


Figure 3.1: System model used for proposed work

The target system model that we have considered is similar to [10] and shown in Figure 3.1 which contains 2 imperative parts: a) Global manager and b) Local manager. First one i.e. global manager performs as a resource manager for the allocation of VMs to available physical machines. The allocation of VMs is based on the predefined characteristics mentioned previously whereas, the second one i.e. local manager acts as a decision maker to decide the exact time and place where a VM should be migrated. Thus, both of these managers play an important role inside data center architecture.

3.3.1 Data center’s power model

Disk storage, CPU performance, Power supply, memory and cooling systems are the few important aspects that influence the power consumption of PM exists inside data centers. CPU is the most important and major source of power consumed by data center therefore, from the existing literature it has been analyzed that linear relationship exists between power consumption and CPU utilization. Here in this work, we have supposed that the total MIPS of a CPU is m *c, where m, c represents the CPU cores as well as MIPS of each core respectively. We have taken the power consumption of machines from SPEC power benchmark [106]. They have provided the power consumption of each machine in every condition i.e. when machine is idle or when it is 100% utilized. We have used 6 different servers and they are HPProLiant ML110 G5, HPProLiant ML110 G4, IBM Server x3550XeonX3470, IBM Server x3250XeonX3480, Acer AR320 F1 and Acer AT150 F1. Table 3.1 shows the configuration of these servers and table 3.2 shows the power consumed by these servers from idle to 100% utilization.

Table 3.1: Six different servers with different configurations

Servers	CPU Model	Cores	Frequency (MHz)	RAM (GB)
HP ProLiant G4	Intel Xeon 3040	2	1,860	4
HP ProLiant G5	Intel Xeon 3075	2	2,660	4
IBM Server x3250	Intel XeonX3480	4	2,933	8
IBM Server x3550	Intel Xeon X3470	12	3,067	16
Acer AT150 F1	Intel Xeon 5670	12	2,933	12
Acer AR320 F1	Intel Xeon 3470	4	2,933	8

3.3.2 Energy model

In this work, the energy consumption has been evaluated with the help of this model. To minimize the energy consumed of data center is the key objective of the current research thus, it is very essential to enlighten this model of energy consumption which can be calculated by the summation of total power consumed by host during each small period of time frame, as shown in equation 3.1.

$$E(t) = \int P(t)dt \tag{3.1}$$

Table 3.2: Power consumption of servers in Watt [106]

Servers	**Idle**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
HPProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HPProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135
IBMServerX3250	41.6	46.7	52.3	57.9	65.4	73	80.7	89.5	99.6	105	113
IBM Server x3550	58.4	98	109	118	128	140	153	170	189	205	222
Acer AT150 F1	65.4	113	125	136	150	165	183	199	215	229	244
Acer AR320 F1	39.6	47.3	55	63.4	71.9	80.3	89.8	97.9	107	116	124

3.3.3 Performance Metrics

For measuring the performance of cloud environment, it is very important to analyze the performance metrics during the proposal of models. Here, in this work, we have considered the QoS (Quality of Servies) as a performance metrics which are formalized in Service Level Agreement (SLA) which has been contracted between the cloud service providers and users. To fulfill the requirements mentioned in QoS plays a significant role in cloud environment. These QoS can vary for different applications in terms of characteristics; therefore, it requires to define some metrics for the calculation of SLA delivered. In this work, we have used following performance metrics: SLATAH (SLA violation time per active host), PDM (performance degradation due to migration) and SLAV (SLA violation) that are introduced in [10] and defined in equation 3.2 and equation 3.3. Multiplication of these SLATAH and PDM metrics introduces SLA violation i.e. SLAV defined in equation 3.4 and its minimization along with energy consumption is the main focus of our study**.**

1. SLATAH (SLA violation per active host): percentage of time when CPU utilization of host machine reaches to 100%.

$$SLATAH = \sum_{i=1}^{N} \frac{T_{si}}{T_{ai}} \tag{3.2}$$

In equation, 3.2 N shows the number of hosts inside data center, whereas T_{si} is the time period when CPU utilization reaches to 100% and T_{ai} is the total time for which host remains active.

2. PDM (Performance degradation due to migration): calculates the amount of percentage the performance is getting degraded during every VM migration. In [24] it has been mentioned that the total performance degradation of system depends upon the host CPU utilization

$$PDM = \frac{1}{M} \sum_{j=1}^{M} \frac{C_{dj}}{C_{rj}} \tag{3.3}$$

In equation 3.3 M shows the number of VMs used inside data center, C_{dj} is the total estimate of the performance degradation of VMj during VM migration and it has been assumed as 10% of CPU utilization during all VMj migration. The total amount of CPU requested by VMj is represented by C_{rj}. These two SLATAH and PDM are equally important for SLA violation; therefore, SLAV is the combination of both SLATAH and PDM shown in equation 3.4

$$SLAV = SLATAH * PDM \tag{3.4}$$

3.4 PROPOSED VM CONSOLIDATION METHOD

Dynamic VM consolidation works in four different steps [10] inside data center and they are described as follows along with its flow chart in figure 3.2 which shows that in this chapter we have provided the solution for first and second step:

1. Selection of the overutilized host which needs the migration of VMs it contains.
2. Selection of VMs from selected over utilized host to perform migrations.
3. Selection of the under-utilized host for migrating it’s all VMs.

4. Designing of new placement policy for selected VMs from selected over and under-utilized hosts.

We have also used above 4 steps in our overall work and this chapter presented a novel method for the first and second step of consolidation process i.e. detection of over and under-utilized host as well as a selection of VMs from selected hosts for migration.

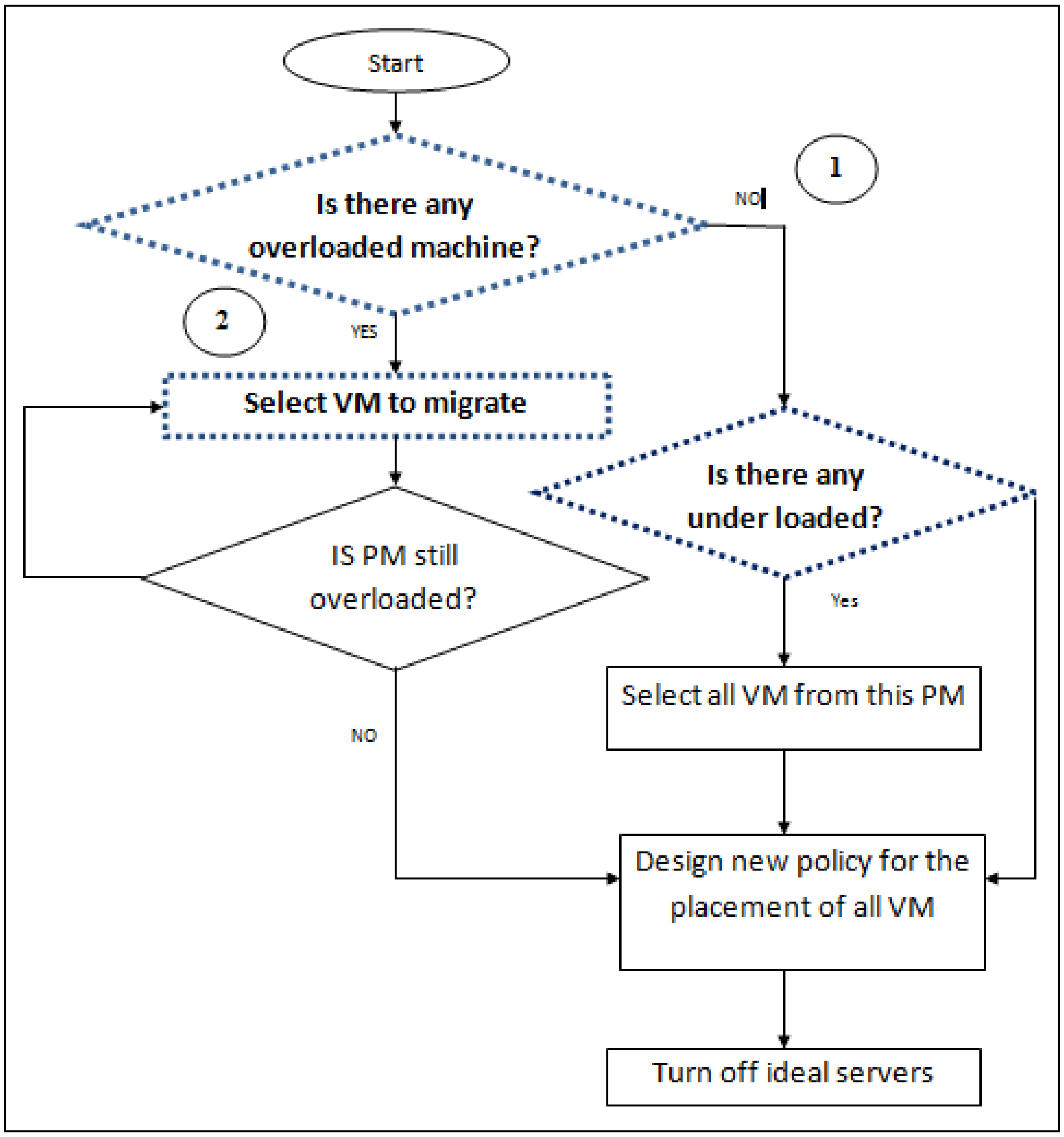

Start
Is there any overloaded machine?
NO
1
2
YES
Select VM to migrate
Is there any under loaded?
IS PM still overloaded?
Yes
Select all VM from this PM
NO
Design new policy for the placement of all VM
Turn off ideal servers

Figure 3.2: Flow chart for VM consolidation process

3.5 PROPOSED APPROACH FOR DETECTION OF HOT SPOTS AND EXCESS CAPACITY SERVERS

The previous approach of Anton et al. [10] for finding underutilized and over-utilized machines i.e. excess capacity servers and host spots for dynamic workload environment are extended by the consideration of the CPU utilization of all physical machines or servers. We have proposed an automatic method for finding over utilized and under-utilized host using Median based approach. If a physical machine is found as an over utilized machine then some of the virtual machines from this machine would be migrated to another machine or server, and if the physical machine is found as underutilized machine then all VMs of this machine would migrated to another physical machine.

3.5.1 Median based threshold approach for finding over utilized and underutilized host machines (MEDTH)

MEDTH method starts with the calculation of the CPU utilization of all physical machines present inside datacenter in the first step. After this, the upper and lower threshold values are to be calculated such that these threshold values will be further used for finding the hot spot i.e. over utilized physical machine and excess capacity server i.e. under-utilized physical machine. For the CPU utilization of all physical machines, we have used a random generator. C_i represents the CPU utilization of P_i numbers of host machines, where $i \in R+$ and P_i number of host machines can be arranged as $\{P_i = P_1, P_2, P_3 \ldots P_i\}$. These physical machines can be even and odd in numbers, therefore median of even numbers of machines i.e. P_{2i} gives two new sets: the first set is (P_1 to P_i represented as X_i), and the second set is ($P_i +1$ to P_{2i} represented as X_j). Similarly, two different sets can also be arranged for odd numbers of physical machines i.e. $P_{2i} +1$. First set is (P_1 to P_i represented as Y_i) and the second set is ($P_i +2$ to $P_{2i} +1$ represented as Y_j). Similarly, upper and lower threshold limits Th_u and Th_l for both even and an odd number of physical machines present inside data centers can be detected using median method formularized in equation 3.5 and equation 6.

$$if\ \frac{P_i}{2} = 2x\ (x\ \in 1,2,3, \ldots \ldots ..\infty)\ \begin{cases} Th_l = median\ (X_i) \\ Th_u = median\ (X_j) \end{cases} \tag{3.5}$$

$$if\ \frac{P_i}{2} = 2x + 1\ (x\ \in 1,2,3, \ldots \ldots ..\infty)\ \begin{cases} Th_l = median\ (Y_i) \\ Th_u = median\ (Y_j) \end{cases} \tag{3.6}$$

With the help of these two equations, the over utilized and under-utilized physical machines can be found as follows:

$$\begin{cases} if\ CPUP_i > Th_u\ (P_i = Over_{host} \\ if\ CPUP_i < Th_l\ \ (P_i = Under_{host} \end{cases} \tag{3.7}$$

Here in above equation 3.7, $Over_{host}$ represents over utilized hosts and $Under_{host}$ represents under- utilized physical machines. If CPU utilization of physical machine is greater than the upper threshold value then the machine will be considered as over utilized and if the value CPU utilization of machine is smaller than the lower threshold then it will be considered as a under-utilized machine.

3.5.2 Results and discussions

For the performance evaluation of proposed approach, we have considered both the real and random workload environment. For random workloads, we have assumed that the data center contains 800 heterogeneous physical machines with the request for the provisioning of 800VMs

Table 3.3: Number of Virtual Machines for real workload environment

Date	**Number of Virtual Machine**
3 March 2011	1,052
6 March 2011	898
9 March 2011	1,061
22 March 2011	1,516

and for real workload we have taken the data from the project of PlanetLab [17] which is a monitoring infrastructure and named as CoMon project. This project contains the data for CPU utilization of thousands of VMs which is obtained from physical servers placed more than 500 locations around the world. Data from these servers are collected after every five minutes. We have selected data of four days from workload traces of the project during March 2011.There are different numbers of VMs for each day, from which each VM has assigned some workload trace randomly for every corresponding day. Table 3.3 shows the number of VMs for each day.

Table 3.4: Percentage of Performance metrics for random workload

Policies for detection of over utilized and underutilized host	SLATAH	PDM	SLAV
MEDTH	0.91%	0.02%	0.0182%
THR	1.78%	0.05%	0.089%
IQR	3.2%	0.11%	0.03872%
MAD	2.11%	0.09%	0.1899%

The simulation of these two different scenarios in CloudSim toolkit [107] provides following results. Table 3.4 and 3.5 shows the SLATAH, PDM and SLA violation occurred during random and real workload from 3rd March to 22nd March using MEDTH and three existing approaches THR, MAD and IQR taken from Anton et al. [10].

Table 3.5: Percentage of Performance metrics for real workload

Policies for detection of over utilized and underutilized host	Date	SLATAH	PDM	SLAV
MEDTH	3 March,2011	2.46%	0.05%	0.123%
	6 March,2011	2.26%	0.06%	0.1356%
	9 March,2011	2.42%	0.06%	0.1452%
	22 March 2011	2.39%	0.05%	0.1195%
THR	3 March,2011	4.95%	0.07%	0.3465%
	6 March,2011	5.08%	0.07%	0.3556%
	9 March,2011	5.21%	0.08%	0.4168%
	22 March 2011	5.11%	0.06%	0.3066%
IQR	3 March,2011	5.01%	0.07%	0.3507%%
	6 March,2011	5.02%	0.07%	0.3154%
	9 March,2011	5.27%	0.08%	0.033728 %
	22 March 2011	4.93%	0.06%	0.2958%
MAD	3 March,2011	5.23%	0.07%	0.3661%
	6 March,2011	5.26%	0.07%	0.3682%
	9 March,2011	5.47%	0.08%	0.4376%
	22 March 2011	5.13%	0.06%	0.318%

Results show that our MEDTH method provides a minimum level of SLA violation with lesser performance degradation and lesser SLA time per active host in comparison to other three policies. These two metrics are equally important for minimizing the SLA violation. Figure 3.3 shows the percentage of SLATAH, SLAV and PDM of the existing methods as well as our proposed method using random workload environment. Similarly, figure 3.4 shows their values using real workload environment. Results show that we have achieved the objective of minimizing the SLA violation.

3.6 PROPOSED APPROACH FOR VM SELECTION USING AHP-VM METHOD

After the detection of host spots and excess capacity servers, the second step is to select VMs from these servers and starts the process of migration. Therefore, for the selection of VMs, we have proposed a novel technique using multi-criteria decision-making process i.e. AHP- Analytic Hierarch Process. Multi criteria decision making is one of the most well-known branches of decision making which is associated with multiple attributes. These attributes are also known as decision criteria along with their importance or weight.

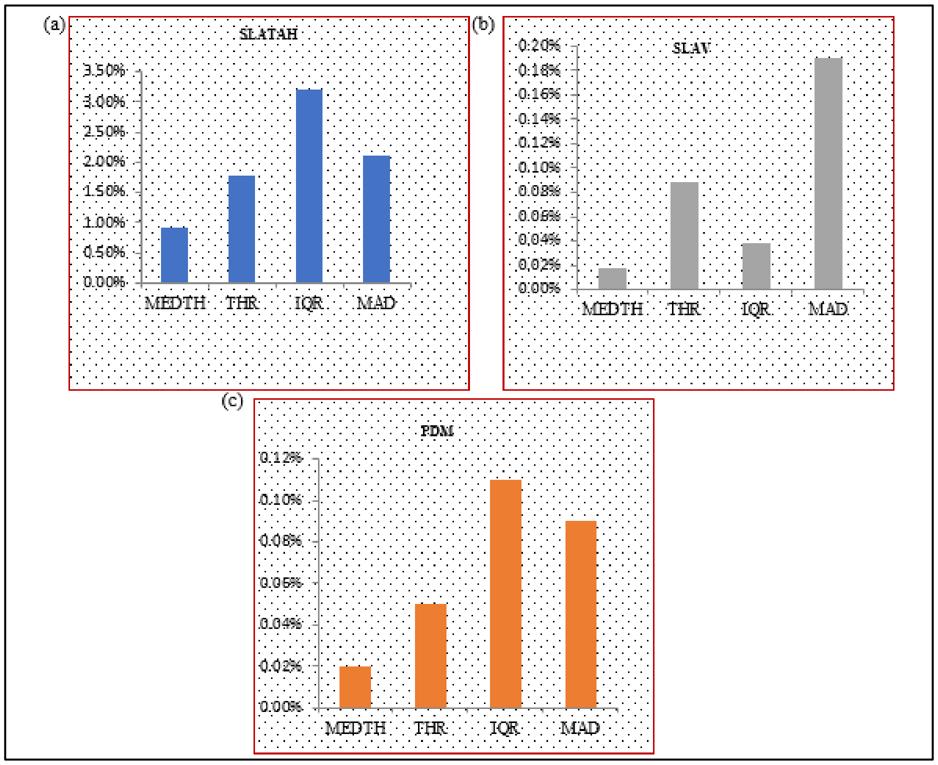


Figure 3.3: (a) Percentage of SLATAH, (b) Percentage of SLAV and (c) Percentage of PDM for random workload

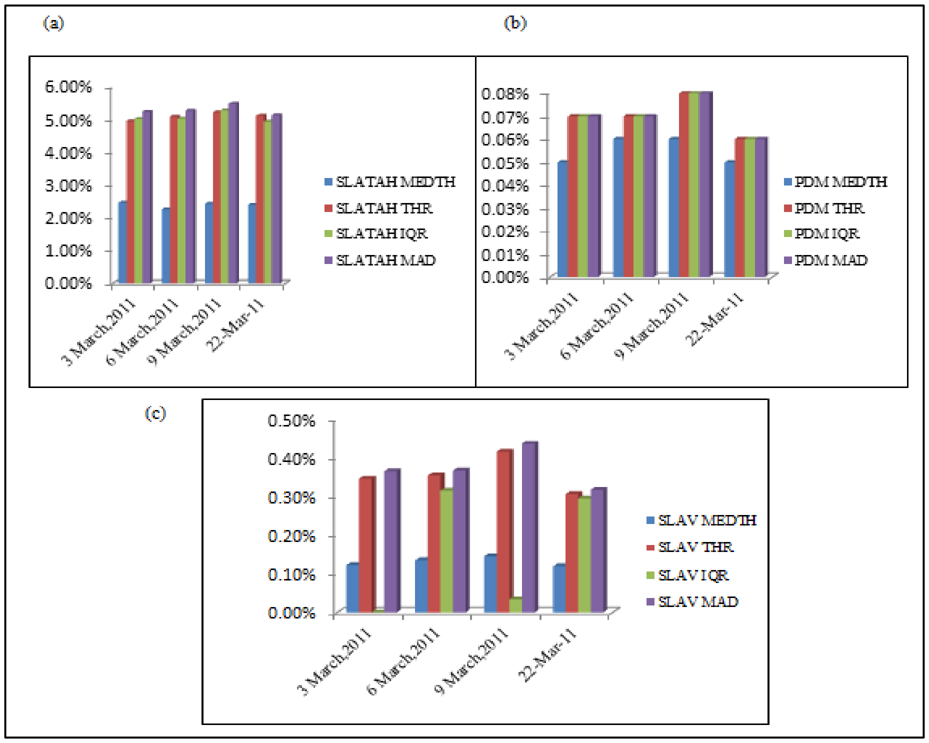


Figure 3.4: (a) Percentage of SLATAH, (b) Percentage of PDM and (c) Percentage of SLAV for real workload

3.6.1 VM selection policy using RAM i.e. memory occupied by VM, CPU utilization and migration time taken by VM.

The excess of VM migration may degrade the performance of data center, therefore the maximum CPU utilization, policy of minimum migration time and maximum memory space(RAM) deals with the problem of minimization of VM migrations along with SLA violation. To fix the lower as well as upper threshold for CPU utilization is the basic idea for the selection of physical machine from where the selected VMs will be migrate and it is also important to keep the CPU utilization of PMs between these threshold values. Therefore, few VMs will be migrated if CPU utilization of PM exceeds the value of upper threshold and all the VMs will be migrated if the value of CPU utilization falls below the lower threshold. Thus, our proposed selection policy i.e. AHP takes advantages of following three policies.

3.6.2 Maximum CPU utilization and Memory (RAM) occupancy

This policy will help to selects the VMs from under-utilized and over utilized machines which consumes maximum memory i.e. RAM as well as CPU utilization of the physical machine. VMs will be migrated in order to avoid the situation of over utilization and under-utilization of machines such that over utilized machine would not persist as over utilized for long time and under-utilized host would be switched in order to reduce the idle power consumption. These policies follows the process of VM selection according to the current utilization of CPU and RAM and will be repeated for all machines.

3.6.3 Minimum time for migration

Second policy for VM selection is according to the migration time of VMs. According to this policy, only those VMs will be selected which requires lesser or minimum time for performing the migrations from one to another host. Here, the migration time has been evaluated by total RAM or memory occupied by VM to the available bandwidth of selected host. Moreover, the process to transfer the contents of memory from one to another host is slow and requires more time therefore, it affects the performance of migration and system as well, therefore, the actual migration process requires minimum time for migration of VMs without its performance degradation. Thus, we have used the policy of minimum migration time proposed in [10] as one of our criteria for decision-making process.

3.6.4 AHP-VM Analytic hierarchy process for VM selection

We have proposed, a novel policy for VM selection using AHP (Analytic Hierarchy Process). It is a multi-criteria decision-making algorithm for the selection of VMs. This policy is based upon the computation of the scores i.e. a score matrix would be created for all VMs present inside the selected host and VM with highest value of score would be selected and so on. Above discussed three policies (maximum CPU utilization, maximum memory occupied by VMs, minimum migration time taken by VMs) would be the three different criteria's for the selection procedure of AHP method. As there are three different criteria's for VM selection, therefore AHP method first chooses the most dominating criteria among all. Here in our case, all these three policies are equally important therefore, we have given equal weights to them. This process is helpful for selecting the most relevant criteria and alternatives and it starts with the creation of N x M matrix, where N represents the number of alternatives for VM and M represents the number of criteria we have taken. The first step of AHP VM selection method is shown in following decision matrix in Equation 3.8.

$$VM_{AHP} = \begin{bmatrix} C_{vm1} & MO_{vm1} & MT_{vm1} \\ C_{vm2} & MO_{vm2} & MT_{vm2} \\ C_{vm_n} & MO_{vm_n} & MT_{vm_n} \end{bmatrix} \tag{3.8}$$

Vm1 to Vm_n represents the available VMs (virtual machines) inside the selected host machine with three different criteria C, MO, MT which represents the CPU utilization, memory occupied and migration time taken by VM. Following several steps will be performed in order to find the best alternative for VM selection:

Step 1: It involves the normalization of initial matrix VM_{AHP} by dividing its each value by sum of the values of every alternative as shown in equation 3.9.

$$Score(q_{ij}) = \begin{bmatrix} \frac{C_{vm1}}{C_{total}} & \frac{MO_{vm1}}{MO_{total}} & \frac{MT_{vm1}}{MT_{toal}} \\ \frac{C_{vm2}}{C_{total}} & \frac{MO_{vm2}}{MO_{total}} & \frac{MT_{vm2}}{MT_{total}} \\ \frac{C_{vm_n}}{C_{total}} & \frac{MO_{vm_n}}{MO_{total}} & \frac{MT_{vm_n}}{MT_{total}} \end{bmatrix} \tag{3.9}$$

Step 2: This step, will create a new matrix VM_{AHPnew} by multiplying the score matrix obtained from above step with some predefined weights. We have considered three different criteria, therefore, W_{ij} is the associated weight for each criterion which shows the status of each criterion and q_{ij} shows the score matrix. To find an optimized weight for each criterion is also a topic of research by itself, therefore; the priority or importance defined by the user is one of the methods for weight optimization. As, we have considered the equal importance for all the resources therefore; we have assigned equal weights to all three criteria's.

$$VM_{AHPnew} = \ max_i \sum_{j=1}^{M} q_{ij} * W_{ij} \ (for\ i = 1,2,3 \dots \dots \dots N) \tag{3.10}$$

Step 3: It involves the ranking of VMs where VMs will be ranked according to the values of VM_{AHPnew} matrix and VM with the greater value of score will be selected for migration.

Table 3.6: Pseudo code for AHP VM selection method

```
1  Input: Overutilized host and under-utilized hosts
2  Output: Selected VMs
3      For(each host h in host list)do
4       Vmlist ← h.getvmlist( )
5       MaxUtil ← vm.getutil( )
6       Max Ram ← vm.getRam( )
7       Min MT ← vm.getRam( )/vm.getBw( )
8       Create matrix Mtr of VM ij which stores MinUtil,MinRam,MinMT
9           For(each VM ij in matrix)do
10              Z ij ← VM ij /j total
11              Score ← Z ij × W ij
12          End For
13      Selectedvm ← max(Score)
14      Migrationlist.add(selectedvm)
15     End For
16 Return List of selected VMs
```

3.6.5 Results and Discussions

For simulation, we have considered the data center's architecture as: 800 heterogeneous host machines or servers with six different types. To use this type of architecture in a real life is very difficult therefore, we have used CloudSim an open source toolkit.

Table 3.7: Workload characteristics

Date	No of VMs	Mean %	SD %
03-03-2011	1052	12.31	17.09
06-03-2011	898	11.44	16.83
09-03-2011	1061	10.70	15.57
22-03-2011	1516	9.26	12.78
25-03-2011	1078	10.56	14.14
03-04-2011	1463	12.39	16.55
09-04-2011	1358	11.12	15.09
11-04-2011	1233	11.56	15.07
12-04-2011	1054	11.54	15.15
20-04-2011	1033	10.43	15.21

Configurations of six different types of servers are mentioned in Table 3.1 and table 3.2 shows the power consumption of these servers. Amazon EC2 [108] provides the VM instance that we have used and they are: 1) High-memory extra-large with (3,000 MIPS, 6000 GB RAM), 2) High-CPU medium with (2,500 MIPS, 850 GB RAM), 3) Extra-large with (2,000 MIPS, 3750 GB RAM), 4) Small with (1000 MIPS, 1700 GB RAM), 5) Micro with (500 MIPS, 633 GB RAM). For the workload of the simulation environment, we have considered 10 days data of CoMon Project [17] and characteristics of these 10 days data is shown in table 3.7.

The key point of VM consolidation is the reduction of energy consumption and SLA violation, thus we have used a combined metric to minimize both energy and SLA i.e. ESV. Equation 3.4 and 3.1 has been used to estimate SLAV and energy consumption respectively. Therefore, the ESV metric can be designed according to equation 3.11. As, the excess of VM migration may consume more energy and SLAV, therefore such points should be considered for the reduction of energy consumption so that excess of SLA violation and VM migration should not occur. Here we have taken the energy consumption, VM migrations, SLAV and ESV metric along with execution time (total time for VM consolidation to take place) for the performance evaluation of algorithms.

$$ESV = E.SLAV \tag{3.11}$$

3.6.5.1 Performance evaluation

For the validation of proposed algorithm, we have performed some experiments using real workload conditions. we have compared AHP VM selection method with other four existing methods of VM selections that are already used in CloudSim [107] such as: 1) Policy of Maximum utilization (MU)- selects a virtual machine which has maximum CPU utilization among all, 2) Policy of Maximum correlation (MC)- migrates VMs which have a higher value of correlation of its CPU utilization or resource usage [10]. 3) Policy of Minimum Migration Time (MMT)- which selects the VM which requires minimum time to migrate in comparison to other VMs. 4) Policy of Random selection (RS)- it randomly selects few VMs for performing live migration. Along with these VM selection methods, there are four different methods for finding the over utilized host inside CloudSim and named as: 1) Median Absolute Deviation (MAD) – a method for the auto adjustment of threshold value of CPU utilization [10], 2) Interquartile Range

(IQR)- a method for setting an upper threshold value of utilization on the basis of robust statistic [10]. 3) Local Regression (LR)- method to find threshold value with the estimate value of future CPU utilization [10]. 4) Static Threshold method (THR)- method to fix the threshold value of host's utilization. Using this method, the host will be considered as over utilized if its CPU utilization exceeds that static threshold. We have taken above mentioned 4 different VM selection policies and all 4 overload detection algorithms for performance evaluation by considering 4 different cases. It has been analysed that AHP VM selection policy provides a better combination of results in comparison to MU, MC, MMT and RS VM selection methods using any of the host overload detection method.

In a real workload environment, for the evaluation of proposed policy we have performed ten experiments using the workload conditions of 10 different days and their average results are also provided with four different test cases. Energy consumption, number of migration, SLA violation and total execution time to complete VM consolidation process are few performance parameters that we have considered. Results for the comparison of AHP VM selection method with rest of the four VM selection algorithms are graphically shown in Fig 3.5 -3.8.

Case 1: Median Absolute Deviation (MAD) as a hot spot detection algorithm along with five different VM selection policies.

In the first case, we have considered MAD - Median absolute deviation as hot spot detection policy i.e. over utilized host and calculate its impact along with existing VM selection policies as well as our proposed method.

Table 3.8: Comparative results using several VM selection policies with MAD

Policy	**Energy consumption (kWh)**	**No. of migration**	**SLAV 10^{-2}**	**ESV 10^{-3}**	**Execution time (Sec)10^{-3}**
MAD\AHP	28.06	12114	6.40	1.78	64.1
MAD\MU	37.88	15312	9.67	3.70	109.3
MAD\MMT	34.90	14214	9.06	3.15	95.0
MAD\MC	34.9	14249	9.07	3.16	101.4
MAD\RS	33.14	13463	8.43	2.77	83.6

Table 3.8 shows the results for MAD\AHP, MAD\MU, MAD\MMT, MAD\MC and MAD\RS. They depict that along with the minimization of energy consumption individually, using MAD as host detection and AHP as a VM selection method also minimizes the ESV metric which means it provides trade-off values for energy and SLA violation using lesser number of migrations. Figure 3.5 shows the graphical representation of the results

Case 2 Interquartile Range (IQR) as a hot spot detection algorithm along with five different VM selection policies.

Here also we have considered IQR – Inter Quartile Range as host overload detection policy and calculate its impact along with existing VM selection policies as well as our proposed method. Table 3.9 shows the results of IQR\AHP, IQR\MU, IQR\MMT, IQR\MC and IQR\RS with their graphical representation in figure 3.6

Table 3.9: Comparative results using several VM selection policies with IQR

Policy	**Energy consumption (kWh)**	**No. of migration**	**SLAV 10^{-2}**	**ESV 10^{-3}**	**Execution time (Sec)10^{-3}**
IQR\AHP	29.34	12505	7.57	2.21	76.86
IQR\MU	38.01	15314	9.8	3.72	105.78
IQR\MMT	34.915	14174	9.32	3.24	93.27
IQR\MC	34.89	14172	9.34	3.22	50.23
IQR\RS	33.24	13575	8.95	2.97	82.5

Case 3: Static Threshold (THR) as a hot spot detection algorithm along with five different VM selection policies.

In this case, again we have considered THR – Static Threshold as host overload detection policy and calculate its effects along with existing VM selection policies as well as our proposed method. Table 3.10 shows the results of THR\AHP, THR\MU, THR\MMT, THR\MC and THR\RS with their graphical representation in figure 3.7.

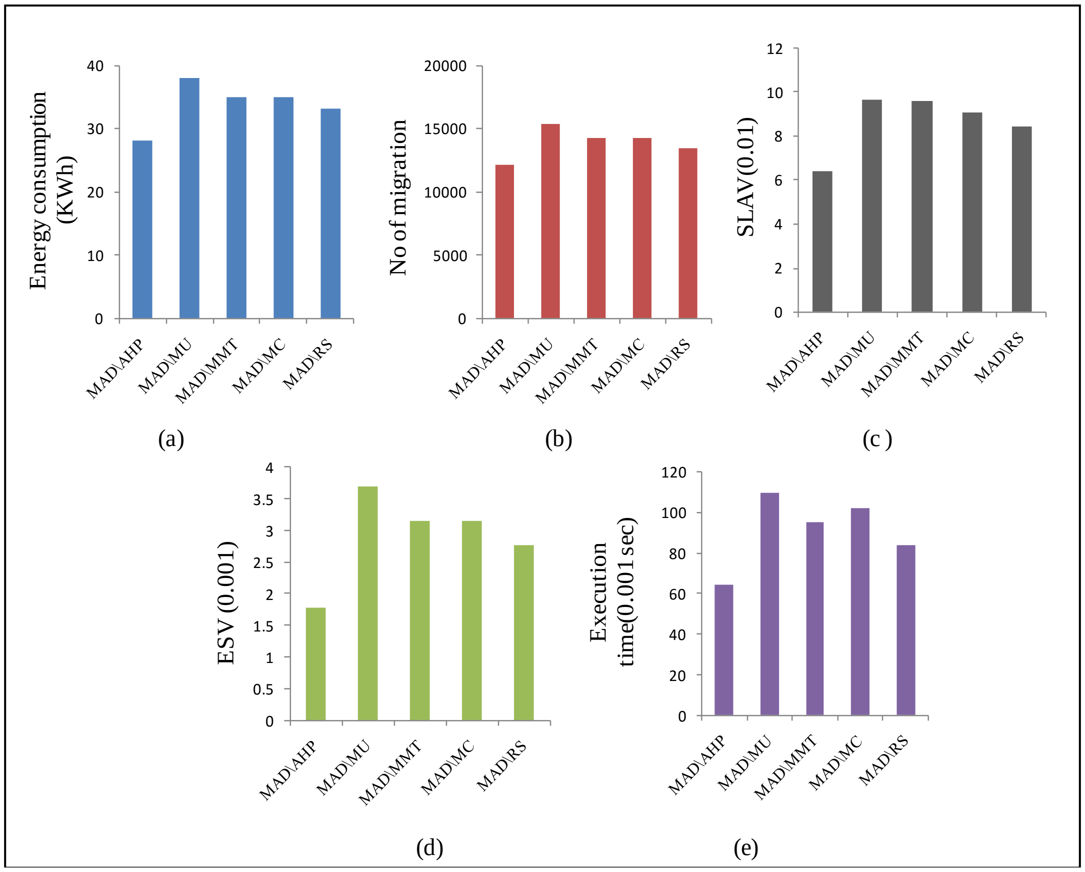


Figure 3.5: (a) Energy consumed by all VM selection policies. (b) Migration count during MAD. (c) Level of SLA violation caused by several VM selection policies. (d) Overall ESV metric and (e) total time to Execute the VM consolidation by using all VM selection policies with MAD.

Table 3.10: Comparative results using several VM selection policies with THR

Policy	**Energy consumption (kWh)**	**No. of migration**	**SLAV 10^{-2}**	**ESV 10^{-3}**	**Execution time (Sec)10^{-3}**
THR\AHP	27.79	12956	9.86	2.88	38.74
THR\MU	39.04	14075	10.00	3.90	46.53
THR\MMT	35.65	14510	9.98	3.55	42.79
THR\MC	36.14	14504	9.99	2.60	46.10
THR\RS	33.69	13242	9.97	3.55	42.36

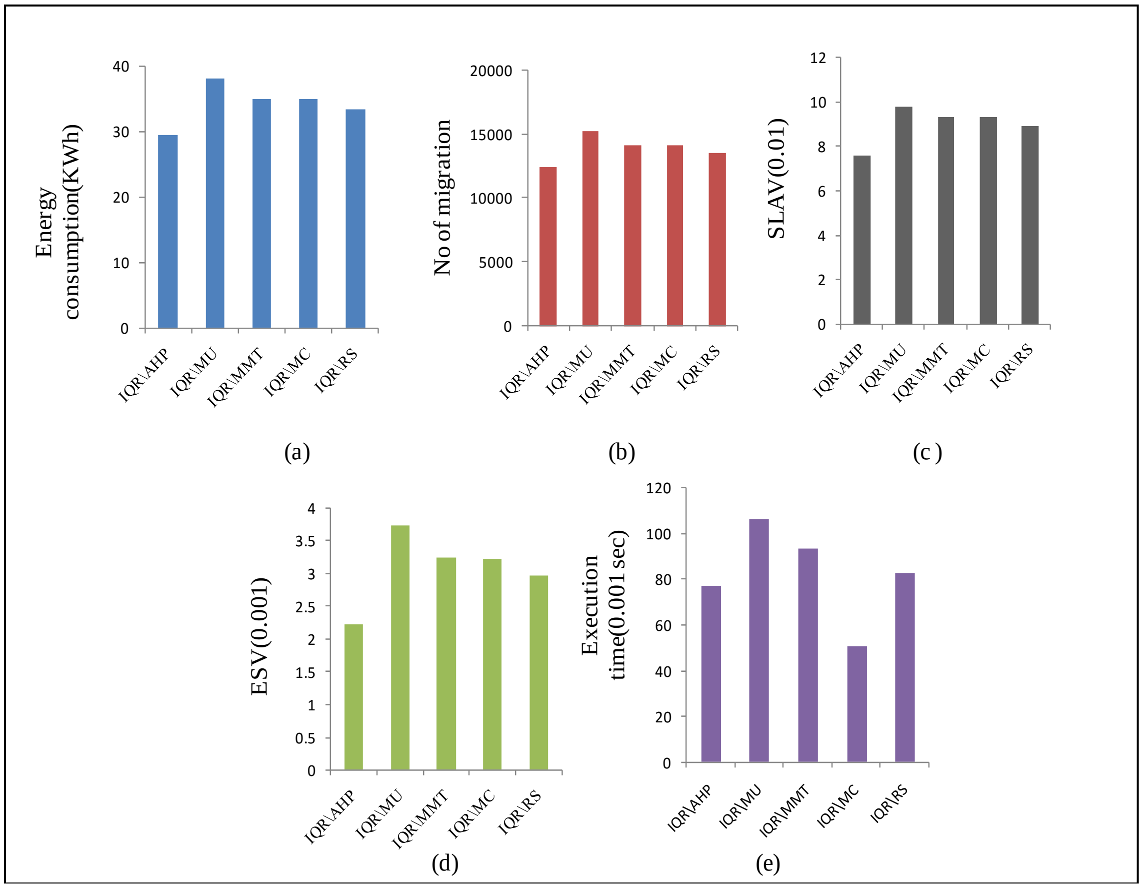


Figure 3.6: (a) Energy consumed by all VM selection policies. (b) Migration count during IQR. (c) Level of SLA violation caused by several VM selection policies. (d) Overall ESV metric and (e) total time to Execute the VM consolidation by using all VM selection policies with IQR.

Case 4: Local Regression (LR) as a hot spot detection algorithm along with five different VM selection policies.

Here we have used LR- Linear Regression based over utilized host detection method and calculate the results of VM consolidation process with the existing VM selection methods and with our proposed method. Table 3.11 shows the results of LR\AHP, LR\MU, LR\MMT, LR\MC and LR\RS along with its graphical representation in figure 3.8

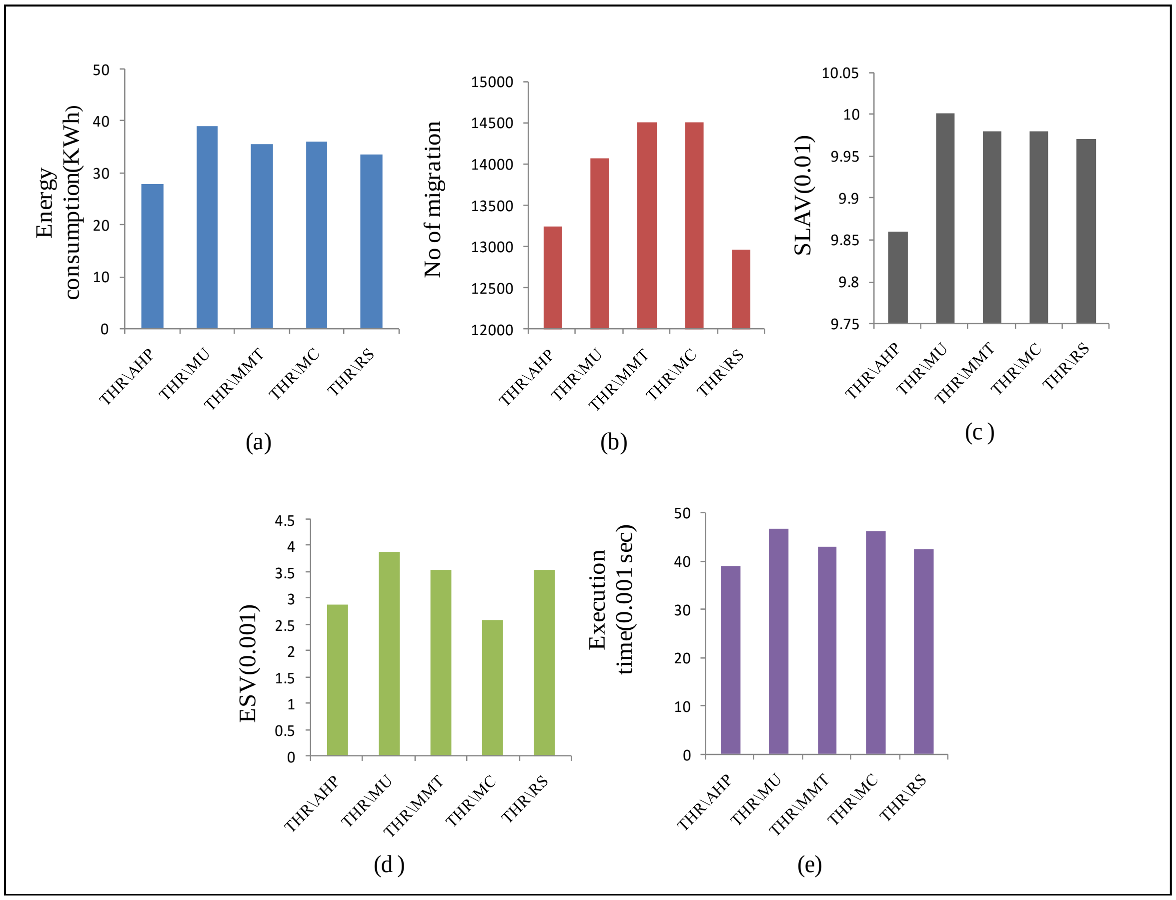


Figure 3.7: (a) Energy consumed by all VM selection policies. (b) Migration count during THR. (c) Level of SLA violation caused by several VM selection policies. (d) Overall ESV metric and (e) total time to Execute the VM consolidation by using all VM selection policies with THR.

Table 3.11: Comparative results using several VM selection policies with LR

Policy	**Energy consumption (kWh)**	**No. of migration**	**SLAV 10^{-2}**	**ESV 10^{-3}**	**Execution time (Sec)10^{-3}**
LR\AHP	28.57	10086	9.76	283.01	71.94
LR\MU	34.71	14592	9.81	337	95.87
LR\MMT	32.08	14234	9.82	312.7	84.98
LR\MC	32.01	14233	9.80	312.10	85.16
LR\RS	29.77	13562	9.78	284.26	84.91

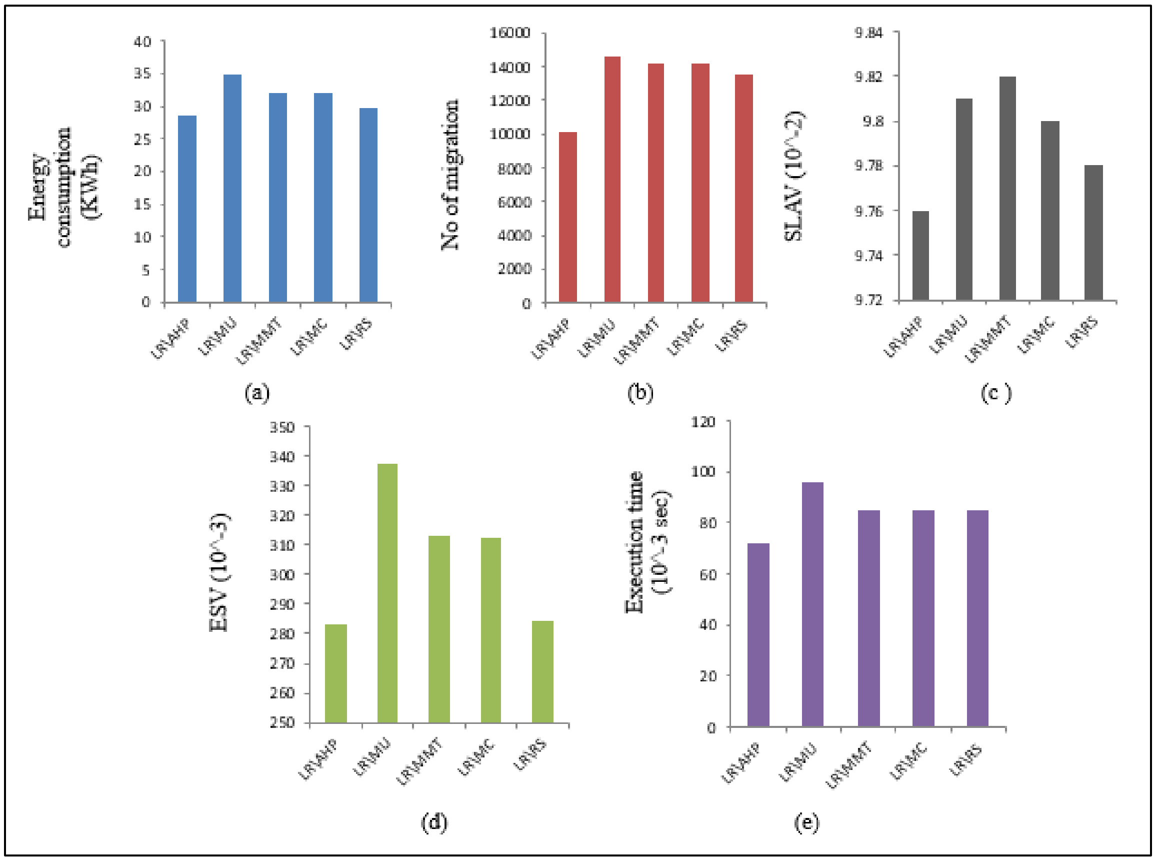


Figure 3.8: (a) Energy consumed by all VM selection policies. (b) Migration count during LR. (c) Level of SLA violation caused by several VM selection policies. (d) Overall ESV metric and (e) total time to Execute the VM consolidation by using all VM selection policies with LR.

From above figures and tables, it has been clear that the use of AHP VM selection policy provides better performance in terms of the energy consumption, number of migrations performed by virtual machine, the level of SLA violation during VM consolidation, overall ESV metric as well as execution time i.e. total time taken for the whole process of VM consolidation.

3.7 SUMMARY

In this chapter, we provide a solution for first two steps of VM consolidation process. We proposed a new method for auto-adjustment of lower and upper threshold values for selection of over utilized and under-utilized servers. We have conducted several experiments for both random and real workload environment for analyzing the performance of proposed approach and results shows that our proposed method provides a minimum level of SLA violation. Now, the main objective is the minimization of energy consumption, therefore, for solving the problem of VM selection we proposed Analytic hierarchy process (AHP) for VM selection which is a multi-criteria decision-making process.

This AHP VM selection method selects the VMs by using the migration time of the VM, by using the CPU utilization of VM along with the memory occupied by VMs. Again, the experiments have been conducted using CloudSim for the performance evaluation of the algorithms. From the experimental results, it has been clear that the proposed AHP method decreases the energy consumption along with other parameters such as: number of VM migrations, level of SLA violation, ESV metric and total time to complete the process. More precisely, we have compared the AHP method of VM selection with MU, MC, MMT and RS by using IQR, LR, MAD and THR as a host detection method. We have concluded that the adoption of AHP VM selection method has reduce the average energy consumption up to 31%, 27%, 27%, and 14% respectively for all four cases.

CHAPTER 4

SLA AND PERFORMANCE EFFICIENT HEURISTICS FOR VM PLACEMENT

This chapter provides three different heuristics for solving the third step of VM consolidation process i.e. the problem of VM placement. Here, the problem has been solved using the basics of classical approach of bin packing algorithms.

4.1 INTRODUCTION

Cloud computing has transformed the working culture of IT company's due to which the demand for cloud resources has been increased in last few years which further raised the level of energy consumption of data centers. This issue of inefficient energy consumption can be resolved by using the features of virtualization technology. Virtualization is the backbone of this cloud environment which makes the cloud resources available to cloud users. It lets the user to use the resources by dividing single physical machine into several virtual machines, in which resources are in the form of logical or virtual. Thus, by generating several VMs instances on a single server, resource utilization of the data centers can be improved. These virtual machines will be placed over some another machine without disturbing the current running applications. This method improves the resource utilization and minimization of the energy consumption, but the use of large numbers of system's resources and migration of the VMs may cause SLA violations. Therefore, there must be some appropriate policies that can minimize the migration count of VMs during VM placement. It is very important for the cloud environment to deliver the reliable QoS-Quality of Services mentioned in the agreement that has been signed between service provider and user known as Service Level Agreement (SLA), so that cloud service providers can efficiently deal with energy-performance trade-offs.

Virtual machine placement can be understood by different aspects in the dynamic consolidation of virtual machines. The algorithms of VM placement are categorized into QoS based approach and power based approach. Besides, each approach is separated into static and dynamic placement. VM placement problem has been solved as a bin packing in which items such as virtual machines are to be packed into variable bins such as physical machines such that minimum number of bins could be used to maintain the infrastructure cost.

4.2 RELATED WORK

Anton et al. [10] separated the VM consolidation into several steps: first is to select the under and overutilized host from data center, second is the selection of VMs from selected hosts for migration and lastly to design a new placement policy for selected virtual machines. Several researchers have presented their ideas for the energy minimization of data centers along with the several steps for efficient performance of data center by improving the various phases of the VM consolidation. Anton et al. used the concept of bin packing method for the mapping of virtual machines over the host by modifying the best fit decreasing heuristic which is known as MBFD. It allocates the virtual machine to that host which has the smallest increase in power consumption after utilization. Related to this method Hung et al. [13] proposed EPOBF heuristic (Energy aware and performance per watt) for the selection of most efficient host for mapping each virtual machine. EPOBF provides the VM-PM mapping on the basis of the ratio of total sum of the MIPS of all cores of the host machine to the total power consumption of host at 100% utilization. In [109, 110] authors presented dynamic round robin (DRR) and Round Robin algorithm (RR) to consolidate virtual machines, according to these algorithms the host machine will not take new virtual machines, if it has already other virtual machines running over it. If these already running VMs are on same machine from longer time then that machine will migrate those virtual machines to some another host and will try to shut down that particular host machine. Hori et al. [6] presented new algorithm for VM placement on the basis of host utilization and minimum correlation. According to which, virtual machines will be migrated to that host where the value of correlation of the CPU utilization among the VMs presented in that host is minimum. Their proposed algorithm provided the trade-off values between energy consumption and performance. Guangjie et al. [11] also presented power aware and remaining utilization algorithm to map the virtual machines over host. They have also revealed the existence of a trade-off values between SLA violation and energy consumption, but VM consolidation can result high level of violation which affects the quality of the services that are mentioned in the SLA.

In above-mentioned literature, the authors have considered the energy minimization and SLAV as their primary objective. From the existing studies and their results, we have analyzed that SLA violation and Energy consumption are indirectly proportional to each other. Moreover, it is important to deliver the better QoS- Quality of Services to cloud users that has been mentioned in the Service Level Agreement (SLA). This SLA level should not be violated in order to maintain the level of trust between the cloud provider and user. Thus, in contrast to this, major objective is to design performance efficient consolidation without compromising the QoS with the minimization of SLA violation, minimization od performance degradation caused by excessive consolidation, reducing the count of extra migration and to minimize the execution time taken by consolidation process. Proposed heuristics for VM placement can strictly handle the QoS and Service Level Agreement (SLA) for heterogeneous data centers in order to maintain the trust level between cloud service providers and cloud service users.

4.3 PROPOSED SOLUTION FOR VM PLACEMENT

Throughout the process of VM consolidation, to select a suitable host for the mapping of VMs is very difficult process. Therefore, we should design some new placement approaches that can improve the utilization of resources without compromising the QoS- Quality of Services. Therefore, to accomplish this objective, we have designed the solution for VM placement problem which is based on the classical bin packing and its details are provided in the following sub section.

4.3.1 VM placement as a bin packing

The problem of bin packing is NP –hard with polynomial time approximation algorithm. It deals with the packing of several items of different sizes into several bins and for efficient packing there should be minimum number of bins to be used. This section shows the model of VM placement as bin packing where each physical machine (PM) represents the bins and virtual machine (VM) represents the items to be packed inside the bin. This VM placement model consists *M* virtual machines with *H* physical machines inside the data center where, *M* VMs can be represented as: $\{M_j\ (MIPS_j, BW_j, RAM_j, PE_{j,})\ |\ j = 1,2, \ldots.. M\}$ and *H* physical machines can be represented as: $\{\ H_i\ (MIPS_i, BW_{i,}\ RAM_{i,}\ PE_i)\ |\ i = 1,2 \ldots..\ H\ \}$. Each virtual machine VM_j needs few amount of $MIPS_j$ (millions instructions per second), network bandwidth (BW_j Kbits/s), RAM_j

(Mbytes of physical memory) and processing elements (PE_j). These resources are provided by the physical machines to virtual machines.

We have assumed the heterogeneous data center's architecture where the linear relationship exists between the power consumption (P) and CPU utilization (U) of every host at every single time frame t as shown in formula given below.

$$P\big(U_{CPU}(t)\big) = P_{idle} + \big(P_{max} - (P_{idle}).U_{CPU}(t)\big) \tag{4.1}$$

Though the objective is to lessen the number of active PMs inside the data center, we have used d_i as a decision variable for every host i which will be set as 1 if i is selected for the mapping of VMs otherwise d_i will be 0. Following equations 4.2- 4.4 shows the objective function with different constraints.

$$\min H = \sum_{i=1}^{H} d_i \tag{4.2}$$

Subject to fulfil the constraint for each VM

Most important constraint during VM mapping is that the sum of all the resources requested by VM should be equal to or less than the maximum available capacity of every host H_i (i = 1,2,….. H):

$$H_i\ (R_{CPU}, t_s)\ \geq\ H_i\ (T_{CPU}\ , t_s) - \sum_{j=1}^{M} VM_j\ \big(D_{CPU\ ,} t_s\big) \tag{4.3}$$

$$H_i\ (R_{ram}, t_s)\ \geq\ H_i\ (T_{ram}\ , t_s) - \sum_{j=1}^{M} VM_j\ \big(D_{ram\ ,} t_s\big) \tag{4.4}$$

In above equations R_{CPU} and R_{ram} represents the available CPU utilization and RAM of ith machine at time frame t_s, T_{CPU} and $T_{ram,}$ represents the total capacity of ith machine, D_{CPU} and D_{ram} represents the requested resources of CPU and RAM by jth VMs.

4.3.1.1 Proposed Heuristics for VM Placement ARBF- Available Resource Best Fit

For providing VM-PM mapping by using the basics of bin packing, we have proposed 3 different heuristics by considering 3 types of computing resources such as physical memory (RAM), MIPS and the power consumption of each host machine during VM placement. These three heuristics are also the modifications of BFD- Best Fit Decreasing and named as (ARBFH1) Available Resources Best Fit Heuristic 1, ARBFH2 (heuristic 2) and ARBFH3 (heuristic 3). Figure 4.1 shows the pictorial view of VM placement using above mentioned resources such as: U, P and M represents the current CPU utilization, rise in power consumption of host after allocation and available Physical memory. VM1, VM2……...VMm represents m different virtual machines. VM list contains the VMs that requires to perform migrations according to current and past utilization of resources.

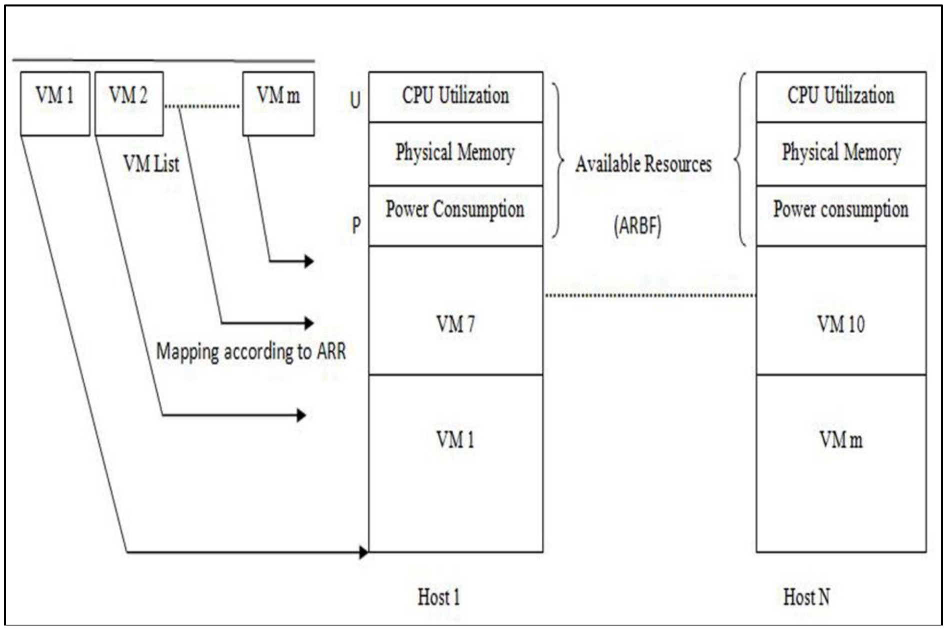


Figure 4.1: Available Resources Best Fit (ARBF) model for VM Placement

Available Resource Best Fit Heuristic 1

This heuristic allocates the host according to MR value with the criteria of maximum available remaining resources. H1 first starts with checking the host whether it has available MIPS and RAM to allocate for VM; if it has available resources, then, it will calculate the utilization factor of the host which is based on the available power of host after the mapping of VMs and total MIPS allocated for VM. As the VM-PM mapping will be conducted using MR (Maximum resources) value, therefore, this MR can be calculated using Utilization that has been calculated in the previous step along with the Available Power of the host. The pseudo-code of ARBF H1 is given below in Table 4.1.

Available Resource Best Fit Heuristic 2

This policy provides the mapping of the VMs by making the use of both current, past utilization and available memory (RAM) of each host machine. We have measured the utilization using both the current as well as MIPS utilization of the CPU in previous time frames. This algorithm starts with the checking of the availability of resources i.e. MIPS and RAM of the host. If the host has sufficient resources to fulfil the requirements of VM than for finding the appropriate host for the VM, utilization of the host will be calculated. Utilization in the case of ARBF H2 considered the previous utilization as well as the available MIPS of the host. Finally, ARBF H2 allocates the VM to host depending upon MR value. This MR value uses both the utilization factor and available RAM for VM-PM mapping. Table 4.2 shows the pseudo-code for ARBF H2.

Table 4.1: Pseudo code for Available Resource Best Fit Heuristic 1

```
1    Input: Host List, VM List
2    Output: Migration Map
3    For each VM in VM List do
4         Allocated Host = NULL
5         Available resource = Max value
6          For each Host in Host List do
7               If (availableMIPSofHost >= RequestedMIPSofVM &&
                 availableRAMofHost >= currentRequestedRamofVM);
8              Util = sqrt (availableMIPS – totalMIPSallocatedfovm)
9              MR = sqrt (Util/Available Power)
10                  If (MR <= Available resource)
11                     Available resource = MR
12                     Allocated Host = Host
13                 End if
14             End if
15             MigrationMap.add (vm, Allocated Host)
16        End for
17   End For
18   Return Migration Map
```

Available Resource Best Fit Heuristic 3

The utilization of the resources, as well as their utilization sequence and criteria along with their effects on power consumption, are the main factors that affect the performance of the system during VM placement. Here in our proposed heuristics, we have considered all these resources one by one or together some time. Moreover, we have also considered their previous and current utilization. ARBF H3 will consider both the current state and past utilization of resources such as MIPS, RAM and power consumption of the host.

Table 4.2: Pseudo code for Available Resource Best Fit Heuristic 2

```
1    Input: Host List, VM List
2    Output: Migration Map
3      For each VM in VM List do
4          Allocated Host = NULL
5          Available resource = Max value
6              For each Host in Host List do
7                  If (availableMIPSofHost >= RequestedMIPSofVM &&
                   availableRAMofHost >= currentRequestedRamofVM);
8                  Util = sqrt (previousUtilizationofMIPSofHost –
                            AvailableMIPSofHost);
9                  AvailRAM = UtilizationofRAMofHost –
                                  CurrentRequestedRAMofVM;
10                 MR = sqrt (AvailRAM + Util)
11                     If (MR <= Available resource)
12                         Available resource = MR
13                         Allocated Host = Host
14                     End if
15                 End if
16             MigrationMap.add (vm, Allocated Host)
17       End for
18   Return Migration Map
```

The consideration of previous utilized resources predicts the behavior of resources in advance and avoid the situation of system load imbalance. The lesser the utilization of CPU, the lesser will be the power consumption, as they both have a direct relationship and accordingly the MR value will be used to find the most suitable host for VM.

4.4 RESULTS AND DISCUSSION

4.4.1 Data center architecture and performance metrics

This section describes the data center's architecture that we have used in our work. Heterogeneous data center has been chosen with H heterogeneous machines consists of different computing resources such as: physical memory, CPU utilization in MIPS, processing elements and network bandwidth. Moreover, these physical machines also contain M heterogeneous VMs (virtual machines) that are also characterized in terms of some computing resources.

Table 4.3: Pseudo code for Available Resource Best Fit Heuristic 3

```
Input: Host List, VM List
Output: Migration Map
 For each VM in VM List do
     Allocated Host = NULL
     Available resource = Max value
         For each Host in Host List do
             If (availableMIPSofHost >= RequestedMIPSofVM &&
                 availableRAMofHost >= currentRequestedRamofVM);
         MR = sqrt
             AvailableUtilizationofMIPSofHost/previousUtilizationofMIPS)
                 If (MR <= Available resource)
                     Available resource = MR
                     Allocated Host = Host
                 End if
             End if
           MigrationMap.add (vm, Allocated Host)
       End for
   End For
Return Migration Map
```

we have used six different types of servers with the different configuration shown in table 3.1 of chapter 3. Each CPU has c cores and each core has m MIPS, therefore total MIPS of CPU is c * m. These six different servers are: AcerAT150 F1, AcerAR320 F1, IBMX3250 XeonX3480, IBMX3250 XeonX3470, HP ProLiantG4 Xeon 3075 and HP ProLiantG4 Xeon3040. We assessed our proposed heuristics in CloudSim that enables simulation and modelling of cloud computing systems. 800 heterogeneous hosts are used to conduct the experiments with four different types of VM instances (High CPU medium instance, small instance, Micro instance and Extra-large instance) are used from Amazon EC2. Here also, we have simulated the experiments using the data provided by PlanetLab using its project for monitoring infrastructure i.e. CoMon project. QoS that we have guaranteed to provide during the VM placement are mentioned in the form of Service level agreement. The concept of energy minimization is very popular nowadays but, the over provisioning of the resources can increase the level of SLA violation and degrade the performance due to excessive migration. The SLA violation (SLAV) can be calculated using two metrics: degradation of performance due to the migration of host (PDM) and the percentage of time when an active host reaches to 100% CPU utilization (SLATAH) defined in equation 3.2 and 3.3 of chapter 3. But in order to assess the performance of proposed heuristics for VM placement, we have used an ESM metric-Energy SLA migration introduced in [65] to make our results comparable with some benchmarks algorithms presented with them and the one defined in [13]. This ESM metric is shown as below equation 4.5:

$$ESM = Energy * SLAV * Number\ of\ migration \tag{4.5}$$

4.4.2 Results and discussions

VM consolidation contains different steps that we have already defined in chapter 3. In this chapter, the combination of best policies presented by Anton et al. [10] are considered as a reference scenario for the comparison. MMT (Minimum migration time), LR (Linear regression) and PABFD are compared to the scenario described in [13] and with our proposed heuristics. Five different performance metrics that may affect from these policies are 1) Migration count, 2) SLA violation, 3) PDM, 4) Energy consumption and 5) Total time taken by consolidation process. But in order to compute the total effect of each policy, the ESM parameter will be calculated and used to find out the best one.

Table 4.4: Comparative results for different VM placement policies

Policy	**SLAV**	**Number of Migrations**	**PDM**	**Execution Time (Sec)**	**Energy consumption (KWh)**	**ESM**
PABFD[10]	0.00242	13448	0.04	0.07009	30.01	976.6
BFD [5]	0.00072	8759	0.02	0.7696	24.39	153.8
EPOBFD[13]	0.00094	13647	0.04	0.11015	25.5	327.1
ARBF H1	0.00035	8235	0.02	0.07707	51	146.9
ARBF H2	0.00033	7283	0.015	0.0635	48	115.3
ARBF H3	0.00065	3646	0.01	0.05976	30.35	71.9

We have done the simulations for the 10 days on ten different workloads provided by COMON project [17] and result of their mean values for SLAV, PDM, numbers of migrations, Energy consumption as well as ESM metric are shown in Table 4.4. The percentage of SLA violations due to overutilization of host and performance degradation due to migration are shown in figure 4.2. Figure 4.3 depicts the count of VM migrations takes place during VM placement; Figure 4.4 shows the percentage of performance degradation during the simulation; Figure 4.5 depicts the average value for the total energy consumed by data center; Figure 4.6 shows the execution time taken by all the policies for the placement of VMs; and Figure 8 depicts the ESM metric which is used to measure the overall performance with effect of energy consumption, SLAV and number of migrations.

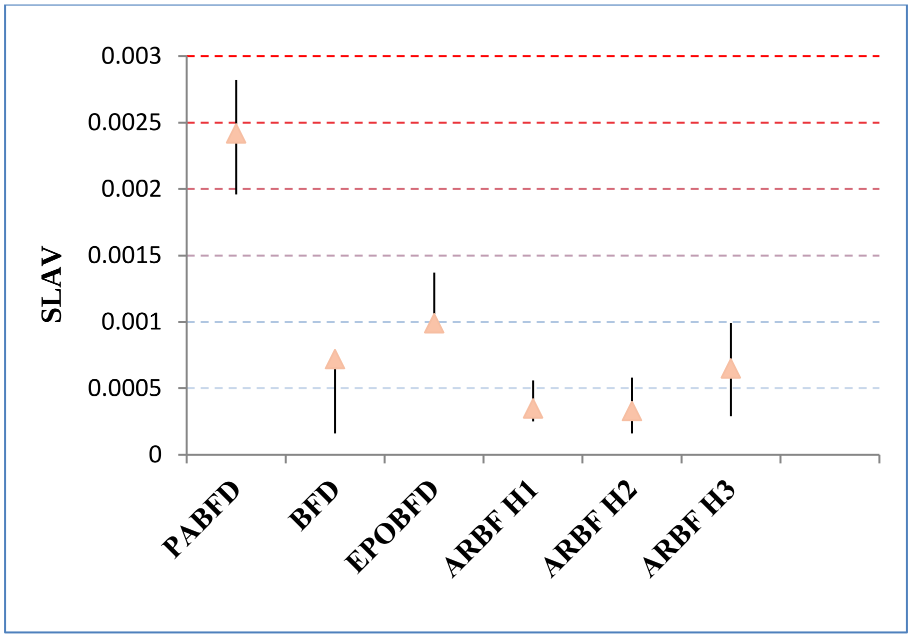


Figure 4.2: SLA violations occurs using several VM Placement policies

Figure 4.2 shows that PABFD provides the maximum level of SLA violation. This SLA violation depends upon over utilization of host and can be calculated as the SLA time per active host with the percentage of performance degradation during migration. Lesser the migration during placement, lesser will be the performance degradation which provides the lesser increase in SLA violation. Moreover, the PABFD and EPOBFD have used only the power consumption of host as a resource for the mapping of VMs therefore; there may be chances that VMs are not provided with the required level of performance. ARB H1 provides a minimum level of SLA violation because of the resources they have used. ARBF H1 used MIPS utilization as well as power consumption both for the mapping of VMs over physical machines.

As depicted in Figures 4.2-4.7, the use of ARBF H1, H2 and H3 provides better performance in terms of SLA violation, migration count, execution time and PDM respectively in contrast to another heuristics. However, these ARBF heuristics consume more energy but simultaneously delivers a good level of ESM metric by reducing the level migration count and SLA violations during simulation.

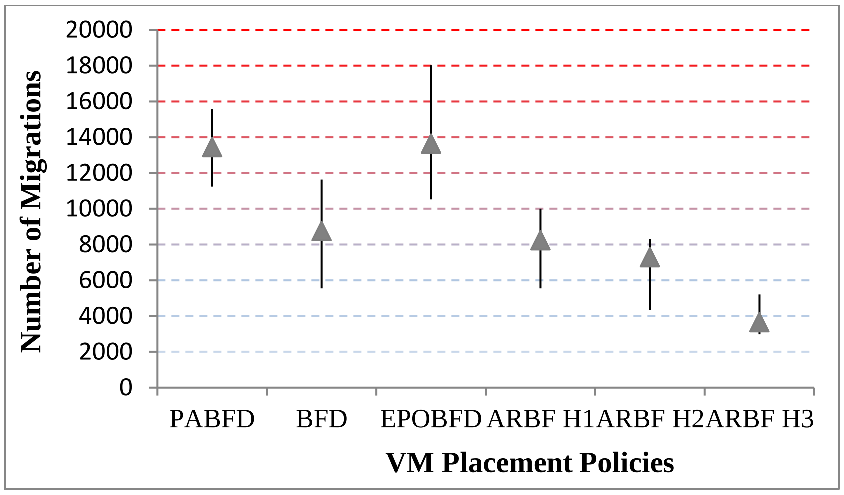


Figure 4.3: Migration count using several VM placement policies

Performance degradation during the Placement depends upon the degradation caused by VM which is estimated as the 10% of CPU utilization during the migration. Thus, the performance degradation reflects the requested resources of VM. ARBF H3 provides minimum performance degradation because of the impact of the previous utilization of MIPS during the allocation of the host.

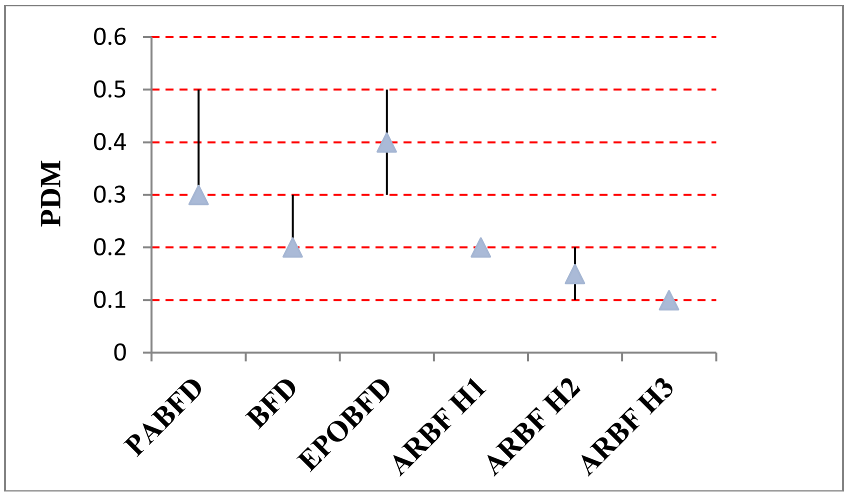


Figure 4.4: Performance degradation during migration using different policies

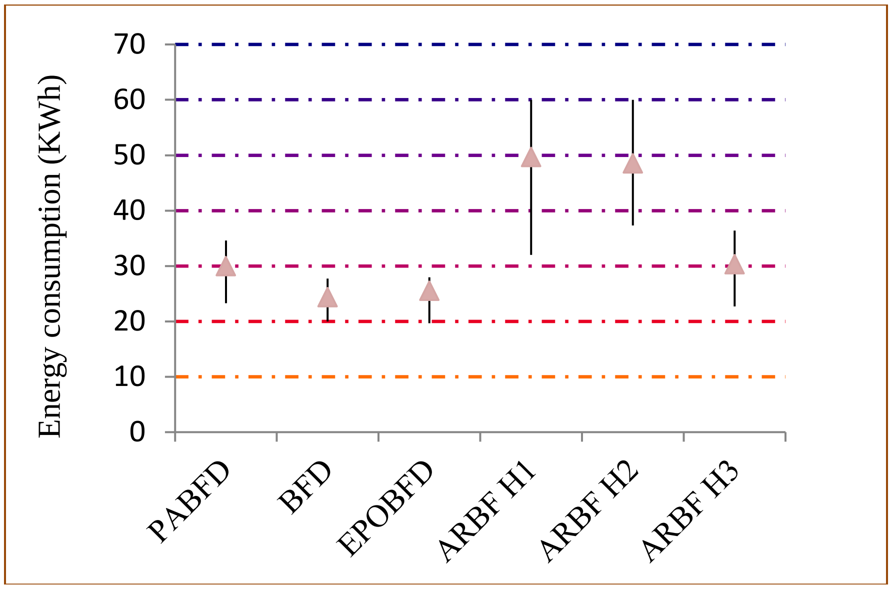


Figure 4.5: Energy consumed by data center during VM placement

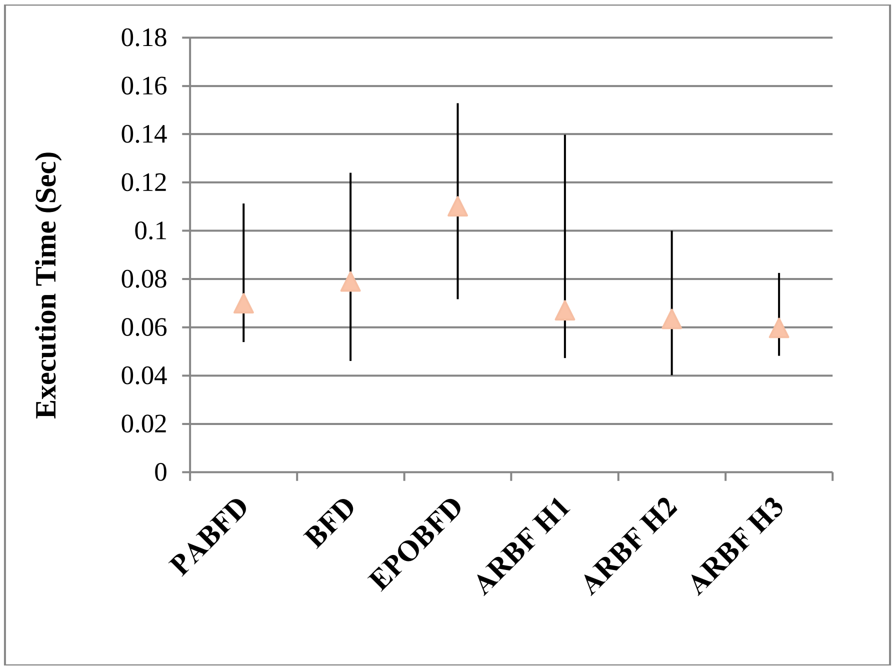


Figure 4.6: Execution time taken by different policies during VM placement

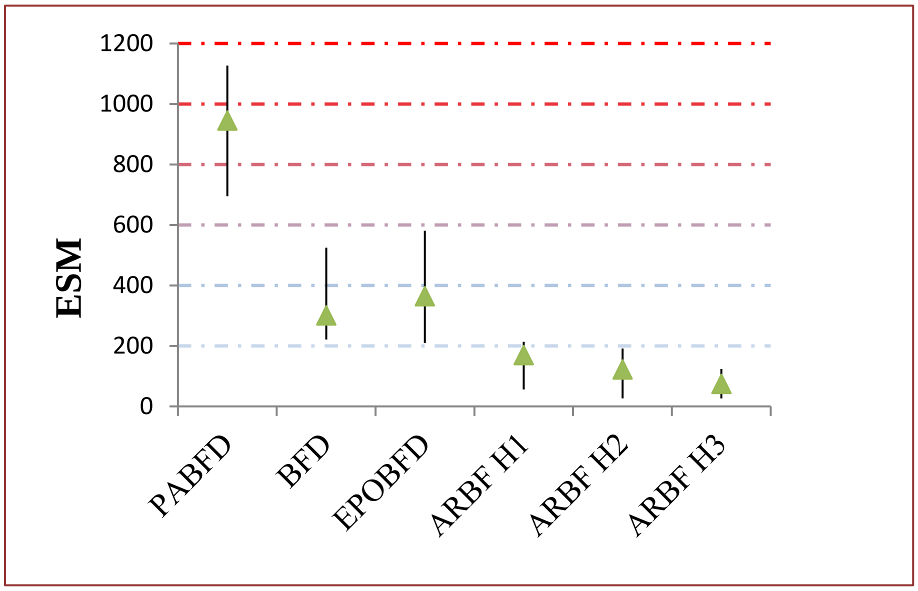


Figure 4.7: Results of ESM for different policies of VM placement

Furthermore, as depicted in Table 4.5, the adoption of ARBF H1, H2, H3 leads to the best ESM improvement as compared to PABFD with 84.9%, 88.9% and 92.6% respectively. ARBF H1, H2, H3 provides ESM improvement by 44.79%, 42.7% and 53.25% respectively and similarly, ARBF H1, H2, H3 leads with 52.9%, 64.7% and 78.01% as compared to EPOBFD.

Table 4.5: Comparative results for improvement of proposed heuristics

Policy	**ESM**	**% (improvement in comparison to PABFD [10])**	**% (improvement in comparison to BFD [5])**	**% (improvement in comparison to EPOBFD [13])**
ARBF H1	146.9	84.9	44.79	52.9
ARBF H2	115.3	88.1	42.7	64.7
ARBF H3	71.9	92.6	53.25	78.01

4.4.3 Statistical analysis

This section presents the statistical analysis of the proposed and benchmark algorithms. According to Ryan-Joiners normality test, ESM values of all these scenarios (LR/MMT/PABFD, LR/MMT/BFD, LR/MMT/EPOBFD, LR/MMT/ARBF H1, LR/MMT/ARBF H2 and LR/MMT/ARBF H3) follows a normal distribution with P value > 0.1. Then we have conducted paired T-test to determine the VM placement policy that minimizes the ESM metric across all algorithm combinations. Table 4.6 provides the results based on paired t- tests for all the scenarios. The T-tests have shown the adoption of LR/MMT/ARBF H1, H2, and H3 leads to the lower value of ESM metric with P-value < 0.001. From the results, we can conclude that our proposed heuristics have the best performance regarding ESM metric. Table 4.7 compares the benchmark algorithms and proposed heuristics regarding the mean values of ESM with 95% confidence interval. From obtained results, it can be concluded that LR/MMT/ARBF H3 has the best performance regarding ESM metric, then ARBF H2 and ARBF H1 has next best performance respectively

Table 4.6: Comparison of all Policies using paired t -tests

Policy 1 (ESM)	Policy 2 (ESM)	Difference	P-value
LR/MMT/PABFD (976.6)	LR/MMT/ARBF H1 (146.9)	808.5(962.3,153.8)	P-value <.001
LR/MMT/BFD (153.8)	LR/MMT/ARBF H1 (146.9)	185(339, 154)	P-value <.001
LR/MMT/EPOBFD (327.1)	LR/MMT/ARBF H1 (146.9)	205.2(359.0,153.8)	P-value <.001
LR/MMT/PABFD (976.6)	LR/MMT/ARBF H2 (115.3)	839.2(962.3,123.2)	P-value <.001
LR/MMT/BFD (153.8)	LR/MMT/ARBF H2 (115.3)	216(339,123)	P-value <.001
LR/MMT/EPOBFD (327.1)	LR/MMT/ARBF H2 (115.3)	235.8(359.0,1232.2)	P-value <.001
LR/MMT/PABFD (976.6)	LR/MMT/ARBF H3 (71.9)	889.4(962.3,72.9)	P-value <.001
LR/MMT/BFD (153.8)	LR/MMT/ARBF H3 (71.9)	266(339,73)	P-value <.001
LR/MMT/EPOBFD (327.1)	LR/MMT/ARBF H3 (71.9)	286.1(359.0,72.9)	P-value <.001

Table 4.7: Comparison of benchmark algorithms with proposed heuristics regarding ESM metric

Policy	ESM	CI (95%)
LR/MMT/PABFD [10]	962.32	(820.6, 1104.0)
LR/MMT/BFD [5]	338.79	(3,675)
LR/MMT/EPOBFD [13]	358.9	(249.1,468.9)
LR/MMT/ARBF H1	153.8	(109.5,198.5)
LR/MMT/ARBF H2	123.17	(78.9, 167.5)
LR/MMT/ARBF H3	72.8	(49.93, 96.03)

4.5 SUMMARY

With the world-wide increase in demand of data centers, the problem of high energy consumption and carbon emission is very popular these days and several studies or theories have been provided to tackle this problem and one of them is VM consolidation. It consolidates the VMs to reduce the energy consumption by performing its different steps. It involves migrations and consolidations of VMs over lesser number of machines. VM consolidation may affect the performance of systems and level of trust between cloud service providers and users by not providing the guaranteed QoS thus, in this chapter our main aim was the improvement of the performance of cloud data centers by improving the ESM metric, SLA violations and number of migrations. This chapter has proposed ARBF heuristics- H1, H2, H3 as a new VM placement heuristics for VM placement. These heuristics select a host for the mapping of virtual machines on the basis of current and past utilization of resources. We have calculated the results of experiments using open simulation framework CloudSim and concluded that proposed heuristics provides better results of the data center's performance with 88%, 84%, 92% improvement in ESM metric.

CHAPTER 5

META HEAURISTICS APPROACH FOR VM PLACEMENT

This chapter provides the meta-heuristic-based solution for VM placement. First, we will discuss the trend of heuristic as well as meta-heuristic approaches and how these both affect the performance of data center. Later on, we will provide nature inspired algorithm for VM placement

5.1 INTRODUCTION

The concept of cloud computing has been defined by Buyya [112]. It is a parallel and distributed system with several virtualized computers where the provisioning of cloud resources is provided according to the mentioned SLA (Service Level Agreement) that has been signed between cloud service user and service provider. Virtualization is one of the main characteristics of cloud environment and it provides elasticity to the environment and helps to minimize the cost of data center's infrastructure. It is also an efficient technology for the sharing of resource with in a cloud environment. One of the virtual machine manager such as hypervisor gives the permission for several OS- operating systems to run on single physical machine. The allocation of several virtual machines on a single host can minimize the cost of infrastructure as well as energy consumed by data center. Thus, the objective of current research is to minimize the energy consumption of datacenter and performance of cloud environment. Our study emphases on the sub part of VM consolidation process which is VM placement process for the management of energy efficient resources. It also helps for the mapping of VMs to most suitable servers. VM placement tries to find most optimam solutions for VM-PM mapping. Thus, there are several techniques to handle this problem such as nature inspired algorithms, bin packing algorithms (already discussed in chapter 4), bin-packing techniques, linear integer programming, constraint programming and many more. In this chapter, we have performed the evaluation of various bin packing methods of VM placement and compared them with our proposed one to understand the trend of VM-PM mapping and how it affects the overall performance of cloud environment. A decision-making system for the placements of VMs using basic Genetic algorithms has been presented by us in this chapter. Their comparison gave us a new direction to study different evolutionary algorithms and thus, we proposed a novel VM placement algorithm using NSGA- Non-Dominated Sorting Genetic Algorithm.

5.2 RELATED WORK

VM placement is a NP hard problem and multi-objective optimization problem which can be solved by both meta-heuristic and heuristic techniques such that: linear programming, Bin packing, constraint programming and using Evolutionary algorithms. Paolo et al. [77] presented the concept of genetic algorithm for VM allocation in distributed systems**.** Though, the traditional algorithms such as bin packing, linear integer programming and many others were not able to offer optimal solutions therefore, several algorithms have been proposed to provide optimal solution for VM placement. Bandi et al. [79] also used genetic algorithm for solving VM placement and minimizing the energy consumption by making the use of current requested resources of VMs as well as the usage of PMS in previous time frame. Shi Chen et al. [80] proposed hybrid genetic algorithm (HGA) which is also the combination of knapsack problem and used the concept of multiple fitness functions for the verification and the effectiveness of their proposed algorithm. Many researchers [83-84] presented GGA (Grouping genetic algorithm) to attain more efficient results and they were better than non-grouping algorithms but they didn’t consider the relationship between VMs, servers and data centers. Therefore, Fereydon et al. [85] also used GGA as a reference algorithm and planned MLGGA (Multi-Level Grouping Genetic Algorithm) to find relationship between every individual and their groups as well. Yu-Shuang. D et al. [86] proposed another approach for VM placement by using the concept of genetic algorithm. They proposed (DPGA) Distributed Parallel Genetic Algorithm which executes in two stages according to which, the initial population is selected from the solution space in the first stage by implementing GA parallely on multiple hosts and solutions obtained from the first stage will become the original population for the second stage thus, the optimal solutions can be found. Joseph C.T et al. [87] proposed a novel approach for VM placement i.e. FGA (Family Genetic Algorithm) which divides the set of population into different families and improved the speed of GA by processing each family parallely and this way, FGA decrease the total execution time of VM placement algorithm.

Evolutionary algorithms are commonly adopted now a day to attain optimal solutions. This study compares the various bin packing PABD, EPOBF and MWFD algorithms of VM placement with the basic genetic algorithm, and analyzed that how this placement affects the performance of cloud environment. Finally, the aforementioned points help to design new VM placement algorithm so that minimization of energy can be achieved using best optimal solution. Genetic Algorithms (GA) are the main class of nature inspired techniques for VM placement thus, we have also proposed a novel approach for VM placement by making the use of non-dominated sorting genetic algorithm (NSGA) which is also the variation of GA and used to deal with the limitations of current techniques to provide optimal results than existing ones

5.3 PROPOSED DECISION-MAKING MODEL FOR VM PLACEMENT

For data center's performance improvement, the VM placement system that we have considered uses the CPU utilization, several PEs (Programming Elements), power consumption of host after VM-PM mapping. It involves some kind of decision-making process for efficient mapping. The proposed model of VM placement has been designed using decision-making system shown in Figure 5.1. These decision-making system can be modelled using linear programming, evolutionary algorithms, neural networks and many others. Evolutionary algorithms provide optimal results for the searching of suitable host for VMs than classical bin packing methods but they take more computational time for the process of searching. Meta-heuristics algorithms such as GA are beneficial to use when it is hard to find some optimal solutions because they have large search space and have the capability to adjust their search space automatically, where as traditional methods don't have. Genetic algorithms start with the random set of populations. As, GA computes the effects that the system has, after the deployment of new VM resources, then GA will choose the set of solutions which will have least effects on the system using its operators, by doing so GA provides better VM placement and accuracy in comparison to existing technologies.

5.3.1 Modeling of GA-VMP (Genetic Algorithm Virtual Machine Placement)

To solve the problem of a multi objective optimization of VM placement, our main objective is the mapping of VMs to the physical host using optimal placement. Virtual machines from 1 to M and physical hosts from 1 to P can be represented as:

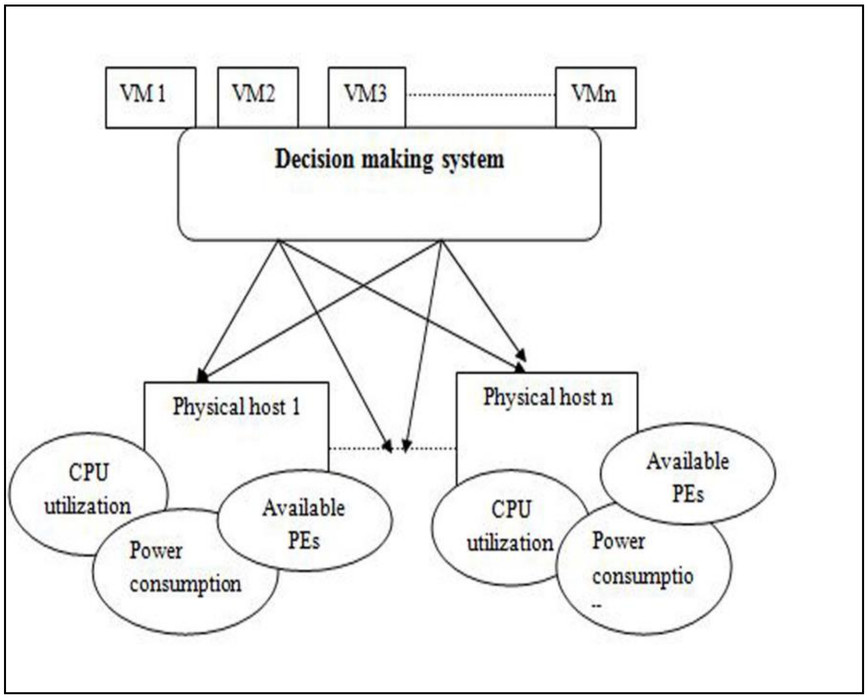


Figure 5.1: VM Placement using Decision making model

$$PM_p = \{CPU_p, MEM_p, PE_p, BW_p\} \tag{5.1}$$

$$VM_m = \{CPU_m, MEM_m, PE_m, BW_m \tag{5.2}$$

From above equation CPU_p is the available processing power of machine and CPU_m is the requested processing power of VMs, the total amount of memory available at host and the total amount of memory requested by VM is represented by MEM_p, MEM_m respectively. Similarly, the number of available programming elements and available amount of bandwidth inside the host is represented by PE_p, BW_p whereas the requested programming elements and network bandwidth by VMs is represented by PE_m , BW_m respectively. The main focus of optimal placement is to maximize the performance of cloud environment and it can be reached by using the following constraints shown in equations 5.4 – 5.7. The value of decision variable X_{ij} is dependent on i and j i.e. it would be 1 if VM i is mapped over PM j otherwise 0. Requested resources of selected VMs should be less than the available resources of host and moreover, the placement of VMs should provide minimum increase in power consumption.

$$\text{Objective: Maximize \{Performance of datacenter\}} \tag{5.3}$$

Subject to fulfill the constraints:

$$\forall_i \sum_{j=1}^{p} x_{ij} = 1 \; \{ \; if \; i \; VM \; is \; allocated \; to \; j \; host \; machine, otherwise \; 0 \; \} \quad (5.4)$$

$$\forall_j \sum_{i=1}^{m} X_{ij} \, . VM_i^{CPU} \leq \; PM_j^{CPU} \quad (5.5)$$

$$\forall_i \sum_{j=1}^{p} X_{ij} \; (Min \; (Power \; consumption)) \quad (5.6)$$

$$\forall_i \; \sum_{j=1}^{p} X_{ij} \; \big(Max(Available \; PEs)\big) \quad (5.7)$$

GA starts with the randomly generated initial set of population i.e. random VM-PM map. Each candidate set of the population represents the mapping of VM and PM present inside data centers. From each population set, the algorithm will select two individuals of the population using tournament selection operator and then perform genetic operations such as crossover and mutation on them. Population set is exchanged by the resulting mapping solution of VMs and physical machines or new individuals that we have obtained after performing genetic operations. This process will be repeated until we obtained the best VM-PM map based on the fitness value, which is nothing but the estimated values of performance metrics. Steps for VM placement using Genetic Algorithm are given below:

```
Step 1: Initialization of the parameters of GA (mp, p, N – mutation, crossover rate and iterations)
Step 2: Random placement all the VMs over host (random initialization of population)
        a) Do
        b) To calculate the fitness value for each chromosome from the population set
        c) Select individuals P1and P2 from population
        d) Perform Mutation (p1, p2, mp)
        e) Perform Crossover (P1, P2, cp)
        f) Replace the population set
        g) Until (best VM-PM map found)
Step 3: Repetition of step 2 util N number of iterations achieved
Step 4: Return VM-PM placement map
```

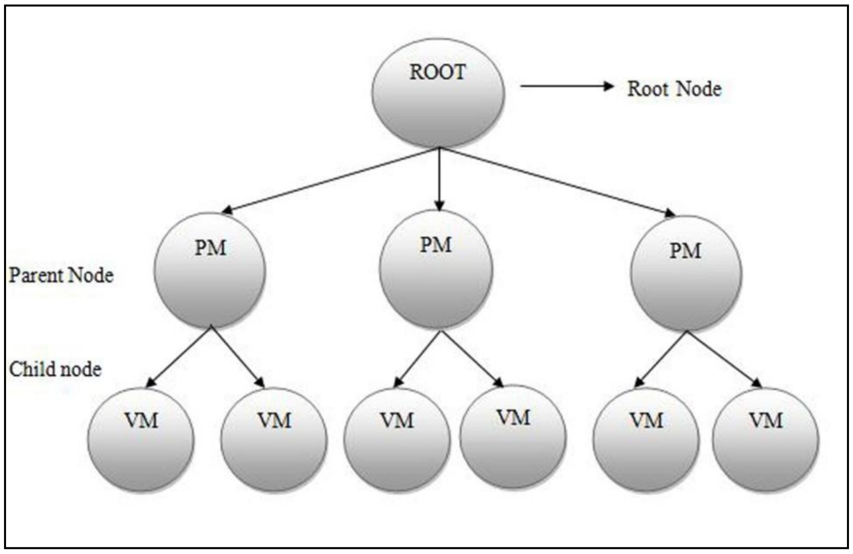


Figure 5.2: Representation of initial population using Tree structure

Figure 5.2 shows the tree like structure or representation of VM placement where every individual or chromosome of the population set is represented as a tree with root node as a global resource manager to monitor the utilization of host machine. It helps for the selection of most suitable host for placement. The next level of root represents the physical machines along with the virtual machines as a child node of the tree.

5.3.2 Performance evaluation and results

To evaluate the performance of VM placement techniques, here we have considered 700 heterogeneous nodes with different CPU model, cores, RAM and different frequency (MHZ) using CloudSim environment. Number of virtual machines that we have used are varying according to dataset provided by PlanetLab [17] and uses random workload conditions of CPU utilization. We have taken six different types of host or servers for our simulations: IBM Server x3550, IBM Server x3250, HP ProLiant G5, HP ProLiant G4, Acer AR320 F1 and Acer AT150 F1. After considering this framework, we have evaluated the performance metrics and analyze the results of experiments that have been obtained on 4 different days with a different number of virtual machines shown in table 5.1. These performance parameters are SLA violation and energy consumption. We have taken the instances of four different types of VMs from Amazon EC2.

These instances are: (3000, 2500, 2000, 1000, 500 MIPS) CPU utilization and (6000, 850, 3750, 1700, 633 GB) RAM.

The process of VM consolidation may degrade the performance of datacenter due to the unnecessary migration of virtual machines which may further increase the level of SLA violations. Therefore, the best VM placement policy will be the one which avoids the unnecessary migrations by providing least SLA violations with minimum energy consumption. Also, the SLA violation may depend upon the degradation of the performance due to VM migration along with the SLA time per active host which represents the percentage of time when CPU utilization reaches to 100% defined in equation 3.2 of chapter 3. As the term of SLA violations and energy consumption are interrelated to each other, therefore, we have used both these factors to compare the performance of data center.

We have simulated the VM placement techniques using CloudSim and analyzed the results of three different bin packing by comparing them with our proposed genetic algorithm decision-making system. We have conducted several experiments using real workloads conditions for their detail analysis. Two performance metrics have been chosen for the comparison or analysis of four different VM placement policies. As, there are many alternatives for the selection of host for the allocation of VMs and selection of VMs for migration but here, we execute LR (Linear regression) policy for host allocation and VMs are to be selected using minimum migration time policy (MMT) proposed by Anton et al. [10]. From literature LR/MMT has been found as the best combination for VM consolidation, thus we have also used this combination during different VM placement techniques. Four different policies are here for VM placement: 1) PBFD, 2) EPOBF 3) SWFDP 4) GA and their results are varying according to the number of VMs used on different days and thus, mentioned in four different tables given below.

Table 5.1: Number of VMs used on four different dates

Dates	No of VMs
9-04-2011	1358
11-04-2011	1233
12-04-2011	1054
20-04-2011	1033

Table 5.2: Energy consumption and SLAV for dataset of (9th April 2011)

Policies	Energy consumption (KWh)	SLAV (%age)
PABFD	34.61	0.00195
EPOBF	32.95	0.00072
SWFDP	93.01	0.00296
GA	28.4	0.00052

Above table 5.2 illustrates the SLA violations and energy consumed by datacenter during VM placement. These results from simulations are obtained for 4 different policies using the data set of PlanetLab which contains 1,358 VMs on 9th April 2011 and similarly, table 5.3 shows the same for 11th April 2011 by considering 1,233 VMs and table 5.4, 5.5 also represents the simulation results for 12th and 20th April using 1,054 and 1,033 VMs.

Table 5.3: Energy consumption and SLAV for dataset of (11th April 2011)

Policies	Energy consumption (KWh)	SLAV (%age)
PABFD	27.63	0.0027
EPOBF	26.74	0.00091
SWFDP	86.1	0.0097
GA	22.2	0.00063

Table 5.4: Energy consumption and SLAV for dataset of (12th April 2011)

Policies	Energy consumption (KWh)	SLAV (%age)
PABFD	29.8	0.00215
EPOBF	28.33	0.00088
SWFDP	89.5	0.0078
GA	24.4	0.000491

Table 5.5: Energy consumption and SLAV for dataset of (20th April 2011)

Policies	Energy consumption (KWh)	SLAV (%age)
PABFD	25.90	0.00281
EPOBF	22.66	0.00107
SWFDP	71.2	0.0102
GA	18.8	0.00048

SWFDP gives exactly opposite results than best-fit placement. It simply shows the worst-case scenario with the maximum amount of energy consumption and SLA violations during placement. Moreover, PABFD has been extended from best fit decreasing, but it gives maximum values for both the energy consumption and level of SLA violation in comparison to both EPOBF and GA. EPOBF provides the second highest level of energy consumption and SLA violations whereas, GA provides minimum values for both of these.

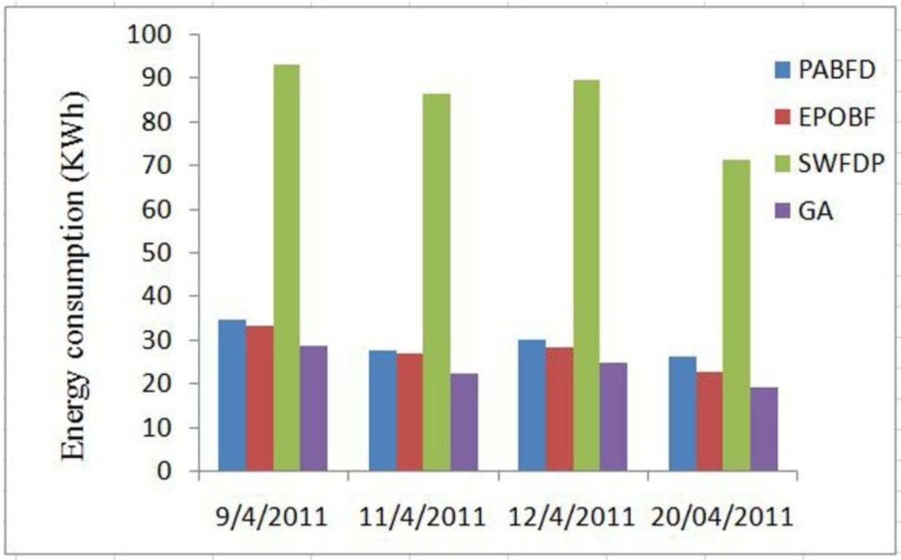


Figure 5.3: Energy consumption for real environment

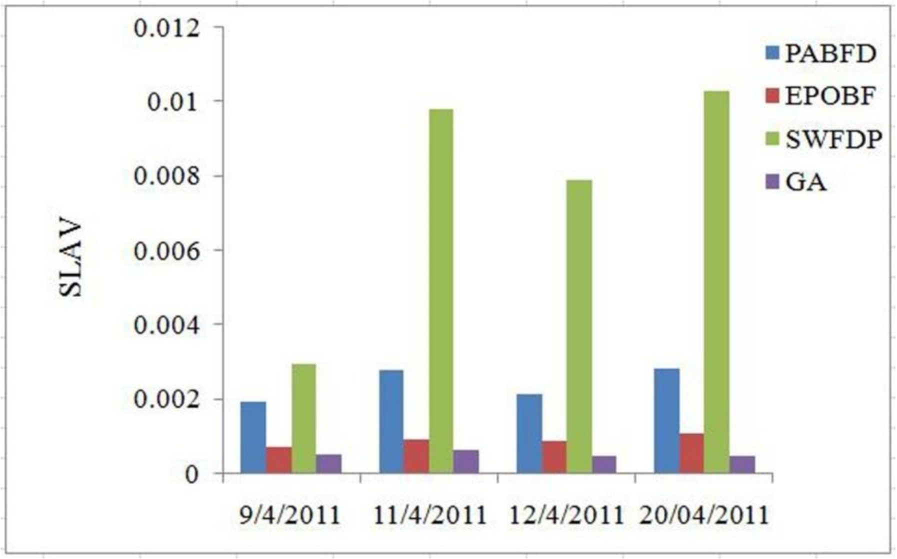


Figure 5.4: SLAV for real environment

From the figures, results reveal that the VM placement using genetic algorithm has minimum energy consumption along with minimum SLA violations during the simulation of all four days data set with different workload conditions. Figure 5.3 depicts the energy consumption of data center on four different days after VM placement using four different policies: PABFD, EPOBF, SWFDP and GA. Similarly, Figure 5.4 depicts the SLA violations occur during VM placement using different policies on four different dates.

5.4 NSGA-VMP (NON-DOMINATED SORTING GENETIC ALGORITHM FOR VIRTUAL MACHINE PLACEMENT)

In this section, we have presented the details of another solution for VM placement i.e. NSGA – VMP (Non-dominated Sorting Genetic Algorithm for VM Placement). NSGA is also the variant of basic GA that we have already described in above section. Single objective optimization requires a single attempt for finding best possible solution but during multi-objective optimization there are set of solutions present in the search space. From several set of solutions, some may be superior during the consideration of all objectives simultaneously and may be inferior during one or more objectives. These types of solution are known as non-dominated solutions or pare to optimal solutions. NSGA provides the set of optimal solutions from a huge search space therefore, we have used NSGA to attain the optimal solution and we have presented the details of NSGA-VMP and its design in following subsections.

5.4.1 Problem formulation for NSGA-VMP

The main objective is to obtain an optimal solution for VM placement using multi-objective optimization and for this, the following key terms and system considerations are defined for data center:

Vm: represents the set of virtual machines.
Pm: represents the set of physical machines.
Vm_iCPU: requested amount of CPU utilization by virtual machines (from i=1 to n).
Vm_iMem:requested amount of memory by virtual machines (from i= 1 to n).
Vm_iBw: requested amount of network bandwidth by virtual machines (from i= 1 to n).
Pm_jCPU: available CPU capacity of physical machines (from j=1 to m).
Pm_jMem: available memory of physical machines (from j=1 to m).

Pm_jBw: available bandwidth of physical machines (from j=1 to m).
$VmPm_j$: complete set of VMs which are to be mapped over physical machines (j=1 to m).

VM 1 can be represented as: *Vm_1 (Mem, CPU, Bw*) and PM 1 can also be represented as: *PM_1 (Mem, CPU, Bw).* Several PMs inside data center are composed of their available resource capacities such as: *Mem, CPU and Bw* and similarly, several VMs also contains various requested resources. Thus, the main purpose of this study is to find the best solution for VM-PM mapping and try to achieve the following 3 objectives simultaneously:

Objective1: Minimization the energy consumption (E)

Energy consumption of each host PM_j has been calculated as the energy consumed per unit time using equation given below:

$$E = \sum_{j=1}^{m}\left(pmax_j - pmin_j\right) \times cpu_j + pmin_j \qquad (5.8)$$

In above equation 5.8, E represents the total energy consumed by hosts, $pmax_j$ represents the maximum power consumed by host j where as $pmin_j$ represents the minimum power consumed by host j similarly, cpu_j represents the CPU utilization of each host j. The power maximum and minimum consumption of host has been provided by SPECpower benchmarks [106] from where, we have chosen 230W as maximum (when host is fully utilized) and 39W (when host is not used at all or in idle position) as minimum power consumption of PM. As, the energy consumed by every host j, can be evaluated as the integral of the power consumed during every period of time frame therefore, power consumption plays an important role.

Objective 2: Minimization of SLA violation (S)

To satisfy the mentioned QoS (Quality of services) in SLA is very imperative in cloud environment where they may vary according to workload and application requirement and thus, it is compulsory to describe some metrics or workload independent metric for the calculation of SLA violation level. Thus, we have considered 2 metrics for determining the level of SLAV and these two metrics are: SLATAH (SLA time per active host) and PDM (performance degradation during migration of virtual machines) shown in equation 3.2-3.4 of chapter 3.

Objective 3: Minimization of VM migration count (M_c)

Unnecessary or Redundant migrations may degrade the performance thus, by reducing the migration count during VM placement plays a significant role. To fulfill the requirements of requested VM resources during VM-PM mapping using selected PMs, following constraints will be considered:

$$\bigcup_{Pm_j \in Pm} Vm\, Pm_j = Vm \tag{5.9}$$

$$Pm_j\, CPU \geq Vm_i\, CP \tag{5.10}$$

$$Pm_j\, Mem \geq Vm_i\, Mem \tag{5.11}$$

$$Pm_j\, Bw \geq Vm_i\, Bw \tag{5.12}$$

Equations 5.9 defines that each VM will be assigned to single PM but one PM can be mapped to several VMs, whereas equations 5.10, 5.11, 5.12 illustrates that available capacity of CPU, bandwidth and memory of a PM should not be less than the requested resource of VM that are going to map selected PM. Moreover, the total capacity of CPU, bandwidth and memory of PM should not beat the resources used by the several set of VMs shown in equations 5.13 - 5.15 given below, where wcpu, wMem and wBw represents the total CPU, memory and bandwidth of physical machines.

$$PM_j wcpu \geq \sum_{i=1}^{n} Vm_{ij}\ (CPU) \tag{5.13}$$

$$PM_j wMem \geq \sum_{i=1}^{n} Vm_{ij}\ (Mem) \tag{5.14}$$

$$PM_j wBw \geq \sum_{i=1}^{n} Vm_{ij}\ (Bw) \tag{5.15}$$

MigCount represents the Migration count and can be evaluated by making the comparisons of initial placement with scheme used in chromosome's placement. Equation 5.16 illustrates the method for the calculation of migration count, where MC_i represents the total VMs that need to perform migrations, ∑MC represents the total sum of chromosomes along with number of individuals in the population represented by P as shown below:

$$\sum MigCount = \frac{1 - \frac{MC_j}{\sum MC}}{P - \sum_{j=1}^{m} \frac{MC_j}{\sum MC}} \tag{5.16}$$

Final Objective function

Every individual that we have obtained from set of new population have to fulfil all 3 objectives such as E (Energy consumption), S (SLA violation) and M_c (migration count). Thus, the value of the final objective function (obj) can be calculated using equation10 where the individual with minimum value of obj will provide optimal solution for the placement

$$objective = Minimize(\ E * S * M_C) \tag{5.17}$$

5.4.2 Description of NSGA

Srinivas et al. [112] presented the concept of Multi-objective optimization using NSGA which is an evolutionary algorithm and works as GA using similar operators such as: mutation and crossover. The only difference between both of them occurs during selection operation. NSGA first ranked the population according to the level of non-dominance of all the individuals before the completion of selection operator. Every non-dominated individual identified from the current set of population can be added to the front which is known as first non-dominated front. After the initialization of population, individuals are sorted on the basis of non-dominance level of every front and process starts with the addition of set of non-dominated solutions into first front and so on. Second front is dominated by every individual present in the first front and likewise, the front goes on. After the evaluation of the similarity between the individuals of every front the resulting individuals are used to promote the front of non-dominated solutions and removed from the population. The same process will be repeated until the entire population is classified. The entire process of NSGA in contrast to VM Placement is shown in figure 5.5 according to which once the population is classified, it will use several genetic operators and produce new set of solutions. During the creation of different fronts and for the identification of their non-dominated level one solution p can be ranked better than another solution q if:

$$non - domination\ level\ of\ p < non - domination\ level\ of\ q \tag{5.18}$$

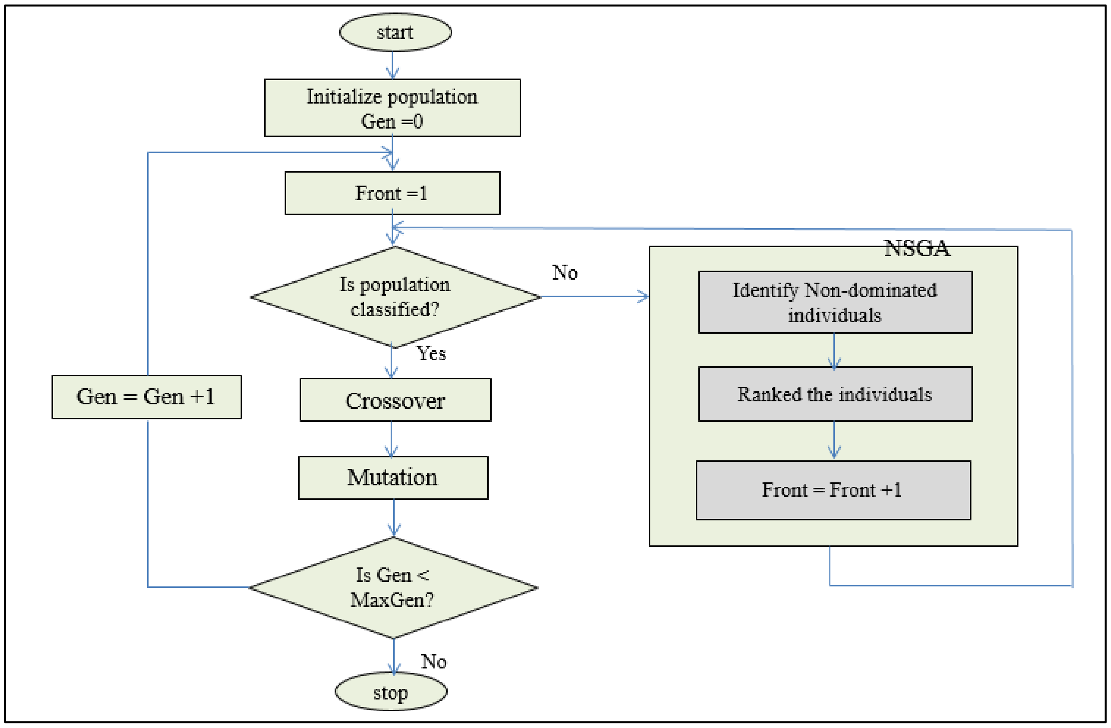


Figure 5.5: Flow chart for NSGA

5.4.2.1 Creation of Initial population

NSGA-VMP starts with the random initialization of the population where all VMs i.e. VM_i (i = 1 to n) will be mapped over PM_j (j= 1 to m) such that none of the VM will be mapped over more than one PM. 1D representation of chromosomes is shown in figure 5.6. in which 1D array demonstrates the assignment of VMs to PMs. For example, this figure shows the chromosome representation of 10 VMs and 5 PMs where VM2 is assigned to PM3.

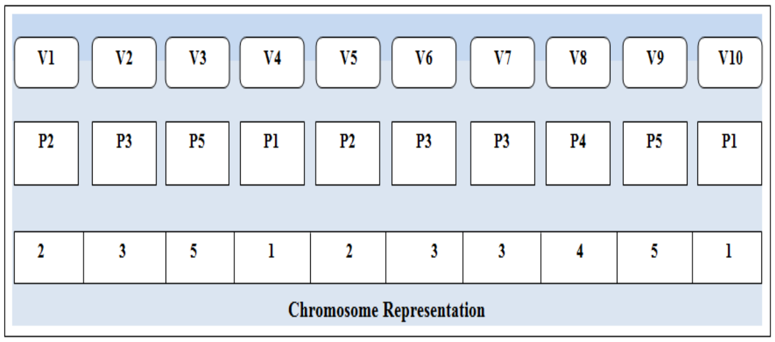


Figure 5.6: Initial population representation

5.4.2.2 Infeasible solutions

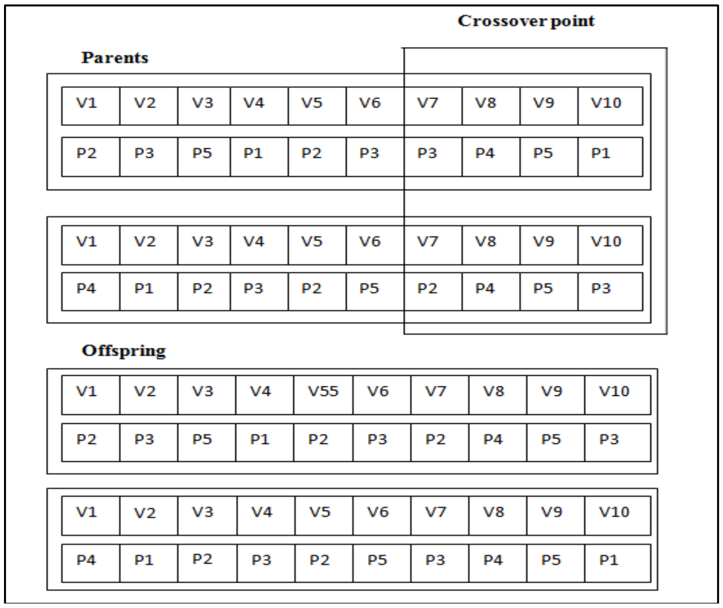
In next section, we have discussed three different constraints and that should be satisfied by every chromosomes and solutions which are not able to satisfy the constraints will be repaired by performing genetic operators such as mutation and crossover. These operators are used to generate new individuals after using genetic operators. Finally, these new individuals will be swapped again with the set of population to again check the constraints whether they are feasible or not. This process will be repeated until the optimal solution not found.

5.4.2.3 Selection operation

The most important step of NSGA-VMP is selection operation in which non-dominance level of every individual is used as a selection operator during multi-objective optimization. All individuals are sorted according to their non-dominance level and one with highest level of dominance will be given as a rank 1 and so on. To consider the individual as better one from any of 2, the one with a minimum value of non-dominance rank will be used. Therefore, all these individuals constitutes Ist non-dominated front where large dummy fitness value will be given to every individual of the front. Moreover, these values are also used by individuals and most importantly, the sharing can be accomplished using the values found by dividing the original values of fitness of every individual to the sum of individuals. Lastly, the set of population is arranged into sub population according to the level of pare to dominance which constitutes the first non-dominated front of the population. Likewise, the process will go on, until the entire population is classified.

5.4.2.4 Genetic operators

Two most important genetic operators are: Crossover and mutation. Crossover operator works according to the grouping of chromosomes which combines the sections from 2 different parents to obtain new off springs. These new offsprings are used to replace the population for example figure 5.7 displays the cross over in which single point crossover is used. We have used uniform crossover to produce new offsprings in this work. Likewise, the mutation operator is efficient to maintain the diversity of chromosomes in the population. Figure 5.8 shows how every gene is mutated with 1/m probability (m represents the number of VMs) by changing the allocation of a VMs to different PMs such that the new allocation may satisfy all the constraints.



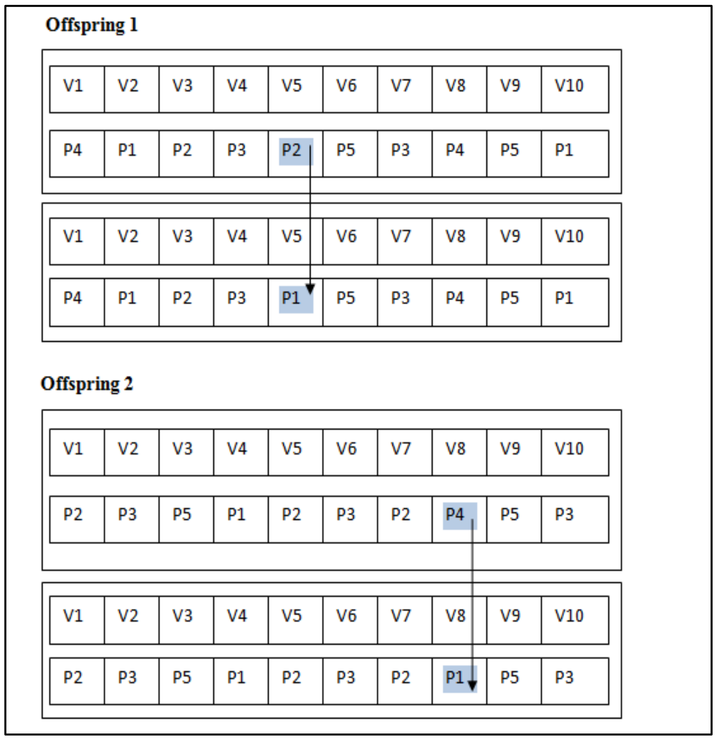
V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
P2	P3	P5	P1	P2	P3	P3	P4	P5	P1

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
P4	P1	P2	P3	P2	P5	P2	P4	P5	P3

V1	V2	V3	V4	V55	V6	V7	V8	V9	V10
P2	P3	P5	P1	P2	P3	P2	P4	P5	P3

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
P4	P1	P2	P3	P2	P5	P3	P4	P5	P1

Figure 5.7: Example of cross over operation



V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
P4	P1	P2	P3	P2	P5	P3	P4	P5	P1

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
P4	P1	P2	P3	P1	P5	P3	P4	P5	P1

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
P2	P3	P5	P1	P2	P3	P2	P4	P5	P3

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
P2	P3	P5	P1	P2	P3	P2	P1	P5	P3

Figure 5.8: Example of Mutation operation

5.4.3 VM placement process using NSGA

The complete process of NSGA-VMP multi-objective optimization is shown in table 5.6 according to which the proposed algorithm starts with the random initialization of population (i.e. VM-PM mapping) in step 3.

Table 5.6: Pseudo-code for VM Placement using NSGA

```
1    Input: population_size, P_crossoverrate, P_mutationrate
2    Output: Children
3      Polpulation←random_initialization of population (population_size)
4    Initialize generation =0
5       Initialize front =0
6        While (stopping criteria (Genaration_size))
7           {
8             If (population classified)
9               {
10                Selected← two best individuals (Population)
11                Children←crossover (Selected, P_crossoverrate)
12                NewChildren← mutation (Children, P_mutationrate)
13                Evaluate_objective_function (NewChildren)
14              }
15            Else
16              {
17                Selection_using_NSGA (Population)
18                  {
19                    Newselection←identify nondominated individuals
                                        (Population)
20                    Rankpop←ranking of individuals (New selection)
21                    Newfront←Rankpop
22                    Newpop← sharing of individuals (Individuals)
23                    Firstfront←Newpop
24                  }
25              }
26            Front++
27           Generation ++
28       }
```

Genetic operators such as: crossover and mutation will be applied once the population is classified as shown in step 8-11 after which the population will be further checked whether it is able to satisfy the objective function or not. Whereas, on the other side step 16 shows that before the selection operation, NSGA will be performed if the population is not classified in which the non-dominated individuals from the population will be identified and ranked according to non-dominance level. Dummy fitness values will be given to every individual of every front and accordingly the sharing function of every individual will be assessed and fronts will be created. With the process of evolution, the fronts will be generated by repeating the steps until the algorithm meets the stopping criteria.

5.5 EXPERIMENTAL SETUP & RESULTS

This section provides the experimental setup and different test cases for the execution of proposed NSGA- VMP. We have implemented it using the open framework for CloudSim environment using JAVA Net beans IDE. For NSGA characteristics, we have selected 0.5 as the crossover and mutation rate along with the population of 500. Whereas, for the simulation of cloud environment, we have used four different test cases with varying numbers of PMs and VMs. They are also varying according to their configuration. These for different test cases are shown in table 5.7.

Table 5.7: Test cases used for performance evaluation

Test case ID	Number of VMs	Number of PMs	Types of Servers used
1	898	500	4
2	1052	600	4
3	1358	700	6
4	1516	800	6

Table 5.8: Performance of data center using GA, ACO and NSGA

Test Cases	Algorithm	Energy consumption (KWh)	SLAV (s)	Migration Count (M_c)
1	GA	23.2	0.0503	220
	ACO	21.5	0.05	198
	NSGA	20.1	0.049	169
2	GA	26.1	0.057	337
	ACO	24	0.054	294
	NSGA	22.5	0.052	271
3	GA	29.3	0.06	402
	ACO	26.3	0.057	376
	NSGA	24.8	0.055	343
4	GA	30.1	0.062	424
	ACO	27.5	0.059	387
	NSGA	25.8	0.057	359

We have compared the proposed NSGA-VMP with existing ACO and GA based VM placement for which we have chosen the Policy of minimum correlation (MC) for VM selection with the value 1.2 as a safety parameter. Moreover, for the comparison of datacenter's performance we have used three performance metrics such as: Energy consumption, SLA Violation and Migration count (number of migration occurs during VM placement) and their respective values are shown in table 5.8 whereas, table 5.9 displays the overall objective i.e. ESM_c – total consumption of Energy SLA and Migration count.

Table 5.9: ESM_c consumption of GA, ACO and NSGA

Test Case	Algorithm	ESM_c
1	GA	268.18
	ACO	212.8
	NSGA	165
2	GA	499.423
	ACO	381.02
	NSGA	317
3	GA	699.48
	ACO	563
	NSGA	467
4	GA	788.64
	ACO	627
	NSGA	527

Figure 5.9, depicts that NSGA-VM placement consumes minimum amount of energy. With the increase in the request of VMs, the consumption of energy will also increase. Whereas, the reason of lesser VM migration during NSGA-VM is that the NSGA already rank the population in different fronts during each iteration and their ranking is based on the sorting of their non-domination level. Thus, the creation of fronts in this approach will reduce the chance of migration for the placement of VM over PM. Since the migration count is less, therefore; performance degradation during migration also will be less. Moreover, the hosts will remain active for a longer time due to the creation of fronts again and again, which leads to lesser SLA time per active host. As the SLA violation is the combination of SLATAH and PDM therefore; this method will provide minimum SLA violation during placement. Due to minimum resources used in this method, minimum will be the power consumption per unit of time and minimum will be the energy consumption. Moreover, figures 5.9-5.11 depicts that the minimum amount of

energy consumption is consumed, with minimum level of SLA violation by performing minimum migrations by NSGA-VMP in comparison to GA and ACO for all four test cases. Figure 5.12 shows the combined ESM_c metric which provides better results during NSGA-VMP.

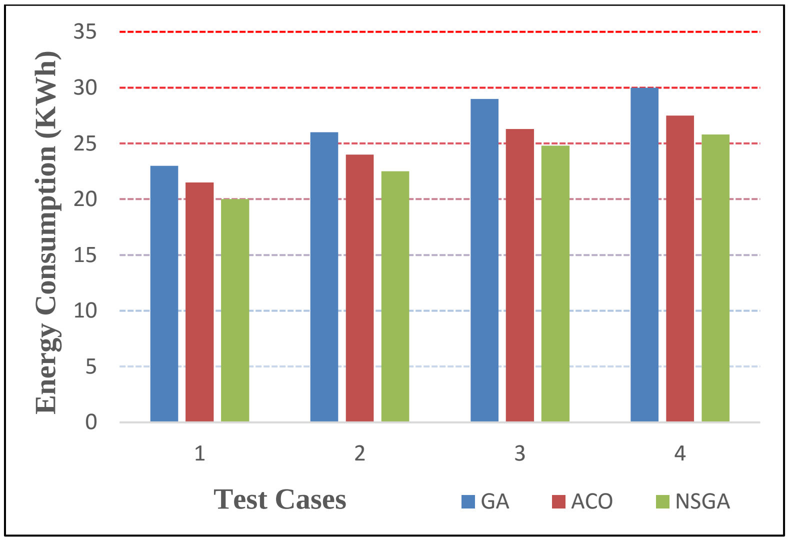


Figure 5.9: Energy consumption using GA, ACO and NSGA VM Placement

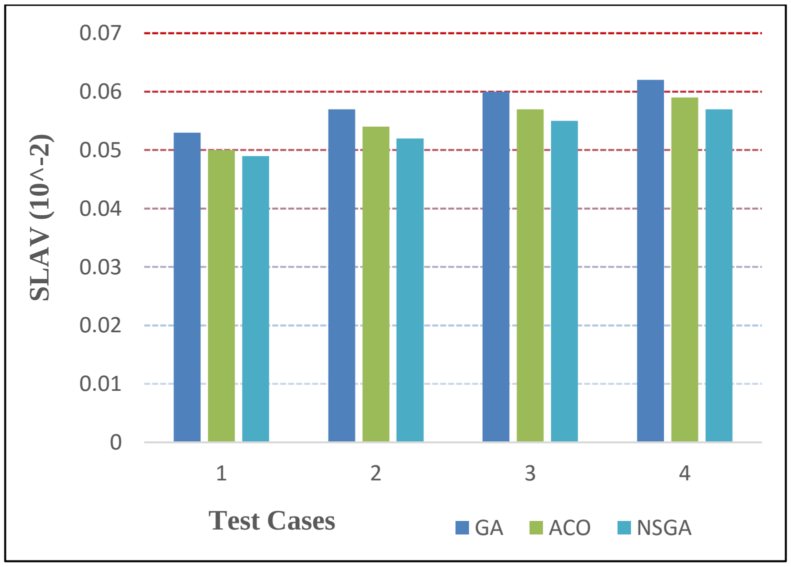


Figure 5.10: SLA Violations during GA, ACO and NSGA VM Placement

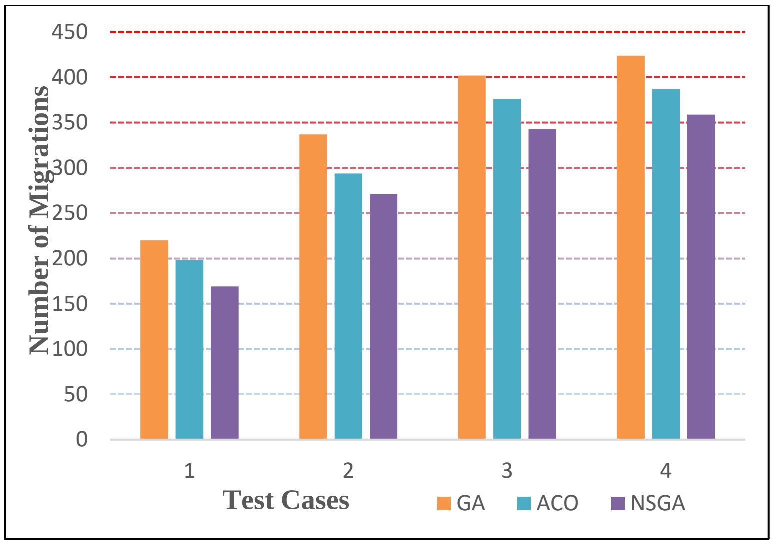


Figure 5.11: Migration count during GA, ACO and NSGA VM Placement

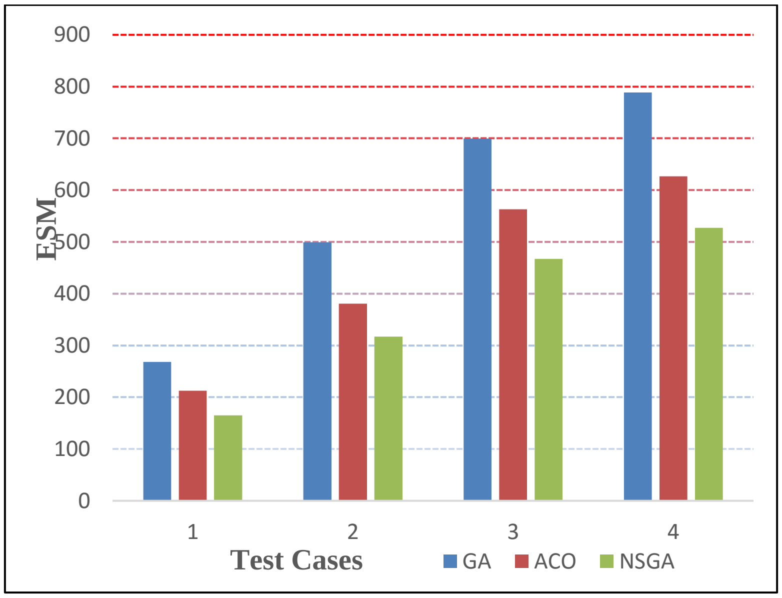


Figure 5.12: ESM_c during GA, ACO and NSGA VM Placement

5.6 SUMMARY

In this chapter, first, we have discussed the several techniques of VM placement involved in VM consolidation process. Moreover, we have analyzed the results of 3 existing VM placement techniques namely, PABFD, EPOBF and SWFD and using genetic algorithm we have proposed a decision-making system for VM placement. Also, we have discussed that how genetic algorithms are different than classical bin packing algorithms. Proposed and predefined techniques have been simulated in the CloudSim simulation environment. Data set from PlanetLab has been chosen for the real workload conditions of VMs on four different days. The promising results obtained from the simulation shows that how the placement of virtual machines using genetic algorithms is helpful for minimizing the consumption of energy inside the data centers and the level of SLA violations at the same time. As, the cloud data centres consume huge amount of energy, therefore, minimization of energy consumptions along with some other performance metrics provides a new direction for the improvement of the cloud environment.

These analyses of VM placement using GA enlighten our work in a new direction. Therefore, our next objective was to explore some other GA technique which would be more efficient than basic GA for solving multi-objective problems. Thus, we have tried to solve the VM placement problem using a NSGA and named the method as NSGA-VMP. As we have discussed in all previous chapters that the main focus of this study is the minimization of energy consumption without raising the level of SLAV and also to improve other performance parameters i.e. reducing the number of unnecessary migrations. Proposed NSGA-VMP sorts the individuals according to their dominance level. It is the first successful attempt for the mapping of VMs over PMs using NSGA for the performance improvement of data center in terms of energy consumption, SLA violation and migration count. According to the experimental results, NSGA-VMP provides an optimal solution of VM-PM mapping for 4 different test cases. Though it takes more time for execution, but it is still acceptable.

CHAPTER 6

BPGA: A NOVEL APPROACH FOR ENERGY EFFICIENT VM PLACEMENT

This chapter provides the optimal placement by improving various performance parameters using multi-objective optimization. We have proposed a BPGA model that will work in two different passes. The first pass makes the use of NSGA for VM placement and the second pass uses the BPNN method for the placement of remaining VMs.

6.1 INTRODUCTION

The proliferation of cloud computing is capable of supporting various computing services such as storage, servers, networks and applications for both e-sciences, e-business and much more over the network. This new era of cloud computing is available with large pool of easily usable and accessible virtualized resources such as applications, hardware, run time platform and services. Large numbers of data centers are required to respond the demands of customers, which results in the consumption of a huge amount of energy. From the previous research, it has been analyzed that 55 percent of energy consumed by data center is only because of the several servers and IT equipment’s and 30 percent is due to cooling equipment’s, therefore, these datacenters are very expensive to maintain and they also have very severe effects on the environment. Moreover, this profitable success of cloud computing environment leads to provide better QoS - Quality of Services that are documented in the SLA- Service level agreement between cloud service providers and users. Virtualization is one of the hot topic in cloud computing, which provides better QoS and deals with auto scaling, server/VM consolidation, energy conservation, load balancing and much more because of its capability to run several operating systems on the single physical machine by sharing the hardware resources. Also, the improper allocation of VMs on unsuitable host affects the interference of the different applications on same physical machines and this leads to the performance degradation with decreased level of Quality of services (QoS) for the applications. Therefore, certain issues should be resolved during VM consolidation process which can improve the utilization of resources, performance and energy consumption of data center. Efficient VM consolidation process deals with the migration problems and the mapping of virtual machines over the suitable physical machine (known as Virtual machine placement) can minimize the energy consumption of data center and also it delivers the better quality of services which may decrease the violation level of SLA agreement. The contribution of this chapter is as follows:

- The approach of BPGA makes the use of bio-inspired optimization technique for the improvement of energy efficiency and other performance parameters inside cloud data centers.
- The proposed BPGA model is scalable for a large heterogeneous cloud environment and we have performed the simulations using open source cloud framework known as CloudSim.
- The efficiency of proposed model has been displayed by comparing it with other techniques (46.5% improvement in energy consumption with 41.3% improvement in hosts usage, and 56.9% improvement in a number of VM migrations as well as the cost of VM placement is also improved by 10%).

6.2 RELATED WORK

Most of the existing work solved the problem of VM placement by using different energy aware heuristics that have been proposed by Anton et al. [5], [10], [59]. These heuristics also used the concept of VM migration by minimizing the number of hosts to lower down the energy consumption. Many improvement and extension can be made using variants of greedy approaches like FFD, BFD, FF BF as mentioned in1 [113] − [114]. VM placement helps for lowering the energy consumption of idle or free resources by keeping them aside and switching them off or into sleep mode. On the basis of previous study, we have concluded that most of the existing work focused on issues related to energy management and performance efficiency of datacenters using VM migration, VM placement i.e. VM consolidation. Similarly, the proposed model also focuses on the performance improvement by considering QoS (Quality of services) metrics and energy enhancement using bio-inspired genetic algorithms as well as artificial neural network.

Presently, the research work of many authors are focusing on the meta heuristics and bio-inspired computing techniques to handle these issues such as: Ant Colony Optimization (ACO) [88], [115], Particle Swarm Optimization (PSO) [115], [116], [117], [118], [119], Fire Fly Optimization (FFO) [120], [121], Genetic Algorithms (GA) etc. Different types or variants of genetic algorithms have been used by researchers in [78]− [122] where Hitosbi et al. [78] & Shi Chen et al. [80]used a genetic algorithm for the allocation of VMs in distributed systems. Also, they have used a set representation for the VM placement in which machines or hosts represents the set and virtual machine represents the set items. Later on, a new approach of a hybrid genetic algorithm for VM placement came into existence and used by Maolin Tang et al. [82]. Grouping Genetic Algorithm (GGA) has been advocated by Xu et al. [84] for efficient management of energy consumption and one year late has been modified by Wilcox et al. [83] named as reordering grouping genetic algorithm (RGGA) which resolved the problem of multi capacity bin packing by considering VMs having multiple weights and servers having multiple capacities. Faruk et al. [123] introduced a new area for VM placement and advocated an iplace an intelligent and tunable power and performance aware VM Placement middleware. According to this middleware, the placement of virtual machines is based on two level artificial neural networks. CPU usage of host machine has been predicted in first level and power consumption, the performance of host in the second level. Giuseppe et al. [124] introduces the allocation of resources using NSGA (Non-dominated Sorting Genetic Algorithm)-II for power efficient cloud environment. For this, they have used the concept of Pareto-optimal solutions. Likewise, in our work, we have also used NSGA multi-objective optimization or set of Pareto-optimal solutions along with back propagation neural network, a method of training artificial neural network. Thus, our model which makes use of these techniques for VM placement is known as BPGA model for the enhancement of performance and management of energy consumption inside data centers.

6.3 SYSTEM MODEL FOR ENERGY AND POWER CONSUMPTION

However, the energy consumption of host inside data center can be calculated from its components such as CPU, memory, network interfaces and disk storage, but CPU is the main source of energy consumption. From a previous study [28], it has been clear that there is a linear relationship between the CPU utilization and power consumption of servers thus, the power consumption represents the function of CPU utilization. Moreover, the server’s CPU utilization may change with time due to variable conditions of workload and thus, the CPU utilization represented as a function of time U(t). Therefore, the total energy consumption (EC) of physical server can be calculated as integral of the power consumption over a period of time [125] shown in following equation 6.1 and in equation 6.2 PCbusy represents the power consumption when machine is fully utilized and PCidle represents the value of power consumption when machine is idle or 0% utilized. For simulations, the power consumption of server when they are idle and fully utilized are taken from SPEC power benchmark [106].

$$EC = \int_{to}^{t1} P\big(U(t)\big)dt \tag{6.1}$$

$$PC = \begin{cases} (PCbusy_i - PCidle_i) \times U_i^{PC} + PC_i^{idle} & ifU_i > 0 \\ 0 & otherwise \end{cases} \tag{6.2}$$

For energy consumption modelling our current work focuses on the energy consumption that changes dynamically. Let ENCji be the energy consumption caused by VMj running on physical machine PMi. Let us assume the energy consumption rate of physical machine PMi is ENRji and thus, the energy consumption ENC can be calculated as follows:

$$ENC_{ji} = ENR_{ji}.E_j \tag{6.3}$$

Hence total energy consumption for executing all the virtual machines is:

$$ENC_{exec} = \sum_{i=1}^{H}\sum_{j=1}^{V} ENC_{ji}$$

$$= \sum_{i=1}^{H}\sum_{j=1}^{V} ENC_{ji} * ENR_{ji} * E_j \tag{6.4}$$

In above Eq4, it is assumed that the physical machine does not consume energy when it is idle [52] however, this assumption could not be possible in real life virtualized environment. When physical machines are idle, they include energy consumption in two parts. One is when all the VMs of that hosts are idle and other is when some of the VMs of hosts are idle. When all the VMs are idle, then the host can be set to lower energy consumption rate using DVFS technology [52]. Thus, the rate of energy consumption of physical machine PM in this case is ENRT ji and ti is the idle time when the host is idle. Thus, the energy consumption when the host is idle with all its idle VMs:

$$ENC_{allidle} = \sum_{i=1}^{H}\sum_{j=1}^{V} ENR_{ji}^{T}.ti \tag{6.5}$$

If some of the VMS in the host is idle, then the rate of energy consumption of VMs are same as that when they are executing on the host. Thus, it means the rate of energy consumption of machine PMi is ENRji. Therefore, the energy consumption in this case is:

$$ENC_{idlepart} = \sum_{i=1}^{H} \sum_{j=1}^{V} ENR_{ji} \,.\, ti \tag{6.6}$$

Total energy consumed by physical machine i which executes virtual machine j is:

$$ENC_{total} = ENC_{exec} + ENC_{allidle} + ENC_{idlepart}$$

$$= \sum_{i=1}^{H} \sum_{j=1}^{V} ENC_{ji} \,.\, E_j + \sum_{i=1}^{H} \sum_{j=1}^{V} ENR_{ji}^{T} \,.\, ti + \sum_{i=1}^{H} \sum_{j=1}^{V} ENR_{ji} \,.\, ti \tag{6.7}$$

6.4 VM PLACEMENT AS A MULTI-OBJECTIVE OPTIMIZATION

VM placement problem has been solved using multi-objective optimization in following sections. Here, in our work, we have used NSGA (non-dominated sorting genetic algorithm) for the optimal mapping of VMs over physical hosts. Moreover, the problem of VM placement along with its constraint that needs to be fulfilled to achieve the objectives are shown in 6.4.2.

6.4.1 Problem formulation

Multi-objective optimization techniques generally use the population based approaches for finding optimal solutions. Also, they use the concept of pareto-dominance during selection operation. For any multi-objective optimization problem, there are set of N number of objectives which need to minimize or maximize (here minimize in our case).

$$f(X) = f(X1, X2, \ldots\ldots Xn) \tag{6.8}$$

Here in equation 6.8, the vector X has a number of decision variables in solution space sp and we have to find particular vector X or different number of trade off vectors which can minimize the objective function. For solving the problem of multi-objective optimization, there are some solutions which can optimize the results for the single objective but do not guarantees to provide optimal results for another objective. For this reason, it is more beneficial to use the concept of pareto-optimal solutions. The concept of pareto-optimality provides the set of trade-off solutions and these sets have all those non-dominated solutions provides the best possible tradeoffs among the best solutions for several objectives [126]. A solution S from solution space sp is considered as pareto-optimal, if no other solution P exists in the search space which dominates solution S. Moreover, solution S dominates solution P or S has better non-dominated rank than P ie. RS < RP (rank of S and P) if both of the following equations are true:

Condition 1: If solution S is as good as solution P for all the objectives

$$\forall j \in [1,2 \ldots \ldots n] f_j(S) \leq f_j(P) \tag{6.9}$$

Condition 2: If solution S is severely better than P for at least one objective

$$\exists\, j \in [1,2 \ldots \ldots n] f_j \leq f_j(P) \tag{6.10}$$

All the solutions which are not dominated by any other solutions are called nondominated solutions and they together constitute a front in the solution space known as non-dominated front also the set of the solution in the non-dominated fronts are known as pare-to optimal solutions. The most tedious step in this concept is to find the set of non-dominated solutions. In our work, to find the set of nondominated solutions the following steps are used [112] with Z number of solutions and each has N number of objectives:

Step 1: Start with I =1

Step 2: Compare solution Si and Pi for their domination rank using above mentioned conditions for all N objective

Step 3: If Si is dominated by Pi, then mark Si as dominated and go to step 2 by incrementing i=i++

Step 4: If all the solution (i =1 to Z) are considered, go to next step otherwise go to step2 by incrementing i

Step 5: Solutions which are not marked as dominated are non-dominated solutions

6.4.2 VM Placement optimization

Here, in this section the VM placement has been optimized as: suppose we have V number of virtual machines and that are to be mapped or placed on M machines and we are assuming that none of the virtual machines requires the resource more than the available resources of physical server. Let Ui be the request of CPU utilization from each VM, Tuj the threshold for the CPU utilization, RMi the memory request for each VM and TRMj the threshold for memory utilization. Moreover, we have taken two binary variables Zij and Xj which will be used to investigate whether the VM i is allocated to server j or not, and whether the server j is currently in use or not. In the current work of multi-objective optimization, the following objectives must be minimized: Total energy consumption (EC) of the data center, Quality of Services such as SLA violations, migration count and Cost of the data center. The minimization of energy consumption is the final objective of current work, but it also affects the level of SLA violation hence, we need to find some trade-off values for these two. Thus, the problem of VM placement can be formalized as:

Energy Consumption:

$$\sum_{j=1}^{M} EC_j = \sum_{j=1}^{M} [X_j \times (PCbusy_j - PCidle_j) \times \sum_{i=1}^{V} (Z_{ij} . U_i) + pcidle_j] \tag{6.11}$$

SLA Violation:

$$\sum_{j=1}^{M} SLAV = \left[\sum_{j=1}^{M} T_{si} - T_{ai}\right] \times [\sum_{i=1}^{V} C_{dj} - C_{rj}] \tag{6.12}$$

Number of the host used:

$$\sum_{j=1}^{M} y_j \ where \begin{cases} y_j = 1 , & if \ \sum_{i=1}^{V} X_{ji} \geq 1 \\ y_j = 0 & otherwise \end{cases} \tag{6.13}$$

Subject to:

$$X_{ji} = \begin{cases} 1, & if \ VM \ i \ is \ allocated \ to \ server \ j \\ O, & otherwise \end{cases} \quad \forall i \in VM \ and \ \forall j \in PM \tag{6.14}$$

$$\sum_{i=1}^{V} VMC_i . X_{ji} \leq PMC_j . Y_j \ \ \forall j \in PM \tag{6.15}$$

$$\sum_{i=1}^{V} VMmem_i . X_{ji} \leq PMmem_j . Y_j \ \ \forall j \in PM \tag{6.16}$$

Migration Count:

$$\frac{1-\frac{MC_j}{\sum MC}}{P-\sum_{j=1}^{M}\frac{MC_j}{\sum MC}} \tag{6.17}$$

The cost of the data center:

$$CT = \sum_{t=t0}^{T}(CTP\ \sum_{j=0}^{M} HA_{tj} + CTV\ \sum_{j=1}^{M} SV_{tj}\ ) \tag{6.18}$$

In equation 6.12, for SLA violation Tsi is the total time for which host experienced 100% utilization, Tai represents the time for which host remains active, Cdj is the estimate of performance degradation caused by migration (here, it is assumed as 10% of CPU utilization during all migrations of virtual machines) and Crj is the total CPU capacity of virtual machine. For minimizing a number of hosts in equation 6.13, we have taken a decision variable Xji, which shows whether host j (yj =1) is used or not (yj=0). Equation 6.15 and 6.16 shows that each host should satisfy the resource requirements of VMs. Also, for migration count, MCj is the total number of VMs that need to be migrated. Moreover, the service providers pay the cost of energy that is consumed by physical machines. Therefore, the cost function is also important for the consideration during live migration and this cost function can be calculated as CTPtp and CTVtp where CTP is the cost of power and tp is a time period (energy per unit time) and CTV is the cost for the level of SLA violated per unit time. SLA violation between the service provider and consumer occurs only if the demand for resources exceeds the available capacity of resources. In equation 6.18, to is the initial time frame and T represents the total time. HAtj indicating whether host j is active on not at the time t and similarly, SVtj indicating whether host j is experiencing SLA violation or not in time frame t.

6.5 PROPOSED BPGA-VMP MODEL FOR VM PLACEMENT

First three steps of VM consolidation process has been already done on our previous work where we have proposed an energy aware algorithm for the selection of VMs for migration and selection of underutilized, over utilized host. Now, in the current work, we have proposed BPGA model which facilitates the energy aware VM placement algorithm for minimization of energy consumption of datacenters without degrading the performance of datacenter. Therefore, by increasing the utilization of server’s resources and by using the lesser number of active servers, energy consumption can be minimized and thus, it directly contributes towards the green computing. The current work presents an energy aware virtual machine placement (VMP) method which is based on NSGA and biological neural network (BPNN) and named as BPGA model shown in Figure 6.1. This technique tries to place the virtual machines to another active host and reduce the usage of number of active host inside datacenter. This BPGA model is working in two passes. In first pass, the process of Non-dominated sorting Genetic Algorithm (NSGA) will be used for the mapping of virtual machines. NSGA contains two objective functions and the physical machine which fulfill all the requirements through NSGA would be selected for the mapping of VMs and rest of the VMS are forwarded to the second pass of BPGA model which follows the process of training the neural network based on the different conditions of the power consumption of machine to resolve the issue of VM selection.

Pass 1: NSGA (Non-dominated Sorting Genetic Algorithm

Step 1: In the initial step of the first pass, there are many options available for the mapping of Virtual machines and any virtual machine can be mapped to any of the physical machines. But it is important to satisfy the equation 6.19 which tells that one VM will be mapped over single physical machine, therefore to select the appropriate one, NSGA will proceed further with step 2 with detailed shown in table 6.1.

$$\sum_{j=1}^{M} X_{ji} = 1 \ \forall i \in PMs \tag{6.19}$$

Step 2: First, the total amount of the resources requested by virtual machines will be calculated along with the capacity of the physical machine that provides the resources and checked weather they satisfy the equation 6.20, 6.21 or not i.e. constraint of fitness function 1. If yes then, the value of fitness function 1 will be calculated and VM clusters will be created in which VMs will be arranged according to their ranks. By VM clusters, we mean that PMs with in these clusters are the options for mapping of VMs. Moreover, if these constraints are not satisfied by the VMs than those pending VMs will be forwarded to pass2 without checking the objective function 2.

$$C_j = \frac{\sum_{j=1}^{M} c_{jm}}{m} \tag{6.20}$$

$$Mem_j = \frac{\sum_{j=1}^{M} Mem_{jm}}{m} \tag{6.21}$$

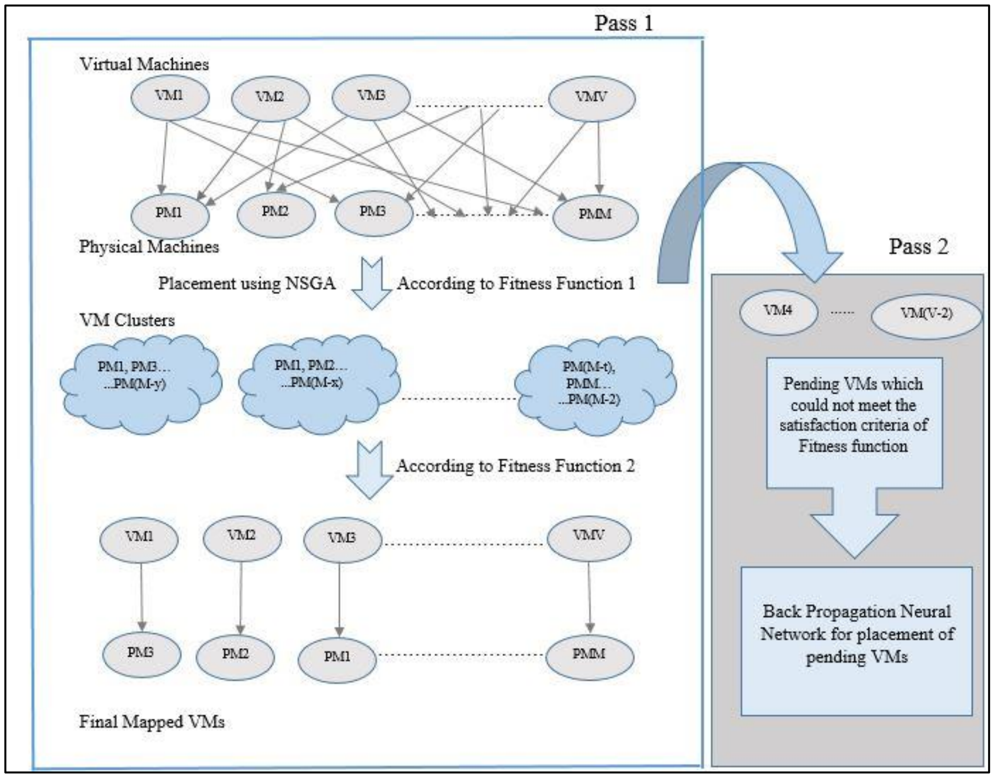

Pass 1
Virtual Machines
VM1
VM2
VM3
VMV
PM1
PM2
PM3
PMM
Physical Machines
Placement using NSGA
According to Fitness Function 1
VM Clusters
PM1, PM3… …PM(M-y)
PM1, PM2… …PM(M-x)
PM(M-t), PMM… …PM(M-2)
According to Fitness Function 2
Final Mapped VMs
Pass 2
VM4
VM(V-2)
Pending VMs which could not meet the satisfaction criteria of Fitness function
Back Propagation Neural Network for placement of pending VMs

Figure 6.1: BPGA model for VM Placement

Fitness Function 1: Resource Capacity CPU (C) and Memory utilization (Mem) are two main resources that we have considered in our work. In the chromosome representation of genetic algorithm, each value of gene array represents the placement destination of the virtual machine. For the jth physical machine, suppose that it can carry m virtual machines, then the resource dimension array for VM m carried on the jth physical machine can be expressed as: [C_{j1}, C_{j2}..C_{jm}] and [Mem_{j1}, Mem_{j2} Mem_{jm}]. The total values of resource dimension are:

$$RC_j = C_j + Mem_j \tag{6.22}$$

$$Fitness_{F2} = \frac{1}{RC_j} \tag{6.23}$$

Equation 6.20 and 6.21 gives the average values of the resource dimensions for the physical machine that carried m VMs. For example, if a physical machine 1 has 2 VMs whose resource dimension represents as [0.6,0.8] and [0.3,0.2] which means 0.6 and 0.3 are CPU and memory utilization of VM1 and 0.8, 0.2 are of VM2 respectively. In this case, the total resource capacity for PM1 is 1.9. Similarly, resource capacity for each PM will be calculated and using equation 6.22, the value of the fitness function will be calculated and physical machines with in the cluster will be arranged according to the fitness values.

Table 6.1: Algorithm for pass 1 of BPGA model

```
Input: Vm list and Host list
VM-PM Placement map
Allocated_host=null, Cando_list = null, minenergy=MAX
Tournament_size[][]=null
Objective1.NSGA= findfit(Hostcapacity, VMrequirements)
    For each Vm in Vm list do
        For host count 1 to host count do
            If (hostfulfill(VMrequirements)) then
            Cando_list[hostcount][0]=hosted
            Cando_list[hostcount][1]=Vmid
            Else
                BPNN( )
            Endif
        Endfor
    Endfor
Objective2.NSGA= selectionbestfit(Cando_list[][])
    If (hostcontaining.Vm >1)
        For each host in Cando_list[][] do
            If (host.energy! minenergy)
                {
                Allocated_host = host
                Minenergy = energy
            Endif
        Endfor
    Endif
Return VM-PM map
```

Step 3 If objective function 1 is satisfied, then from above-generated VM clusters every virtual machine has more than one physical machines available for the mapping. Therefore, this step provides the final mapping of virtual machines over physical machines by calculating the value of fitness function 2 and physical machines which could be the destinations for the VMs will be arranged according to the ranking of their fitness value within the clusters and according to that value, the physical machine will be chosen from every VM clusters for the placement of VMs. The value of energy consumption has been calculated in fitness function 2 using equation 6.7.

Fitness Function 2: Energy Consumption

$$Fitness_{F2} = \frac{1}{\sum_{j=1}^{M}[ENCtotal]} \tag{6.24}$$

Pass 2: BPNN (Back Propagation Neural Network) In pass 2, the forwarded VMs which were not able to satisfy the constraints of NSGA will be mapped on physical machines using neural network, where we have three main sections: a) Training data of input layer b) hidden layer for processing of weights and errors c) output layer for the allocation. In the process of back propagation neural network, initially, weights are randomly assigned to all the edges. From the training dataset, the output is observed after activating the neural network for every input. This output will be compared with the desired output and errors will be propagated back to the previous layer (hidden layer) and weights will be adjusted accordingly. The process is repeated until the gradient is satisfied or the error is below some threshold value explained in table 6.2. After the termination of this process, we have trained a neural network which is ready to work for new inputs (or for testing). Following different steps are used in the second pass of BPGA model:
Step 1: First, we will initialize the variables that we have used for the training and testing of the neural network. Here, we have generated the random value for the gradient and initialize the value of gradientsatisfied = 0 which will become 1 if gradient will be satisfied. Termweight [ ][ ] matrix contains the weight of input data by calculating the value of utilization of servers on 11 different conditions from idle to 100% utilization. The constraint of gradient satisfaction is shown in following equation 6.25 where generate Termweight is a function to generate linear weights (a.x+b) with random integer values for a and b.

$$(if\ gradientsatisfied == 0)$$

$$Termweight = generate_{termweight(data)}$$

$$= a * (data) + b \tag{6.25}$$

Table 6.2: Algorithm for Pass 2 of BPGA model

```
Input: Vm list, Host list
Output: VM-PM placement map
Train_Set[][]
    For each host in Hostlist do
        For i = 1to 11 do
            Train_Set[host][i] =host[condition(i)]
        Endfor
    Endfor
Total Epochs = 500
Initialize_gradientsatisfied = 0
Termweight[][]
  While(gradientnotsatisfied) do
      For each data in Train_Set([][]) do
          Termweight[i][f]= a*data+b
      Endfor
  Endwhile
MT = Mean(Termweight)
  If MT < gradient
      Termweight = Termweight + s
  Else
      Gradientsatisfied=1
  Endif
For each vm in Test_Set do
    Allocated_Host = simulate(Test_Set, Train_Set)
Endfor
Assign Vm to Allocated_Host
Return VM-PM map
```

Step 2: Second step follows with the processing of weights within the hidden layer by checking the average value of weights and gradient. If the random change in the average weight is less than the value of gradient, then the value of weight will be increased by the value of s calculated using equation 6.26 for which first, we will find the mean value of Termweight known as MT and compare it with a gradient.

$$\begin{cases} Termweight = termweight + s \;\; (if\ MT < gradient) \\ gradientsatisfied = 1 \qquad\qquad\qquad otherwise \end{cases} \tag{6.26}$$

Table 6.3: Simulate function for pass 2

```
Tts = Test_Set
Trs = Train_Set
[r] = size(Trs)
For (I =1 to r)`
MR = Mean(Tts)
  If (MT < MR)
  Allocated_Host = i
End For
```

Step 3: The Neural network has been trained from previous steps and ready to work for new inputs or test data i.e. Test Set[ ][ ] matrix. As similar to Train Set, the matrix of Test Set contains energy consumption of VMs during the eleven different conditions of severs from idle to 100% utilized. Energy consumption of VMs has been calculated using equation 6.1 in with random values of power consumption for VMs and finally, the Test Set matrix will be further passed to simulate function along with Train Set matrix for the allocation of most appropriate host mentioned in table 6.3.

Theorem 1: *The time complexity of the BPGA model for VM Placement is O(NvmNpm(Npm+Nvm)+Npm), where Nvm is the number of virtual machines, Npm is the number of physical machines and w is the weights used in the neural network.*

Proof: The time complexity for checking the requirements of each virtual machine is O(Nvm) as the process is linear. It takes O(Npm) time complexity for checking whether a physical machine can be added to Cando_List or not (Lines 6-15, Algorithm1). Also, the time complexity for finding the host with minimum energy is O(Npm) (Lines 18-23, Algorithm1). In algorithm2 (Lines 4-8), complexity for the training of dataset containing physical machines is O(Npm). The time complexity for the selection of the most suitable host for mapping using BPNN is O(W3), this is because of the total number of passes required to update the weights between each communication. Where first pass will compute the error at the output layer while the second pass back propagates the error to lowest weight and the last pass is for the updating of each weight (Lines 10-22). For allocating the host to VMs time complexity is O(Nvm) (Lines 30-31, Algorithm2) and or the rest of lines time complexity is O(1). Thus, the total complexity of algo2 turns out to be O(Npm+Nvm +w3). since the value of weights is very small so the value of w3 is incomparable to the rest of the complexity and can be eliminated. therefore, the final complexity of algo2 is O(Npm+Nvm). Hence, total time complexity for our BPGA model is calculated as O(NvmNpmO(algo2)+Npm) = O(NvmNpm(Npm+Nvm)+Npm)

For the better understanding of BPGA algorithm, an example has been provided. Suppose there are 6 physical machines (M1, M2......M6) and 12 VMs (V1, V2. V12) that has to be mapped over these physical machines, such that fewer number of PMs are used for their mapping (as similar in the case of bin packing). As discussed earlier, now these VMs have multiple options for their mapping, therefore, requested resources of VMs and capacity of PMS will be checked using equation 6.15, 6.16 for the creation of VM clusters. Suppose the array for the capacity of resource dimension of PMs are [1.5, 0.9, 1.2, 0.5, 1.0, 0.7] which is the total resource dimension of CPU and memory utilization and the array dimension for requested resources of VMs are [1.1, 0.4, 1.4, 1.6, 0.5, 0.8, 1.8, 1.7, 0.6, 1.3, 1.2, 1.5]. Thus, VM clusters will be created as VM1 (PM1, PM3) i.e. VM cluster 1 contains two physical machines for the mapping of VM1 and similarly VM cluster 2: VM2 (PM1, PM2, PM3, PM4, PM5, PM6), VM cluster 3: VM3 (PM1), VM cluster 4: VM5 (PM1, PM2, PM3, PM5), VM cluster 5: VM6 (PM1, PM2, PM3, PM5), VM cluster 6: VM9: (PM1, PM2, PM3, PM5, PM6), VM cluster 7: VM10 (PM1), VM cluster 8: VM11 (PM1, PM3). No VM cluster has been obtained for VM4, VM7, VM8 and VM12 and therefore, these VMs will be forwarded to pass 2 of BPGA model in which appropriate host for these VMs should be selected using BPNN. Using NSGA VM clusters will be arranged according to the ranking of physical machines such as: VM1 (PM3, PM1) with1.2 and 1.5 resource capacity for PM3, PM1 respectively and similarly others clusters are: VM2 (PM4, PM6, PM2, PM5, PM3, PM1), VM3 (PM1), VM5 (PM2, PM5, PM3, PM1), VM6 (PM2, PM5, PM3, PM1), VM9 (PM6, PM2, PM5, PM3, PM1), VM10 (PM1), VM11 (PM3, PM1). Furthermore, the value of fitness function 2, equation 6.24 will be calculated for these VM clusters and it will provide the value of energy consumption of each PM and thus, a physical machine which provides minimum energy consumption will be selected for that VM from every cluster. Suppose values of energy consumption for each PM are: [5Kwh, 3Kwh, 7Kwh, 8Kwh, 2Kwh, 4Kwh] and accordingly VM clusters will be again arranged and provide the final mapping of VMs. (VM1-PM1) i.e. VM1 will be mapped over PM1 and similarly, (VM2-PM5), (VM3-PM1), (VM5-PM5), (VM6-PM5), (VM9-PM5), (VM10-PM1), (VM11-PM1). Thus, the final mapping using NSGA minimize the number of the host used after mapping which directly minimizes the power consumption of datacenter. Moreover, the pending VMs which could not satisfy the above-mentioned equation 6.15, 6.16 and could not find the host for mapping will be forwarded to pass 2. In which optimization using BPNN will provide a most appropriate host for mapping along with minimizing the level of SLA violation.

6.6 PERFORMANCE EVALUATION

To evaluate the performance improvement made by BPGA VM Placement model, we have compared it with three existing reference algorithms such as GA (Genetic Algorithm), ACO (Ant Colony Optimization) and our proposed NSGA. These reference algorithms are briefly described as follows:

Genetic algorithm: GA an evolutionary optimization technique tries to provide the optimal solution. The problem of these algorithms deals with computation time. GA are beneficial to use instead of greedy algorithms because they have huge search space and also, they have the ability to automatically adjust the search space with the help of their genetic operators. Thus, GA computes the effect of the system after the deployment of new resources and chooses the solution which will have least effects on the system. Thus, GA provides optimal results for VM placement problem by taking more computational time.

ACO: Ant Colony Optimization is also a meta-heuristic approach for the search of optimal solutions by using a probabilistic technique which solves the problems of NP class. ACO algorithm deals with the process of food discovering of actual ants. Here the probabilistic technique is practised by the ants for the searching of their food. They choose the routes which have high pheromone. During the discovery of their food they dreg the pheromone on their way back to provide direction to other ants to trail the food. Although, they have a positive feedback mechanism to get optimal solutions these ants act as multi agent system and thus, create some complex solutions for solving the problems like bin packing.

NSGA: Non-dominated Sorting Genetic Algorithm considers pare to optimal solutions for finding the optimal solution for multi-objective problem. The process of VM placement has been already solved by NSGA in previous chapter. Here, we have also compared it with the proposed BPGA model to understand how it affects the VM mapping when we consider it alone and when we use it with combination of BPNN.

6.6.1 Experimental Set up

Since it is very difficult and expensive to perform repeatability of experiments in real time cloud environment which provide the infrastructure of the large scale virtualized data center. Therefore, we have chosen the way of simulation for performance evaluation and for which we have used CloudSim toolkit. It is an existing simulation framework having the capability to implement energy aware algorithms by providing the services to allocate physical machines to virtual machines according to customized procedures and it also helps to model the data center network topologies and many more. We have performed the simulations considering a data center environment which contains 800 machines i.e. servers of six different types and their configuration has been already provided in chapter 3. Similarly, VM instances are taken from Amazon EC2 shown in table 6.4 which corresponds to five different types of instances that show a number of resources requested by VMs. Moreover, for simulation environment, we have considered real time workload conditions of 10 different days with data provided by CoMoN Project. Their data contains CPU utilization of more than thousand VMs (each in 5-minute intervals) located on 500 different servers around the world.

Table 6.4: Instances of VMs taken from Amazon EC2

VM Type	CPU(MIPS)	RAM(GB)
Extra-large (high memory)	3000	6
Medium (high CPU)	2500	0.85
Extra large	2000	3.75
Small	1000	1.75
Micro	500	0.613

6.6.2 Results and Discussion

800 hosts have been used for the simulations, whereas a number of runs for the simulation are 10 with a different number of VMs on every run (with different workload condition). Total energy consumption, the number of the host used after placement, migration count, SLA Violations occur during VM placement and Cost of data center have been calculated through proposed BPGA-VMP model. The obtained results have been compared with GA, ACO and NSGA based techniques for VM placement, and their comparison are shown in figures 6.2−6.7.

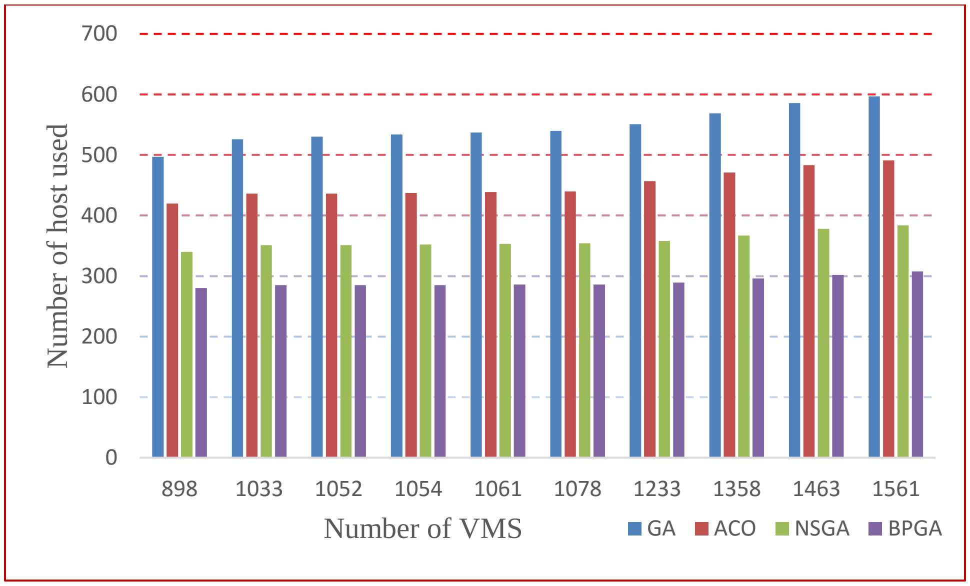


Figure 6.2: Number of host V/s number of VMs

Figure 6.2 shows the comparative view of four VM placement techniques i.e. GA, ACO, NSGA and BPGA on the basis of a number of the host used by VMs after VM placement during ten different runs of simulation. Numbers of virtual machines are increasing on every run for a fixed number of hosts. As, most of the servers inside data center are idle and consumes unnecessary power, which will increase the energy consumption of data center therefore, it is important to minimize the usage of hosts and to prevent this situation by setting the idle hosts to sleep or hibernate mode. Obtained results depict that BPGA technique uses minimum number of machines for the mapping of virtual machines in comparison to other three techniques. This is because BPGA optimization model provides the global search and chooses the most appropriate host for VMs using multi-objective constraints along with BPNN optimization which results in the optimal utilization of hosts with minimum migration count which reduces the wastage of energy consumption.

Similarly, Figure 6.3 depicts the number of migration occurred during each four techniques. BPGA performs a lesser number of migration counts in contrast to GA, NSGA and ACO on every run of simulation. BPGA is capable for finding the best physical machine for the mapping of VMs by taking lesser number of VM migrations without compromising the energy consumption, as the excess of VM migration degrades the performance of the system and leads to increase in energy consumption. Thus, with lesser number of migration used for mapping of VMs within a lesser number of hosts, BPGA VM Placement consumes lesser amount of energy consumption in comparison to GA, NSGA and ACO as shown in Figure 6.4. Thus, the tendency to minimize the energy consumption depends upon the reduction of active host and migration count of VMs, otherwise more energy been wasted during the migrations of VMs. BPGA VM Placement technique attains the lesser amount of energy consumption using minimum migrations and maps the VMs on a fewer number of hosts as possible.

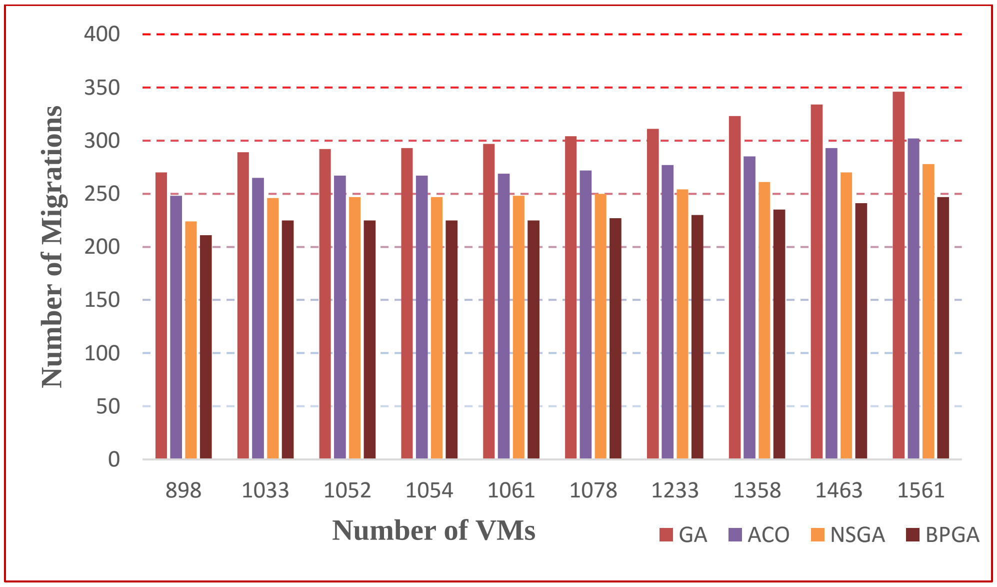


Figure 6.3: Migration count V/s Number of VMs

Both the terms energy and SLA are interrelated, utilization of resources with minimization of energy always increases the risk of SLA violations. Therefore, dealing with both of these factors simultaneously is very important. Here, in our simulations level of SLA violation is also lesser in the case of BPGA VM Placement technique in contrast to others. Thus, figure 6.5 shows the level of SLA violation for four different techniques, where the level of SLA Violations is the minimum for BPGA followed by NSGA, ACO, and GA. The increase in the value of lower threshold of CPU utilization increases the energy consumption and which further increases the level of SLA Violations and therefore, we have used median based approach for the selection of threshold values for the utilization of host thus, the chances of increase in SLA Violations reduces. Furthermore, figure 6.6 (a) shows the overall ESM value for the combined results of Energy consumption, SLA Violation and migration count and (b) depicts the cost of VM Placement for all four techniques which is also calculated from the power consumption of host and SLA violation occurs by the host for a particular period of time frame using equation 6.18.

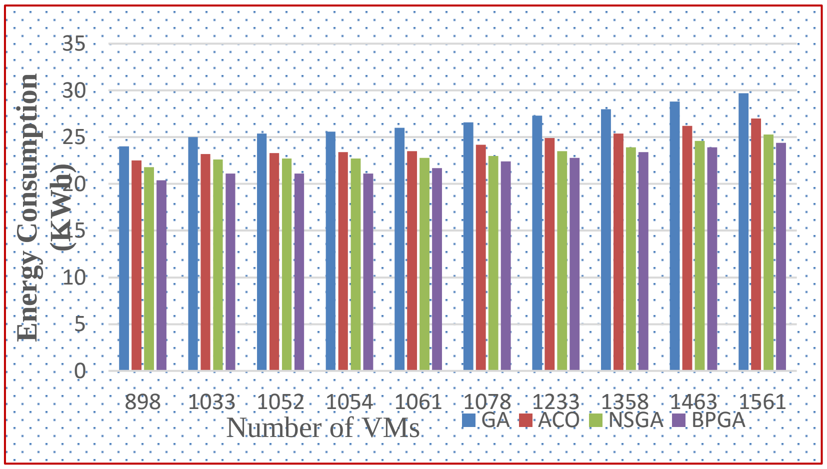


Figure 6.4: Energy Consumption during different VM Placement policies

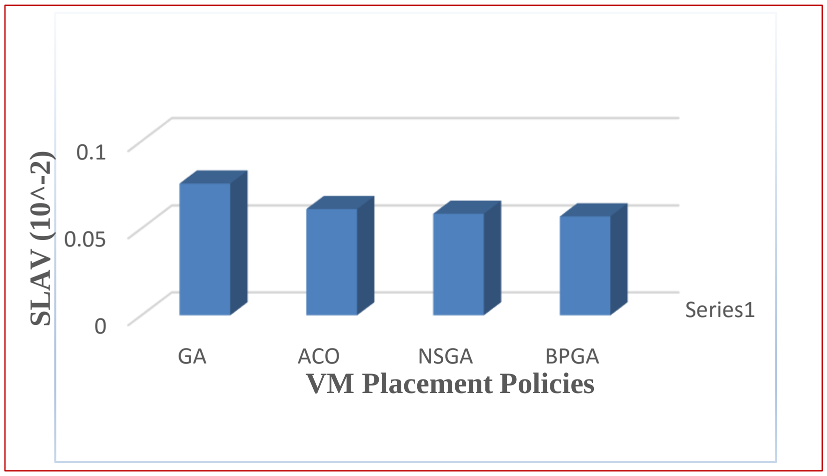
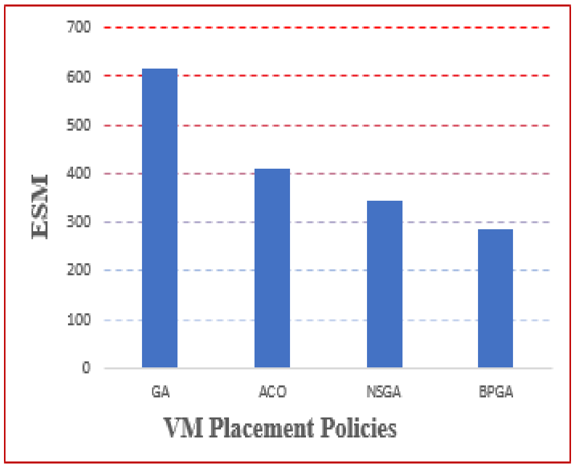


Figure 6.5: Level of SLA Violation during different VM Placement policies

(a)



(b)

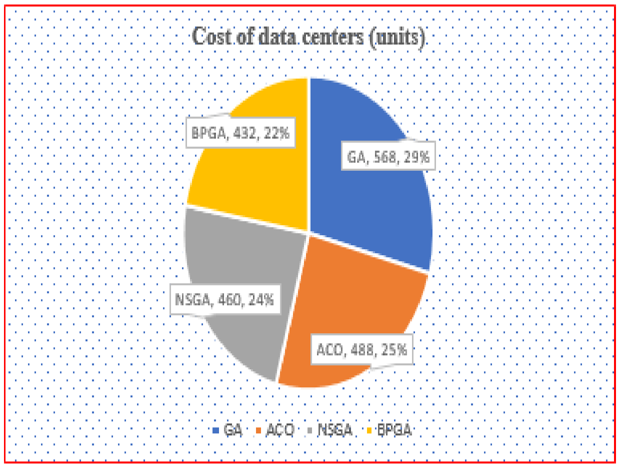


Figure 6.6: (a) Comparative values for ESM metric taken by different VM Placement policies. (b) Cost of data center using various VM Placement Policies

Figure 6.2-6.6 shows the reliability and proficiency of proposed technique for VM placement. The outcomes of the simulations show that an average 19.8% of energy has been improved in comparison to basic GA-VMP, 9.58% and 4.5% of energy has been saved in comparison to ACO based VM placement and NSGA-VMP. Thus, the improvement of the energy consumption using minimum hosts shows the effectiveness of proposed solution.

6.7 SUMMARY

Energy efficiency of cloud environment became a popular area for research in recent years. Growing demand for cloud computing increases the extensive usage of data centers which consumes large amount of energy and causes large emission of carbon. Therefore, effective management of energy consumption is important for data centers. Presently, there are many researchers trying to implement the bio-inspired techniques for handling these growing energy crises in different areas. Similarly, for current work, we have chosen multi-objective optimization algorithm along with neural network training algorithm for finding the most suitable hosts for virtual machines. The reasons for choosing them are their huge searching space and faster convergence speed.

Furthermore, this chapter proposed an energy aware VM placement model i.e. BPGA which performs migration of VMs from single active host to other in order to minimize the usage of hosts. BPGA model follows the divide and conquer approach and solving the problem of VM placement in two passes to provide the optimal solutions. Therefore, proposed model can be effectively used for VM placement and improvement in results provides their efficacy with respect to other three existing algorithms. BPGA provides 71.7%, 62.2% and 46.5% improvement in energy consumption in comparison to PABFD, GA and ACO respectively. similarly, 61.9%, 59.1% and 41.3% improvement in the number of host usage and finally 79.3%, 72.04% and 56.9% improvement in minimizing the number of VM migrations. Along with this BPGA also minimizes the cost of VM Placement by 28%, 26% and 10% with respect to other three. Thus, the proposed technique contributes toward green computing by minimizing the energy consumption of data center by reducing the number of migrations and host usage.

CHAPTER 7

CONCLUSION AND FUTURE DIRECTIONS

This chapter summarizes the work presented in a thesis on energy efficient virtual machine consolidation. It also highlights the main findings as well as discusses the future research directions and research problem in the area of energy efficient cloud data centers.

7.1 CONCLUSION

The working environment of IT industry has revolutionized the trend of utility based computing i.e. cloud computing. Due to this increasing trend, the demand for cloud data centers are increasing day by day which consumes world's main computing resources and results a huge amount of energy consumption. Therefore, the efficient management of data center's resources and CO_2 emission are two important issues to handle for environmental sustainability.

The process of dynamic consolidation has been considered as one of the most effective methods for energy minimization and improvement of resource utilization within the data centers. Therefore, this thesis has presented different novel algorithms or methods for dynamic VM consolidation while dealing with QoS. Proposed approaches minimize the energy consumption or we can say that provide a tradeoff values for energy consumption as well as SLA violation while dealing with other QoS metrics. Also, our proposed model minimizes the cost of the data center. This thesis has achieved all the objectives mentioned in chapter 1.

Chapter 2 presented the analysis and taxonomy for energy efficient data centers and VM consolidation. The study of existing literature helped us to identify the research gaps and provides us with the research direction. From chapter 3 we have proposed a solution for every step of VM consolidation process and proposed a solution for over utilized and under-utilized host detection method on the basis of the threshold value of their CPU utilization. Here, the objective of improvement of QoS has achieved. Along with this, this chapter also deals with the problem of VM selection and presented a multi-criterion based decision-making method i.e. AHP VM selection. With the proposal of this solution, we achieved the objective of energy minimization while dealing with QoS such as SLA violation and Migration count

To achieve our next objective, chapter 4 presented solution for the most interesting problem of VM consolidation i.e. VM placement. Here, we have discussed the concept of bin packing algorithm for VM placement and presented 3 different heuristics for VM placement on the basis of BFD-Best Fit Decreasing. The utilization of the resources either their current utilization or their previous utilization are two main points of consideration. However, this proposed method consumes more energy but minimizes the level of SLA violation and migration count. Thus, it improves the overall ESM metric that we have used to calculate the performance of data center and this way we have achieved our objective of data center performance improvement.

For enlighten our work in some new direction we have analyzed the trend of bin packing algorithms of VM placement and compare them with one of nature inspire algorithm i.e. basic GA- Genetic Algorithm. This performance analysis has been conducted in chapter 5, from which we have analyzed that GA provides better placement of VMs in small extent. Thus, by getting an idea of nature inspire algorithms, we have formulated the problem of VM placement as multi-objective optimization and solved it with NSGA- Non-dominated Sorting Genetic Algorithm. The main purpose of this method is to provide most appropriate host to VM. The main objective of this thesis has been achieved here i.e. Energy minimization while dealing with SLA violation.

For the achievement of our next objective i.e. cost of the data center. Chapter 6 presented a novel model for VM placement named as BPGA model. This model works with the principal of NSGA and BPNN-Back Propagation Neural Network method. Both of these NSGA and BPNN work simultaneously but in two different passes or we can say that they work parallel to achieve the objective of minimization of data center's cost but energy consumption should not be inconsiderable. Therefore, the proposal of this model is able to achieve all the objectives presented in this thesis such as energy minimization of the data center while dealing with the QoS delivered to the users.

7.2 FUTURE RESEARCH DIRECTION

Even though this thesis presented its contribution towards the area of energy efficient VM consolidation but still there are several open research challenges that have not been considered in this thesis and need to be addressed in future:

- The process of VM consolidation involves the communication of VMs and establishment of virtual networks. During VM migrations, these VMs may be hosted on far located physical machines and increase the cost of data transfer. Thus, these communicating VMs should be mapped in such a manner that there should be the minimum cost of data transfer.

- The increase of the utilization of computing resources consumes huge amount of electrical energy which is transformed into heat and leads to the problems such as life time of hardware, availability of hardware and others. Therefore, to keep the components of computing devices or hardware in a safe operation state or prevent them from hardware failure or system crashes, it is important to deal with the cooling of these components. But to minimize the cost of cooling operation at a same time is also important. Therefore, it is necessary to investigate some new methods for the reallocation of VMs in order to avoid the problem of overheating of computing resources.

- The most important benefit of cloud computing is to deliver the QoS services to users mentioned in the SLA agreement signed between user and providers. The requirements of these users may vary over time therefore, it is very important to design some new algorithms that consider the time variations in SLA with the usage of a minimum number of physical servers i.e. without increasing the cost of the infrastructure of the data center. These heterogeneous requirements of users may also make the process of VM consolidation a little bit complex. Thus, the design of new algorithms that can consider this heterogeneity of the resources to meet the requirements of cloud user by enlightening the concept of energy efficiency with in data centers will be the most significant solution for the problems related to these issues.

- The growth of energy consumption due to data centers has become tremendous issue and therefore, the minimization of energy consumption is very important. As similar to the consideration of computing resources of data centers, the hardware components of data center also play a significant role for this growing energy consumption. Therefore, it is very important and beneficial to deal with the hardware devices such as: racks, switches, cooling components for energy minimization.

- Future work is planned for the evaluation of proposed models or algorithms in Cloud Stack a real infrastructure for the cloud environment. Moreover, the direction for future research will be implementing energy aware resource allocation algorithms using different configurations of data centers such as network topology, cooling structure and many others.

APPENDIX

The appendix of this thesis includes the code of the accomplished work. This thesis reports the VM consolidation in cloud environment which has been done in four different subparts discussed in chapter 3 to 6. Thus, the complete code is also provided in different parts which includes: selection of over utilized and under-utilized host machine, selection of VM for migration, placement of VMs over most optimal host machine and finally turn off the idle machines.

```
package org.cloudbus.cloudsim.power;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Set;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.HostDynamicWorkload;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.power.lists.PowerVmList;
import org.cloudbus.cloudsim.util.ExecutionTimeMeasurer;
import org.cloudbus.cloudsim.util.MathUtil;
// Prints the over utilized hosts.
protected void printOverUtilizedHosts(List<PowerHostUtilizationHistory> overUtilizedHosts)
        {
                if (!Log.isDisabled()) {
                        Log.printLine("Over-utilized hosts:");
                        for (PowerHostUtilizationHistory host : overUtilizedHosts) {
                                Log.printLine("Host #" + host.getId());
                        }
                        Log.printLine();
                }
        }
        // Checks if is host over utilized after allocation.
        // host the host
        //vm the vm
        // return true, if is host over utilized after allocation
        protected boolean isHostOverUtilizedAfterAllocation(PowerHost host, Vm vm)
        {
                boolean isHostOverUtilizedAfterAllocation = true;
                if (host.vmCreate(vm))
                {
                        isHostOverUtilizedAfterAllocation = isHostOverUtilized(host);
                        host.vmDestroy(vm);
                }
                return isHostOverUtilizedAfterAllocation;
```

```
          }
// Gets the over utilized hosts.
protected List<PowerHostUtilizationHistory> getOverUtilizedHosts()
          {
          List<PowerHostUtilizationHistory>overUtilizedHosts=new
LinkedList<PowerHostUtilizationHistory>();
                    for(PowerHostUtilizationHistory host : this.<PowerHostUtilizationHistory>getHostList())
                    {
                              if (isHostOverUtilized(host))
                              {
                                        overUtilizedHosts.add(host);
                              }
                    }
                    return overUtilizedHosts;
          }
// Gets the under utilized host.
protected PowerHost getUnderUtilizedHost(Set<? extends Host> excludedHosts)
          {
                    double minUtilization = 0;
                    PowerHost underUtilizedHost = null;
                    for (PowerHost host : this.<PowerHost> getHostList())
                              {
                              if (excludedHosts.contains(host))
                              {
                                        continue;
                              }
                    double utilization = host.getUtilizationOfCpu();
                    minUtilization= getHostUtilizationMedlower(host);
                              if(utilization < minUtilization)
                              {
                                        underUtilizedHost = host;
                              }
                              }
          return underUtilizedHost;
          }
protected double getHostUtilizationMedlower(PowerHost host)
          {
                    double[] data = new double[host.getUtilizationHistory().length];
                    data = host.getUtilizationHistory();
                    System.out.println(data.length);
                    return MathUtil.medlower(data);
          }
public class MedianMethod extends PowerVmAllocationPolicyMigrationAbstract
          {
          private double safetyParameter = 10;
          /** The fallback vm allocation policy. */
private PowerVmAllocationPolicyMigrationAbstract fallbackVmAllocationPolicy;
public MedianMethod(List<? extends Host> hostList, PowerVmSelectionPolicy, vmSelectionPolicy,
double safetyParameter, PowerVmAllocationPolicyMigrationAbstract fallbackVmAllocationPolicy,
double utilizationThreshold)
```

```
 {
   super(hostList, vmSelectionPolicy);
  setSafetyParameter(safetyParameter);
  setFallbackVmAllocationPolicy(fallbackVmAllocationPolicy);
}
      public MedianMethod( List<? extends Host> hostList, PowerVmSelectionPolicy vmSelectionPolicy,
double safetyParameter, PowerVmAllocationPolicyMigrationAbstract fallbackVmAllocationPolicy)
        {
                super(hostList, vmSelectionPolicy);
                setSafetyParameter(safetyParameter);
                setFallbackVmAllocationPolicy(fallbackVmAllocationPolicy);
        }
//Override
protected boolean isHostOverUtilized(PowerHost host)
        {
                PowerHostUtilizationHistory _host = (PowerHostUtilizationHistory) host;
                double upperThreshold = 0;
                try
                        {
                        upperThreshold = 1 - getSafetyParameter() * getHostUtilizationMed(_host);
                        }
                           catch (IllegalArgumentException e)
                              {
                                 return getFallbackVmAllocationPolicy().isHostOverUtilized(host);
                              }
                addHistoryEntry(host, upperThreshold);
                double totalRequestedMips = 0;
                for (Vm vm : host.getVmList())
                        {
                        totalRequestedMips += vm.getCurrentRequestedTotalMips();
                        }
                double utilization = totalRequestedMips / host.getTotalMips();
                return utilization > upperThreshold;
       }
//Gets the host utilization med.
Protected double getHostUtilizationMed (PowerHostUtilizationHistory host) throws
IllegalArgumentException
      {
        double[] data = host.getUtilizationHistory();
        //if (MathUtil.countNonZeroBeginning(data) >= 12)
            {
                // 12 has been suggested as a safe value
                return MathUtil.med(data);
            }
      }
// to calculate lower threshold
public double getHostUtilizationMedlower(PowerHostUtilizationHistory host) throws
IllegalArgumentException
{
        double[] data = host.getUtilizationHistory();
		return MathUtil.medlower(data);
}
}
// selection of VMs using AHP method
package org.cloudbus.cloudsim.power;
import java.util.List;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.core.CloudSim;
public class VmselectionPolicyAHP extends PowerVmSelectionPolicy
{
		private PowerVmSelectionPolicy fallbackPolicy;
		protected List<PowerVm> vmList1;
		public VmselectionPolicyAHP()
		{
				super();
				setFallbackPolicy(fallbackPolicy);
		}
		public PowerVmSelectionPolicy getFallbackPolicy()
		{
				return fallbackPolicy;
		}
		private void setFallbackPolicy(PowerVmSelectionPolicy fallbackPolicy)
		{
				this.fallbackPolicy = fallbackPolicy;
		}
		public Vm getVmToMigrate(final PowerHost host)
		{
		List<PowerVm> migratableVms = getMigratableVms(host);
				if (migratableVms.isEmpty())
				{
						return null;
				}
				double[][] metrics = null;
				try
				{
				metrics= getMaxscore(getminUtil(migratableVms), getminRam(migratableVms),
getmigrationtime(migratableVms));
				}
				catch (IllegalArgumentException e)
				 {
						return getFallbackPolicy().getVmToMigrate(host);
				}
				double max=0;
				int maxIndex =0;
				for(int i=0;i<metrics.length;i++)
				{
						if(metrics[i][0]>max)
						{
						max=metrics[i][0];
						maxIndex=i;
```

```
                        }
                }
                return migratableVms.get(maxIndex);
                                        }
private double[][] getMaxscore(double[][] getminUtil,double[][] getminRam, double[][] getmigrationtime)
               {
                int j= vmList1.size();
                int i=0,l;
                double[][] AHP= new double[j][3];
                double[][] AHPnew= new double[j][3];
                double[][] score= new double[j][1];
                for(i=0;i<j;i++)
                        {
                        for(l=0;l<3;l++){
                                AHP[i][l]= getminUtil[i][l]+getminRam[i][l]+getmigrationtime[i][l];
                        }
                }
        double mex= 0;
        double mex1= 0;
        double mex2= 0;
        for(i = 0;i<j;i++)
                {
                  if(AHP[i][0] > mex)
                  mex=AHP[i][0];
                }
                 for(i = 0;i<j;i++)
                   {
                        if(AHP[i][1] > mex1)
                        mex1=AHP[i][1];
                   }
                     for(i = 0;i<j;i++)
                        {
                        if(AHP[i][2] > mex2)
                        mex2=AHP[i][2];
                        }
                  i=0;
                  while(i<j)
                  {
                         l=0;
                        AHPnew[i][l]= AHP[i][l]/mex;
                        AHPnew[i][l]=(AHPnew[i][l]*0.50);
                        i++;
                  }
                  i=0;
                  while(i<j)
                  {
                        l=1;
                        AHPnew[i][l]= AHP[i][l]/mex1;
                        AHPnew[i][l]= (AHPnew[i][l]*0.25);
                        i++;
                        }
                        i=0;
                        while(i<j)
                        {
                                l=2;
                                AHPnew[i][l]= AHP[i][l]/mex2;
                                AHPnew[i][l]= (AHPnew[i][l]*0.25);
                                i++;
                        }
                        for(i=0;i<j;i++)
                        {
                                score[i][0]=(AHPnew[i][0]*30)+(AHPnew[i][1]*30)+(AHPnew[i][2]*40);
                        }
                                    return score;
                      }
protected double[][] getminUtil(final List<PowerVm> vmList)
            {
                vmList1=vmList;
                int n = vmList.size();
                double[][]utilization = new double[n][3];
                for(int i=0;i<n;i++)
                  {
                    Vm vm;
                    vm= vmList.get(i);
                    double vmutilization = m.getTotalUtilizationOfCpuMips(CloudSim.clock())/vm.getMips();
                        utilization[i][0] = vmutilization;
                        utilization[i][1] = 0;
                        utilization[i][2] = 0;
                  }
            return utilization;
            }
protected double[][] getminRam(final List<PowerVm> vmList)
            {
            int n=vmList.size();
            double[][]ramutilization = new double[n][3];
            for(int i=0;i<n;i++)
                  {
                    Vm vm;
                    vm= vmList.get(i);
                    Double vmutilization = (double) vm.getRam();
                      ramutilization[i][1]= vmutilization;
                      ramutilization[i][0]= 0;
                      ramutilization[i][2]= 0;
                  }
            return ramutilization;
           }
protected double[][] getmigrationtime(final List<PowerVm> vmList)
            {
            int n= vmList.size();
            double[][]migtime = new double[n][3];
            for(int i=0;i<n;i++)
                {
                    Vm vm;
                    vm= vmList.get(i);
                    double vmutilization = vm.getRam()/vm.getBw();
                    migtime[i][2]= vmutilization;
                    migtime[i][0]= 0;
                    migtime[i][1]= 0;
                }
          return migtime;
        }
}
//VM Placement using bin packing techniques ARBF H1
public PowerHost findHostForVm (Vm vm, Set<? extends Host> excludedHosts)
{
      double minPower = Double.MAX_VALUE;
      PowerHost allocatedHost = null;
      Double MR= null;
      for (PowerHost host : this.<PowerHost> getHostList())
          {
            if (excludedHosts.contains(host))
            {
            continue;
            }
                 if(host.isSuitableForVm(vm))
                      {
                      if (getUtilizationOfCpuMips(host) != 0 && isHostOverUtilizedAfterAllocation(host, vm))
                                {
                                continue;
                                }
                                try
                                   {
                                      double hutil =sqrt(host.getAvailableMips()-host.getTotalMipsallocatedforVm(vm));
                                       MR= sqrt(hutil/host.getPreviousUtilizationMips());
                                          if ( MR < minPower)
                                                    {
                                                    minPower = MR;
                                                    allocatedHost = host;
                                                    }
                                          }
                                 catch (Exception e)
                                {

                                }
                      }
            } return allocatedHost;
}
//VM placement using ARBF H2
public PowerHost findHostForVm (Vm vm, Set<? extends Host> excludedHosts)
{
      double minPower = Double.MAX_VALUE;
      PowerHost allocatedHost = null;
      double MR= null;
```

```
        double AvailRAM= double.MAX_VALUE;
        for (PowerHost host : this.<PowerHost> getHostList())
            {
              if (excludedHosts.contains(host))
              {
              continue;
              }
                    if(host.isSuitableForVm(vm))
                          {
                          if (getUtilizationOfCpuMips(host) != 0 && isHostOverUtilizedAfterAllocation(host, vm))
                                      {
                                      continue;
                                      }
                                      try
                                         {
                                  double hutil =sqrt(host.getPreviousUtilizationMips()-host.getAvailableMipsofHost());
                                 AvailRAM = (host.getPreviousUtilizationofRam() – host.getCurrentUtilizationofVm(vm));
                                 MR= sqrt(hutil+AvailRAM);
                                                if ( MR < minPower)
                                                            {
                                                            minPower = MR;
                                                            allocatedHost = host;
                                                            }
                                                }
                                       catch (Exception e)
                                      {

                                      }
                          }
              } return allocatedHost;
}
//VM placement using ARBF H3
public PowerHost findHostForVm (Vm vm, Set<? extends Host> excludedHosts)
{
        double minPower = Double.MAX_VALUE;
        PowerHost allocatedHost = null;
        double MR= null;
        double AvailRAM= double.MAX_VALUE;
        for (PowerHost host : this.<PowerHost> getHostList())
            {
              if (excludedHosts.contains(host))
              {
              continue;
              }
                    if(host.isSuitableForVm(vm))
                          {
                          if (getUtilizationOfCpuMips(host) != 0 && isHostOverUtilizedAfterAllocation(host, vm))
                                      {
                                      continue;
                                      }
                                      try
                                         {
                                    MR= sqrt(host.getAvailableUtilizationofMipsofhost()/host.getPreviousUtilizationofMips());
                                                if ( MR < minPower)
                                                            {
```

```
                                                   minPower = MR;
                                                   allocatedHost = host;
                                                   }
                                          }
                                 catch (Exception e)
                                {

                                }
                      }
            } return allocatedHost;
}
//VM placement using GA
package org.cloudbus.cloudsim.power;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Random;
import java.util.logging.Logger;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.util.ExecutionTimeMeasurer;

public class PowerVMPlacementPolicyUisngGA extends PowerVmAllocationPolicyMigrationAbstract
{
//final static double crossoverrate =0.2;
public static Random rnd;
List<GA> pop;
Public PowerVMPlacementPolicyUisngGA(List<? extends Host> hostList,PowerVmSelectionPolicy
vmSelectionPolicy)
 {
  super(hostList, vmSelectionPolicy);
   Log.setDisabled(true);
}
 public List<Map<String, Object>> optimizeAllocation(List<? extends Vm> vmList)
  {
     ExecutionTimeMeasurer.start("optimizeAllocationTotal");
     List<Map<String, Object>> migrationMap = new LinkedList<Map<String, Object>>();
      // populate migrationMap here
      ExecutionTimeMeasurer.start("optimizeAllocationHostSelection");
      initGA();
getExecutionTimeHistoryHostSelection().add(ExecutionTimeMeasurer.end("optimizeAllocationHostSelection"));
while(true)
{
    try
       {
          migrationMap = pop.get(rnd.nextInt(pop.size())).getMap();
           break;
        } catch (Exception e)
              {
                }
        }
getExecutionTimeHistoryTotal().add(ExecutionTimeMeasurer.end("optimizeAllocationTotal"));
return migrationMap;
```

```
}
protected boolean isHostOverUtilized(PowerHost host)
{
     return false;
}
 private void initGA()
          {
             rnd = new Random();
             pop = new ArrayList<GA>();
             for (int i = 0; i < 5; i++)
                      {
                          try {
                                   addToPopulation(new GA(this));
                               } catch (Exception e)
                                      {
                                      System.out.println(e.getMessage());
                                      }
                      }
                          for (int i = 0; i < 10; i++)
                            {
                                      mutation();
                                      crossover();
                            }
             }
private List<GA> addToPopulation(GA ind)
         {
             if (pop.size() <50)
                    {
                          pop.add(ind);
                    }
             return pop;
          }
private void crossover()
         {
             GA p1, p2;
             p1 = pop.get(rnd.nextInt(pop.size()));
             do
                 {
                  p2 = pop.get(rnd.nextInt(pop.size()));
                 }
             while (p1.equals(p2));
                   try {
                          addToPopulation(new GA(p1, p2));
                          }
                          catch (Exception e)
                          {
                          }
             }
private void mutation()
         {
             for (GA ind : pop)
                   {
                      if (( rnd.nextInt(100) < 10))
                           {
                              try
```

```
                                    {
                                      ind.Mutation();
                                     } catch (Exception e)
                                                    {
                                                    }
                              }
                      }
             }
}
//VM Placement using NSGA
package org.cloudbus.cloudsim.power;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Random;
import java.util.logging.Logger;
import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.util.ExecutionTimeMeasurer;

public class PowerVmAllocationPolicyMigrationGA extends PowerVmAllocationPolicyMigrationAbstract
 {
  public static Random rnd;
  List<GAInd> pop, pareto;
  public PowerVmAllocationPolicyMigrationGA(List<? extends Host> hostList, PowerVmSelectionPolicy
vmSelectionPolicy)
       {
         super(hostList, vmSelectionPolicy);
        }
public List<Map<String, Object>> optimizeAllocation(List<? extends Vm> vmList)
    {
        ExecutionTimeMeasurer.start("optimizeAllocationTotal");
        List<Map<String, Object>> migrationMap = new LinkedList<Map<String, Object>>();
        // populate migrationMap here
        ExecutionTimeMeasurer.start("optimizeAllocationHostSelection");
        initGA();
        getExecutionTimeHistoryHostSelection().add(
ExecutionTimeMeasurer.end("optimizeAllocationHostSelection"));
           while(true)
            {
                 try {
                          migrationMap = pareto.get(rnd.nextInt(pareto.size())).getMap();
                          break;
                      }
                catch (Exception e)
                {
                 }
            }
   getExecutionTimeHistoryTotal().add(ExecutionTimeMeasurer.end("optimizeAllocationTotal"));
     return migrationMap;
}

protected boolean isHostOverUtilized(PowerHost host)
```

```
   {
            return false;
   }
private void initGA()
            {
              rnd = new Random();
              pop = new ArrayList<GAInd>();
              pareto = new ArrayList<>();
                        for (int i = 0; i < 50; i++)
                                    {
                                    try {
                                          addToPopulation(new GAInd(this));
                                         } catch (Exception e)
                                                {
                                                System.out.println(e.getMessage());
                                                 }
                                     }
                        for (int i = 0; i < 500; i++)
                                    {
                                    mutation();
                                    crossover();
                                     }
            }

            private void crossover()
                        {
                        GAInd p1, p2;
                        p1 = pop.get(rnd.nextInt(pop.size()));
                        do
                        {
                        p2 = pop.get(rnd.nextInt(pop.size()));
                        }
                        while (p1.equals(p2));
                        try
                         {
                           addToPopulation(new GAInd(p1, p2));
                         } catch (Exception e)
                         {
                        }
                        //crossover takes too much time
                        for (int i = 0; i < pop.size(); i++)
                                    {
                                    if (rnd.nextInt(100) < 90)
                                     {
                                       GAInd p1, p2;
                                        p1 = pop.get(rnd.nextInt(pop.size()));
                                                do {
                                                      p2 = pop.get(rnd.nextInt(pop.size()));
                                                    } while (p1.equals(p2));
                                                try {
                                                      addToPopulation(new GAInd(p1, p2));
                                                     } catch (Exception e)
                                                {
                                                }
                                      }
```

```
                    }
                }
private void mutation()
 {
       for (GAInd ind : pop)
             {
                  if (!ind.isPareto && rnd.nextInt(1000) < 10)
                         {
                         try
                             {
                               ind.Mutation();
                              } catch (Exception e)
                                     {
                                     }
                         }
             }
}
private void removeIndividual(GAInd ind)
{
             pop.remove(ind);
             pareto.remove(ind);
}

private boolean addToPareto(GAInd ind)
             {
             List<GAInd> dominatedInds = new ArrayList<>();
             for (GAInd target : pareto)
                    {
                      if (ind.dominates(target) == Domination.True)
                          {
                           dominatedInds.add(target);
                          }
                     else if (ind.dominates(target) == Domination.False)
                         {
                         return false;
                         }
                     }
             for (GAInd gaInd : dominatedInds)
             {
                         removeIndividual(gaInd);
             }
             if (pareto.size() < 20)
                 {
                         pareto.add(ind);
                         ind.isPareto = true;
                         return true;
                 }
             return false;
          }
private boolean addToPopulation(GAInd ind)
             {
             if (pop.size() < 50)
                 {
                   pop.add(ind);
                   addToPareto(ind);
```

```
                        return true;
                    }
                List<GAInd> dominatedInds = new ArrayList<>();
                for (GAInd gaInd : pop)
                        {
                         if (ind.dominates(gaInd) == Domination.True)
                            {
                                    dominatedInds.add(gaInd);
                            }
                        }
                if (dominatedInds.size() > 0)
                        {
                            GAInd random = dominatedInds.get(rnd.nextInt(dominatedInds.size()));
                            removeIndividual(random);
                            pop.add(ind);
                            addToPareto(ind);
                            return true;
                          }
                return false;
            }
```

```
//VM Placement using BPNN (this code has been developed in Netbeans using cloudSim libraries)
import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Scanner;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.core.CloudSim;
public class BPNN
{
static TermProcessing trp=new TermProcessing();
static File[] listOfFiles = null;
statpic String InputFiles = "C:\\Database\\BPNN\\Vms";
public static void main(String[] args)
   {
      MachineLearning mchn=new MachineLearning();
      String[] machinefeatures=new String[66];
      int hostcount=0;
      long currentneuroelapsed=0;
      double slavoilation=0;
      double energyconsumed=0;
      double totalservercost=0;
      machinefeatures=mchn.readmachineprop();
      long startTime = System.currentTimeMillis();
      long elapsedTime;
      elapsedTime = 0L;
      Scanner in = new Scanner(System.in);
      List<Vm> vmlist = null;
      BPNN bp=new BPNN();
```

```
bp.listFiles();
vmlist = new ArrayList<Vm>();
int changefitness=0;
int num_user = 1; // number of cloud users
Calendar calendar = Calendar.getInstance(); // Calendar initialized with the current date and time.
boolean trace_flag = false; // trace events
int[][] cando=new int[1000][2];
int cancount=0;
int allocationcount=0;
CloudSim.init(num_user, calendar, trace_flag);
String cpupath="";
String[] TxttoWord_1;
TxttoWord_1 = trp.processDoc(listOfFiles[0].getAbsolutePath());
String currentvalue=TxttoWord_1[0];
int slen=currentvalue.length();
int vmss=TxttoWord_1.length;
int Machines = 7;
int hosts=0;
System.out.println("Enter total desired Hosts :");
hosts=in.nextInt();
double[] allhosts=new double[1000];
vmss=currentvalue.length();
int[] vmssId = new int[vmss];
double[] vmssenergy = new double[vmss];
double[] vmssCpu = new double[vmss];
double[] vmssTime = new double[vmss];
double[][] allocated_final=new double[50][50];
vmsSpecification JS = new vmsSpecification();
vmssId = JS.Id(vmss);
vmssenergy = JS.energyUtilization(vmss);
  for(int iso=0;iso<currentvalue.length();iso++)
       {
         vmssCpu[iso]=(double)(currentvalue.charAt(iso));
       }
vmssTime = JS.vmsTime(vmss);
int[] MachinesId = new int[Machines];
double[] Machinesenergy = new double[Machines];
double[] MachinesCpu = new double[Machines];
double[] MachinesTime = new double[Machines];
double[] Servercost=new double[Machines];
int types=Math.round(hosts/Machines);
MachineSpecification MS = new MachineSpecification();
Servercost=MS.MachineCost(Machines);
MachinesId = MS.Id(Machines);
Machinesenergy = MS.energyUtilization(Machines);
MachinesCpu = MS.CpuUtilization(Machines);
MachinesTime = MS.MachineTime(Machines);
 //initialize GA
 int gapopulation=0;
 gapopulation=MachinesId.length;
```

```
  // initializing the forward pass
  double fitvalue=0;
fitvalue=Machinesenergy[0];
int currentelement=0;
double Fitness_value=0;
gaobj1 ga1=new gaobj1();
for(int i=0;i<hosts;i++)
   {
     allhosts[i]=9999;
   }
System.out.println("Fitness Value:"+fitvalue);
  for(int i=0;i<gapopulation;i++)
     {
        currentelement=MachinesId[i];
        double mym=0;
        mym=Machinesenergy[i];
           for(int sp=0;sp<Machinesenergy.length;sp++)
             {
                if(Machinesenergy[sp]>mym)
                  {
                     mym=Machinesenergy[sp];
                  }
             }
    Fitness_value=mym;
    int flag_fit=0;
    int stop_searching=0;
         for(int j=0;j<vmssId.length;j++)
             {
                if(stop_searching==0)
                 {
                    double checkvalue=0; // this value would be compared with the fitness value ;
                    int flag_newfit=0;
                    checkvalue=vmssenergy[j];
                    flag_fit=ga1.findfit(Fitness_value,checkvalue);
                        if(flag_fit==1)
                         {
                           int crvms=vmssId[j];
                                try
                                    {
                                       cando[cancount][0]=currentelement;
                                       System.out.println(currentelement);
                                       System.out.println(cando[cancount][0]);
                                       cando[cancount][1]=crvms;
                                       cancount=cancount+1;
                                     }
                                catch(Exception err)
                                   {
                                   }
                         }
           else
```

```
                    {
                    flag_newfit=ga1.crossover(Fitness_value*Math.random(),checkvalue*Math.random());
                     if(flag_newfit==1)
                         {
                           stop_searching=1;
                           Fitness_value=checkvalue;
                           changefitness=changefitness+1;
                           System.out.println("Fitness Value Changed "+changefitness+"times .. New Fitness
Value:"+Fitness_value);
                           }
                     }
              }
         }
       }
for(int i=0;i<cancount;i++)
       {
         System.out.println("Server ID:"+cando[i][0]+" can perform "+cando[i][1]+" vms");
        }
         for(int i=0;i<cancount;i++)
          {
         System.out.println("Server ID:"+cando[i][0]+" can perform "+cando[i][1]+" vms");
          }
         int current_vms=0;
      for(int i=0;i<vmssId.length;i++)
         {
         current_vms=vmssId[i];
         int[] vmscluster=new int[10];
         int jbscount=0;
      // now creating the vms clusters
           for(int j=0;j<cancount;j++)
               {
                 if(current_vms==cando[j][1])
                  {
                    try
                     {
                        vmscluster[jbscount]=cando[j][0];
                         jbscount=jbscount+1;
                     }
                    catch(Exception err)
                     {
                     }
                  }
               }
      double[] currentenergy=new double[10];
          if(jbscount>0)
             {
               for(int ii=0;ii<jbscount;ii++)
                {
                   int currentjb=0;
                   currentjb=vmscluster[ii];
               try
                {
                      currentenergy[ii]=Machinesenergy[currentjb];
                   }
              catch(Exception err)
                 {
                  currentenergy[ii]=Math.random();
                  }
          }
   }
 double currentgaelement=0;
 double gacrossover=0;
 for(int y=0;y<jbscount;y++)
     {
     gacrossover=gacrossover+currentenergy[y];
     }
    gacrossover=(gacrossover/jbscount)*Math.random();
    int flag_dont=0;
      for(int kp=0;kp<jbscount;kp++)
        {
           currentgaelement=currentenergy[kp];
            if(currentgaelement>gacrossover*Math.random())
            {
               if(flag_dont==0)
              {
                 try
                  {
                    allocated_final[i][0]=vmscluster[kp];// server
                    allocated_final[i][1]=i;// vms
                    allocationcount=allocationcount+1;
                    flag_dont=1;
                  }
                 catch(Exception err)
                 {
                  }
              }
          }
      }
}
int rcount=0;
double[] remainingjobs = new double[900];
int rjobcount=0;
for(int sg=0;sg<allocationcount;sg++)
   {
      if(allocated_final[sg][0]>MachinesId.length)
         {
            remainingjobs[rjobcount]=allocated_final[sg][1];
          }
      else
         {
          System.out.println("vms :"+allocated_final[sg][1]+" Allocated to "+allocated_final[sg][0]);
          long currentelapsed=System.currentTimeMillis();
          currentelapsed=currentelapsed-startTime;
        }
   }
      for(int is=0;is<vmssId.length;is++)
          {
           int found=0;
           int found2=0;
            for(int ps=0;ps<allocationcount;ps++)
             {
              if (allocated_final[ps][1]==ps)
               {
                found=1;
               }
             if(allocated_final[ps][1]==is)
               {
                found2=1;
               }
             }
          if (found==0)
          {
            System.out.println("vms :"+is+" Allocated to "+0);
          }
           if(found2==0)
           {
              remainingjobs[rcount]=is;
              rcount=rcount+1;
           }
        }
    elapsedTime = (new Date()).getTime() - startTime;
    double[] makespan=new double[vmssTime.length];
    double[] resourceutil=new double[vmssTime.length];
    double totalmakespan=0;
    double totalcpu=0;
  for(int i=0;i<vmssTime.length;i++)
    {
     resourceutil[i]=vmssCpu[i]*elapsedTime;
     makespan[i]=vmssTime[i]*elapsedTime;
     totalmakespan=totalmakespan+makespan[i];
     totalcpu=totalcpu+resourceutil[i];
    }
  for(int is=0;is<vmssId.length;is++)
   {
    double found=0;
    for(int ps=0;ps<allocationcount;ps++)
      {
         if (allocated_final[ps][1]==ps)
          {
          found=Math.round((ps)*Math.random());
```

```
                }
            }
             if (found==0)
               {
                System.out.println("vms :"+is+" Allocated to "+0);
               }
                else
                 {
                    int found1=0;
                    found1=(int)found;
                 }
             }
      System.out.println("Total make span :"+totalmakespan/2000);
      System.out.println("Total Cpu Utilization :"+totalcpu/100000);
      int mccount=1;
      int nextmachine_value=10;
      double[][]training_features=new double[6][11];
         for(int i=0;i<rcount;i++)
           {
             System.out.println(remainingjobs[i]+",");
           }
             int condition=1;
                for(int i=0;i<machinefeatures.length;i++)
                   {
                     System.out.println("Power        Consumption        for        machine        "+mccount+"under
Condition"+condition +"is :"+machinefeatures[i]);
                      try
                         {
                          training_features[mccount-1][condition-1]=(Double.parseDouble(machinefeatures[i]));
                          }
                     catch(Exception err)
                        {
                         }
                   condition=condition+1;
                    if(condition==12)
                       {
                        condition=1;
                       }
                    if(i==nextmachine_value)
                      {
                        mccount=mccount+1;
                        nextmachine_value=nextmachine_value+11;
                      }
             }
       System.out.println(Arrays.toString(training_features));
       double[] group=new double[6];
       for(int i=0;i<6;i++)
         {
          group[i]=i+1;
         }
```

```
        startTime=System.currentTimeMillis();
        int neuralnetworkepochs=0;
GenerateHiddneLayer ghl=new GenerateHiddneLayer();
// the generate hidden layer class contains the values of the epochs / greadient and the values of the
arbritrary constants
neuralnetworkepochs=ghl.totalepochs();
int gradientsatisfied=0;
double gradient=0;
double[][] termweight=new double[6][11];
System.out.println("Total term weight :"+(termweight)) ;
termweight=training_features;
double[][] previoushwt=new double[6][11];// to store the previous state of the data
double[][] newwt=new double[6][11];
int epochrun=0;
gradient=ghl.generategradient();
   for(int i=0;i<neuralnetworkepochs;i++)
        {
            if(gradientsatisfied==0)
              {
                if (i==0)
                 {
                   newwt=ghl.generateweight(termweight);
                   previoushwt=newwt;
                  }
               else
                 {
                   newwt=ghl.generateweight(termweight);
                       for(int k=0;k<6;k++)
                           {
                             for(int l=0;l<11;l++)
                                  {
                                    newwt[k][l]=newwt[k][l]+previoushwt[k][l];
                                  }
                            }
                  }
              for(int p=0;p<newwt.length;p++)
                  {
                      for(int l=0;l<11;l++)
                           if(newwt[p][l]>=gradient)
                             {
                               gradientsatisfied=epochrun;
                              }
                          else
                            {
                              slavoilation=slavoilation+1;
                             }
                  }
            }
         epochrun=epochrun+1;
       }
```

```
        for(int i=0;i<6;i++)
          {
            for(int j=0;j<11;j++)
              {
               System.out.println("The weight of "+i+"th item is :"+newwt[i][j]);
              }
            }
        System.out.println("Gradient value is "+gradient);
        System.out.println("Gradient satisfied at  "+gradientsatisfied+1);
        System.out.println("Neural network training complete");
       double[] testset=new double[11];
          for(int i=0;i<rcount;i++)
            {
             for(int j=0;j<11;j++)
               {
                 double currente=0;
                 double currentt;
                 currente=vmssenergy[(int)(remainingjobs[i])];
                 currentt=vmssTime[(int)(remainingjobs[i])];
                 double total=currente+currentt;
                 testset[j]=total+(total*(j*10)/100);
                }
             GenerateHiddneLayer gnrt=new GenerateHiddneLayer();
             double[][] testweight=new double[1][11];
                 for(int k=0;k<11;k++)
                   {
                     testweight[0][k]=testset[k];
                   }
                    testweight=gnrt.generateweight(testweight);
                    Simulateneural simneuro=new Simulateneural();
                    double result=0;
                    result=simneuro.simneural(newwt, testweight, mccount);
                    totalservercost=totalservercost+Servercost[(int)result];
                    allhosts[hostcount]=result*types+Math.round((double)Math.random()*types);
                    System.out.println("Best suited for VM:"+remainingjobs[i]+"is :"+allhosts[hostcount]);
                    currentneuroelapsed=System.currentTimeMillis();
                    hostcount=hostcount+1;
                    currentneuroelapsed=currentneuroelapsed-startTime;
        energyconsumed= energyconsumed+intg.Integrate (0,1,(vmssenergy[(int)remainingjobs[i]]+
Machinesenergy[(int)result])* elapsedTime);
                 }
              energyconsumed=energyconsumed*(currentneuroelapsed)/60;
              long totalelapsed=System.currentTimeMillis();
              totalelapsed=totalelapsed-startTime;
              totalelapsed=totalelapsed;
              energyconsumed=energyconsumed/(2000*2);
              System.out.println("Total Energy Consumed :"+(energyconsumed)+"KWh");
              totalelapsed=(totalelapsed/(60));
             double tell=(double)totalelapsed;
             double totalhostused=0;
```

```
                    for(int i=0;i<hosts;i++)
                      {
                        double currenthost=i;
                        int hostfound=0;
                           for(int j=0;j<hostcount;j++)
                              {
                               if(allhosts[j]==currenthost)
                               {
                                 hostfound=1;
                               }
                              }
                             if(hostfound==1)
                               {
                                totalhostused=totalhostused+1;
                               }
                        }
            tell=tell/300;
            slavoilation=slavoilation/30000;
            System.out.println("Total makespan :"+(tell)+" sec");
            System.out.println("SLA VOILATION :"+slavoilation);
            System.out.println("Migration Count :"+rcount*3);
            System.out.println("Total Server Side Cost:"+totalservercost+" Units ");
            System.out.println("Total Host Used is :"+totalhostused*4);
        }
public void listFiles()
   {
       File folder = new File(InputFiles);
       listOfFiles = folder.listFiles();
       System.out.println("Getting input database...");
   }
}
//simulation of neural network
package BPNN;
public class Simulateneural {
   public double simneural(double[][] traindata,double[][]testdata,int machinecount)
   {
       double value=0;
       double testrec=0;
       machinecount=machinecount-2;
       double[] diff=new double[machinecount];
       double machinevalue=0;
          for(int is=0;is<machinecount;is++)
            {
             for(int i=0;i<11;i++)
              {
                machinevalue=machinevalue+traindata[is][i];
                testrec=testrec+testdata[0][i];
              }
             machinevalue=machinevalue/11;
             testrec=testrec/11;
            try
             {
                diff[is]=Math.abs(machinevalue-testrec)*Math.random();
              }
              catch(Exception err)
              {
                System.out.println(err);
              }
           }
        double currentmin=diff[0];
        double currentpos=0;
        for(int i=0;i<machinecount;i++)
           {
              if(currentmin>diff[i])
                 {
                   currentmin=diff[i];
                   currentpos=i;
                 }
            }
        value=currentpos;
        return value;
    }
 }
//generation of hidden layer
package BPNN;
public class GenerateHiddneLayer {
public double[][] generateweight(double[][] data)
    {
        double[][] myweight=new double[6][11];
        Generateab gab=new Generateab();
        double[] ab=new double[2];
           for(int i=0;i<6;i++)
            {
                for(int j=0;j<11;j++)
                  {
                   ab=gab.returnab();
                   // the formual for the linear weight is ax+b
                   try
                    {
                     myweight[i][j]=ab[0]*data[i][j]+ab[1];
                     }
                   catch(Exception err)
                   {
                   myweight[i][j]=Math.random();
                   }
                 }
            }
        return myweight;
     }
  public int totalepochs()
```

```
    {
        return 100;
    }
   public double generategradient()
       {
        double gradient=0;
        gradient=50+10*Math.random();
        return gradient;
    }
}
//weights processing
package BPNN;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
public class TermWeight
             {
             float dl[]=new float[2];
             float rdl[] = new float[2];
             public float termWeight(int nO,int docLen ,int termF, int total_doc)
                {
                  float W = 0, tf = 0, idfji = 0, df = 0;
                  tf = nO;
                  df = termF;
                  idfji = (float) (Math.log(total_doc / df) / Math.log(2));
                  W = Math.abs(tf * idfji);
                  return W;
                }
            public float[] getLengthValue(ArrayList<Float> weight,ArrayList<Float> weight2)
              {
               dl[0]=0;
               dl[1]=0;
               for (int docIn = 0; docIn < weight.size(); docIn++)
                     {
                        if(weight.get(docIn)>=0)
                          {
                            dl[0] +=  (Math.pow(weight.get(docIn), 2));
                          }
                      }
               for(int docIn=0;docIn < weight2.size();docIn++)
                      {
                        if(weight2.get(docIn) >=0)
                           {
                            dl[1] +=(Math.pow(weight2.get(docIn), 2));
                           }
                      }
                dl[0] = (float) Math.sqrt(dl[0]);
                dl[1] = (float) Math.sqrt(dl[1]);
                return dl;
              }
public float getCosineMetric(ArrayList<Float> Vec1,ArrayList<Float> Vec2)
{
        float cosSim = 0, numTerm = 0, denoTerm = 0;
        for (int index = 0; index < Vec2.size(); index++)
             {
             numTerm += Vec1.get(index) * Vec2.get(index);
              }
        rdl = getLengthValue(Vec1, Vec2);
        denoTerm = (rdl[0] * rdl[1]);
        if ((denoTerm > 0))
             {
               cosSim = Math.abs(numTerm / denoTerm);
              }
       return cosSim;
 }
public String[] formatDoc(String path)
{
        BufferedReader br = null;
        String expr = ",";
        String[] values = null;
           try
             {
               String CLine;
               br = new BufferedReader(new FileReader(path));
               while ((CLine = br.readLine()) != null)
               {
                values = CLine.split(expr);
                }
               br.close();
             }
              catch (Exception ert)
                {
                }
         return values;
   }
   public float findLargest(float[] data)
      {
        float largest = data[0];
        for (int x = 0; x < data.length; x++)
             {
               if (data[x] > largest)
                  {
                    largest = data[x];
                  }
             }
         return largest;
 }
```

```
public int inDocument(String term, String path)
    {
        int occ = 0;
        BufferedReader br = null;
        String[] mat = null;
        String expr = "\\s*(=>|,|\\s)\\s*";
           Try
             {
              String sCurrentLine;
              File checkFile=new File(path);
                 if(checkFile.isFile())
                  {
                      br = new BufferedReader(new FileReader(path));
                        while ((sCurrentLine = br.readLine()) != null)
                          {
                            mat = sCurrentLine.split(expr);
                          }
                   for (int wr = 0; wr < mat.length; wr++)
                         {
                            String val = mat[wr].toString().trim();
                             if (val.equalsIgnoreCase(term))
                              {
                                  occ++;
                               }
                          }
                         br.close();
                    }
                }
               catch (IOException e)
              {
                e.printStackTrace();
                System.out.println("Error in word count !");
               }
           return occ;
      }
public int outDocument(String term, String path)
 {
        int Out_cnt = 0;
        BufferedReader br = null;
        String[] mat = null;
        String expr = "\\s*(=>|,|\\s)\\s*";
        try
            {
              String sCurrentLine;
              br = new BufferedReader(new FileReader(path));
              while ((sCurrentLine = br.readLine()) != null)
                 {
                   mat = sCurrentLine.split(expr);
                  }
            for (int wr = 0; wr < mat.length; wr++)
              {
                  String val = mat[wr].toString().trim();
                      if (val.equalsIgnoreCase(term))
                         {
                          Out_cnt++;
                         }
                 }
           }
         catch (IOException e)
          {
           }
 return Out_cnt;
}
public double LinkFunction(ArrayList<Float> wt1,ArrayList<Float> wt2)
{
    double lf = 0;
        for (int i = 0; i < wt1.size(); i++)
             {
            for (int j = 0; j < wt2.size(); j++)
              {
               lf = wt1.get(i) * wt2.get(j);
              }
             }
       return lf;
    }
 public double Fmeasure(double p,double r)
    {
    double f1_measure=0;
    f1_measure=(2*(p*r)/(p+r));
    return f1_measure;
    }
}
```

REFERENCES

[1] K. Marimuthu, D. G. Gopal, K. S. Kanth, S. Setty & K. Tainwala, “Scalable and secure data sharing for dynamic groups in cloud”, *In Advanced Communication Control and Computing Technologies (ICAC-CCT), 2014 International Conference on*, pp. 1697-1701, 2014.

[2] P. Mell and T. Grance, “The NIST Definition of Cloud Computing, National Institute of Standards and Technology”, *Information Technology Laboratory, Technical Report* Version 15, 2009.

[3] M. Armbrust, A. Fox, R. Griffth, A. D. Joseph & R. Katz, “Above the Clouds: A Barkeley View of Cloud Computing”, *UC Berkeley Reliable Adaptive Distributed Systems Laboratory* White Paper, 2009.

[4] Y. Gao, H. Guan, Z. Qi, T. Song, F. Huan & L. Liu, “Service level agreement based energy-efficient resource management in cloud data centers”, *Computers & Electrical Engineering*, vol. 40, no. 5, pp. 1621-1633, 2014.

[5] R. Buyya, A. Beloglazov & J. Abawajy, “Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges”, *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications*, 2010.

[6] A. Horri, M. S. Mozafari & G. Dastghaibyfard, “Novel resource allocation algorithms to performance and energy efficiency in cloud computing”, *The Journal of Supercomputing,* vol. 69, no. 3, pp. 1445-1461, 2014.

[7] M. Poess & R. O. Nambiar, “Energy cost, the key challenge of today's data centers: a power consumption analysis of tpoc results”, *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1229-1240, 2008.

[8] K. Maurya & R. Sinha, “Energy conscious dynamic provisioning of virtual machines using adaptive migration thresholds in cloud data center”, *International Journal of Computer Science and Mobile Computing*, pp. 74-82, 2013.

[9] A. E. Ezugwu, S. M. Buhari & S. B. Junaidu, “Virtual machine allocation in cloud computing environment”, *International Journal of Cloud Applications and Computing (IJCAC),* vol. 3, no. 2, pp. 47-60, 2013.

[10] A. Beloglazov & R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers", *Concurrency and Computation: Practice and Experience,* vol. 24, no. 13, pp. 13971-420, 2012.

[11] Z. Á. Mann, "Allocation of virtual machines in cloud data centers a survey of problem models and optimization algorithms", *ACM Computing Surveys*, vol. 48, no. 1, 2015.

[12] S. K. Garg, S. K. Gopalaiyengar, R. Buyya, "SLA based Resource Provisioning for heterogenous workloads in a virtualized cloud datacenter", *In 11th international conference on Algorithms and architectures for parallel processing*, pp. 371-384, 2011.

[13] Q. H. Nguyen, T. Nam & T. Nguyen, "EPOBF: Energy Efficient Allocation of Virtual Machines in high performance computing", *Journal of Science and Technology,* vol. 51, no. 4B, pp. 173-182, 2013.

[14] K. Nakku, C. Jungwook & S. Euiseong, "Energy-credit scheduler: An energy-aware virtual machine scheduler for cloud systems", *Future Generation Computer Systems,* vol. 32, pp. 128-137, 2014.

[15] A. Khosravi, S. K. Garg & R. Buyya R, "Energy and Carbon-Efficient Placement of Virtual Machines in Distributed Cloud Data Centers", *Proceedings of Euro-Par Parallel Processing*, pp. 317-328, 2013

[16] H. F. Sheikh, H. Tan, I. Ahmad & S. Ranka, "Energy-and performance-aware scheduling of tasks on parallel and distributed systems", *ACM Journal of Emerging Technologies in Computing Systems*, vol. 8, no. 4, pp. 1-37, 2012.

[17] K. S. Park & V. S. Pai, "CoMon: a mostly-scalable monitoring system for PlanetLab," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 1, pp. 65–74, 2006.

[18] O. Sharma & H. Saini, "State of Art for Energy Efficient Resource Allocation for Green Cloud Data centers", *International Journal of Control Theory and Application, v*ol. 9, no. 11, pp. 5271-5280, 2016.

[19] O. Sharma & H. Saini, "VM Consolidation for Cloud Data Centers using Median Based Threshold Approach", P*roceedings of 12th International Multi-Conference on Information Processing (IMCIP), v*ol. 89, pp. 27-33, 2016.

[20] O. Sharma & H. Saini, “Energy Efficient Virtual Machine Consolidation for Cloud Data Centers Using Analytic Hierarchy Process”, *In International Journal of Advanced Intelligence and Paradigms.* (In Press).

[21] O. Sharma & H. Saini, “SLA and Performance Efficient Heuristics for Virtual Machine Placement inside Cloud Data Centers”, *In International Journal of Grid and High-Performance Computing,* vol. 9, no. 3, 2017.

[22] O. Sharma & H. Saini, “Performance Evaluation of VM Placement Using Classical Bin Packing and Genetic Algorithm for Cloud Environment”, *In International Journal of Business Data and Communication Network*, vol. 13, no. 1, pp. 45-57, 2016.

[23] O. Sharma, H. Saini, “Energy & SLA Efficient Virtual machine placement in Cloud Environment using NSGA (Non-dominated Sorting Genetic Algorithm)”, *In International Journal of Information Security and Privacy*. 2017

[24] O. Sharma, H. Saini, “BPGA: A Novel Approach for Energy Efficient Virtual Machine Placement in Cloud Data Centers”, *In Journal of Computing*.

[25] Report to congress on server and data center energy efficiency, environmental protection agency, Available at: www.energystar.gov/ia/partners/prod_development /downloads/ EPA_ Datacenter _Report_ Congress_Final1. Pdf, retrieved on 21/11/2016.

[26] Open Compute Project, “Energy efficiency,” Available at: http://opencompute.org /about /energy-efficiency/, retrieved on 3/02/2017.

[27] L. A. Barroso & U. Holzle, “The case for energy-proportional computing,” *Computer,* vol. 40, no. 12, pp. 33–37, 2007.

[28] X. Fan, W. D. Weber & L. A. Barroso, “Power provisioning for a warehouse-sized computer,” *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA),* pp. 13–23, 2007.

[29] Natural resources defense council. Available at: http://www.nrdc.org/energy, retrieved on 12/1/2017.

[30] Scaling up energy efficiency a cross the data center industry: evaluating key drivers and barriers, nrdc, Available at: http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf , retrieved on 12/2/2017.

[31] A. Beloglazov, “Energy Efficient Management of Virtual Machines in Data Centers for Cloud Computing” A thesis, Available at: http://beloglazov.info/thesis, retrieved on 25/1/2017.

[32] L. Minas & B. Ellison, "Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers", *Intel Press*, 2009.

[33] A. C. Orgerie & L. Lefevre, "When Clouds become Green: the Green Open Cloud Architecture", Proceedings of International Conference on Parallel Computing, vol. 19, pp. 228 - 237, 2010.

[34] J. Stoess, C. Lang & F. Bellosa, "Energy management for hypervisor-based virtual machines", *In USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference,* pp. 1-14, 2007.

[35] L. Cherkasova & R. Gardner, "Measuring cpu overhead for i/o processing in the xen virtual machine monitor", *In Proceedings of the Annual Conference on USENIX Annual Technical Conference*, pp. 1-24, 2005.

[36] X. Fan, W. D. Weber & L. A. Barroso, "Power provisioning for a warehouse-sized computer," P*roceedings of the 34th Annual International Symposium on Computer Architecture (ISCA),* pp. 13–23, 2007.

[37] S. H. Lim, B. Sharma, G. Nam, E. K. Kim & C. R. Das, "MDCSim: A multi-tier data center simulation, platform" *In Cluster Computing and Workshops,* pp. 1-9, 2009.

[38] M. Pedram & I. Hwang, "Power and performance modeling in a virtualized server system", *International Conference on Parallel Processing Workshops*, pp. 520-526, 2010.

[39] X. Fan, W. D. Weber & L. A. Barroso, "Power provisioning for a warehouse-sized computer", *In Proceedings of the 34th Annual International Symposium on Computer Architecture, ISCA*, pp. 13-23, 2007.

[40] W. Dargie, "A stochastic model for estimating the power consumption of a processor", *IEEE Transactions on Computers,* vol. 99, ISSN 0018-9340, 2014.

[41] D. Economou, S. Rivoire & C. Kozyrakis, "Full-system power analysis and modeling for server environments", *In Workshop on Modeling Benchmarking and Simulation (MOBS),* 2006.

[42] R. Basmadjian, N. Ali, F. Niedermeier, H. D. Meer & G. Giuliani, "A methodology to predict the power consumption of servers in data centers", *In Proceedings of the 2nd International Conference on Energy efficient Computing and Networking*, pp. 1-10, ISBN 978-1-4503-1313-1, 2011.

[43] A. Kansal, F. Zhao, J. Liu, N. Kothari & A. A. Bhattacharya, “Virtual machine power metering and provisioning”, *In Proceedings of the 1st ACM symposium on Cloud computing*, pp. 39-50, 2010.

[44] W. Dargie, “A stochastic model for estimating the power consumption of a processor”, IEEE *Transactions on Computers*, ISSN 0018-9340, 2014.

[45] Gartner, “Gartner estimates ICT industry accounts for 2 percent of global CO2 emissions,” Available: http://www.gartner.com/it/page.jsp?id=503867, retrieved on 17/01/2017).

[46] G. Dhiman, K. Mihic & T. Rosing, “A system for online power prediction in virtualized environments using gaussian mixture models”, *In Proceedings of the 47th Design Automation Conference*, pp. 807-812, ISBN 978-1-4503-0002-5, 2013.

[47] K. Rybina, W. Dargie, A. Strunk & A. Schill, “Investigation into the energy cost of live migration of virtual machines”, *In Sustainable Internet and ICT for Sustainability, Performance of Communication Systems"*, pp. 1-8, 2013.

[48] A. Strunk & W. Dargie, “Does live migration of virtual machines cost energy”, *In 27th IEEE International Conference on Advanced Information Networking and Applications*, pp. 514-521, 2013.

[59] W. Hu, A. Hicks, L. Zhang, E. M. Dow, V. Soni, H. Jiang, R. Bull & J. N. Matthews, “A quantitative study of virtual machine live migration”, *In Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, pp. 1-10, 2013.

[50] B. Guenter, N. Jain & C. Williams, “Managing cost, performance and reliability trade-offs for energy-aware server provisioning” *In Proceedings. of the 30st Annual IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1332-1340, 2013.

[51] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba & J. L. Hellerstein, “Dynamic energy-aware capacity provisioning for cloud computing environments”, *In Proceedings of the 9th International Conference on Autonomic Computing, ICAC*, pp. 145-154, 2012.

[52] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao & F. Zhao, “Energy-aware server provisioning and load dispatching for connection-intensive internet services”, *In Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pp. 337-350, 2013.

[53] A. C. Orgerie & L. Lefevre, “ERIDIS: Energy-efficient Reservation Infrastructure for large scale Distributed Systems”, *Parallel Processing Letters*, vol. 21, no. 2, pp. 133-154, 2011.

[54] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy & G. Jiang, “Power and performance management of virtualized computing environments via lookahead control”, *In Proceedings of the 2008 International Conference on Autonomic Computing*, pp. 3-12, 2008.

[55] A. Beloglazov, R. Buyya, Y. C. Lee & A. Y. Zomaya, “A taxonomy and survey of energy-efficient data centers and cloud computing systems”, *Advances in Computers*, vol. 82, pp. 47-111, 2011.

[56] Y. Ding, X. Qin, L. Liu & T. Wang, “Energy efficient scheduling of virtual machines in cloud with deadline constraint”. *In Future Generation Computer Systems*, vol. 50, no. C, pp. 62-74, 2015.

[57] M. Mishra**,** A. Das, P. Kulkarni & A. Sahoo, “Dynamic Resource Management Using Virtual Machine Migration**s”,** *In the Proceedings of Cloud Computing: Networking and Communication Challenges.* IEEE Communication Magazing, 2012.

[58] G. J. Popek and R. P. Goldberg, “Formal Requirements for Virtualizable third generation architectures,” Communications of ACM, vol. 17, no. 7, pp. 412-421, 1974.

[59] A. Beloglazov & R. Buyya, “Energy Efficient Resource Management in Virtualized Cloud Data Centers”, *In the Proceedings of 10th IEEE International Conference on Cluster, Cloud and Grid Computing*, pp. 826-831, 2010.

[60] “VMware distributed power management concepts and use,” *Technical Report*, 2010.

[61] Z. A. Mann, “Allocation of Virtual Machines in Cloud Data Centers: A Survey of Problem Models and Optimization Algorithms”, *ACM Computing Survey*, vol. 48, no. 1, 2015.

[62] M. Jeffrey, Galloway, K. L. Smith, S. V. Susan, “Power aware load balancing for Cloud Computing,” *In Proceedings of the World Congress on Engineering and Computer Science,* 2011.

[63] D. M. Erceghurn, V. M. Mirosevich, “Modern Robust Statistical Method: An easy way to maximize the accuracy and power of your research,” *American Psychologist*, vol. 63, pp. 591-601, 2008.

[64] M. S. Hasan, E. N. Huh, “Heuristic based Energy-aware Resource Allocation by Dynamic Consolidation of Virtual Machines in Cloud Data Center”, *KSII Transactions on Internet and Information Systems*, vol. 7, no. 8, 2013.

[65] E. Arianyan, H. Taheri & S. Sharifian, “Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers”, *In Computers and Electrical Journal*, vol. 47, no. C, pp. 222-240, 2015.

[66] X. Wang, X. Liu, L. Fan & X. Jia, “A Decentralized Virtual Machine Migration Approach of Data Centers for Cloud Computing”, *Hindawi Publishing Corporation Mathematical Problems in Engineering,* Article ID 878542, 2013.

[67] M. Andreolini, S. Casolari, M. Colajanni & M. Messori, “Dynamic load management of virtual machines in a cloud architecture”, *International Conference on Cloud Computing,* pp. 201-214, 2009.

[68] M. Yue, “A simple proof of the inequality FFD (L)< 11/9 OPT (L)+ 1, for all l for the FFD bin-packing algorithm”, *Acta Mathematicae Applicatae Sinica (English Series),* vol. 7, no. 4, pp. 321–331, 1991.

[69] W. Song, Z. Xiao, Q. Chen & H. Luo, “Adaptive Resource Provisioning for the Cloud Using Online Bin Packing”, *IEEE Transaction of Computers vol. 63, no. 11, pp. 2647-2660,* 2014.

[70] G. Lovasz, F. Niedermeier, H. D. Meer, “Performance Tradeoffs of Energy-Aware Virtual Machine Consolidation”, *In Cluster Computing,* vol. 16, no. 3, pp. 481-486, 2013.

[71] C. Ghribi, M. Hadji & D. Zeghlache, “Energy Efficient VM Scheduling for Cloud Data Centers: Exact allocation and migration algorithms”, *In the Proceedings of Cluster, Cloud and Grid computing,* 2013.

[72] A. Verma, G. Dasgupta, T. K. Nayak, P. De & R. Kothari, “Server workload analysis for power minimization using consolidation”, *In: Proceedings of the 2009 conference on USENIX annual technical conference.* Pp. 28–28, 2009.

[73] H. Abdi, “Multiple correlation coefficient”, *In Salkind NJ (ed) Encyclopedia of measurement and statistics,* pp. 648–651.

[74] S. Esfandiarpoor, A. Pahlavan & M. Goudarzi, “Structure-aware online virtual machine consolidation for datacenter energy improvement in cloud computing”, *In Computers and Electrical Engineering,* vol. 42, no. C, pp. 74-89, 2014.

[75] M. R. Chowdhury, M. R. Mahmud & R. M. Rahman, “Implementation and performance analysis of various VM placement strategies in CloudSim”, *In Journal of Cloud Computing, v*ol. 4, no. 20, 2015.

[76] Z. H. Zhan, X. F. Liu, Y. J. Gong, J. Zhang, H. S. H. Chung & Y. Li, "Cloud Computing Resource Scheduling and a Survey of its Evolutionary Approaches", *ACM Computing Surveys*, vol. 47, no. 4, Article 63, 2015.

[77] P. Campegiani, "A genetic algorithm to solve the virtual machines resources allocation problem in multi-tier distributed systems", *In: Second International Workshop on Virtualization Performance: Analysis, Characterization, and Tools*, 2009.

[78] H. Iima & T. Yakawa, "A new design of Genetic Algorithm for Bin Packing". *In IEEE conference on Evolutionary computing*, pp. 1044-1049, 2003.

[79] B. Madhusudhan & K. C. Sekaran, "A genetic algorithm approach for virtual machine placement in cloud", *In the Proceedings of International Conference on Emerging Research in Computing, Information, Communication and Application (ERCICA),* 2013.

[80] S. Chen, J. Wu, Z. Lu, "A cloud computing resource scheduling policy based on genetic algorithm with multiple fitness", *In the Proceedings of CIT 12th International conference on computer and information Technology,* pp. 177-184, 2012.

[81] T. Thiruvenkadam and P. Kamalakkannan, "Energy efficient multi-dimensional host load aware algorithm for virtual machine placement and optimization in cloud environment", *Indian journal of science and technology,* vol. 8, no. 17, 2015.

[82] M. Tang & S. Pan, "A hybrid genetic algorithm for the energy efficient virtual machine placement problems in data centers", *Neural Processing Letters,* vol. 41, no. 2, pp. 211-221, 2015.

[83] D. Wilcox, D. McNabb, K. Seppi, "Solving virtual machine packing with a reordering grouping genetic algorithm", In*: Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 362–369, 2011.

[84] J. Xu, J. Fortes, "Multi-objective virtual machine placement in virtualized data center environments", *In: Proceedings of the IEEE/ACM International Conference on Green Computing and Communications and International Conference on Cyber, Physical and Social Computing, 2*010.

[85] F. F. Moghaddam, R. F. Moghaddam, M. Cheriet, "Carbon –aware distributed cloud: multilevel grouping genetic algorithm". *Cluster computing,* vol. 18, no. 1, pp. 477-491, 2015.

[86] Y. S. Dong, G. C. Xu, X. D. Fu, "A distributed parallel genetic algorithm of placement strategy for virtual machines deployment on cloud platform", *The scientific world journal,* Article ID 250139, 2014.

[87] C. T. Joseph, K. Chandrasekaran & R. Cyriac. A novel family genetic approach for virtual machine allocation", In the proceedings of International Conference on Information and communication Technologies (ICICT), vol. 46, pp. 558-565, 2014.

[88] Y. Gao, H. Guan, Z. Qi & L. Liu, "A multi-objective Ant colony system algorithm for virtual machine placement in cloud computing", *In Journal of computer and system Sciences*, vol. 79, no. 8, pp. 1230-1242, 2013.

[89] E. Feller, L. Rilling, C. Morin, "Snooze: a scalable and autonomic virtual machine management framework for private clouds", *Proceedings of International Symposium on Cluster, Cloud and Grid Computing*, pp. 482-489, 2012.

[90] E. Feller, L. Rilling & C. Morin, "Energy-aware ant colony based workload placement in clouds," *Proceedings of the12th IEEE/ACM International Conference on Grid Computing,* pp. 26–33, 2011.

[91] M. A. Tawfeek, A. B. Elsisi, A. E. Keshk & F. A. Torkey, "Virtual machine placement based on ant colony optimization for minimizing resource wastage", *In Advanced Machine Learning Technologies and Applications,* pp. 153–164, 2014.

[92] A. Layeb and Z. Benayad, "A novel firefly algorithm based ant colony optimization for solving combinatorial optimization problems. *International Journal of Computer Science and Applications,* vol. 11, no. 2, article19, 2014.

[93] B. Perumal & A. Murugaiyan, "A Firefly Colony and Its Fuzzy Approach for Server Consolidation and Virtual Machine Placement in Cloud Datacenters", *Hindawi Publishing Corporation Advances in Fuzzy Systems*, Article ID 6734161, 2016.

[94] C. Dupont, G. Giuliani, F. Hermenier, T. Schulze & A. Somov, "An energy aware framework for virtual machine placement in cloud federated data centers", *In Future Energy Systems: Where Energy, Computing and Communication Meet,* 2012.

[95] L. Zhang, Y. Zhuang, & W. Zhu, "Constraint Programming based Virtual Cloud Resources Allocation Model", *International Journal of Hybrid Information Technology,* vol. 6, no. 6, pp. 333-344, 2014.

[96] J. Dong, H. Wang, & S. Cheng, “Energy-performance tradeoffs in IaaS cloud with virtual machine scheduling”, *Communications*, vol. 12, no. 2, pp. 155-166, 2015.

[97] Z. Usmani & S. Singh, “A survey of Virtual Machine Placement Techniques in a cloud Data Center”, *In the Proceeding of International Conference on Information Security & Privacy (ICISP),* vol. 78, pp. 491-498, 2016.

[98] N. Bobroff, A. Kochut & K. Beaty, “Dynamic placement of virtual machines for managing sla violations”, *In Internation Symposium on Integrated Network Management,* pp. 119-128, 2007.

[99] B. Speitkamp & M. Bichler, “A mathematical programming approach for server consolidation problems in virtualized data centers”, *IEEE Transactions on system and services*, vol. 3, no. 4, pp. 266-278, 2014.

[100] C. Clark, K. Fraser, K. S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt & A. Warfield, “Live migration of virtual machines”, *In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation*, pp. 273–286, 2005.

[101] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai & F. Xia, “A survey on virtual machine migration and server consolidation frameworks for cloud data centers”. *In Journals of Network and Computer Applications, v*ol. 52, pp. 11-25, 2015.

[102] R. H. Michael, D. Umesh, and G. Kartik, "Post-copy live migration of virtual machines," *SIGOPS Operating System Review*, vol. 43, pp. 14-26, 2009.

[103] R. H. Michael & G. Kartik, "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning”, *ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, 2009.

[104] M. H. Ferdaus & M. Murshed, “Energy-aware Virtual Machine Consolidation in IaaS Cloud Computing”, In Computer Communication and Networks, pp. 179-208, 2014.

[105] F. Farahnakian, P. Liljeberg & J. Plosila, “LiRCUP: Linear Regression based CPU Usage Prediction Algorithm for Live Migration of Virtual Machine”, *Proceedings of the 39th Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA),* pp. 357–364, (2013).

[106] SPEC power benchmarks, Standard Performance Evaluation Corporation. Available at http://www.spec.org/benchmarks html#power. Retrieved on 23/1/2016.

[107] N. Rodrigo, Calheiros, R. Ranjan, A. Beloglazov, A. F. Cesar & R. Buyya, “CloudSim :a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms”, *In Software Practice and Experience, v*ol. 41, pp. 23-50, 2011.

[108] Amazon elastic computing cloud (EC2), in, Available at: http://aws.amazon.com/ec2/instance-types. Retrieved on: 12/8/2015.

[109] E. Pakbaznia, M. Ghasemazar & M. Pedram, “Temperature aware dynamic resource provisioning in a power optimized datacenter”, *In proceedings of Design, automation and test in Europe conference and exhibition (DATE10),* pp. 124–130, 2010.

[110] A. Pahlavan, M. Momtazpour & M. Goudarzi, “Data center power reduction by heuristic variation-aware server placement and chassis consolidation”, *In Proceedings of the 16th CSI International Symposium on Computer Architecture and Digital Systems (CADS),* 2012.

[111] G. Han, W. Que, G. Jia, G L. Shu, “An efficient virtual machine consolidation scheme for multimedia cloud computing”, *Journal of Sensors,* vol. 16, no. 2, pp. 1–17, 2016.

[112] R. Buyya, “Market oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities”, *In Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications,* 2009.

[112] N. Srinivas & D. Kalyanmoy, “Multi-objective optimization using nondominated sorting in genetic algorithms”, *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[113] A. Anand, J. Lakshmi & S. K. Nandy, “Virtual machine placement optimization supporting performance SLA”, *In 5th IEEE International Conference on Cloud Computing Technology and Science,* vol. 298-305, 2013.

[114] J. Sekhar & G. Jeba, “Energy efficient VM live migration in cloud data centers”, *International Journal of Computer Science and Network*, vol. 2, no. 2, pp. 71-75, 2013.

[115] S. A. Ludwig & A. Moallem, “Swarm intelligence approaches for grid load balancing”, *Journal of Grid Computing,* vol. 9, no. 3, pp. 279-301, 2011.

[116] H. Li, G. Zhu, C. Cui, H. Tang, Y. Dou & C. He, “Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing”, *Journal of Computing*, vol. 98, no. 3, pp. 303-317, 2016.

[117] X. S. Yang, “Nature-Inspired Metaheuristic Algorithms”, Luniver Press ISBN:1905986106 9781905986101, 2008.

[118] S. Wang, Z. Liu, Z. Zheng, Q. Sun & F. Yang, “Particle Swarm Optimization for Energy Aware Virtual Machine Placement Optimization in Virtualized data centers”, *In the Proceedings of International Conference on Parallel and Distributed Systems,* dOI: 10.1109/ICPADS.2013.26

[119] M. A. Rodriguez & R. Buyya, “Deadline Based Resource Provisioning and Scheduling Algorithms for Scientific Workflows on Clouds”, *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222-235, 2014.

[120] N. J. Kansal & I. Chana, “Energy-aware Virtual Machine Migration for Cloud Computing- A firefly Optimization”, *Journal of Grid Computing*, vol. 14, no. 2, pp. 327-345, 2016.

[121] X. S. Yang & X. He, “Firefly algorithm: recent advances and applications”, *International Journal of Swarm Intelligence,* vol. 1, no. 1, pp. 36-50, 2013.

[122] F. F. Moghaddam, R. F. Moghaddam & M. Cheriet, “Carbon-aware distributed cloud: multi-level grouping genetic algorithm”, *cluster computing*, vol. 18, no. 1, pp. 477-491, 2015.

[123] F. Caglar, S. Shekhar & A. Gokhale, “iPlace: An intelligent and Tunable Power- and Performance Aware Virtual Machine Placement Technique for CloudBased Real-Time Applications”, In the proceedings of International Symposium on Object/Component/Service Oriented Real-Time Distributed Computing. doi: 10.1109/ISORC. 2014.35, 2012.

[124] G. Portaluri, S. Giordano, D. Kliazovich & B. Dorronsoro, “A Power Efficient Genetic Algorithm for Resource Allocation in Cloud Computing Data Centers”, *In the Proceedings of 3rd International Conference on Cloud Networking*, 2014. DOI: 10.1109/CloudNet.2014.6968969

[125] A. Beloglazov, J. Abawajy & R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing”, *Future Generation Computing System,* vol. 28, no. 5, pp. 755-768 (2012).

[126] D. Kumar & K. K. Mishra, “Multi-objective optimization using co-variance guided Artificial Bee Colony”, *Journal of Information Science and Engineering,* vol. 32, 2016.

[127] A. C. Adamuthe, R. M. Pandharpatte & G. T. Thampi,“ Multi-objective virtual machine placement in cloud environment”, *In the proceedings of International Conference on Cloud and Ubiquitous computing and emerging technologies,* 2013

LIST OF PUBLICATIONS

Published Journal Papers

- O. Sharma and H. Saini. "Energy Efficient Virtual Machine Consolidation for Cloud Data Centers Using Analytic Hierarchy Process". *In International Journal of Advanced Intelligence and Paradigms*. (In Press)
- O. Sharma and H. Saini. "Performance Evaluation of VM Placement Using Classical Bin Packing and Genetic Algorithm for Cloud Environment". *In International Journal of Business Data and Communication Network (IJBDCN)*, vol. 13, no.1, pp.45-57, 2016.
- O. Sharma and H. Saini. "SLA and Performance Efficient Heuristics for Virtual Machine Placement inside Cloud Data Centers". *In International Journal of Grid and High-Performance computing (IJGHPC)*, vol. 9, no.3, pp. 17-33, 2017.
- O. Sharma and H. Saini. "State of Art for Energy Efficient Resource Allocation for Green Cloud Data centers". *In International Journal of Control Theory and Application (IJCTA)* vol.9, no.11, pp. 5271-5280, 2016.

Published Conference Proceedings

- O. Sharma and H. Saini. "Experimental Analysis for Energy Management Techniques for Mobile Devices Using Cloud Computing". *In the Proceeding of International Conference on Green Computing and Internet of Things (ICGCIoT)*, pp. 737-742, 2015.
- O. Sharma and H. Saini. "VM Consolidation for Cloud Data Centers using Median Based Threshold Approach". *In the proceedings of 12th International Multi-Conference on Information Processing (IMCIP-2016)*, vol. 89, pp.27-33, 2016.

Communicated Journal Papers

- O. Sharma and H. Saini. "Energy & SLA Efficient Virtual machine placement in Cloud Environment using NSGA (Non-dominated Sorting Genetic Algorithm)". *In Journal of Cases on Information Technology (JCIT). [Major indexing: SCOPUS, ESCI, DBLP, ACM digital Library, web of sciences, Google scholar]*.
- O. Sharma and H. Saini. "BPGA: A Novel Approach for Energy Efficient Virtual Machine Placement in Cloud Data Centers". *In Journal of Computing (JOC).*

<u>Communicated Book Chapter</u>

- O. Sharma and H. Saini. “Performance Evaluation for Energy Aware Virtual Machine Placement Techniques for Cloud Data Centers”. In Advances in Data Communications and Networking for Digital Business Transformation. *IGI Global*.

Candidate Signature