Energy Efficient Resource Management in Cloud Computing

Submitted in partial fulfillment of the requirements

for the award of the degree of

DOCTOR OF PHILOSOPHY

Submitted by

A Sudarshan Chakravarthy

(Roll No. 701432)

Under the guidance of

Dr. Ch. Sudhakar and Prof. T. Ramesh

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL

TELANGANA - 506004, INDIA

SEPTEMBER 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA - 506004, INDIA

THESIS APPROVAL FOR Ph.D.

This is to certify that the thesis entitled, Energy Efficient Resource Management in Cloud Computing, submitted by Mr. A Sudarshan Chakravarthy [Roll No. 701432] is approved for the degree of DOCTOR OF PHILOSOPHY at National Institute of Technology Warangal.

Examiner

Research Supervisors
Dr. Chapram Sudhakar and Prof. T. Ramesh
Dept. of Computer Science and Engg.
NIT Warangal
India

Chairman
Prof. P. Radha Krishna
Head, Dept. of Computer Science and Engg.
NIT Warangal
India

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
TELANGANA - 506004, INDIA

CERTIFICATE

This is to certify that the thesis entitled, Energy Efficient Resource Management in Cloud Computing, submitted in partial fulfillment of requirement for the award of degree of DOCTOR OF PHILOSOPHY to National Institute of Technology Warangal, is a bonafide research work done by Mr. A Sudarshan Chakravarthy (Roll No. 701432) under my supervision. The contents of the thesis have not been submitted elsewhere for the award of any degree.

Research Supervisors

Dr. Chapram Sudhakar
Associate Professor

Prof. T. Ramesh
Professor
Dept. of CSE
NIT Warangal
India

Place: NIT Warangal
Date: 28 September, 2020

DECLARATION

This is to certify that the work presented in the thesis entitled "*Energy Efficient Resource Management in Cloud Computing*" is a bonafide work done by me under the supervision of Dr. Chapram Sudhakar and Prof. T. Ramesh and was not submitted elsewhere for the award of any degree.

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/date/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

A Sudarshan Chakravarthy

(Roll No. 701432)

Date: 28-09-2020

ACKNOWLEDGMENTS

It is with great pleasure that I acknowledge my sincere thanks and deep sense of gratitude to my supervisor, Dr Chapram Sudhakar sir, from whom I have learnt so much about how to conduct oneself being a teacher. His values as a teacher and as a human being, are something I would like to take forward both in my professional and personal life. He has been very supportive to me throughout my research period which has been full of 1's and 0's, and constantly encouraged me to do good work as a researcher. He has always given ample time to me for research discussions, technical help, and the revision of the work. I feel lucky to be associated with such a great human being in my life.

I also would like thank my co-supervisor, Prof T Ramesh sir, who have inspired me immensely directly or indirectly throughout my association with NIT, Warangal. His integrity and the core values with which he works as a teacher, are something I aspire to imbibe into myself. He has been very supportive to me through out the Ph.D period and his insightful comments have immensely improved the quality of the research work.

I extend my gratitude to all my Doctoral Scrutiny Committee members Prof. R.B.V Subramanyam, Dr. K. Ramesh, and Prof. D.Srinivasacharya for their insightful comments and suggestions during oral presentations.

I am immensely thankful to Prof. D.V.L.N Somayajulu, Dr. K. Ramesh, Dr. Ch. Sudhakar, Prof. R.B.V. Subramaanyam, and Prof. P. Radha Krishna Heads of Dept. of CSE during my stay in the department, for providing adequate facilities. I wish to express my thanks to faculty members of Computer Science and Engineering department.

I thank my colleagues and friends at NITW, Sai Krishna, Sangamesh, Venu, Abhilash, Manoj, Satyanarayanan, Pavan Kumar, Vinay Raj, Preethi, Hem Kumar and Spoorthy who has made my life at NITW easy and helped in their own ways to better my research work.

I thank my close friends outside of NITW, Divakar Bokam, Vikramaditya, Harsha, Govardhan, and Balu just for being there for me always.

I thank my brothers Ravi kumar and Harsha Vardhan who always motivated and encouraged me to achieve good things in life. I am sure if not for their support, I wouldn't have the courage to take this career route. They have given me all kinds of support throughout this PhD period, be it emotional or financial. I will be forever thankful for the way they back me in every endeavor of mine.

During this research period only, my beloved wife, P Aswini has entered into my life. She is the pillar of my strength. Her love, patience and understanding have helped me to stay calm in difficult periods of research. My son Vihaan, came as an angel into our life and has brought immense joy to us. He has made my life full of happiness due to his innocent acts. I love both of you so much!!

Finally, I thank my parents who brought me on to this earth and who have given their blood and sweat to make me what I am today. They have made many sacrifices to provide quality education to me and my brothers. From my childhood, my father consistently reiterated the importance of education in life. I am always in awe of the way my father planned my education with all the financial constraints our family had. My mother took up the job of bus conductor, which is a very difficult job for women in India, so that they can afford quality education for me and my brothers. I will never forget the several sleepless nights she had to go through due to the leg pains for standing entire day in bus. The dreams of my parents and sheer determination with which they have worked to provide for us, has made me to attain this level of education. I love you amma and nanna!

A Sudarshan Chakravarthy

Dedicated to

My beloved parents, my lovely wife & little Son

ABSTRACT

Cloud computing is a computing paradigm that is built upon the features of computing as a utility, on-demand access to resources, elasticity in scaling the resources, and virtualization technology. It offers a cost effective and highly scalable approach to run applications. Proliferation in the usage of cloud services has resulted in establishment of large scale cloud data centers comprising vast number of servers and storage devices. However, these large-scale data centers consume massive amounts of energy. Due to the high operational costs and the heavy carbon emissions get released from these data centers, the high energy consumption has become a major concern for the cloud providers. The work loads of the data center need to be allocated to it's resources efficiently such that the energy consumed by the data center is reduced as much as possible.

This dissertation focuses on energy efficient resource allocation schemes for efficient management of cloud data centers. The key technique used in the thesis to reduce the energy consumption is server consolidation by intelligent VM placement and then powering down the remaining servers. In this thesis, first, a comprehensive survey of the energy efficient resource allocation schemes in cloud computing is presented. Second, an off-line phase wise approach for VM placement and routing mechanism that optimizes the energy consumed by both server and networking elements is designed. As a part of the solution a heuristic called affinity is proposed that quantifies the benefit of placing two virtual machines together in terms of the load that would be removed from the networking elements. Third, two online algorithms based on spectral graph partitioning are proposed for VM placement to achieve joint host and network energy optimization. The proposed algorithms use spectral graph partitioning to recursively find the group of virtual machines with most intra communication and least inter communication and place them together. Fourth, a new problem called ”Intermediate node selection for scatter-gather VM migration” is introduced and the intractable nature of the problem is proved. Subsequently, two greedy solutions are proposed to solve the problem to minimize energy and eviction time. Finally, energy efficient VM placement in federated cloud is modelled as a co-operative game and a novel cloud federation formation mechanism that gives the optimal federation structure is designed. The performance of the proposed solutions is evaluated for standard data center network architectures through simulation.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

VM	Virtual Machine
PM	Physical Machine
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
DVFS	Dynamic Voltage Frequency Scaling
DRAM	Dynamic Random Access Memory
CPU	Central Processing Unit
MDFD	Modified Best Fit Decreasing
kWh	kiloWatt per Hour
INS - SGM	Intermediate Node Selection in Scatter-Gather Migration
MSSP	Minimum Subset Sum Problem
EC2	Elastic Compute Cloud
IT	Information Technology
VMM	Virtual Machine Monitor
VMD	Virtual Memory Device
ECR	Energy Consumption Ratio
QoS	Quality of Service
EMRSA	Energy Aware Map Reduce with Service Level Agreements
DVS	Distributed Voltage Scaling

SLA	Service Level Agreements
ECG	Efficient Green Control
BFD	Best Fit Decreasing
DSLH	Dynamic heuristic demand selection with lowest common father
DSFN	Dynamic heuristic demand selection with fewest nodes
RES	Renewable Energy Source
JOSNE	Joint Optimization of Server and Network elements Energy
ACO	Ant Colony Optimization
DC	Data Center
LP	Linear Programming
GA	Genetic Algorithm
PSO	Particle Swarm Optimization

Chapter 1

Introduction

Cloud computing has revolutionized the computing industry by offering computing as an ever present on-demand service to the users. According to NIST [1], cloud computing is defined as “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”. The key objective of cloud computing is to share data and computations over a scalable network of nodes. Users can utilize the services that are offered by third party cloud providers, without any knowledge and expertise of the technology and infrastructure used to support those services. They can also outsource the computation, storage to a third party and pay only for the services used. The essential characteristics of cloud computing are on demand self service, broad network access, elasticity and measured service.

The popularity of cloud computing is growing day by day and there is a rapid and widespread adoption of cloud services to support many sought-after internet applications of different fields like e-commerce, social networking, on-demand video streaming, big-data analytics and so on. The reasons for this growing trend are the alleviation of the users from the burden of owning and maintaining the server resources, reduction in total cost of ownership, ability to access the resources and data from anywhere, and flexibility to scale up and down the resources required as per their dynamically changing needs.

To service the ever-increasing demand for cloud services, cloud providers are deploying large-scale data centers containing thousands of servers and network switches across the world. With the rapid growth in the number and the size of the data centers, the energy consumption and the costs associated with them have increased dramatically. According to McKinsey [2] report on data center energy efficiency, an average data center consumes as much energy as 25000 households. As per the latest study on the U.S data center energy usage, the data centers in the U.S are estimated to consume 73 billion kWh, which is nearly 2% of the total energy consumed by the U.S [3]. As the electricity prices are also rapidly increasing with time, the cloud service providers are experiencing substantial energy costs. Apart from the operational costs, exorbitant levels of energy consumption lead to adverse effect on the environment through the large volumes of carbon emissions get released from these data centers. Hence reducing the energy usage has become a key concern for the cloud providers in the design and operation of large-scale data centers.

1.1 Research problems and objectives

The aim of the research work done in this thesis is to reduce the energy consumed by the data center resources. The problems addressed in the thesis are:

- ***How to place the virtual machines of communication intensive applications on to the physical servers in a cloud data center by exploiting data center network topologies such that the cumulative energy consumed by both servers and networking elements is minimized?***

  At a time, in a cloud data center, many composite applications belonging to different tenants need be deployed. These composite applications are made of several subtasks which are logically connected by data and flow dependencies. Generally, a virtual machine in a cloud can serve one or more tasks of an application. Each tenant requests several virtual machines to service the tasks belonging to their application. Virtual machines, belonging to a tenant, communicate and exchange data with each other during their execution. If the behavior of the application belonging to the tenant is known in advance, the communication patterns of the virtual machines can be predicted. If two communicating virtual machines are placed very far in the data center network, then the data that is being exchanged would go through many network elements resulting in higher network energy consumption. Here, the idea is to consolidate the virtual machines onto minimal number of servers in such a way that any two communicating virtual machines are placed very close to each other in the proximity of the data center network.

- ***How to consolidate the virtual machine communication flows on the network elements of the cloud data center such that the total network energy is minimized?***

  The network switches and the routers in the data center network contribute to a significant portion of the total power consumed by the data center. The networking components also consume a sizable power even when they are idle, almost equal to 30% of the power consumed when they are fully loaded [4]. Like in the case of physical servers, a prudent approach to save energy is allocating the network flows of the virtual machines to a few number of networking components. The communication flows of the virtual machines placed on the physical servers need to be consolidated onto fewer network elements to achieve reduction in network energy.

- ***How to select intermediate nodes for scatter-gather VM migration in cloud data centre such that energy consumed by intermediate nodes is minimized while keeping the eviction time of the virtual machine being migrated under a threshold?***

  The scatter-gather migration is a new type of migration that is aimed at reducing the time to evict the state of a virtual machine during migration process. It uses some pre-selected nodes as intermediate nodes to stage the VM content for migrating a virtual machine from one host to another. For efficient implementation of the scatter-gather migration mechanism, the set of intermediate nodes selected should consume the least amount of energy while maintaining the eviction time under a given threshold.

- ***How a subset of the cloud providers of a federated cloud be formed as a federation to service incoming VM requests such that the social welfare of reducing the energy is maximised?***

  Cloud federation is an effective solution for reducing the energy expenditure of the cloud providers by offering a way to make profit out of the idle resources residing in their data centers. In a federation, a cloud provider earns income by offering services to the requests either by it's own resources or by leasing in/out resources from the other providers. It gains profit if the income from providing the services to the users is more than the operational costs of resources to provide those services like energy costs or the leasing costs. Cloud provider, being a selfish agent, always tries to maximize its profit and would join the federation if its profit is maximised or at least gets more profit than what it gets when acting alone. However, achieving the social welfare of the federation, i.e, the total sum of the profits obtained by all the providers participating in the federation helps the cloud providers to have better sustainability in the long run. So an important challenge is how to achieve the global energy sustainability of the federation and in turn prompt the cloud providers to take part in the federation.

Based on the above research problems, the following objectives have been defined for the research work done in the thesis.

- **To design an energy efficient VM scheduling and routing mechanism that jointly optimizes the energy consumed by servers and network in a cloud data center.**
- **To design an online communication aware VM placement that jointly optimizes the energy consumed by servers and network in a cloud data center.**
- **To define and model a new problem called "Intermediate node selection for Scatter-Gather VM Live Migration" and prove the intractable nature of the problem.**
- **To develop a greedy solutions for the problem of intermediate node selection for Scatter-Gather VM live migration that optimally selects the intermediate nodes with respect to energy and eviction time.**
- **To define the resource allocation in a federated cloud as a co-operative game and design a federation formation mechanism for servicing a set of incoming requests that maximizes the social welfare, i.e., reducing the total energy costs.**

1.2 Overview of the contributions of the thesis

In thesis, to achieve the desired objectives listed above for energy efficient resource allocation in cloud computing, solutions for four inter-related problems have been presented. First, we have proposed a mechanism for scheduling the virtual machines that are requested in advance for reservation by multiple tenants in energy efficient way by jointly optimizing the energy consumed by both server and network elements. Second, a mechanism for placing the set of virtual machine requests that are being requested on the go on to the servers is designed such that it minimizes joint energy of server and network elements. Third, as virtual live migration is used for opportunistic power savings, we have identified the problem of selecting the intermediate nodes for Scatter-Gather migration with least increase in energy consumption and have designed a mechanism to select the intermediate nodes for

Scatter-Gather migration that causes least increase in energy. To achieve better resource utilization for achieving high profits and high energy savings, many cloud providers form as a federation to serve the VM requests of the users. We have designed a mechanism for sharing of resources among the cloud providers of the federation to service the virtual machine requests of the users, that achieves good overall social welfare of the federation in terms of energy costs.

In the following, an overview of the chapter wise contributions is presented.

1.2.1 Energy efficient VM scheduling and routing in multi-tenant cloud data center

We have formulated the problem of jointly optimizing the energy consumption of servers and network elements by optimal VM scheduling and routing as an integer programming problem. To solve it a phase-wise optimization approach with two ant colony based meta-heuristic algorithms is proposed. The topology features of the data center network and the communication patterns of the applications are considered in the construction of the solution.

The proposed mechanism is a phase wise optimization approach. It is divided into two phases. The first stage of the problem is finding an optimal placement of the virtual machines onto the physical machines, which minimises the power consumption of the servers while considering the communication among the virtual machines in placement decisions. As a solution to the sub-problem, an ant colony based meta-heuristic static algorithm is proposed. As part of the solution a heuristic called affinity is defined. It indicates the goodness of virtual machine being placed in data center entity. It is defined differently for a physical machine, a rack and a cluster. The pheromone values give the historical preference of two virtual machines to get allocated together onto a physical machine. The data center is logically and hierarchically divided into several clusters, racks and physical machines. Using the heuristic values and the pheromone values, the algorithm probabilistically selects a cluster first. A rack out of all the racks belonging to that cluster, is selected probabilistically based on the affinity values of the racks. Then a physical machine belong to that rack is selected probabilistically based on the heuristic values and pheromone values. This process is repeated till the solution is found out for all the virtual machines. This completes the first phase of the solution. In the second phase, the communication flows of the virtual machines are consolidated onto the minimum number of links and switches using ant colony based algorithm.

The proposed solution is simulated in a custom made cloud simulator. It is tested for three standard data center networks, 3-tier, B-Cube and Hyper-tree of different sizes and compared against two standard algorithms, first-fit and round-robin algorithms. The results show that our solution improves energy savings on an average by 15% and 20% when compared with first fit and round robin respectively across all data center networks. The proposed algorithm performs much better for 3-tier data center networks among the three standard data center networks considered. Moreover, the algorithm is scalable as the proposed mechanism is achieving similar percentage of savings even when the load on the data center is increasing and with increase in the data center size.

1.2.2 Online energy efficient VM placement using spectral graph partitioning

In this work, the problem of minimizing the energy consumed by both sever and networking elements by an efficient online VM placement is addressed. The problem is modelled as a mixed integer programming problem. As a solution two greedy algorithms called best-fit-spectral and least-increase-spectral are proposed. Here the virtual machines request for one application at a time is considered. The communication patterns of the multi-tier composite applications can be easily predicted and they are represented as a graph G. The proposed algorithms use spectral graph partitioning to identify the groups of virtual machines with high intra and low inter communication to place the virtual machines in an energy efficient manner. In best-fit-spectral, recursively the communication graph is partitioned till we identify a maximal subset of the virtual machines that can be placed on to a physical machine in the best-fit way. In least-increase-spectral, for all the physical machines that have the capacity to hold the set of the virtual machines belonging to the graph G, the energy required to host these virtual machines is estimated. Based on those estimations, the group of virtual machines is placed on the server that causes least increase in energy. If there is no physical machine with the capacity to host all the virtual machines, then the communication graph of the virtual machines is partitioned in to two graphs G_1 and G_2 using spectral graph partitioning that gives the least normalized cut and the algorithm is called recursively on these sub-graphs.

The proposed greedy algorithms are simulated in a custom made cloud simulator. It is compared with first-fit-decreasing heuristic for three different data center network architectures of three different sizes. The three standard data center networks considered are 3-tier, B-cube, and Hyper-tree. It is observed through the simulation results that the solutions best-fit-spectral and least-increase-spectral reduce the energy consumption by 17% and 21% on an average respectively when compared with first-fit-decreasing heuristic.

1.2.3 Intermediate node selection for achieving energy efficient Scatter-Gather Migration

In this work, we define the problem of intermediate node selection in Scatter-Gather migration and prove the intractable nature of the problem. We establish that the problem of Intermediate Node Selection in Scatter-Gather Migration (INS-SGM) is NP hard by reducing 0-1 knapsack problem to Minimum Subset Sum Problem (MSSP) and then MSSP to INS-SGM. The problem is mathematically modelled as an integer programming problem based on two optimality criteria - minimizing eviction time and minimizing energy. Two algorithms called maximum-decrease-in-eviction-time and least-increase-in-energy are proposed to solve the problem for the optimality criterion of eviction time and energy respectively. The basic idea of the first algorithm that aims at decreasing the eviction time of the migration is to select a node that has least expected transfer time first as an intermediate node. The second algorithm aims at reducing the additional energy consumed due to addition of intermediate nodes while keeping the eviction time of the migration under a given threshold. The algorithm design is based upon a 5/4 approximation algorithm for minimum subset sum [5].

The performance of the proposed solutions is analyzed with respect to three parameters - eviction time, energy and total migration time using simulation results. A 3-tier data center network is considered in the simulation. Four different types of virtual machines are considered in the simulation whose configurations are defined as per the standard amazon EC2 instances. We have considered three data centers of varying sizes to check the scalability of the solution. The performance analysis of the proposed solutions is done by comparing them with three basic solutions namely random selection policy, farthest node first, and closest node next.

1.2.4 Federation formation mechanism for achieving energy efficient resource sharing in federated cloud

We formulate the energy aware resource allocation for a set of requests in federated cloud as a co-operative game. Then the properties of the modelled co-operative game like stability and fairness are studied. Shapley value is used to distribute the profit among the cloud providers participating in the federation to service the requests. We show that the proposed game is not cohesive and grand coalition, i.e, federation containing all the cloud providers, may not form. So to find the optional federation for a set of requests from the users, a novel federation formation mechanism is designed. It is a branch and bound technique, where the division of the search space is based upon integer partitioning of the number of the cloud providers. For each subspace represented by the permutation of an integer partitioning, the average profit and maximum profit are calculated. Then an ordered exploration and pruning of the sub spaces is done based on their maximum and average profit values to find out the optimal federation structure.

The proposed solution is simulated and compared against two variations of best fit greedy approach, one that tries to find federation for one request at a time and the other, that finds the federation for set of virtual machine requests in each time epoch. The performance analysis is done with respect to the following parameters: total profit, individual profit, percentage of requests serviced, overall utilization, average federation size, VM utilization.

1.3 Organization of the thesis

The thesis mainly focuses on achieving energy efficient resource management in cloud. The contributions and findings of the research work are organized into chapters as below.

Chapter 1: A brief introduction to the issue of energy efficient management in the cloud computing is presented. A overview of the research contributions of the thesis with respect to achieving energy efficiency in the cloud computing is given.

Chapter 2: The preliminaries related to cloud computing and detailed literature survey of the energy efficient resource management schemes in cloud are presented.

Chapter 3: A mathematical model of the joint host-network energy optimization problem through VM scheduling and routing is presented. An ant colony based phase wise optimization approach for solving the problem is presented in detail.

Chapter 4: An on-line VM placement mechanism that is based on spectral graph partitioning of the communication graph to minimize the cumulative energy consumed by servers and network elements is presented in detail.

Chapter 5: A new problem called intermediate node selection for Scatter-Gather VM migration is introduced, and the mathematical model and the proof for the hardness of the problem is presented. Two heuristic algorithms called maximum-decrease-in-eviction-time and least-increase-in-energy for solving the problem with respect to reducing eviction time and energy are presented.

Chapter 6: A federation formation mechanism to achieve social welfare of reducing the overall energy of the cloud providers that are part of cloud federation is presented.

Chapter 7: A summary of the research outcomes of the thesis work is presented. Future directions of the research and scope for extensions of the works done in this thesis are presented.

Chapter 2

Literature Survey

Cloud Computing that offer various services under one roof is emerging as a popular computing paradigm in the computing industry. It has revolutionized the IT industry with its pay-per-use model of service, ability to allocate the resources as and when the users require and alleviating them from the operational issues of the resources. It offers a cost effective and highly scalable approach for running the applications. So the popularity of the cloud is ever increasing resulting in huge number of customers deploying their workloads, data on to the cloud instead of the local resources. Due to the growing popularity for cloud services expansive cloud data centers comprising vast number of servers and storage devices are being built around the world by cloud providers. However, these cloud data centers consume a massive amount of energy. According to a report by Lawrence Berkely National laboratory, the data centers in united states accounts for 70 billion kilowatt-hours of energy consumption. As per a study, the global data center electricity consumption is 205 tera watt-hours in 2018 and it is equivalent to 1% of the total world's power consumption[3]. The exorbitant power consumed by the data centers incurs huge operational costs for the cloud providers. Typically a data center energy cost is equivalent to it's infrastructure set up cost or 75% of the cloud provider operational cost.

Apart from the energy costs, there is a growing concern over the significant amount of CO_2(carbon di-oxide) gets released from these data centers and it's adverse effect on the environment. Hence, considering energy efficiency in designing efficient data center management mechanisms is an important issue.

In the past few years, there has been an extensive research on designing efficient resource management schemes for reducing the energy consumed by the cloud data centers. In this chapter we present a detailed survey of the existing resource management mechanism in the literature. First we introduce preliminaries related to cloud computing and then present the detailed survey of the energy efficient resource management mechanisms.

2.1 Virtualization

The cloud data centers contain a massive number of physical machines and storage devices. The physical resources in a cloud data center are virtualized to distribute their services among many users effectively [6][7][8]. This leveraging of virtualization technology maximizes the utilization of hardware resources by sharing them among multiple users. A thin layer of software called Virtual Machine Monitor (VMM) or hypervisor running between operating system and system hardware manages and controls the virtual machines running on a platform . To improve data center efficiency many resource management techniques like load balancing, server consolidation, and power management are applied by migrating one or multiple virtual machines from one physical machine to another physical machine .

2.1.1 VM Live Migration

VM Live migration is the process of moving a virtual machine that is servicing an application from one host to another host with minimal disruption to the service. In general, there are two variants of live migration, pre-copy and post-copy.

2.1.1.1 Pre-Copy Migration

The pre-copy migration [9][10] contain two phases, Iterative pre-copy and Stop-copy. In the iterative pre-copy phase, in the first round, entire memory pages of the virtual machine being migrated are copied to the destination while ensuring the VM keep running on the source. In the subsequent rounds, only the dirtied pages are copied to the destination. Once the dirtied memory contents become relatively small, the virtual machine is stopped at the source, and the dirtied pages are copied to the destination. Then the virtual machine would be started at the destination. This phase is called Stop-Copy phase. In this phase the application that is running on the migrated virtual machine experiences service unavailability.

2.1.1.2 Post-Copy Migration

In the post-copy migration [11][12], the virtual machine is directly resumed at the destination. The memory contents of the virtual machines are actively pushed from the source to the destination by a process called pre-paging. Usually whatever the content the virtual machine requires would have reached the destination, but if any memory page is missing that the VM requires, it would be fetched from the source through network page fault.

2.1.1.3 Scatter Gather VM Live Migration

It is a variant of post copy VM migration. The Scatter-Gather VM migration, contains two phases, first scatter phase and second the gather phase. In the scatter phase the memory contents of the virtual machine that is being migrated is sent to not only to the destination but also to some intermediate nodes. It allows the source to get decoupled quickly from the migration process. The scattering of the content is achieved through an abstracted layer called virtual memory device (VMD). The VMD collects all the free memory available at the intermediate nodes and presents as a block device one per the migration. The source migration manager then scatters the memory contents to multiple intermediates by writing them to the block device allocated to it and each page written by the source on the block device would be sent to an intermediate node. This information of where each page is stored is sent to the destination. In the gather phase, the destination migration manager collects the contents concurrently from the intermediate nodes at its own pace.

Here the problem of how to select intermediate nodes is kept as an open problem. For effective implementation of the Scatter-Gather VM migration, the intermediate nodes should be selected carefully. The algorithms for intermediate node selection should be implemented in the virtual device module over the intermediate nodes.

2.1.2 VM migration parameters

The typical metrics that are used for virtual machine migration are total migration time and downtime. Their definitions are are as follow.

Total migration time: It is the time from when the process of migration begins from the source machine until the time when the destination VM has complete control.

Downtime: It is the time during the migration when the application experiences service unavailability.

Recently, Deshpande *et al.* [13] proposed a new metric called eviction time that measures how fast the resources of the source machine used for hosting the VM being migrated are released, and it is defined as follows.

Eviction Time: It is the time taken to decouple the source from the destination during VM migration.

2.2 Data Center Networks

A data center is a pool of computational, storage and network devices connected through a communication network. The data center network has to be scalable, and efficient to connect tens of thousands of servers in the data center. The following are the data center networks considered for simulating the works presented in the thesis.

2.2.1 Three Tier Data Center Network

A 3-Tier data center network consists of three layers namely, edge, aggregation and core [14]. The network can be divided into k pods. Each pod in a k-ary fat-tree consists of $\frac{k^2}{2}$ severs and two layers of $\frac{k}{2}$ k port switches. Each edge switch is connected to $\frac{k}{2}$ servers and $\frac{k}{2}$ aggregate switches. Each aggregate switch connects to $\frac{k}{2}$ edge and $\frac{k}{2}$ core switches. k ary fat-tree contains $\frac{k^2}{2}$ core switches, where each core switch connects to k pods.

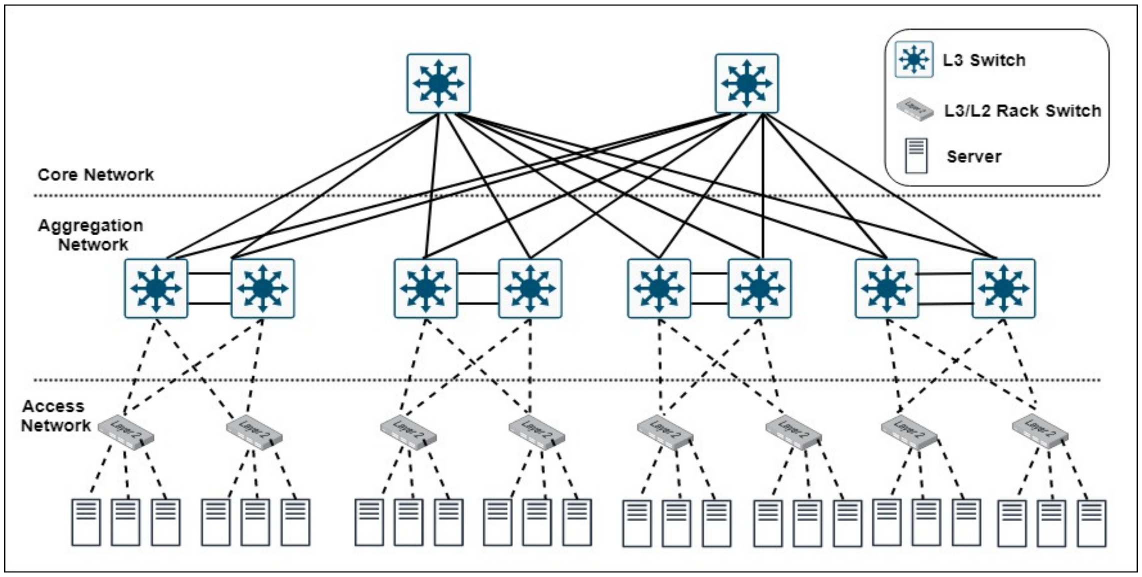


Figure 2.1: Three tier data center network architecture

2.2.2 B-Cube Data Center Network

B-Cube data center network is a server centric and recursively constructed data center network architecture [15]. A $Bcube_k$ is constructed from $Bcube_{k-1}$ and n^k, n port switches. Each server in a B-Cube has $K + 1$ ports, which are numbered from level 0 to level-k. A $Bcube_k$ has n^{k+1} servers and $k + 1$ level of switches, with each level having n^k, n-port switches. The B-Cube data center network uses lot of wires and it's complexity stops it from growing to a data center network size bigger than the one used for container based data center (CDC).

2.2.3 Hyper-Tree Data Center Network

It is a cost effective and scalable data center network architecture. The first layer of hyper-tree has n nodes and one n-port switch [16]. From the second layer on wards, a k layer hyper-tree consists of $n^2(k-1)$-layer hyper-trees. Specifically a k layer hyper-tree can be treated as a matrix containing n^2 rows and n^{2k-3} columns where each row specifies a $k-1$ layer hyper-tree having n^{2k-3} nodes.

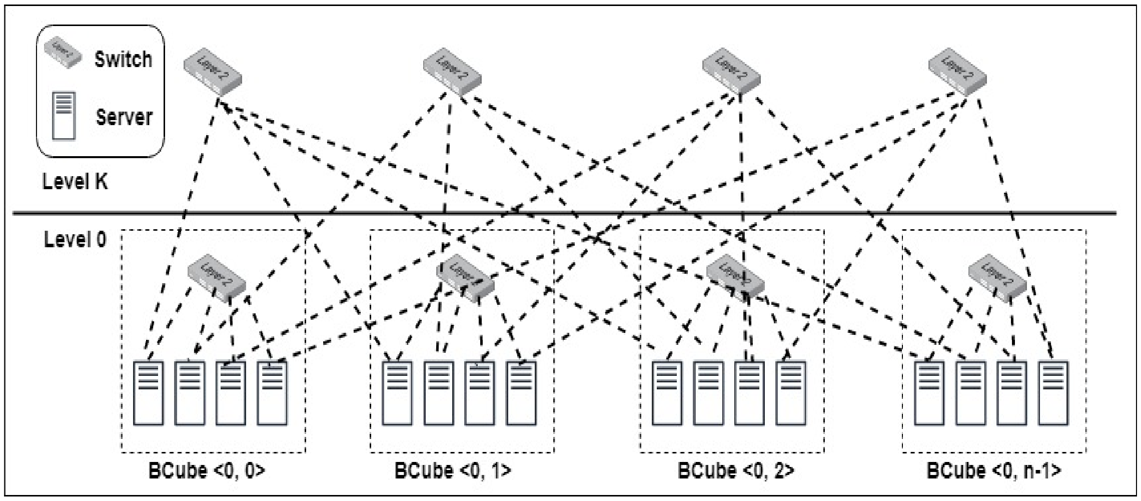


Figure 2.2: B-Cube data center network architecture

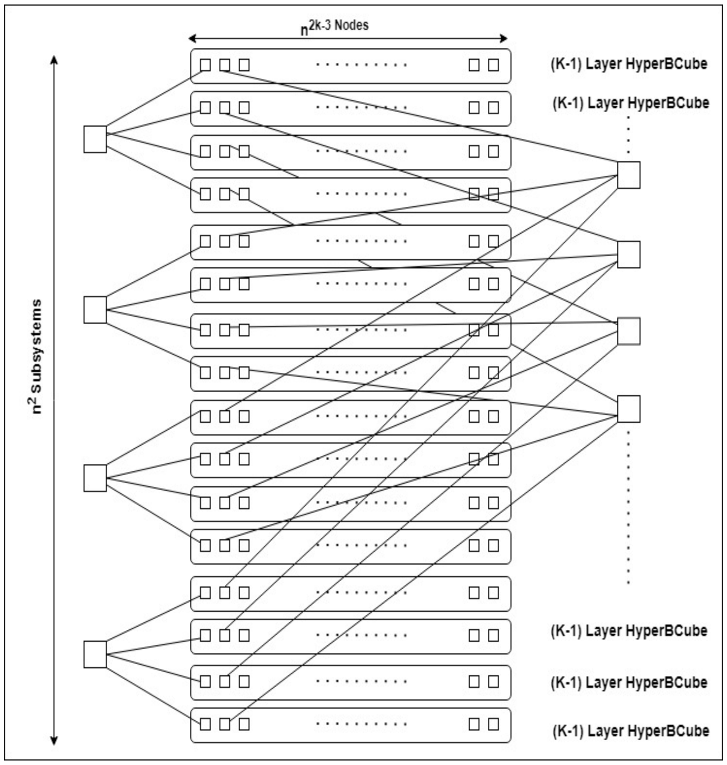


Figure 2.3: Hyper-tree data center network architecture

2.3 Survey on energy efficient resource management in cloud computing

Although data centers are huge consumers of power, only half of that total power is used by the IT load [17]. This power inefficiency of the data centers incurs huge amount of losses to the providers. The reasons for this data center energy inefficiency are peak provisioning of the resources, low deployment of virtualization technology and presence of energy disproportionate servers. The data center providers generally over estimate it's IT load and accordingly put high number of physical resources active to support the peak demand. However, the severs present in most of the current data centers are energy disproportionate, i.e., the energy consumed by these servers is not proportional to the amount of load on them. The idle servers consume nearly 60-70% of the peak load energy [18] [19] [20]. So effective energy efficient resource allocation schemes should be developed so as to mitigate the energy wasted because of these in-efficiencies.

The energy efficient cloud resource management schemes can be classified based on the way they are achieving energy efficiency. However, some mechanisms are hybrid, which have employed more than one way of saving energy, and they are put into a class, as per their primary approach to save energy.

2.3.1 Dynamic Voltage Frequency Scaling Based Provisioning Schemes

Dynamic Voltage-Frequency Scaling(DVFS) is a general technique used to achieve energy efficiency in server processors of a data center [21] [22][23] [24]. The power consumption equation of a complementary metal–oxide–semiconductor(CMOS) circuit has two parts, static and dynamic. The dynamic power consumption depends on the frequency the circuit is clocked at. If the circuit is clocked faster, it will consume more energy, and in the same way, if it is clocked slower, it will consume less energy [25][26][27]. Hence by intelligently operating the frequency of the circuit, significant energy savings can be achieved. Through DVFS, the core of a processor can run in discrete performance states called as P-States. If the core is in the lower numbered performance state, then the core runs at higher frequencies, resulting in faster execution times and higher power consumption. In the higher numbered states a DVFS enabled core will run at lesser frequencies drawing less power giving slower execution times for a task in that state. In effect, a DVFS enabled core can be tuned to different performance states depending on the performance requirements to achieve energy efficiency.

Oxley *et al.* [28] proposed a resource allocation scheme that uses DVFS technique to allocate cores of heterogeneous computer system to a bag of independent tasks. Through this, two problems; optimising makespan under power budget constraint and optimising energy under performance(makespan) constraints are addressed. The execution times of the workloads are modeled stochastically and the arithmetic intensity(the number of operations performed per word transferred) of the workloads is considered to take a decision on what should be the core and it's state to execute that workload for saving the energy. The idea here is to run the memory intensive tasks at lower frequencies exploiting the presence of lots of idle times because of memory or disk accesses and run compute-intensive tasks at higher frequency states. Several heuristics are proposed to decide the core and it's state for each workload. It is proved that this approach is robust under uncertain task execution times.

Wanneng *et al.* [29] proposed a resource allocation mechanism using an improved clonal selection algorithm which minimises both makespan and energy consumption. Here, the servers are assumed to be DVFS enabled. The idea here is to identify the frequency and voltage values for all the resources to achieve optimal value for the combined objective of makespan and energy consumption and set those values accordingly. The objective functions for energy and makespan are formulated using the variables representing the possible frequency levels and voltage levels the devices can be in. The problem is formed as a multi-objective optimisation problem where both the objectives are conflicting in nature. As a solution, an immune clonal optimisation based algorithm is proposed.

Alahmadi *et al.* [30] integrated VM reuse and Dynamic Frequency Scaling to design an energy-aware scheduling framework for a homogeneous cloud data center. The workloads considered here are a bag of independent tasks and initially for each task the lowest frequency it can run without missing deadlines is estimated. Then to service the task, VM re-use is used to save the VM launching overhead. If any of the VMs has adequate capacity or extension space for future accommodation, then the task would be allocated to that VM. Otherwise, a new VM would be launched. Here, First Fit Decreasing heuristic is used to allocate the VMs to the cores of the PM's and the cores would run in the optimal frequency for the tasks, that is estimated in the initial phase.

Deng *et al.* [31] use DVFS mechanism to save memory energy by lowering the frequency of DRAM devices, memory channels and memory controllers at the time of low memory activity. It is based on an observation that at low bandwidth utilisation reducing the frequency of the memory will not impact the performance much. The frequency of the memory will be adjusted based on the bandwidth utilisation. If the utilisation crosses a predetermined threshold value, the frequency of the memory will be increased to reduce the impact on the performance. As an extension, a technique for coordinating DVFS across multiple memory controllers, memory devices and channels is proposed to reduce the overall system power consumption. However, the interaction between memory voltage scaling and CPU- voltage scaling is not considered in this work.

The principle of ”Low frequency results in lower energy consumption” may not always work as running a task at low frequency can increase it's execution time considerably resulting in higher energy consumption. So Wang *et al.* [32] addressed the problem of assigning tasks to processors with minimal energy consumption while maintaining QoS constraints. Here, a new task model is proposed to represent QoS with a frequency instead of a deadline. A ratio called Energy Consumption Ratio (ECR) is defined which gives the relative energy consumption of a task at different frequencies to the maximum frequency of a processor. The goal here is to find optimal frequencies for the tasks at which the cumulative sum of Energy Consumption ratios for all the tasks is minimized. To solve this two heuristics are proposed for a single task and batch jobs respectively. The draw backs to this approach are, representing frequency for a task as it's QoS parameter is impractical.

All the above methods are developed for a single tier web server which service periodic tasks. Horavath *et al.* [33] presented a DVFS mechanism for multi-tier web server systems which hosts complex web services. It dynamically adjusts the server voltages to minimize the total system power consumption in a multi-tier web service environment while meeting end-to-end delay constraints. The proposed web service architecture has a pipeline of many processing stages. Each stage contains a single server which use DVFS capable processors. A coordinated Distributed Voltage Scaling(DVS) policy is introduced, where decisions on frequency adjustments are made on each stage locally while minimizing overall end to end delay. The DVS policy presented here is based on assumption that the CPU delay at a stage, is a convex function of CPU utilization. Based on the assumption, two rules are defined for adjusting the CPU clock speed to maximize the power savings under delay constraints. First, the frequency of the most loaded machine will be increased to the next discrete level, whenever the end to end delay surpasses an upper threshold value. In the same way, whenever the end to end delay goes below a lower threshold, the frequency of the highly loaded machine would be decreased by one discrete level. Compared to other methods dynamic voltage frequency scaling incurs less overhead.

2.3.2 VM Consolidation and Migration based provisioning

One of the ways to reduce power consumption by a data center is by VM consolidation [34] [35] [36][37][38][39]. In this, all the virtual machines are consolidated to few number of servers, thus reducing the amount of the hardware in use. The goal here is to allocate the virtual machine requests to minimal number of physical nodes in an optimal way while satisfying the Quality of Service (QoS) agreements with the user [40][41]. The remaining physical machines can be turned off or switched to low power mode, thereby reducing the energy [42][43][44][35][36]. Once this initial placement is done, at any point of servicing these requests if any physical node is underutilized, the virtual machines running on this lightly loaded node can be allocated to other active nodes through live migration, so as to turn off this lightly loaded node to low power mode. In the same way, virtual machines can be migrated from highly loaded nodes to other active nodes if the virtual machines located on the highly loaded node are experiencing performance degradation.

To achieve this, Beloglazov *et al.* [45] proposed an efficient resource management policy for cloud data centers that is based on VM consolidation. This approach handles heterogeneity of both physical machines and virtual machines and considers QoS requirements of the user while achieving VM consolidation to save energy. In this, the VM consolidation process is divided into two phases: first, Initial Placement of VM's and second, Optimization of the Initial Allocation. The first phase is modeled as bin packing problem with variable bin sizes and prices. To solve this, Modified Best Fit Decreasing (MDFD) heuristic is proposed, in which all the VMs are sorted in decreasing order of current utilization. Each VM in the sorted order is picked one after the other and allocated to a host that provides the least increase of power consumption due to this allocation. In the second phase, VMs are identified for migration to optimize the current allocation. To decide on the VM to migrate, four workload independent heuristics are proposed. These heuristics are based on the idea of setting upper and lower utilization thresholds for hosts and keeping total utilization of CPU by all VMs between these thresholds. Then, these selected VMs for migration are reallocated again using MDFD.

Neeraj *et al.* [46] proposed a hybrid algorithm combing Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Euclidean distance to multi-objective energy efficient virtual machine allocation. The multiple objectives are energy efficiency, minimization of resource waste and minimization of SLA violations. It contains three phases: VM allocation, task scheduling and VM migration. First, an initial VM placement that optimizes energy and resource wastage is done and then Migration is applied for the initial Placement to reduce SLA violations. The SLA policy considered is strong and the SLA parameters considered are purchased resources, response time and deadline time. Initial Placement is modeled as multi-objective multi-dimensional combinatorial optimization problem and euclidean distance is used to identify the optimal point between the two objectives of energy minimization and resource utilization maximization. The power model considered in this work is DVFS based, and it assumes that the servers are DVFS enabled.

Although the basic VM consolidation approaches save considerable energy, workloads may experience significant performance degradation due to virtual machine co-location and migration. When few virtual machines are co-located on the same physical server each VM experiences performance degradation because of the contention for shared resources such as caches and networks. The extent of the performance losses depends on the types of workloads that are co-located. On the other hand, some types of workloads like memory intensive loads, experience more performance loss during migration process than the other types of workloads. It's a challenging issue to mitigate these losses during VM consolidation and migration processes. To address this, K. Ye *et al.* [47] proposed a profiling based frame work for server consolidation. In this, initially detailed profiling of the performance losses that occur to different types of workloads due to VM co-location and migration is done. This profiling information will be used by the server consolidation module to put thresholds on performance losses when finding the optimal consolidation scenario to serve the current workloads. A new problem called, migration planning is defined whose aim is to find the optimal number of migrations for changing the current work load placement to target workload placement, while maintaining the performances losses of each workload under a threshold. It is mapped to linear sum assignment problem and a polynomial time algorithm which uses the migration performance-loss profiles of the different workloads in placement decisions is proposed to solve the problem.

According to Versick and Tavangarian [48], there are certain hard ware and software properties of virtual machines to measure the live migration decisions qualitatively. These properties describe whether decision of migrating a VM from one PM to another PM satisfy certain necessary requirements of the VM and also improve the performance of the VM or not. Based on these properties, a novel dynamic load aggregation mechanism is proposed, which outputs an optimal migration scenario for the current workload. In this, a graph is constructed with PMs and VMs as nodes and the relationships derived through the migration properties are represented as edges. The weight of the edge decides the desirable nature of VM being hosted on a particular PM. K-Means clustering algorithm is applied on this graph to find out the optimal number of physical machines to cover all the virtual machines. Remaining physical machines can be switched to low power mode to save the energy. K -Means algorithm doesn't fare well if the no of VMs and PM's is large, as is the case generally in a normal cloud data centre.

The above approaches do not suit well for real time tasks or applications, as the real time tasks require guaranteed deadlines and above approaches tend to consolidate the workloads on less number of physical machines to save energy [49], resulting in lack of the schedulability for some of the real time tasks. To address this, Zhu *et al.* [50] proposed an energy aware scheduling scheme that optimally balances the energy savings through server consolidation and dead line guarantees of the real time workloads. The core of the idea is to have a rolling horizon, which holds the tasks coming into the system for a while till the need arises for them to be scheduled to meet the deadlines. A new VM is created only when the existing virtual machines cannot serve the current task to be scheduled. If the newly created VM's host doesn't have the capacity to host it, then VM migration is applied to create space for the new VM or if all the active hosts are full, then a new physical machine is turned on to active mode. So at a time only few physical machines would be in active mode, hence reducing the energy consumption of the data centre, while achieving the schedulability of the real time tasks by allocating more resources for the tasks near to deadline. It is assumed that the tasks are independent which is not the case normally, so this approach has to be improved for the case without this key assumption.

As the majority of the applications that run in the data centers are big data applications, and they consume large amounts of energy, Mashayekthy *et al.* [51] proposed a frame work to improve the efficiency of the Map-Reduce jobs for a heterogeneous data center. As each job, consumes different energy on different machines under different time slots, the idea is to schedule them on time slots for which expected energy consumed is less. The deadlines and dependencies of the Reduce jobs, with respect to the Map jobs, are maintained. Here, two scheduling algorithms called, Energy Aware Map Reduce with Service Level Agreements(EMRSA)-I and EMRSA-II are proposed based on a separate heuristic each to characterize the energy consumption of the different machines and also to induce an order relation among the machines. This approach reduces the energy cost of the execution of a single big data application on a data center by intelligently scheduling the map-reduce jobs on the heterogeneous machines.

The technique of saving energy by switching servers to low power modes has negative effects under improper system controls. First, There is a chance that burst arrivals may not get the service or at least they will experience considerable delays. Second, If the switching from low power mode to active mode or vice-versa happens too frequently, it may happen that the energy consumption due to these switching over heads may be more than the energy saved by keeping the servers in low power mode. Third, SLA violations may occur. So to avoid the switching too often, Chiang *et al.* [52] applies a control approach called N-Policy for resource provisioning. In the Queuing systems with N Policy, a server in sleeping mode gets turned on only when the no of items in a queue is greater than some pre defined threshold N, rather than turning it on immediately after an arrival. But if the value of N is too large it may result in poor performance as server will be staying in low power mode for longer durations. Based on this N-Queuing policy, three power saving policies are proposed. First, an ISN policy, in which each server has 3 modes of operation: Busy, Idle and Sleep. Busy mode indicates that it is servicing one or more requests. When there are no remaining requests to be serviced it will be switched to idle mode for a short period of time before switching to sleep mode. When it is in idle mode any arrival of service request causes switch back to active mode. When no requests come during the short idle period, it will attain sleep mode. It will stay in sleep mode, as long as the no of the requests waiting in the queue are less than the predefined threshold N. When the no of requests in the queue are N or more it switches to active mode. Second, SN Policy, doesn't have any mode called as idle mode. It has only two modes, i.e., active and sleep mode. Whenever the server doesn't have any requests to handle it will go to sleep mode. And as per N-Policy a server in a sleep mode will be switched to active mode if and only if the number of waiting requests in the queues exceeds or equals to N. Third SI policy, in which a server stays in sleep mode for a fixed period. Once that fixed time expires, it will be switched to either active mode or busy mode depending on whether there are waiting requests or not. It doesn't follow N-Policy. An algorithm called EGC (Efficient Green Control), calculates the optimized values for service rate and the threshold N while satisfying response time constraints. It is validated that the ISN power policy is better than the remaining two power policies. Though this method decreases the additional energy consumed for restarting the servers to certain extent, still better approaches need to applied to reduce this start up latency, leading to energy efficiency by load prediction.

Markus *et al.* [53] extended the cutting stock problem to solve the problem of consolidating the virtual machines with stochastic workloads on to physical machines. The workload of each VM is considered to be stochastic, i.e, represented with a random variable. Here a convolution of the probability mass functions of the workload of different virtual machines is used to identify the cumulative workload characteristics when those virtual machines are placed onto the same machine. This convolution operation is used to identify the probability that the total workload demand by the virtual machines placed on a physical machine exceeds the physical machine capacity by a pre-defined tolerable value. The probabilities are used in assigning the virtual machines onto the physical machine using first-fit-decreasing manner. The power saving mechanism used here is turning down the physical machines, those do not have any virtual machine on them. The limitations of this work are, the workload is considered in only one dimension, i.e, computing capacity. It is desirable to extend it to other dimensions of the capacity like memory and bandwidth. However, the independence nature of the workloads on these different dimensions is unknown.

2.3.3 Load Prediction Based Provisioning Approaches

In the VM consolidation based resource provisioning, the current set of VM requests will be allocated to few physical machines that are sufficient enough to meet the desired performance requirements. Remaining physical machines will be switched to low power mode. If the set of VM requests in the next instant of time are more, some of the physical machines which were put to low power mode in the previous time point, should be turned on again. Switching off and on a PM also requires considerable amount of energy. If the idle times of the PM's are low or in other words this switching off and on happens frequently, then the energy consumed by them for the switching will be more than the energy consumed if they were kept idle [54][34]. To reduce this re-start latency, Dabbagh *et al.* [55] proposed a framework for resource allocation and provisioning in which, the number of VM requests in the future and their requirements are predicted and based on that prediction, it estimates the number of physical machines required in the near future. The remaining physical machines are turned off to low power mode. It uses K Means clustering algorithm [56] to partition the VM requests into different clusters and uses stochastic wiener filter [57] prediction to estimate the work load of each cluster. To find out the number of physical machines needed to be on to serve these estimated workloads, Best Fit Decreasing (BFD) heuristic is used by the power management module. Based on the output given by this module, some of the physical machines will be turned off to save energy. An adaptive version of the stochastic wiener filter prediction is also proposed to incorporate the varying workload characteristics over the time.

In server consolidation, frequent switching of servers on/off incurs some cost, either in the form of profit lost while waiting for the server to be turned on or the power consumed for this switching process. The power saving mechanism of turning down hosts in an environment with unpredictable and size varying workload is a risky decision in terms of quality of service. Kusic *at el.* [58] proposed a risk aware controller to address these issues. In this, the resource provisioning problem is formulated as sequential optimization under uncertainty and solved using a limited look ahead control. Here, the resource provisioning in the cloud is formulated as multi-objective problem, i.e. maximizing the cloud provider profit by minimizing the power consumption and SLA violations. Power minimization and SLA violations minimization are conflicting objectives. As a solution a predictive control approach called as limited look ahead control is proposed. The objective of maximizing profit is converted into risk aware utility function which indicates the controller preference between different resource provisioning configurations. For each resource provisioning configuration the risk aware utility is calculated and out of that optimal provisioning is selected. This process is done periodically as the workload volume changes with time.

2.3.4 Topology Aware Provisioning Policies

All the above approaches consider consolidation of virtual machines to reduce the energy consumption of physical machines but they ignore the energy consumption of the network elements. However the data center network energy constitutes nearly 10-20% of the total data center energy consumed, which is a significant quantity. Hence, there is a need to develop topology aware VM scheduling techniques.

To address this, Jiankang *et al.* [59] presented a VM allocation approach that optimizes both physical machine and network resource utilization simultaneously. It's scheduling policy has two stages called static VM placement and dynamic migration. In the first stage a static placement of the virtual machines on the physical machines is done by considering the traffic flows and network topology to reduce the energy consumed by the network devices. In the second stage using first fit decreasing heuristic an optimal no of migrations is calculated to adopt to the changes in the work load while considering the network topology and the communication among the virtual machines. Even though effort is made to reduce the energy consumption of both physical machines and network elements, the performance constraints of the virtual machines based on the QOS agreements are not considered in this work.

Heller *et al.* [60] proposed a network wide power manager called Elastic Tree, that dynamically adjusts the sets of active switches and links to suit the changing demands of the traffic load. It continuously monitors data center traffic conditions and chooses the set of network elements that must stay active to meet performance and fault tolerance goals; then it powers down as many unneeded links and switches as possible. To choose the switches and links to power down several methods like greedy bin packer, topology aware heuristics and prediction methods are proposed.

Wang *et al.* [61] addressed the problem of optimal bandwidth allocation and energy aware routing problem in cloud data center. The energy aware routing problem is modelled as Multi-Commodity-Flow Problem and it is proved NP-Hard by reducing the problem to 0-1 knapsack problem. Here Blocking Island paradigm derived from artificial intelligence is used for representing Bandwidth resource at different levels of network abstraction. For each node β-Blocking Islands are calculated using a greedy algorithm. These blocking islands are used to construct Blocking Island Graph which is an abstraction of the available resources of the entire network. The Blocking Island Graph is again decomposed into multiple levels in the decreasing order of the bandwidth demand, and it is called as Blocking Island Hierarchy. Bandwidth allocation problem is solved by two heuristics called dynamic heuristic demand selection with lowest common father (DSLH) and dynamic heuristic demand selection with fewest nodes(DSFN) which uses the blocking island hierarchy. The Blocking Island Hierachy is also used in calculating optimal routes for the network flows.

Son *et al.* [62] proposed an energy efficient VM consolidation with dynamic overbooking in software defined cloud data center. In software defined data center network as the control plane is separated from the data plane, the routing decisions can be easily adopted to the changing workload demands. The traffic of the communication flows between different virtual machines can easily be routed in efficient manner based on the changing workload and can be consolidated onto minimum number of network links and switches. Here, the objectives are minimizing SLA violations and energy consumed by both hosts and network elements.

2.3.5 Thermal Aware Provisioning Approaches

Resource virtualization and consolidation achieve energy efficiency by increasing the system utilization and decreasing the number of active servers in the data center. But the improved system utilization increases the amount of heat dissipated from the severs. This in turn would result in higher power consumption for the cooling systems and also lower system reliability by persistent system hardware failures, thus incurring more cost to the provider [63][64][65]. Hence, the VM consolidation approaches should find out an optimal trade-off between system utilization and thermal efficiency to achieve energy efficiency.

Yao *et al.* [65] proposed a work load assignment method that minimizes the data center power consumption by considering the correlation between servers and cooling system. A novel adaptive control approach is presented where both quality of control constraints and thermal constraints are satisfied simultaneously. In this, decisions on workload assignment to different racks are taken based on the co-relation between the thermal characteristics of the racks estimated by an online recursive method and the performance constraints represented in the form of control inputs.

A significant portion of the energy consumed in the data center is due to the cooling systems employed at the data centers. The energy consumed by the cooling system partially depends on the ambient temperature surrounding the data center. Xu *et al.* [66] exploited this property to develop a temperature aware work load management scheme for the geo-distributed data centers. The workloads are allocated to the data centers that have cooler ambient temperatures varying with geographic location and time, while considering latency to ensure the desired performance to the users. In addition to that, the varying electricity prices is also considered in the scheduling decisions to achieve reduction of energy-cost. The delay tolerant property of the batch workloads is exploited to achieve energy-cost savings by delaying the batch jobs to execute them at times and places where the ambient temperature and the electricity prices are low . But this scheme doesn't consider resource virtualization, the most prominent feature of the cloud along with the complex issues that comes with it during workload assignment.

To minimize the total energy costs of data-center operation while achieving a desired level of system reliability, Tang *et al.* [67] proposed a thermal-aware task-scheduling technique. In a blade server, multiple blades are integrated into each chassis. All these blades in a chassis share a common supply and cooling fan. Each blade may itself have many processors. Operation of a chassis incurs chassis start up power consumption, along with actual power consumption of the blades. Hence, the power consumption cost of adding a task to one chassis may be different depending upon whether it involves waking up an idle chassis or an idle blade. Based on this observation, several thermal-aware task-scheduling heuristics are proposed, which exercise the trade-off between cost of start-up and power saving coming from turning off the blade servers.

2.3.6 Renewable Energy Aware Resource Provisioning Approaches

As the cooling costs contribute to larger portion of the energy consumed by the data centers lately, cloud providers are building the data centers powered by renewable energy resources [68]. Hence the VM scheduling approaches should be able to exploit this by placing the workloads onto the servers of these data centers supported by renewable energy resources.

In that direction, Mandal *et al.* [68] proposed a novel VM placement technique, that uses Renewable Energy Source (RES) aware migration heuristics. In the RES aware placement initially the VM's are allocated onto the data centers which are powered by renewable energy sources. After such data centers get exhausted only, the VM's are placed on non RES data centers. After the initial placement, VMs can be migrated across the data centers to relocate the energy demand to the data centers with RES. Three migration heuristics are proposed. First, in Renewable energy-aware migration heuristic, at the time of each energy prediction period, VM is migrated from one data center to another data center if the later is having more green energy sources. Second, in Migration cost aware metric, a threshold is set for data center brown energy consumption and whenever it exceeds that threshold VM's are migrated to other data centers with better green energy resources. Third, traffic aware migration heuristic considers renewable energy, migration cost and also future load prediction to take a decision on VM migration.

In self sustained data centers, completely powered by renewable energy sources, the amount of the power available varies with time. On the other hand the amount of power required by the data centre depends on the resource requirements of the hosted workloads. Generally, to improve the system utilization different types of workloads with different system requirements like transactional workloads, batch jobs are co located. Cheng *et al.* [69] proposed an elastic power aware resource provisioning approach called ePower which uses the characteristics of the transactional and batch workloads to adjust the resource allocations based on the power available to meet the performance constraints. As transactional workload intensities vary with time and batch jobs can tolerate delays, when the intensities of the transaction workloads are more and the deadline of the batch job is far, ePower allocates more resources to the transactional workloads out of the resources limited by the power supplied at that instant of time. When the deadline for the batch job is approaching, more resources are added to the batch job. This adjustment is done at each defined time point by using a power aware simulated annealing algorithm which designs a cooling schedule by considering dynamic power supply and characteristics of heterogeneous workloads. As the resource allocations are constrained by the power supply, only few resources will be in active mode serving the workloads. Remaining resources could be turned off to low power modes to save the energy.

Bruneo *et al.* [70] proposed an analytical frame work that is based on stochastic reward nets. In this, a layered approach is followed to separate the issues related to virtual machines and physical machines. Both the layers are modeled using Petri nets and the dependencies between both the layers under different energy saving policies are presented and analyzed.

2.3.7 Energy Efficient Resource Management in Federated Cloud

The notion of a federated cloud and it's essential qualities are first introduced by Rochwerger *et al.* [71]. In this work, the essential characteristics of a cloud federation and the issues in forming a cloud federation are presented in detail. To address the requirements of a federated cloud a model called RESERVOIR is proposed for the cloud providers to form as a cloud federation by pooling their resources together to service the requests of the users [72]. However, the mechanism for the cloud federation formation is not presented in this work.

Buyya *et al.* [73] discusses the challenges, and architectural issues in building a federated cloud. Moreover, a framework is presented for facilitating efficient management of federated clouds. In this work also cloud federation formation mechanism is not presented.

Goiri *et al.* [74] presented equations that can characterize decisions a cloud provider makes about outsourcing tasks to the other cloud providers as part of a federated cloud. The incentives the other cloud provider gets for offering their resources is not considered in this study.

Mashayekhy *et al.* [75] presented a federation formation mechanism to provide services to the user requests. In this work, the resource sharing in federated cloud among multiple cloud providers is modelled as a hedonic game called cloud federation formation game and the properties of the game like fairness and stability are studied. A cloud federation formation mechanism is presented based on merge-split rules and it is individually stable with respect to cloud providers. The fairness in the profit distribution is achieved by normalized Banzhaf value.

Mehedi Hassan *et al.* [76] proposed a game theoretic based resource and revenue sharing mechanism in federated cloud. The proposed mechanism aims at maximizing the social welfare of the federated cloud by reducing the overall energy consumption. In this work, the internal demand of a cloud provider is analyzed using $M/G/m/m + r$ queuing system and the amount of resources offered to the cloud federation by the participating cloud provider is based on the internal demand and it's capacity. The proposed mechanism is proved to be individually stable.

Chapter 3

Off-line VM Scheduling and Routing in Multi-tenant Cloud Data Center

The energy costs of a data center mainly arises from the energy consumed by physical servers, networking components and cooling equipment. Typically, an idle server in a data center consumes more than 50% of the energy it consumes when it is fully loaded [4]. Based on this finding a popular approach to save energy is server consolidation [77] [45] [28] [78] [47]. Server consolidation is the process of allocating virtual machines that service one or many tasks onto a few servers. The remaining servers are switched off entirely or switched to a low power mode to save energy, as the energy-disproportionate servers in data centers consume a considerable amount of power even when they are idle. Moreover, the network switches and the routers in the data center network contribute to a significant portion of the total power consumed by the data center. The networking components also consume a sizable power even when they are idle, almost equivalent to 30% of the power consumed when they are fully loaded [4]. Like in the case of physical servers, a prudent approach to save energy is allocating the network flows of the virtual machines to a few number of networking components and turning off the remaining ones as they do not have any load on them.

At a time, in a cloud data center, many composite applications belonging to different tenants need be deployed. These composite applications are made of several sub-tasks which are logically connected by data and flow dependencies. Generally, a virtual machine in a cloud can serve one or more tasks of an application. Each tenant requests several virtual machines to service the tasks belonging to their application. Virtual machines, belonging to a tenant, communicate and exchange data with each other during their execution.

An user or tenant, specifically who wants to deploy dead-line sensitive application on the cloud, submits a request for a set of virtual machines to the cloud provider for booking them in advance. As the behavior of the application belonging to the tenant is known in advance, the communication patterns of the virtual machines is known. If two communicating virtual machines are placed very far in the data center network, then the data that is being exchanged would go through many network elements resulting in higher network energy consumption. Cloud provider receives many such requests from multiple tenants for advance booking of the virtual machines to run their deadline-sensitive applications. It leaves an opportunity for the cloud provider to plan for an effective consolidation of the virtual machines belonging to multiple tenants onto the physical machines such that the energy consumed for provisioning these vm's is minimized. In this work, the idea is to consolidate the virtual machines onto minimal number of servers in such a way that any two communicating virtual machines are placed very close to each other in the proximity of the data center network. Then the communication data among the virtual machines is routed through or consolidated onto minimal number of links and switches to save the energy by turning off the remaining idle links and switches.

Contributions of the chapter

- We model the joint optimization of server and network element energy consumption during VM scheduling and routing as an integer programming problem
- A phase wise optimization approach containing two meta heuristic algorithms based on ant colony optimization is proposed as a solution to the problem.
- The proposed solution is tested for three standard data center network topologies namely 3-Tier, B-Cube and Hyper-Tree of different sizes and it's effectiveness is compared with two standard heuristic solutions, first-fit and round robin.

3.1 System Model

3.1.1 Data Center Model

A heterogeneous single site cloud data center, consisting a set of m heterogeneous servers $\mathbb{H}\{H_1, H_2, \dots H_m\}$ is considered here. The capacity of a server H_i in terms of computing power, ram size and storage is given by $Z_i^{\texttt{comp}}$,$Z_i^{\texttt{mem}}$ and $Z_i^{\texttt{storage}}$ respectively. The data center network topology considered here is a hierarchical topology that connects the switches in different layers namely core, aggregate and edge. The set of switches is represented as $\mathbb{S}\{S_1, S_2, \dots\}$. As the switches are heterogeneous and the capacity in terms of the number of ports varies with the switch. The capacity of a switch S_s is given by $Z_s^{\texttt{port}}$. The data center network is represented as a weighted graph $\mathcal{G}\left(\mathbb{H} \cup \mathbb{S}, \mathbb{E}\right)$, where vertices of the graph include both servers and switches, and the edges are either between a server and a switch or two switches. The weight $w\langle a, b\rangle$ on an edge $\langle a, b\rangle \in \mathbb{E}$ gives the bandwidth capacity of the link between the two nodes a and b. The data center is logically partitioned into several clusters where each cluster contains physical machines that are arranged into several racks. The cluster set is given by $\mathbb{C}$ and racks in a cluster c is given by $\mathbb{R}_c$. A rack contains a Top-of-Rack(ToR) switch, to which all the physical machines in the rack are connected. The set of physical machines belonging to rack r is denoted as $\mathbb{H}_r$. Each top of the rack switch is connected to all the aggregation switches in its cluster, and the aggregation switches of each cluster are connected to all the core switches of the data center.

3.1.2 Tenants Model

At any time a cloud data center hosts multiple tenants and they are represented by a set $\mathbb{N}\{N_1, N_2, \dots N_n\}$. The set of virtual machines requested by a tenant N_j is represented by $\mathbb{V}_j$. The virtual machines belonging to a tenant are of different types and accordingly their resource requirements vary. The resource requirement of a virtual machine k belonging to a tenant j is given by $R_{jk}^{\texttt{comp}}$, $R_{jk}^{\texttt{mem}}$ and $R_{jk}^{\texttt{storage}}$ concerning its computing power, ram size and storage respectively. The communication among the virtual machines belonging to a tenant j is given by the weighted graph $\mathcal{C}_j\left(\hat{\mathbb{V}}_j, \hat{\mathbb{E}}_j\right)$, where the nodes represent the virtual machine belonging to tenant j, and an edge represents the potential communication between the two virtual machines that the endpoints of the edge represent. The weight $\hat{w}_j\langle\hat{a}, \hat{b}\rangle$ on the edge $\langle\hat{a}, \hat{b}\rangle$ indicate the amount of bandwidth required for the communication between the virtual machines $\hat{a}$ and $\hat{b}$. Let $\mathbb{F}_j$ is the set of communication flows constructed from the communication graph of tenant N_j. Let the set $\mathbb{V}=\bigcup_{j=1}^{n} \mathbb{V}_j$, contain all the virtual machines belonging to different tenants and are indexed sequentially.

3.1.3 Energy Consumption Models

The energy costs of a cloud provider depend mainly on the energy consumption by physical servers, networking components, and cooling equipment of the data center. The power consumed by all servers and network components of a data center constitute approximately 40% of the total data center power consumption, while the cooling equipment consumes approximately 35% of the total data center power consumption [79]. If the power consumption of the servers and networking devices is reduced, the power drawn by the cooling equipment also decreases. Here the focus is on reducing the power consumption of only the physical servers and the networking devices. Reducing the power consumption of cooling system is not considered here. The power consumption models considered here of a server and a switch are described below.

3.1.3.1 Power consumption of a server

The power consumption model considered here is utilization-based power consumption. A server in the data center consumes some fixed amount of power when it is on and any additional power consumption depends on its utilization. The utilization of a server depends on how many fractions of its resources are used by the virtual machines that are placed on it. So the power consumption of a physical server in a data center can be divided into two parts: first, a baseline static consumption and second, dynamic power consumption that is load dependent. Here the dynamic power consumption is based on VM power metering that is followed in [80]. The total power consumption of a server at a time t is given by

$$\mathbf{P}_i(t) = \mathbf{P}_{\texttt{baseline}} + \sum_{\hat{v} \in \mathbb{V}_i^t} \mathbf{P}(\hat{v}) \tag{3.1}$$

where $\mathbf{P}_{\texttt{baseline}}$ is the base line power consumption, $\mathbb{V}_i^t$ gives the set of virtual machines that are placed on he physical machine i at time t, $\mathbf{P}(\hat{v})$ is the power consumption of the virtual machine $\hat{v}$ on the physical machine i.

If the power consumed by a VM is expanded further in terms of the utilization of the VM in cpu, memory and i/o as in [81], then the power consumption of a server is expressed as

$$\begin{aligned}\mathbf{P}_i(t) = \alpha \cdot \sum_{\hat{v} \in \mathbb{V}_i^t} \mathbf{U}_{\texttt{cpu}}(\hat{v}) + \beta \cdot \sum_{\hat{v} \in \mathbb{V}_i^t} \mathbf{U}_{\texttt{mem}}(\hat{v}) \\ + \gamma \cdot \sum_{\hat{v} \in \mathbb{V}_i^t} \mathbf{U}_{\texttt{io}}(\hat{v}) + n_i \cdot e + \mathbf{P}_{\texttt{baseline}}\end{aligned} \tag{3.2}$$

where α, β, γ are the weight-ages for utilization of the server by the VM in CPU, memory and IO respectively.

3.1.3.2 Power consumption of a switch

The energy consumption of a networking component depends on the number of active line cards, ports and the traffic flowing through it. The current data centers mostly comprise of commodity switches, so only the energy consumption model of a switch is considered here. The power saving mechanisms applied for switch here are, turning off a port as it saves 0.5% of the total energy consumed by the switch [82], turning off line card entirely when there are no active ports on it and turning off the switch altogether. So the power consumption of a switch with one line card, which is the case generally in commodity switches of the data center can be written as

$$\mathbf{P}_s(t) = \mathbf{P}^s_{idle} + \mathbf{P}^s_{port} \times n^s_{port}(t) \tag{3.3}$$

where P^s_{idle} is the basic power consumed with all of its ports disabled, P^s_{port} is the power

Table 3.1: EVMSR: Notations

Notation	Meaning
m	Total no of physical servers
n	Total no of tenants
$\mathbb{T}$	Set of time epochs
$\mathbb{H}\{H_1, H_2, \dots H_m\}$	Set of physical Servers
Z_i^{comp}, Z_i^{mem}, $Z_i^{storage}$	Capacities of i^{th} server in terms of computational power, main memory and storage
$\mathbb{N}\{N_1, N_2, \dots N_n\}$	Tenant Set
$\mathbb{V}_j\{V_j^1, V_j^2, \dots\}$	Set of virtual machines belonging to j^{th} tenant
$\mathbb{V}$	Set of virtual machines of all the tenants
R_{jk}^{comp}, R_{jk}^{mem}, $R_{jk}^{storage}$	Requirements of k^{th} VM of j^{th} tenant
$\mathcal{C}_j\left(\mathbb{V}_j, \hat{\mathbb{E}}\right)$	Communication graph of j^{th} tenant
$\mathbb{S}\{S_1, S_2, \dots\}$	Set of Physical Switches
Z_s^{port}	The capacity of S_s switch
$\mathcal{G}\left(\mathbb{H} \cup \mathbb{S}, \mathbb{E}\right)$	Data center network graph
$O_i(t)$	Indicate whether server H_i is active at time t or not
$O_s(t)$	Indicate whether switch S_s is active at time t or not
$O_{\langle a,b\rangle}(t)$	Indicate whether link $\langle a, b\rangle$is active at time t or not
$\mathbb{F}_j$	the set of communication flows constructed from the communication graph of tenant N_j
f_j^l	l^{th} flow belonging to tenant j
$route_t(f_j^l, \langle a, b\rangle)$	Whether flow f_j^l goes through link $\langle a, b\rangle$ or not
$route_t\left(f_j^l, S_s\right)$	Whether flow f_j^l goes through switch S_s or not.
$\mathbb{V}_i^t$	Set of virtual machines that are placed on he physical machine H_i at time t
$\mathbb{U}$	VM schedule

consumed by a single port when it is on, $n_{port}^s(t)$ is the total number of active ports that are present in the switch.

3.2 Problem Formulation

Given a data center network $\mathcal{G}(\mathbb{H} \cup \mathbb{S}, \mathbb{E})$ and virtual machine requests from n number of tenants, with a communication graph $\mathcal{C}_j\left(\hat{\mathbb{V}}, \hat{\mathbb{E}}\right)$ for each tenant indicating the communication pattern of the requested virtual machines, our aim is to consolidate the VM requests onto minimal number of servers and switches. The remaining servers and switches are turned off to save energy. So the objective here is to consolidate the virtual machines belonging to multiple tenants onto the physical servers in such a way that the total energy cost of servers and the network elements is minimised while satisfying the capacity constraints of servers and links. We model this **J**oint **O**ptimization of **S**erver and **N**etwork elements **E**nergy (**JOSNE**) consumption during VM scheduling and routing in cloud as an integer programming problem in the following way.

Let $\mathbf{X}_i^{v_j^k}(t)$ is a boolean variable indicating whether v_j^k, k^{th} virtual machine of tenant j, placed on server H_i at time t. Let $\mathbf{O}_i(t)$ indicate whether the server H_i is switched on or off at time t. In the same way, $\mathbf{O}_s(t)$ and $\mathbf{O}_{\langle a,b\rangle}(t)$indicate whether the switch S_s and link $\langle a, b\rangle$ is switched on or off at time t respectively. Let $\texttt{route}_t\left(f_j^l, \langle a, b\rangle\right)$ is a Boolean variable indicating whether the communication flow f_j^l belonging to tenant j, goes through the link $\langle a, b\rangle$ at time t or not. Then the problem can be formulated as

3.2.1 Objective Function

$$\textbf{min} \quad \sum_{\forall t\in\mathbb{T}} \left(\sum_{\forall i\in\mathbb{H}} \mathbf{P}_i(t) \times \mathbf{O}_i(t) + \sum_{\forall s\in\mathbb{S}} \mathbf{P}_s(t) \times \mathbf{O}_s(t) \right) \tag{3.4}$$

Here, the first term is the total energy consumed by the server machines and second term is the total energy consumed by the network elements in the total duration of time. So the objective is minimization of the energy consumed by both physical machines and network elements.

3.2.2 Constraints

$$\sum_{\forall i\in\mathbb{H}} \mathbf{X}_i^{V_j^k}(t) = 1, \qquad \forall t \in \mathbb{T}, \forall j \in \mathbb{N}, \forall k \in \mathbb{V}_j \tag{3.5}$$

$$\sum_{\forall i\in\mathbb{H}} \mathbf{X}_i^{V_j^k}(t) \leq \mathbf{O}_i(t), \forall t \in \mathbb{T}, \forall j \in \mathbb{N}, \forall k \in \mathbb{V}_j \tag{3.6}$$

$$\sum_{\forall i\in\mathbb{H}} \mathbf{X}_i^{jk}(t) * R_{jk}^{comp} \leq Z_i^{comp}, \forall t \in \mathbb{T}, \forall j \in \mathbb{N}, \forall k \in \mathbb{V}_j \tag{3.7}$$

$$\sum_{\forall i \in \mathbb{H}} \mathbf{X}_i^{jk}(t) * R_{jk}^{mem} \leq Z_i^{mem}, \forall t \in \mathbb{T}, \forall j \in \mathbb{N}, \forall k \in \mathbb{V}_j \tag{3.8}$$

$$\sum_{\forall i \in \mathbb{H}} \mathbf{X}_i^{jk}(t) * R_{jk}^{storage} \leq Z_i^{storage}, \forall t \in \mathbb{T}, \forall j \in \mathbb{N}, \forall k \in \mathbb{V}_j \tag{3.9}$$

$$\sum_{\forall l \in \mathbb{F}_j} bd\left(f_j^l\right) * \texttt{route}_t\left(f_j^l, \langle a, b \rangle\right) \leq W\langle a, b \rangle, \forall t \in \mathbb{T}, \forall j \in \mathbb{N} \tag{3.10}$$

$$\texttt{route}_t\left(f_j^l, \langle a, b \rangle\right) \leq \mathbf{O}_t\langle a, b \rangle, \forall j \in \mathbb{N} \forall l \in \mathbb{F}_j \tag{3.11}$$

$$\begin{aligned} \forall (a, b) &\in \mathbb{E}, \\ \mathbf{O}_{\langle a,b \rangle}(t) &\leq \mathbf{O}_a(t) \\ \mathbf{O}_{\langle a,b \rangle}(t) &\leq \mathbf{O}_b(t) \\ \mathbf{O}_{\langle a,b \rangle}(t) &= \mathbf{O}_{\langle a,b \rangle}(t) \end{aligned} \tag{3.12}$$

$$n_{port}^s(t) = \sum_{\forall (\langle s,a \rangle \vee \langle a,s \rangle) \in \mathbb{E}} \mathbf{O}_t\langle a, s \rangle, \forall s \in \mathbb{S} \tag{3.13}$$

$$\sum_{\forall l \in \mathbb{F}_j} \texttt{route}_t\left(f_j^l, \langle i, s \rangle\right) \leq \mathbf{X}_i^{src\left(f_j^l\right)}(t), \forall t \in \mathbb{T}, \forall j \in \mathbb{N}, \forall i \in \mathbb{H}, \forall s \in \mathbb{S} \tag{3.14}$$

$$\begin{aligned} &\sum_{\forall l \in \mathbb{F}_j} (\mathbf{X}_s^{src\left(f_j^l\right)}(t) - \mathbf{X}_s^{dest\left(f_j^l\right)}(t)) = \\ &\sum_{\forall i \in \mathbb{H}} \texttt{route}_t\left(f_j^l, \langle s, i \rangle\right) - \sum_{\forall i \in \mathbb{H}} \texttt{route}_t\left(f_j^l, \langle i, s \rangle\right), \forall t \in \mathbb{T}, \forall j \in \mathbb{N}, \forall s \in \mathbb{S} \end{aligned} \tag{3.15}$$

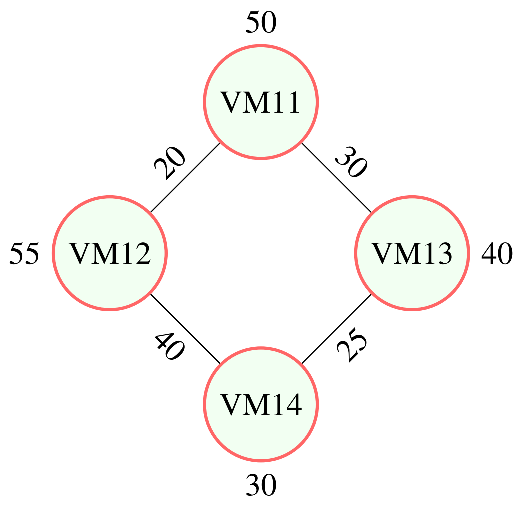
$$\sum_{\forall l \in \mathbb{F}_j} \mathtt{route}_t\left(f_j^l, \langle a, s\rangle\right) = \mathtt{route}_t\left(f_j^l, \langle s, a\rangle\right), \forall t \in \mathbb{T}, \forall j \in \mathbb{N}, \forall s \in \mathbb{S}, \forall a \in (\mathbb{S} \cup \mathbb{K}) \tag{3.16}$$

Equations 3.5 and 3.6 are the assignment constraints of server. Equation 3.5 ensures that each VM is allocated to only one server. Equation 3.6 makes sure that each VM is allocated only to an active server. Equations 3.7, 3.8, 3.9, and 3.10 are the capacity constraints of server and link respectively. Equations in 3.12 are the link constraints, where a link would be active only when both the end point of the link are active and maintains the bidirectional property of the link. Equation 3.13 is used to calculate the number of active ports of a switch. Equations 3.11, 3.14, 3.15, and 3.16 are the constraints related to flows. Equation 3.11 ensures that flow goes through only on active switch. Equation 3.14 guarantee that the flow starts from a server s only if the source (VM) is placed on the server s. If the source and destination VM's of a flow are placed on to the same server then the flow should not go through any switch and that is achieved by the Equation 3.15. Equation 3.16 maintains the flow conservation property for a switch.

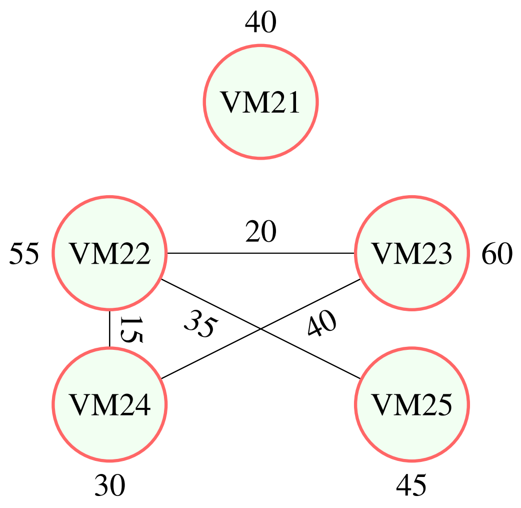
3.2.3 Explanation with an Example

The energy aware VM Placement and routing problem can be explained with an example. Consider two tenants, where the application of the first requires 4 virtual machines namely $VM11$, $VM12$, $VM13$, and $VM14$ and the application of the second tenant requires 5 virtual machines namely $VM21$, $VM22$, $VM23$, $VM24$, and $VM25$. The capacity requirements and the communication patterns of the virtual machines of both the tenants are given as for the Figure.3.1. The data center network along with capacities of the physical machines and the physical links are given by a generic graph as in Figure.3.2 without adhering to any particular data center network topology. Suppose these virtual machines are placed on the physical machines as given in the Figure.3.3. As it can be observed from the diagram this placement causes significant load on the switches and links as some of the communicating virtual machines are placed on different physical machines that are located far apart. Due to the placement A, the physcial machines that need to be kept active are PM1, PM2, PM4 and PM5 and the switches that need to kept on are S2, S3, S4, S5, S6, S7 if the communication flows are routed as per the given Figure.3.3. A better placement of the virtual machines than the placement A is given by the Figure.3.4. It is resulting in less load on the network elements. The physcial machines that need to be kept active because of the placement B are PM1, PM2, and PM3 and the switches that need to be kept on are S3, S6, and S7 if the communication flows are routed as per the given Figure.3.4. So as the number of physical machines and switches that need to be active because of the placement B are less, it is more energy efficient.

Once the placement of the virtual machines is done, the communication flows need to be routed through minimal number of links and switches. So for the placement B, two flows from PM1 to PM2 need to routed through minimal number of links and switches. The possible paths for two flows (VM14, VM12) and (VM14, VM13) are PM1, S3, PM2, PM1, S3, S6, S7, PM2, PM1, S2, S6, S7, PM2,PM1, S2, S6, S3, PM2. In Figure.3.4, the paths selected for the two flows are PM1, S3, PM2 and PM1, S3, S6, S7, PM2 respectively resulting in minimum number of switches and links saving energy of the network elements.



(a) Tenant 1



(b) Tenant 2

Figure 3.1: EVMSR Example 1: Communication Graphs of Virtual Machines

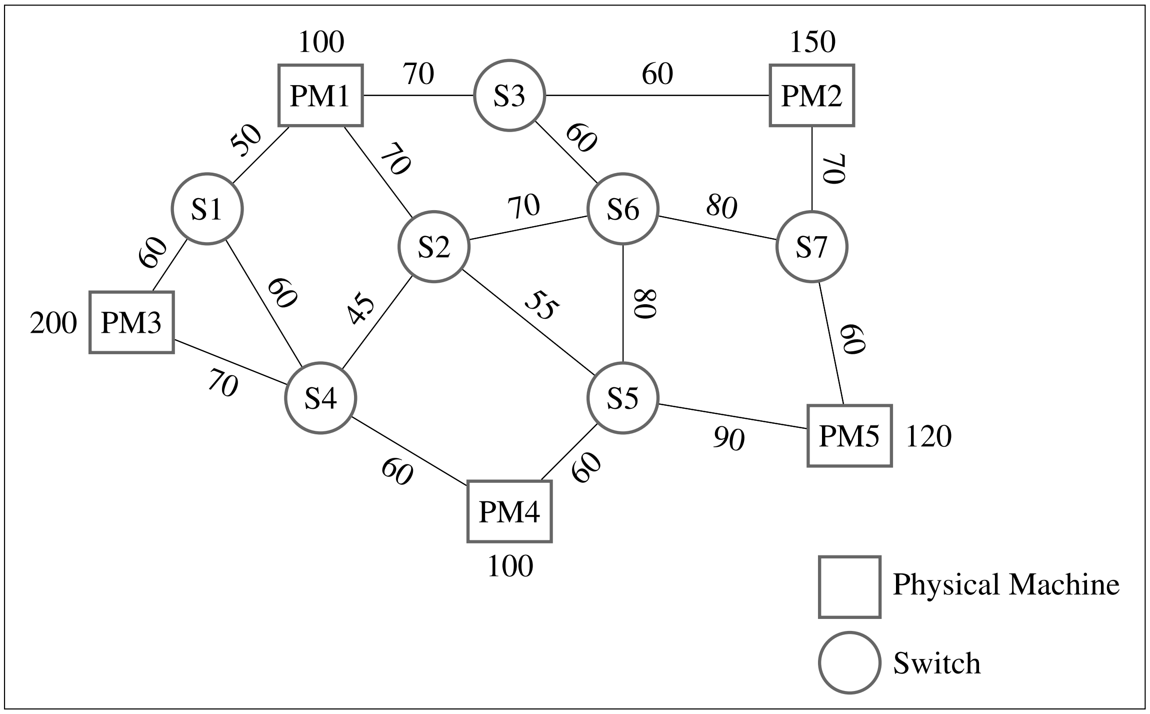


Figure 3.2: EVMSR Example 1: Data Center Network

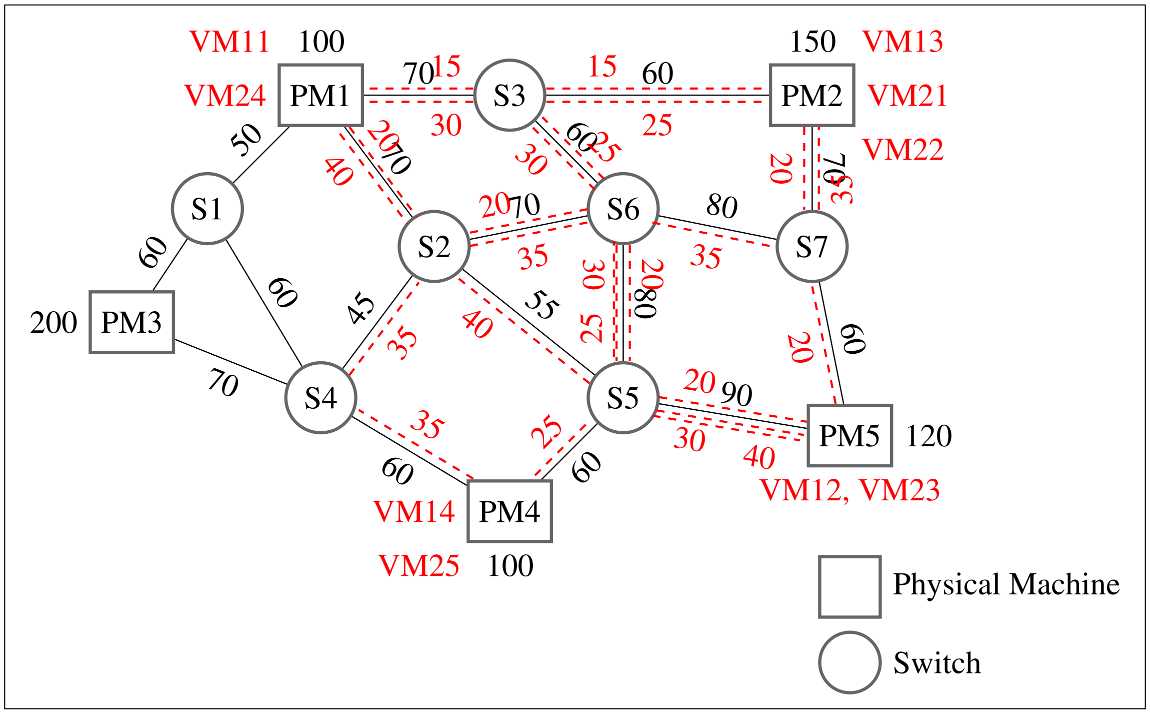


Figure 3.3: EVMSR Example 1: VM Placement A

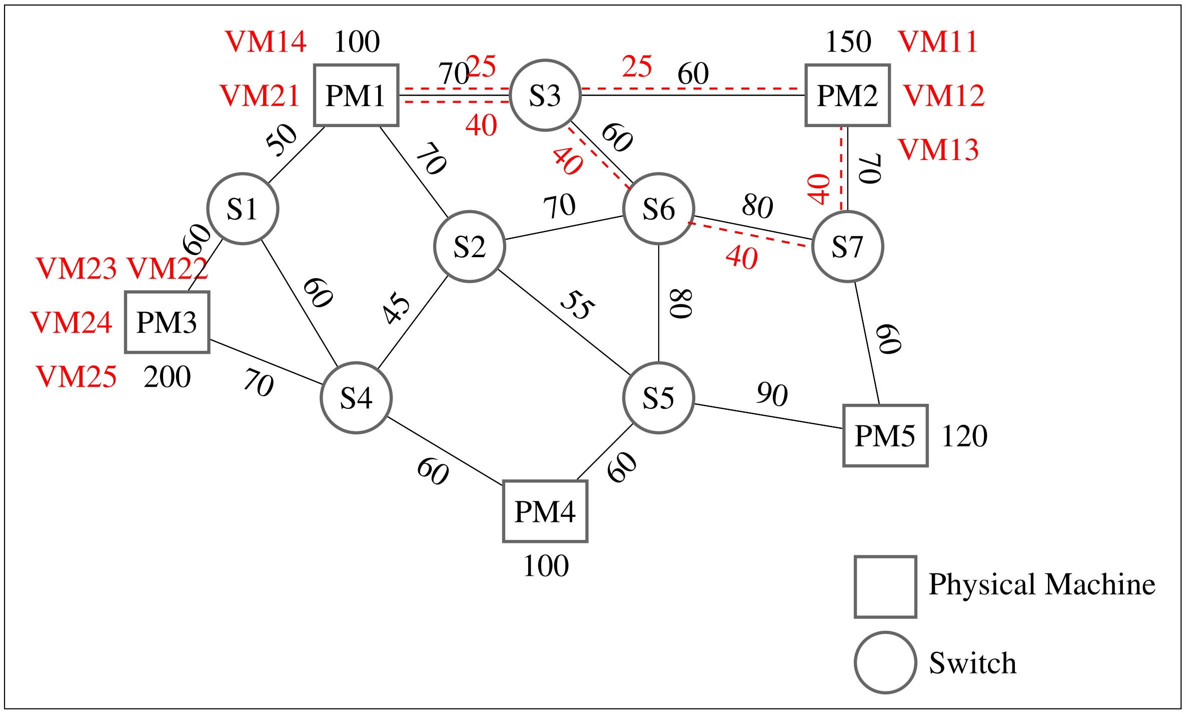


Figure 3.4: EVMSR Example 1: VM Placement B

3.3 Energy Efficient VM Scheduling and Routing (EVMSR)

The solution to the problem is divided into two stages. The first stage is finding an optimal schedule of virtual machines belonging to different tenants. Once the virtual machines are placed optimally, the communication flows belonging to different tenants need to be routed through the links in such a way, that reduces the total power consumption of the network switches. This is termed as the second stage.

3.3.1 Communication Aware VM Scheduling

The first stage of the problem is finding an optimal placement of the virtual machines onto the physical machines, which minimises the power consumption of the servers while considering the communication among the virtual machines in placement decisions. This subproblem of JOSNE can be proved to be NP-Complete by reducing it to bin-packing problem [83]. So as a solution to the subproblem, an ant colony based meta-heuristic static algorithm is proposed here. In Ant Colony Optimization (ACO), a set of artificial ants construct the solution to a given optimisation problem by searching the solution space intelligently with effective communication among themselves regarding better solutions using pheromone values [84]. In ACO, each ant finds a solution by adding sub-components of it iteratively to the partial solution. The sub-component is selected probabilistically, and its probability depends on two things. One, the attractiveness of the sub-component, which is computed based on a heuristic indicating its prior desirability and second, trail value of the sub-component, that gives the historical value of the sub-component. The trail values, also called as pheromone values are updated with the best solution after the completion of the solution. A detailed explanation regarding the heuristic, pheromone values and solution construction by an ant is given below.

3.3.1.1 Heuristic

To find an optimal placement of virtual machines on to the physical machines, a heuristic called affinity is proposed. The heuristic is defined differently for physical machine, rack and cluster. For a physical machine H_i and a virtual machine set $\mathbb{V}$ it is defined as

$$\texttt{Affinity}(\mathbb{V}, H_i) = \sum_{\forall \hat{k} \in \mathbb{V}} \texttt{Affinity}\left(\hat{k}, H_i\right) \tag{3.17}$$

where

$$\texttt{Affinity}(\hat{k}, H_i) = \sum_{\forall v \in \mathbb{X}_i^{jv}(t)=1} \hat{w}_j(v, \hat{k}) \tag{3.18}$$

It gives the total amount of load that could be removed from the switches if the VM $\hat{k}$ is placed on the server H_i, with some virtual machines are already allocated to it.

Similarly, affinity of a rack r for a virtual machine set $\mathbb{V}$ is defined as

$$\texttt{Affinity}(\mathbb{V}, r) = \sum_{\forall H_i \in \mathbb{H}_r} \texttt{Affinity}(\mathbb{V}, H_i) \tag{3.19}$$

It gives the benefit in terms of the load that would be removed from the network components outside the rack, if a virtual machine is to be allocated to some machine belonging to the rack. More the affinity value, more the benefit we get if we place the VM on any machine in rack r. Affinity of a cluster is given by

$$\texttt{Affinity}(\mathbb{V}, c) = \sum_{\forall r \in \mathbb{R}_c} \texttt{Affinity}(\mathbb{V}, r) \tag{3.20}$$

It gives the affinity of the virtual machine set to a cluster. Like in the case of rack, here also if affinity value is more , more benefit we would get if we place the virtual machines on any machine in cluster c.

3.3.1.2 Pheromone Values

Pheromone values give the historical information accumulated by the solutions constructed till now. These values direct the search towards the best solutions. Here, the pheromone trail is left between two virtual machines. It gives the historical preference of two virtual machines to get allocated together onto a physical machine. For any virtual machines v_1 and v_2 it is indicated by $\tau[v_1, v_2]$ and is given as below.

$$\tau[v_1, v_2] = \begin{cases} \hat{w}_j[v_1, v_2] & \text{, if } v_1 \text{ and } v_2 \text{ gets packed in } \mathbb{U} \\ 0 & \text{, otherwise;} \end{cases} \tag{3.21}$$

The pheromone values between a virtual machine and a server are calculated using the pheromone values between the virtual machines. For a virtual machine v and physical machine H_i, it is given as

$$\tau[v, H_i] = \begin{cases} \sum_{\forall \hat{v} \in \mathbb{V}_{H_i}} \tau[v, \hat{v}] & \text{, if } \mathbb{V}_{H_i} \text{ is not empty} \\ 0 & \text{, otherwise;} \end{cases} \tag{3.22}$$

3.3.1.3 Construction of the Solution by an ant

In ant colony optimization, each artificial ant iteratively constructs a solution. One iteration of an ant in the proposed algorithm can be explained as follows. Initially, the mode of all the physical machines is kept to be inactive. The pheromone values and heuristic values are initialized to zero. The total time is divided into small epochs. At each time step, the ant calculates the VM's that are ready based on their start time. It adds them to a set called

RVMSET. It contains all the virtual machines that are ready to get scheduled at this point. Then it calculates the affinities for cluster, rack and the physical machine. Based on these values, it selects a cluster c probabilistically as per the equation given below.

$$P(c) = \frac{\texttt{Affinity}\,(RVMSET, c)}{\sum_{\forall \hat{c} \in C} \texttt{Affinity}\,(RVMSET, \hat{c})} \tag{3.23}$$

Once a cluster c is selected, then out of all the racks of cluser c, a rack r is selected probabilistically as per the equation below.

$$P(r) = \frac{\texttt{Affinity}\,(RVMSET, r)}{\sum_{\forall \hat{r} \in R(C)} \texttt{Affinity}\,(RVMSET, \hat{c})} \tag{3.24}$$

After the rack is selected, then a physical machine H_p is selected probabilistically from all the physical machines that are in the rack by the equation

$$P(H_p) = \frac{\texttt{Affinity}\,(RVMSET, H_p)}{\sum_{\forall H_{\hat{p}} \in P(r)} \texttt{Affinity}\,(RVMSET, H_{\hat{p}})} \tag{3.25}$$

Once a physical machine is selected, it's mode is changed to active mode. Then for this physical machine, feasible virtual machines out of the RVMSET are calculated based on the physical machine capacity and are added to the feasible set $\mathbb{A}$. From the virtual machines belonging to the feasible set, one VM v is selected probabilistically by the following equation.

$$P\left(v\right) = \frac{{\tau_{(v,H_p)}}^{\alpha_1} {\eta_{v,H_p}}^{\beta_1}}{\sum\limits_{\forall u \in \mathbb{A}} {\tau_{(u,H_p)}}^{\alpha_1} {\eta_{(u,H_p)}}^{\beta_1}} \tag{3.26}$$

where $\tau_{(v,H_p)} = \texttt{Size of v}{+}\texttt{Affinity}\,(H_i, V)$ is the heuristic and $\tau_{(v,H_p)}$ is the pheromone value between the virtual machine v and the physical machine H_p.

The selected virtual machine v is placed on to the physical machine H_p. The ant repeats this process over all the time epochs, and it constructs the solution. Then that solution is evaluated regarding the energy saved as per the first term of the objective function given in eq.5, and if it is the best solution among all the ants in the current iteration, then it is saved as the best local solution. Then the fitness value of the solution is compared with the best solution found till now, and if it has better fitness value, then the current solution is saved as the best solution found after all iterations.

3.3.1.4 Pheromone Update

After each iteration, the pheromone values are updated using the local best solution, and after every 5 iterations, the pheromone values are updated using the global best. This is done for exploitation and exploration of the solution space intelligently by the ants.

Algorithm 3.1 Communication Aware Energy Efficient VM Scheduling

```
1: **Input**: $\mathcal{G}(\mathbb{S} \cup \mathbb{H}, \mathbb{E})$,$\mathbb{V}$, $\mathbb{F}$, ants,$\alpha_1$,$\beta_1$,$\gamma_!$, $\mathbb{T}$;
2: **Output**: $\mathbb{U}$;
3: while Stopping Criterion is not met do
4:     $G\_best = \phi$, $L\_best = \phi$;
5:     for $a = 1$ to ants do
6:         $t = 0$,$\mathbb{R} = \phi$;   ▷ $\mathbb{R}$ is the set of virtual machines that are ready to be scheduled
7:         for $t = 0$ to $\mathbb{T}$ do
8:             if $t == 0$ then
9:                 $O_(H_i) = 0 \; \forall i \in \mathbb{H}$;
10:            end if
11:            for $j = 1$ to $n$ do
12:                for $k = 1$ to $\|\mathbb{V}_j\|$ do
13:                    if $stime(VM_{jk}) == t$ then
14:                        $\mathbb{R} = \mathbb{R} \cup VM_{jk}$;
15:                    end if
16:                end for
17:            end for
18:            while $\mathbb{R} \neq \phi$ do
19:                Pick a cluster $c$ from the available clusters with probability $P(c)$;
20:                Pick a rack $r \in \mathbb{R}_c$ with probability $P(r)$;
21:                Pick a physical machine $H_p \in \mathbb{H}_r$ with probability $P(H_p)$;
22:                $O_t(H_p) = 1$;
23:                $\mathbb{A}$= $FEASIBLE - SET(\mathbb{S}, \mathbb{R}, \mathcal{G}, p)$;   ▷ Finds feasible set of VMs
       according to dependency graphs and remaining capacity of PM P
24:                Pick a virtual machine $v$ from the feasible VM set $\mathbb{A}$ with probability
       $P(v)$;
25:                Add an element $(v, H_p, t)$ to $U$;
26:                $RC_t(H_p) = RC_t(H_p) - Req(v)$;
27:                $\mathbb{R} = \mathbb{R} - v$;
28:            end while
```

```
29:             $t = t + 1$;
30:             $\forall(v, H_p, t')$ in $\mathbb{U}$;
31:             **if** $(Ttime(v) + t') == t$ **then**
32:                 $RC_t(H_p) = RC_t(H_p) + Req(v)$;
33:             **end if**
34:         **end for**
35:         Calculate $F(U)$ based on the first term of Eq.3.4;
36:         **if** $F(U) < F(L\_best)$ **then**
37:             $L\_best = U$;
38:         **end if**
39:         **if** $F(U) < F(G\_best)$ **then**
40:             $G\_best = U$;
41:         **end if**
42:         pheromone update $(G_best, L_best, V)$;
43:     **end for**
44: **end while**
```

Algorithm 3.2 Pheromone Update

```
1: **Input**: $G_best, L_best, \mathbb{V}$
2: **for** $V_1 = 1$ to $|\mathbb{V}|$ **do**
3:     **for** $V_2 \in \mathbb{V} - V_1$ **do**
4:         **if** $(v_1, v_2)$get packed in G_best **then**
5:             $m_1 = 1$;
6:         **else**
7:             $m_1 = 0$;
8:         **end if**
9:         **if** $(v_1, v_2)$get packed in L_best **then**
10:            $m_2 = 1$;
11:        **else**
12:            $m_2 = 0$;
13:        **end if**
14:        **if** $noofiterations\%5 == 0$ **then**
15:            $\tau[i, j] = (\gamma * \tau[i, j]) + (1 - \gamma)(m_1 * F(G\_best))$;
16:        **else**
17:            $\tau[i, j] = (\gamma * \tau[i, j]) + (m_2 * F(L\_best))$
18:        **end if**
19:    **end for**
20: **end for**
```

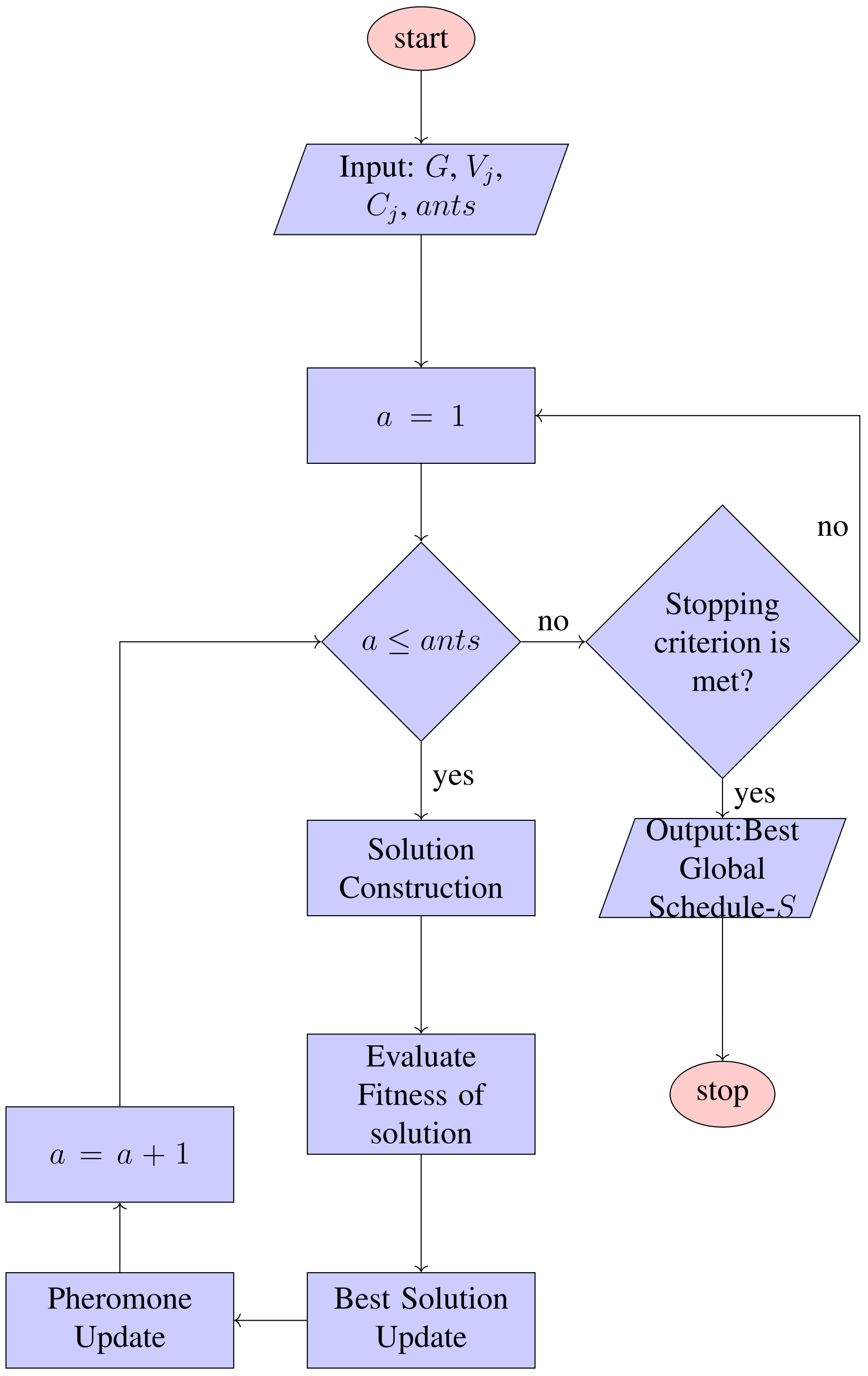

start
Input: G, V_j, C_j, $ants$
$a = 1$
$a \leq ants$
no
Stopping criterion is met?
no
yes
yes
Solution Construction
Output:Best Global Schedule-S
Evaluate Fitness of solution
stop
$a = a + 1$
Pheromone Update
Best Solution Update

Figure 3.5: Flow Chart for Algorithm: Communication Aware Energy Efficient VM Scheduling

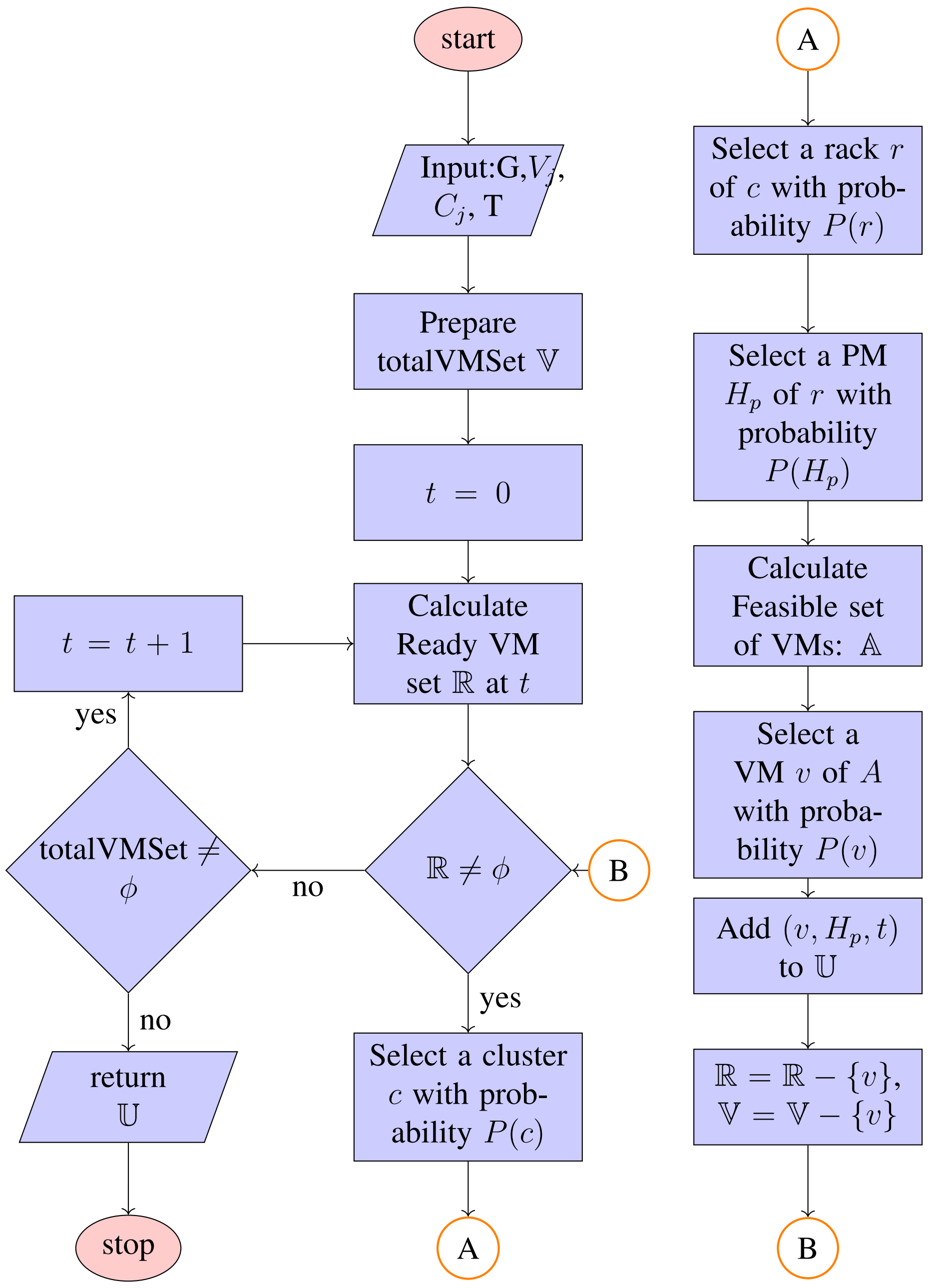

start
Input:G,V_j, C_j, T
Prepare totalVMSet $\mathbb{V}$
$t = 0$
Calculate Ready VM set $\mathbb{R}$ at t
$t = t + 1$
$\mathbb{R} \neq \phi$
no
totalVMSet $\neq \phi$
yes
no
return $\mathbb{U}$
stop
yes
Select a cluster c with probability $P(c)$
A
B
A
Select a rack r of c with probability $P(r)$
Select a PM H_p of r with probability $P(H_p)$
Calculate Feasible set of VMs: $\mathbb{A}$
Select a VM v of A with probability $P(v)$
Add (v, H_p, t) to $\mathbb{U}$
$\mathbb{R} = \mathbb{R} - \{v\}$, $\mathbb{V} = \mathbb{V} - \{v\}$
B

Figure 3.6: Flow Chart: Solution Construction by an ant

3.3.2 Energy Efficient VM Communication routing

The second phase of the solution deals with optimizing the energy consumption of network elements for a given VM placement. After placing the virtual machines on to the physical machines optimally with respect to energy consumption of the server elements in the first phase, the communication or data flows among the virtual machines need to be scheduled on to the switches, in a way that minimises the number of the active ports and switches. The inactive ports and switches are turned off to save energy.

The set of flows belonging to a tenant N_j is given by $\mathbb{F}_j$. These set of flows are derived from the communication graph $\mathcal{C}_j\left(\hat{\mathbb{V}}, \hat{\mathbb{E}}\right)$ of each tenant j. Let the set $\mathbb{F}=\bigcup_{j=1}^{n} \mathbb{F}_j$ contain the flows of all the tenants where a flow f_r of F is represented with a triplet $f_r\left(s_r, d_r, b_r\right)$. Here s_r is the source of the flow, d_r is the destination of the flow and b_r is the required bandwidth of the flow. A data center graph $\mathcal{G}\left(\mathbb{H} \cup \mathbb{S}, \mathbb{E}\right)$ is given, where the set $\mathbb{H}$ contains the servers which are the sources and destinations of flows and set S represents the intermediate switches in the paths between sources and destinations. The weight on any edge gives the bandwidth capacity of the link that the edge is representing. Now, the aim is to find out an optimal routing of the flows in such a way that the total no of active ports and the switches are minimised.

NP-Completeness of this subproblem can be proven by reducing Multi-Commodity flow problem to this problem [85]. Hence, we have proposed a meta-heuristic algorithm based on ant colony optimisation.

The total number of ants considered here is equal to number of flows. An ant constructs a path iteratively for the flow assigned to it. It selects a node probabilistically as the next node the flow goes through and adds that node to the partially constructed path.

The basic definitions of the variables used in the algorithm are as follows. Let P^{f_r} is the constructed path of a flow f_r by an ant r. $\mathbb{P}$ is set of the paths for all the flows computed by the ants in an iteration of the algorithm. $\mathbb{P}_{best}$ is the set of paths for all the flows in the best solution found till now. D is a weighted matrix indicating the residual capacity of each link in the data center graph. It is initialized to the capacities of the data center of the graph at the start of each iteration of the algorithm. i_0 indicate the number of iterations without improvement in the solution fitness value while running the algorithm and i_{n0} is the maximum number of iterations allowed without any improvement in the solution fitness value.

A detailed description of heuristic, pheromone trails and construction of the solution by an ant is given in the following sections.

3.3.2.1 Heuristic

The heuristic information $\eta(i^*, j^*)$ is associated with each link (i^*, j^*) of the data center network. It indicates the attractiveness of selecting a node j^* as the next node to visit after visiting a node i^*. Node j^* is the neighboring node belongs to $N(P^{f_r})$. If i^* is the last node of the partially constructed path P^{f_r} for the flow f_r, then we define two sets, $\mathbb{N}_1$ = $\{i|(i^*, j^*) \in \mathcal{G}\ \&\ j^* \notin p^{f_r}\}$ and $\mathbb{N}_2$ = $\{i|(i^*, j^*) \in \mathcal{G}\ \&\ j^* \in p^{f_r}\}$

$$N(P^{f_r}) = \begin{cases} \mathbb{N}_1, & if \mathbb{N}_1 \neq \phi \\ \mathbb{N}_2, & \text{Otherwise} \end{cases} \tag{3.27}$$

Here, the ant picks the node among all the nodes that are yet to be visited and if there are so such nodes, then we allow it go back to an already visited node. Then the heuristic for each edge (i^*, j^*) it is defined as

$$\eta_{i^*j^*} = (\eta^1_{i^*j^*})^{\alpha_2}(\eta^2_{i^*j^*})^{\beta_2} \tag{3.28}$$

where $\eta^1_{i^*j^*}$ is given as

$$\eta^2_{i^*j^*} = \frac{\text{no of active ports in } S_{j^*}}{Z^{port}_{j^*}} \tag{3.29}$$

and $\eta^2_{i^*j^*}$ is given as

$$\eta^2_{i^*j^*} = \frac{1}{D(i^*, j^*)} \tag{3.30}$$

The heuristic $\eta_{i^*j^*}$ depends on two factors. One, the utilization of the switch given by $\eta^1_{i^*j^*}$ and second, the residual capacity of the link (i^*, j^*) given by $\eta^2_{i^*j^*}$. The importance of each factor depends on the parameter values α_2 and β_2.

3.3.2.2 Pheromone Values

For each flow, an independent pheromone value indicated by τ_{f_r} is maintained. It is defined as τ_{f_r} = $\{\tau_{i^*j^*f_r} | (i^*, j^*) \ \in \mathcal{G}\}$. $\tau_{i^*j^*f_r}$ indicates the learned desirability of an ant to move from node i^* to node j^* in the construction of the path for the flow.

3.3.2.3 Pheromone Update

The pheromone updates are done by only the best solution out of all the solutions constructed by ants in one iteration in the following way.

$$\tau_{i^*j^*f_r} = \gamma * \tau_{i^*j^*f_r} + (1-\gamma) * \Psi(i^*, j^*, P_{best}^{f_r}) * \tau_{i^*j^*f_r} \tag{3.31}$$

$$\Psi(i^*, j^*, P_b^{f_r} est) = \begin{cases} 1, & \text{if edge } (i^*, j^*) \in P_{best} \\ 0, & Otherwise \end{cases} \tag{3.32}$$

3.3.2.4 Construction of a Solution

Each ant iteratively construct a path for the flow that is assigned to it. P^{f_r} is the partial path calculated by the ant for the flow f_r. Taking the source node of the flow as the first node, it iteratively selects a node probabilistically till the destination node of the flow is reached. The following equation gives the probability of ant going from node i^* to node j^*.

$$P_{i^*j^*}^{f_r} = \begin{cases} \dfrac{(\tau_{i^*j^*f_r})^{\alpha_2} (\eta_{i^*j^*})^{\beta_2}}{\sum\limits_{j^* \in \mathbb{N}(P^{f_r})} (\tau_{i^*j^*f_r})^{\alpha_2} (\eta_{i^*j^*})^{\beta_2}}, \text{if } j^* \in \mathbb{N}(P^{f_r}) \\ 0 \qquad\qquad \text{Otherwise} \end{cases} \tag{3.33}$$

An ant constructs a path for the flow assigned to it by picking one node after another probabilistically. Likewise, all the ants construct a path for the flows assigned to them. A set P represents all the paths constructed by the ants till now. Then, it is evaluated using the energy function of the network elements given by the second term of the equation 3.4. If the energy consumed by this solution is less than the energy consumed by the best solution found in previous iterations, then the current solution is updated as the best solution. Otherwise, i_0 indicating the number of iterations that didn't see any change is updated. If it crosses a predefined value i_{n0} then, the pheromone values are reset to some predetermined values. If not, the pheromone values are updated using the best solution. This process is continued until the stopping criterion is met, which usually is the no of iterations without change. Once the stopping criterion is met, the algorithm is terminated, and the output is returned.

Algorithm 3.3 Energy efficient VM Communication Routing

```
Output: $\mathbb{P}_{best}$ and $F(P_{best})$
1:  The problem instance is taken as the input.
2:  Initialization
3:  Set parameters
4:  while Termination criteria is not met do
5:      $\mathbb{P} \leftarrow \emptyset$
6:      Initialize_Network_Capacity($D$,$\mathcal{G}$)
7:      for $r = 1$ to $|\mathbb{F}|$ do
8:          $P^{f_r} = \{s_{f_r}\}$
9:          $i^* \leftarrow s_{f_r}$
10:         while $i^* \neq d_{f_r}$ do
11:             Choose the next node $j^*$ from $\mathbb{N}(P^{f_r})$ to $i^*$ according to Eq.3.33
12:             $P^{f_r} \leftarrow P^{f_r} \cup j^*$
13:             $i^* \leftarrow j^*$
14:         end while
15:         Update_Capacity($D$,$P^{f_r}$)
16:         $\mathbb{P} \leftarrow \mathbb{P} \cup P^{f_r}$
17:     end for
18:     Calculate $F(P)$ based on second term of the objective function in Eq.3.4
19:     if $F(\mathbb{P}_{best}) > F(P)$ then
20:         $P_{best} \leftarrow P$, $i_0 = 0$
21:     else
22:         $i_0 = i_0 + 1$
23:     end if
24:     if $i_0 \geq i_{n0}$ then
25:         Reset_Pheromone_Trails($\tau$,$a$)
26:     else
27:         Pheromone_Trails_Update($\tau$,$\mathbb{P}_{best}$)
28:     end if
29: end while
```

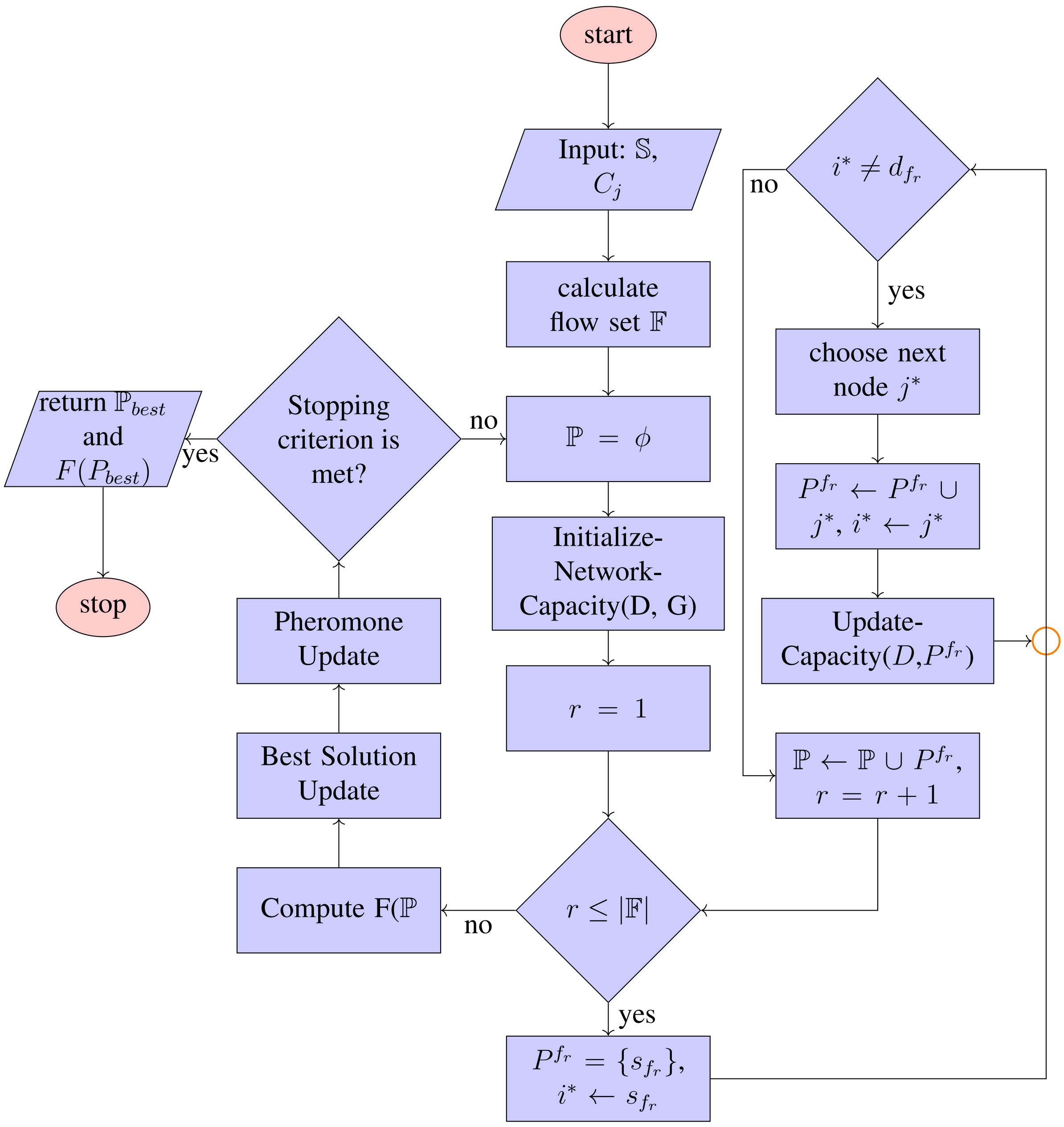

start
Input: $\mathbb{S}$, C_j
calculate flow set $\mathbb{F}$
$\mathbb{P} = \phi$
Initialize-Network-Capacity(D, G)
$r = 1$
$r \leq |\mathbb{F}|$
yes
$P^{f_r} = \{s_{f_r}\}$, $i^* \leftarrow s_{f_r}$
$i^* \neq d_{f_r}$
yes
choose next node j^*
$P^{f_r} \leftarrow P^{f_r} \cup j^*$, $i^* \leftarrow j^*$
Update-Capacity(D,P^{f_r})
no
$\mathbb{P} \leftarrow \mathbb{P} \cup P^{f_r}$, $r = r + 1$
no
Compute F($\mathbb{P}$
Best Solution Update
Pheromone Update
Stopping criterion is met?
no
yes
return $\mathbb{P}_{best}$ and $F(P_{best})$
stop

Figure 3.7: Flow Chart for Algorithm: Energy Efficient VM communication Routing

3.4 Performance Evaluation

This section presents a detailed discussion of the results of the experiments carried out to test the proposed solution. The solution is implemented on a system with intel core i7-4790U with four cores and 8GB RAM. Java is used to implement the algorithms proposed in the solution. The efficiency of the solution is tested for three data centers of different sizes. The results of the above experiments are compared with two standard heuristic algorithms for VM scheduling, first fit and round robin.

A round robin is a naive approach that picks virtual machines one by one and allocates to the physical machines in a round robin manner. First Fit heuristic approach sorts the VMs in descending order, and in that order, each VM is allocated to a PM that can accommodate it first. Once the VM schedule is found based on these two approaches, for routing the data of the VM's that are located in any two active physical machines, the shortest path is considered as the routing path. So the remaining switches which don't route any communication data of the virtual machines are turned off to save the energy. The first fit approach performs well when compared with other deterministic algorithms. Hence it is taken as a representative algorithm for deterministic algorithms.

The virtual machines considered here are of four types, where the requirements of each type of virtual machines are as shown in the Table 3.2. The physical machines are assumed to be heterogeneous. They are classified into four types of physical machines, and the capacities of each type of physical machine are given in the Table 3.3. The power consumption values of a server for each type of physical machine are given in the Table 3.4. The peak power consumption of a commodity switch is taken as 40 watts, which is a standard value for a typical commodity switch. Here, three data centers of different sizes are taken to test the proposed solution. Each data center contains a different number of clusters, comprising of a different number of physical machines arranged in racks.

For all the three data centers considered here, the proposed solution is tested for three different standard data center network architectures namely 3-tier, B-Cube and Hyper-tree. These results are compared with the first fit and the round-robin values for the same.

A 3-Tier tree data center network consists of three layers namely, edge, aggregation and

Table 3.2: EVMSR: VM Types

VM type	MIPS	Memory	Disk	Net	Degree
VM1	100	500	50000	10	5
VM2	200	1000	100000	20	10
VM3	300	2000	200000	30	20
VM4	400	4000	400000	40	40

Table 3.3: EVMSR: Physical Server Capacities

PM type	MIPS	Memory	Disk	Net	Degree
PM1	1000	4096	102400	20	8
PM2	2000	8768	204800	40	16
PM3	4000	32768	512000	100	32
PM4	8000	131072	1024000	1000	64

core [14]. The network can be divided into k pods. Each pod in a k-ary fat-tree consists of $\frac{k^2}{2}$ severs and two layers of $\frac{k}{2}$ k port switches. Each edge switch is connected to $\frac{k}{2}$ servers and $\frac{k}{2}$ aggregate switches. Each aggregate switch connects to $\frac{k}{2}$ edge and $\frac{k}{2}$ core switches. k ary fat-tree contains $\frac{k^2}{2}$ core switches, where each core switch connects to k pods.

B-Cube, is a recursively constructed data center network architecture [15] . A $Bcube_k$ is constructed from $Bcube_{k-1}$ and n^k, n port switches. Each server in a B-Cube has $K+1$ ports, which are numbered from level 0 to level-k. A $Bcube_k$ has n^{k+1} servers and $k+1$ level of switches, with each level having n^k, n-port switches.

Hyper-tree contain n nodes and one n-port switch in the first layer of it [16]. From the second layer, a k layer hyper-tree consists of $n^2(k-1)$-layer Hyper-treess.

We evaluate the proposed solution by varying the number of applications that are needed to deploy in the cloud. The number of applications is ranged from 10 to 400. Each application contains a different number of tasks. For each application, the number of tasks it contains is generated randomly and its dependency graph is generated randomly accordingly. The execution time of each task is picked randomly between two predetermined values, and the type of the virtual machine that it requires is randomly picked from the different types of virtual machines as mentioned above. The assumption here is one to one correspondence between a task and a virtual machine, i.e., each task is managed by only one virtual machine and vice-versa. The maximum number of virtual machines that require to service an application at any time depends on the maximum concurrency

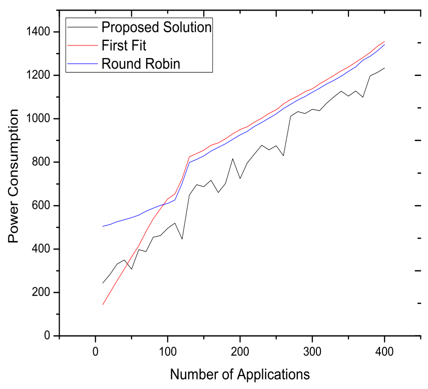
Table 3.4: EVMSR: Physical Server Power Consumption(in Watts)

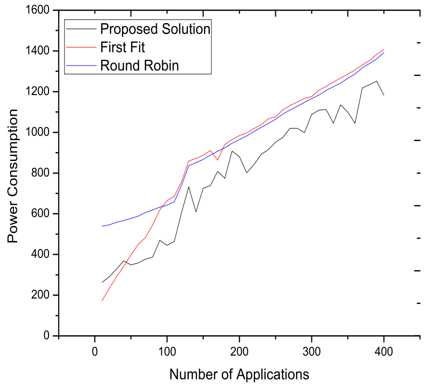
PM type	Max Power Consumption
PM1	58
PM2	79
PM3	101
PM4	152

in the dependency graph of the application. The communication graph of an application indicates the potential communication pattern among the virtual machines, which we assume could be known in advance. Here the key assumption is, the communication happens only among the virtual machines of an application that are concurrently running. So based on the earlier generated dependency graphs, the communication graphs are generated randomly and the edge weight, indicating the size of the data that is being exchanged during the communication is assigned some random value between 1 Mb to 100 Mb for each edge.

The plots in Figures 3.8, 3.9 and 3.10, depict the effectiveness of the proposed solution in saving energy when compared with the other two approaches, first fit and round robin. It can be observed that the proposed solution saves the energy significantly when compared with the first fit and the round robin for all the three standard data center networks considered here. For smaller workloads, the first fit heuristic performs better and as the load increases performance of the proposed solution improves and consume much less energy than the First fit. The round-robin algorithm works poorly for all the sizes of the workloads when compared with both the proposed solution and the first fit. The algorithms for both the phases have been tested by varying the parameters. The optimal values of the parameters for the algorithm in the first phase are α_1, β_1, γ_1 are 0.5, 0.4 and 0.6 respectively, whereas for the second phase the parameter values are 0.3, 0.4 and 0.4 respectively. The value of n_0 for the algorithm in the second phase is set at 4. The algorithm for the first phase converges at around 25 iterations and the algorithm for the second phase converges at around 20 iterations, even when the workload on the data centers is very high.

The proposed solution is achieving significant power savings over other approaches because the first phase of the proposed solution uses a parameter called affinity, which gives the expected amount of data center traffic that would be removed from the data center network if two virtual machines are placed together, in placement decisions to reduce





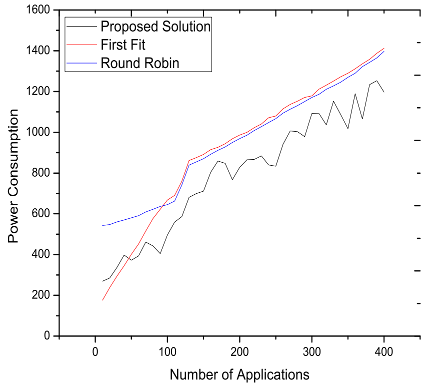
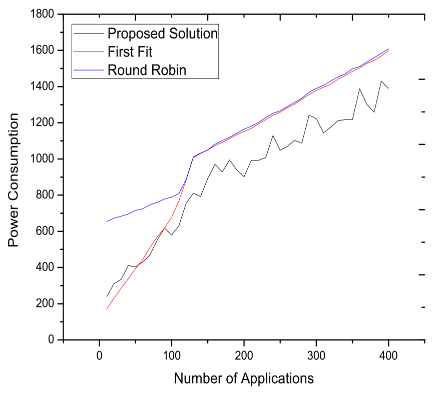
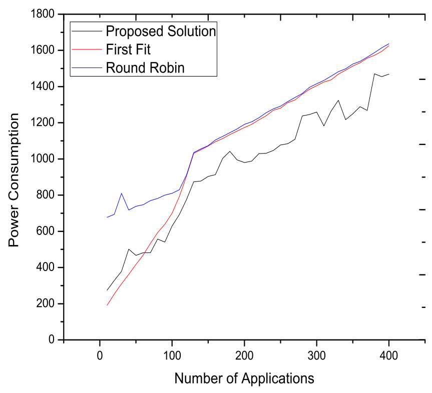


Figure 3.8: EVMSR: Energy Consumption in DC1 (a) 3-Tier (b) B-Cube (c) Hyper-Tree





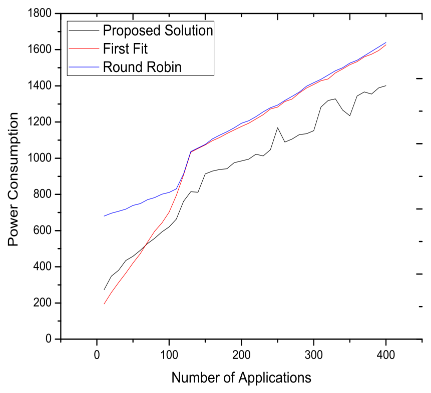
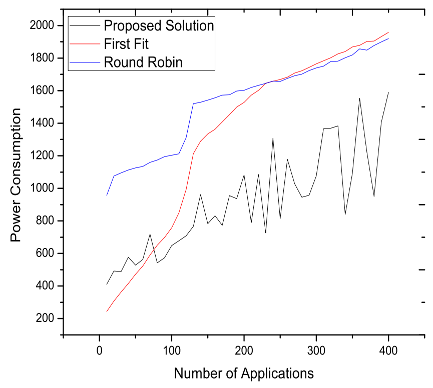
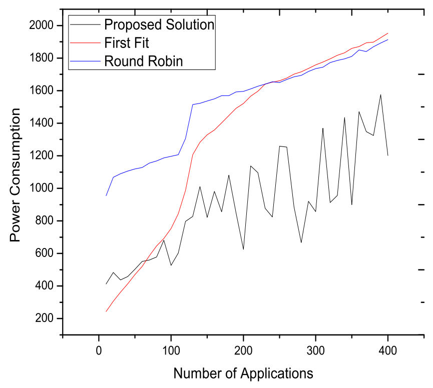


Figure 3.9: EVMSR: Energy Consumption in DC2 (a) 3-Tier (b) B-Cube (c) Hyper-Tree





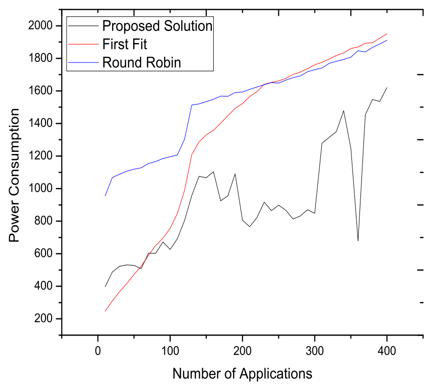


Figure 3.10: EVMSR: Energy Consumption in DC3 (a) 3-Tier (b) B-Cube (c) Hyper-Tree

the energy consumption of networking elements. In first-fit and round robin solutions no such parameter is used ,so two communicating virtual machines can be placed on different physical machines that are far apart in the data center, resulting in high network energy. Moreover, the proposed solution consolidates the communication flows of the virtual machines onto a few number of networking elements, hence reducing the energy consumed by the networking elements furthermore. In first-fit and round robin solutions, for each flow shortest path is selected, so the communications flows of the virtual machines are consolidated on to higher number of networking elements when compared with the proposed method.

3.5 Summary

In this chapter, communication aware energy efficient VM scheduling and routing problem in cloud data center is addressed. To save energy, the proposed two-phase solution first consolidates the virtual machines on to few servers while placing communicating virtual machines in close proximity. Then it consolidates the communication flows of virtual machines on to few switches and links. The solution is tested for three different standard data center network architectures of three different sizes. On average, the proposed solution improves the energy savings by 15% and 20% when compared with first-fit and round-robin solutions respectively. The proposed solution is more effective for data centers of large sizes and it is scalable as the percentage of energy savings are similar even when the load on the data center is high

Chapter 4

On line VM Placement using Spectral Graph Partitioning

In this chapter the online version of the joint host-network energy efficient VM placement problem in cloud computing is addressed. The users of deadline-insensitive applications requests virtual machines on the go. These applications are usually multi-tier applications and they consist of many independent sub tasks. Each of these sub tasks need to be serviced by one or more virtual machines. These virtual machines exchange data and communicate during the execution of the application. If the communicating virtual machines are placed far in the data center network, then the energy consumed by the networking elements would increase. In this version of the problem, tenant requests a set of virtual machines on the go by describing the communication patterns of the virtual machines with a graph. The cloud provider has to optimally place these newly provisioned virtual machines on the physical machines. Here the idea is to use graph partitioning on the communication graph of the virtual machines requested by the tenant to identify groups of virtual machines with least inter and high intra-communications and place the groups optimally on to the physical machines.

Contributions of this chapter are as follows

- We model the problem of online version of Joint Host-Network Energy Minimized VM placement problem as an mixed integer programming problem.

- As solutions, we propose two greedy algorithms called best-fit-spectral and least-increase-in-energy-spectral as a solution to the problem. The proposed algorithms apply spectral graph partitioning to identify the groups of virtual machines with high intra-communication and less inter-communication.
- The proposed solutions are tested against first-fit-decreasing heuristic for three standard data center networks namely 3-tier, B-cube, and hyper-tree of three varying sizes.

The remainder of the chapter is organized as follows. In section 4.1, the system model and the problem formulation are presented. The section 4.3.1 contains preliminaries related to spectral graph partitioning and the detailed description of the two algorithms which are presented as solutions to the problem. In the section 4.4, simulation results are presented and in section 4.5, chapter is summarized.

4.1 System Model

In this section, the system model is presented in detail. The mathematical formulation of the problem using mixed integer programming is also presented.

4.1.1 Data Center Model

The data center model considered here is same as the one in section 3.1.1

4.1.2 Request Model

A virtual machine request is represented with $R(\mathbb{V}, C)$, where $\mathbb{V}$ is the set of virtual machines requested and $C(\mathbb{V}, \hat{\mathbb{E}})$ is the communication graph that depicts the communication patterns of the virtual machines. The requirements of a virtual machine $v \in \mathbb{V}$ is given by $R_v^{\texttt{comp}}$, $R_v^{\texttt{mem}}$ and $R_v^{\texttt{storage}}$ in terms of computation, memory and storage respectively. The weight $\hat{C}\langle\hat{a}, \hat{b}\rangle$ on the edge $\langle\hat{a}, \hat{b}\rangle$ gives the required bandwidth for VMs $\hat{a}$ and $\hat{b}$.

4.1.3 Energy Models

The energy models considered here is same as the one in section 3.1.3

4.2 Problem Formulation

Problem statement: A tenant requests a set of virtual machines V to a data center whose network is depicted as a graph $\mathcal{G}(\mathbb{H} \cup \mathbb{S}, \mathbb{E})$. The communication patterns of the vms of the tenant is represented with $\mathcal{C}\left(\hat{\mathbb{V}}, \hat{\mathbb{E}}\right)$. Now the objective is to find out an efficient placement of the virtual machines of the tenant on to the physical machines such that the total energy consumed by the physical machines and switches is minimized.

Let $\mathbf{X}_i^k(t)$ is a binary variable specifying whether k^{th} vm of the request, is placed on server H_i at time t or not. Let $\mathbf{O}_i(t)$ specifies whether the server H_i is on or off at time t. Similarly the boolean variables $\mathbf{O}_s(t)$ and $\mathbf{O}_{\langle a,b\rangle}(t)$ denote whether the switch S_s and link $\langle a,b\rangle$ is switched on or off at time t respectively. Let $\texttt{route}_t\left(f^l, \langle a,b\rangle\right)$ is a binary variable depicting whether the communication f^l goes through the link $\langle a,b\rangle$ at time t or not. Then the problem can be modelled as

$$\textbf{min} \quad \sum_{\forall t\in\mathbb{T}} \left(\sum_{\forall i\in\mathbb{H}} \mathbf{P}_i(t) \times \mathbf{O}_i(t) + \sum_{\forall s\in\mathbb{S}} \mathbf{P}_s(t) \times \mathbf{O}_s(t) \right) \tag{4.1}$$

4.2.1 Constraints

4.2.1.1 Assignment Constraints

$$\sum_{\forall i\in\mathbb{H}} \mathbf{X}_i^k(t) = 1, \qquad \forall t\in\mathbb{T}, \forall k\in\mathbb{V} \tag{4.2}$$

$$\sum_{\forall i\in\mathbb{H}} \mathbf{X}_i^k(t) \leq \mathbf{O}_i(t), \forall t\in\mathbb{T}, \forall k\in\mathbb{V} \tag{4.3}$$

4.2.1.2 Capacity Constraints

$$\sum_{\forall i\in\mathbb{H}} \mathbf{X}_i^k\,(t) * R_k^{comp} \leq Z_i^{comp}, \forall t\in\mathbb{T}, \forall k\in\mathbb{V} \tag{4.4}$$

$$\sum_{\forall i \in \mathbb{H}} \mathbf{X}_i^k\left(t\right) * R_k^{mem} \leq Z_i^{mem}, \forall t \in \mathbb{T}, \forall k \in \mathbb{V} \tag{4.5}$$

$$\sum_{\forall i \in \mathbb{H}} \mathbf{X}_i^k\left(t\right) * R_k^{storage} \leq Z_i^{storage}, \forall t \in \mathbb{T}, \forall k \in \mathbb{V} \tag{4.6}$$

$$\sum_{\forall l \in \mathbb{F}} bd\left(f^l\right) * \mathbf{route}_t\left(f^l, \langle a, b\rangle\right) \leq W\langle a, b\rangle, \forall t \in \mathbb{T} \tag{4.7}$$

$$\mathbf{route}_t\left(f_j^l, \langle a, b\rangle\right) \leq \mathbf{O}_t\langle a, b\rangle, \forall l \in \mathbb{F} \tag{4.8}$$

4.2.1.3 Link Constraints

$$\begin{aligned} \forall\left(a, b\right) &\in \mathbb{E}, \\ \mathbf{O}_{\langle a,b\rangle}(t) &\leq \mathbf{O}_a(t) \\ \mathbf{O}_{\langle a,b\rangle}(t) &\leq \mathbf{O}_b(t) \\ \mathbf{O}_{\langle a,b\rangle}(t) &= \mathbf{O}_{\langle a,b\rangle}(t) \end{aligned} \tag{4.9}$$

$$n_{port}^s(t) = \sum_{\forall(\langle s,a\rangle \vee \langle a,s\rangle) \in \mathbb{E}} \mathbf{O}_t\langle a, s\rangle, \forall s \in \mathbb{S} \tag{4.10}$$

4.2.2 Flow constraints

$$\sum_{\forall l \in \mathbb{F}} \mathbf{route}_t\left(f^l, \langle i, s\rangle\right) \leq \mathbf{X}_i^{src\left(f^l\right)}\left(t\right), \forall t \in \mathbb{T}, \forall i \in \mathbb{H}, \forall s \in \mathbb{S} \tag{4.11}$$

$$\begin{aligned} \sum_{\forall l \in \mathbb{F}} (\mathbf{X}_s^{src\left(f^l\right)}\left(t\right) - \mathbf{X}_s^{dest\left(f^l\right)}\left(t\right)) &= \\ \sum_{\forall i \in \mathbb{H}} \mathbf{route}_t\left(f^l, \langle s, i\rangle\right) - \sum_{\forall i \in \mathbb{H}} \mathbf{route}_t\left(f^l, \langle i, s\rangle\right), \\ \forall t \in \mathbb{T}, \forall s \in \mathbb{S} \end{aligned} \tag{4.12}$$

$$\sum_{\forall l \in \mathbb{F}} \mathbf{route}_t \left(f^l, \langle a, s\rangle\right) = \mathbf{route}_t \left(f^l, \langle s, a\rangle\right), \forall t \in \mathbb{T}, \forall s \in \mathbb{S}, \quad \forall a \in (\mathbb{S} \cup \mathbb{K}) \tag{4.13}$$

The energy efficient VM placement problem in cloud data center is NP-complete as it can be reduced from a known NP-Complete problem called bin packing problem with different bin sizes.

4.3 Joint Host and Network Energy Efficient VM Placement Using Spectral Graph Partitioning

In this section, we present the vm placement algorithm that reduces the energy consumed by both server and network elements. It uses spectral graph partitioning to identify groups of virtual machines with high intra communication.

4.3.1 Spectral Graph Partitioning

Spectral graph theory is the study of the graph combinatorial properties by the eigen values and the eigen vectors of matrices representing the graph [86][87][88][89]. The graph is represented as a 3-tuple ($\mathbb{V}$, $\mathbb{E}$, w) where $\mathbb{V}$, $\mathbb{E}$, and w are vertex set, edge set and weight function defined on edge set respectively. For a communication graph of a tenant k, the vertices are the virtual machines servicing the application, edge represents a communication between two virtual machines, and the weight on edge gives the amount of data gets exchanged as part of the communication. If every node is reachable from any other node then the graph is said to be connected. The adjacency martix of a graph is defined as:

$$A_{ij} = \begin{cases} w(i,j), & \text{if}(i,j) \in E \\ 0, & \text{Otherwise} \end{cases} \tag{4.14}$$

The degree of a vertex i is defined as $d_i = \sum_{j=1}^{m} [A]_{ij}$. Then the graph Laplacian is defined as:

$$L = diag(d_1, d_2, \ldots, d_m) - A \tag{4.15}$$

When a graph is partitioned into two sets, the weight of the cut is given by

$$cut(V_1, V_2) = \sum_{u \in V_1, v \in V_2} w(u, v) \tag{4.16}$$

However, partitioning the graph by above cut can results in partitions with isolated vertices which is not desirable for our problem. A better way to partition the graph is by minimizing the following normalized cut [90] defined as:

$$ncut(V_1, V_2) = \frac{cut(V_1, V_2)}{assoc(V_1, V)} + \frac{cut(V_1, V_2)}{assoc(V_2, V)} \tag{4.17}$$

where $assoc(V_1, V)$ is the sum of weights between the nodes in V_1 to to all the nodes in V.

Minimization of this normalized cut is equivalent to find y that minimizes the following equation:

$$\frac{y^T(D - W)y}{y^T D y} \tag{4.18}$$

where $y = \{0, 1\}^N$ is binary indicator vector indicating the partition of the each node. Observe that if the binary condition on y is relaxed and allowed it to take real values, solving equation.4.18 is equivalent to:

$$(D - W)y = \lambda D y \tag{4.19}$$

Here $(D - W)$ is graph Laplacian and by applying $z = D^{1/2}y$ it can be re-written as

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}z = \lambda z \tag{4.20}$$

Hence, the problem of minimizing the normalized cut is equivalent to solving the general eigenvalue system for normalized graph Laplacian.

$$\mathcal{L} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} \tag{4.21}$$

The normalized graph Laplacian is a symmetric matrix and semi-definite matrix. We are interested in analysing the eigenvalues and eigenvectors of the Laplacian matrix to find the partitions that gives the optimal normalized cut. If the communication graph is connected, then $\mathcal{L}$ has 0 as it's eigenvalue and all the other eigen values are above zero. The eigenvalues are arranged in the following way, $\lambda_1 = 0 < \lambda_2 < \cdots < \lambda_m$. The eigenvector belonging to the second largest eigenvalue λ_2 is called as fidel vector. This vector is used in identifying the optimal partition in the following way. For a connected graph, partitioning of vertex set $\mathbb{V}$ into two sets $\mathbb{V}_1$ and $\mathbb{V}_2$ is given by

$$i \in \mathcal{V}_1 \text{ if } [\mathbb{V}_2]_i \geq 0, \quad i \in \mathcal{V}_2 \quad \text{if } [\mathbb{V}_2]_i < 0 \tag{4.22}$$

In this section, two heuristic algorithms are proposed as solution to the problem of Joint Host and network energy efficient VM placement in cloud data center. The spectral graph partitioning presented in the above section is used in the construction of the solution.

4.3.2 Best Fit with Spectral Graph Partitioning

The key idea here is to place all the group of communicating virtual machines on a physical machine in the best fit way. If there is no physical machine that has the capacity to all the virtual machines, then partition the virtual machine set into two sub-groups with minimal intra communication between the groups and repeat the process on the sub-groups till all the virtual machines are allocated to a machine. The proposed algorithm 4.1, takes communication graph of the virtual machines along with their requirements and the capacities of the hosts in H are taken as input. Initially, the physical machines in H are sorted in non-ascending order of their capacities as H_1, H_2,..H_n. i,e; cap$(H_1) \geq$ cap$(H_2) \geq \cdots \geq$ cap(H_m). Then it checks whether the capacity of the first physical machine in the sorted order can able to satisfy the requirements of all the virtual machines of the graph. If yes, place all the virtual machines on to that physical machine and return S the set containing the placement solution. Otherwise, partition the communication graph G into two graphs with minimal normalized cut between the two graphs using spectral graph partitioning. Then apply the same algorithm recursively on these two sub-graphs till all the virtual machines are placed on the hosts.

Algorithm 4.1 Best-Fit-with-Spectral-Graph-Partitioning(G, H)

```
Output: $\mathbb{S}$                                              ▷ VM Placement
 1: Sort the physical machines in $H$ in non-ascending order of their capacities. Let us
    assume they are indicated as $H_1$, $H_2$,..$H_n$;
 2: flag=0, Index-LastMachine-with-Capacity=0;
 3: for $H_k \in H$ do
 4:     if $\sum_{V_i \in G} req(V_i) \leq cap(H_k)$ then
 5:         flag=1;
 6:         Index-LastMachine-with-Capacity= k;
 7:     end if
 8: end for
 9: Assign all virtual machines in $G$ to physical machine $H_{\text{Index-LastMachine-with-Capacity}}$
10: for $V_i \in G$ do
11:     Add $(V_i, H_{\text{Index-LastMachine-with-Capacity}})$ to $S$;
12: end for
13: if flag $= 0$ then
14:     Partition the VM communication graph $G$ into two graphs $G_1$ and $G_2$ using spectral
    graph partitioning;
15:     $S_1$= Best-Fit-with Spectral-Graph-Partitioning($G_1$, H);
16:     Update capacities of the hosts in $H$ based on the placement in $S_1$
17:     $S_2$= Best-Fit-with Spectral-Graph-Partitioning($G_2$, H);
18:     $S = S_1 \cup S_2$;
19: end if
20: return $S$;
```

4.3.3 Least Energy Increase with Spectral Graph Partitioning

The idea here is to place the all the virtual machines of the graph onto the physical machine that gives least increase in energy because of this placement. If there is no physical machine that could host all the virtual machines together, partition the virtual machines set into two groups with least intra-communication between the groups and apply the same process again on these sub-groups. In the given algorithm 4.2, for all the physical machines that have the capacity to hold all the virtual machines, the energy required to host all the virtual machines is estimated. The power estimation include the expected energy expenditure of the physical machine for hosting these set of virtual machines and also expected energy consumed by the network elements due to the communication of the current set of virtual machines with the virtual machines already placed. Based on those estimations, the set of virtual machines is placed on the machine which causes least increase in energy. If there is no physical machine with the capacity to host all the virtual machines, then the communication graph of the virtual machines is partitioned into two graphs G_1 and G_2 using spectral graph partitioning that gives the least normalized cut and algorithm is called recursively on these sub-graphs.

Algorithm 4.2 Least-Energy-Increase-with-Spectral-Graph-Partitioning(G, H)

```
Output: $\mathbb{S}$                                   ▷ VM Placement
 1: minPower=Max;
 2: minPowerIndex=-1, flag=0;
 3: for $H_k \in H$ do
 4:     if $\sum_{V_i \in G} req(V_i) \leq cap(H_1)$ then
 5:         flag=1
 6:         power=estimatePower( $G, H_k$)
 7:         if $power \leq minPower$ then
 8:             minPower=power
 9:             minPowerIndex=$k$
10:         end if
11:     end if
12: end for
13: Assign all virtual machines in $G$ to physical machine $H_{minPowerIndex}$
14: for $V_i \in G$ do
15:     Add $(V_i, H_{minPowerIndex})$ to $S$;
16: end for
17: if flag $= 0$ then
18:     Partition the VM communication graph $G$ into two graphs $G_1$ and $G_2$ using spectral
    graph partitioning;
19:     $S_1$= Least-Energy-Increase-with-Spectral-Graph-Partitioning($G_1$, H);
20:     Update capacities of the hosts in $H$ based on the placement in $S_1$
21:     $S_2$= Least-Energy-Increase-with-Spectral-Graph-Partitioning($G_2$, H);
22:     $S = S_1 \cup S_2$;
23: end if
24: return $S$;
```

4.4 Performance Evaluation

In this section detailed analysis of the experimental results is presented. For evaluating the proposed algorithms, we consider an IaaS cloud computing environment. As the system aimed at is IaaS cloud computing environment, it is desirable to evaluate it on a large scale data center. However, it is very hard to get access to and repeat the experiments on such large data centers. Hence simulation is selected as a method to evaluate the proposed solution.

The simulations are conducted on a system with intel core i7-4790U with four cores and 6GB RAM. The proposed algorithms are implemented in java using a custom built discrete event simulator. To implement spectral graph partitioning of the graph, clustering package of Statistical Machine Intelligence and Learning Engine(SMILE) [91] is used.

The proposed solutions are compared with first-fit-decreasing heuristic. It initially sorts the virtual machines in non decreasing order of their requirement and places one by one in the first-fit manner onto the physical machines till all the VMs are placed. This heuristic is known to perform better than any other deterministic solutions present in the literature.

To validate the effectiveness of the proposed solutions, we consider three data centers of varying sizes and configurations. They are given in the Table 4.1. The first data center contains 90 physical machines in 4 racks of a single cluster. The second data center contain 240 physical machines organized into 8 racks and one cluster. These 240 physical machines are connected by 14 switches. The third data center contains 2 clusters, where the first cluster contain 480 machines arranged into 16 racks and the second cluster contains 160 physical machines arranged into 4 racks.

Table 4.1: Online Spectral: Data Centers of different sizes

Type	Clusters	racks	PMs	Switches
DC1	1	4	90	7
DC2	1	8	240	14
DC3	2	20(16, 4)	640	35

For each data center, we consider three different data center networks namely 3-tier, B-cube and Hyper-tree to evaluate how the proposed algorithm is performing for different data center networks.

3-tier is a hierarchical data center network architecture containing three layers called edge, aggregate and core [14]. A n-ary fat tree consists of n pods where: each pod consists of $\frac{n^2}{2}$ servers and two layers of $\frac{n}{2}$ n-port switches. Each edge switch is connected to $\frac{n}{2}$ servers and $\frac{n}{2}$ aggregate switches. Each aggregate switch is connected to $\frac{n}{2}$ edge switches and $\frac{n}{2}$ core switches. Each of the $\frac{n^2}{2}$ core switches connects to n pods.

B-cube is a server centric data center network architecture [15]. It has a recursive structure. Basic module of this architecture is $Bcube_0$ which is a connection of n servers to switch containing n ports. $Bcube_1$ is constructed from n $Bcube_0$ and n n-port switches. Similarly a $Bcube_k$ is constructed from n $Bcube_{k-1}$ and n^k n-port switches.

Hyper-tree [16] contains several layers. The first layer contains n servers and one n port switch. From the second layer, a k layer hyper-tree consists of $n^2(k-1)$ layer hyper-cubes.

We have considered four types of virtual machines and their requirements are given by the Table 4.2.

Table 4.2: Online Spectral: VM Types

Type	MIPS	RAM	Disk	Net
VM1	100	500	50000	10
VM2	200	1000	100000	20
VM3	300	2000	200000	30
VM4	400	4000	400000	40

We have have considered four different types of physical machines and their configurations are as given in the Table 4.3.

Table 4.3: Online Spectral: PM Capacities

Type	MIPS	RAM	Disk	Net	Degree
PM1	1000	4096	102400	20	8
PM2	2000	8768	204800	40	16
PM3	4000	32768	512000	100	32
PM4	8000	131072	1024000	1000	64

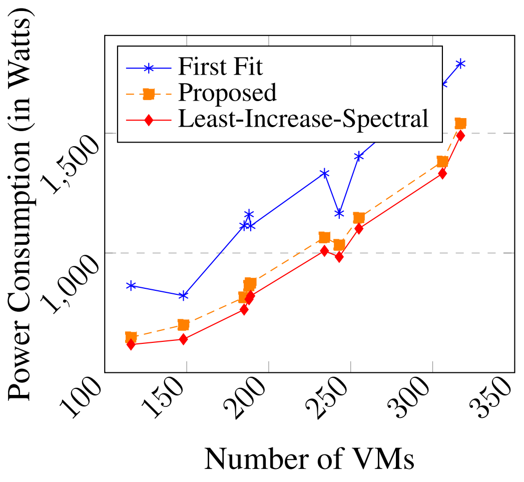
The power consumption profiles of the physical machines are given by the Table 4.4. The idle power is 0.5% of the max power.

Table 4.4: Online Spectral: PM Power Consumption(in Watts)

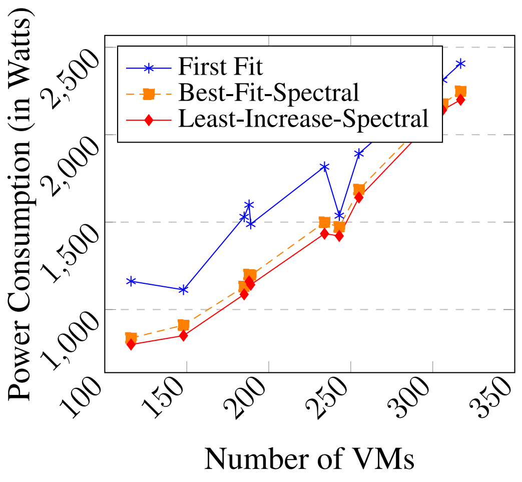
PM type	Max Power
PM1	58
PM2	79
PM3	101
PM4	152

The proposed solution is evaluated for 10 applications, each containing a varying number of virtual machines. The communication graph of each application is generated randomly with 0.3 probability that an edge exists between any two virtual machines. The virtual machine execution times are generated by picking a random value between 60 and 120 with uniform probability. The data transfer between any two communicating virtual machines is picked a value between 1MB to 50 MB with uniform probability. The requests for a set of virtual machines comes on the fly and the the graphs are plotted as per the load on the data center in terms of the number of virtual machines.

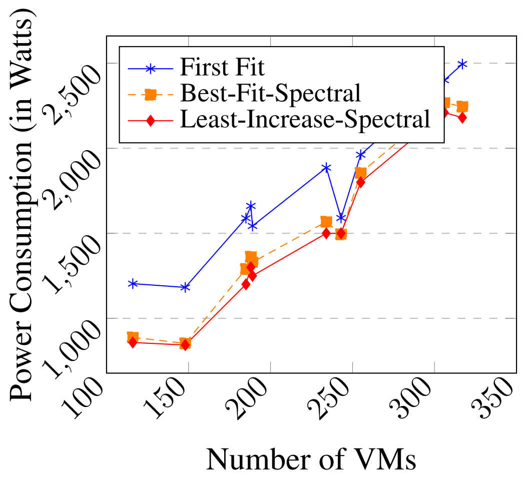
The graphs in Figure 4.1, Figure 4.2 and Figure 4.3 present the effectiveness of the proposed solution against the first-fit heuristic. The proposed solutions are scalable as they performing better as the number of virtual machines on the data center is increasing. The percentage of energy savings by the proposed solutions on an average is maintained the same for all the three data center configurations of varying sizes. Concerning data center network architectures, both the proposed solutions are giving better energy savings for 3-tier data center network architecture. The percentage of energy savings is the least for hyper-cube data center network architecture. On an average, best-fit spectral is achieving 18 % of energy savings and least-increase-spectral 21 % of energy savings when compared with the first-fit decreasing heuristic. Both the proposed greedy solutions use spectral graph partitioning to identify the clusters of virtual machines with high intra VM communication and low inter VM communication and each such identified cluster of VM's is placed onto a single machine, thereby reducing load on the networking elements and in turn reducing the network energy. Whereas, the first fit approach consolidates virtual machines on to the physical machines without considering the communication patterns of the virtual machines, resulting in higher network energy.



(a)

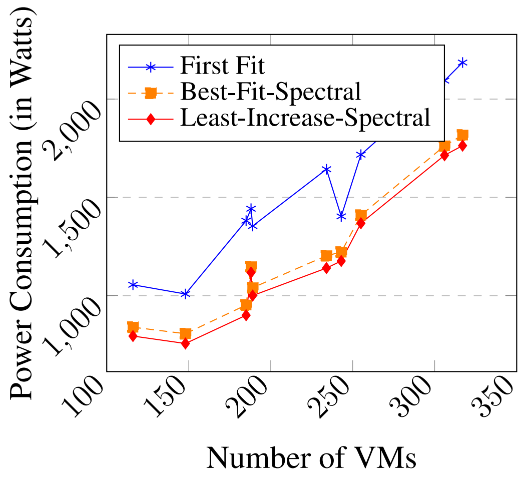


(b)

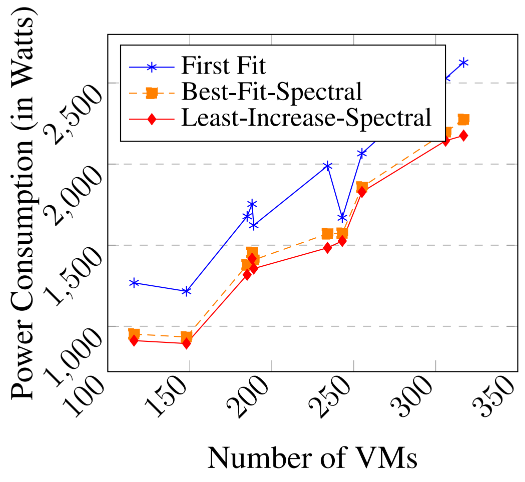


(c)

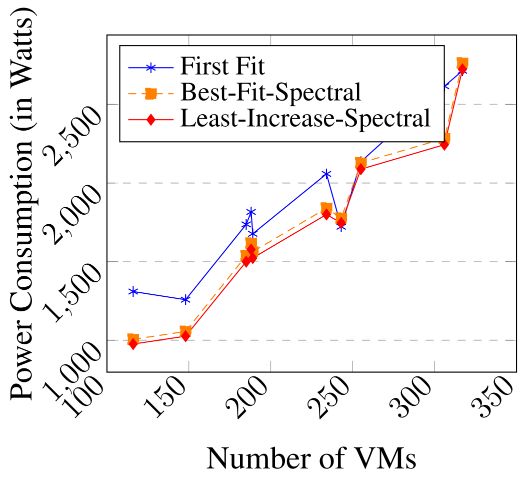
Figure 4.1: Online with Spectral : Energy Consumption for DC1 (a) 3-Tier (b)B-cube (c) Hyper-Cube



(a)

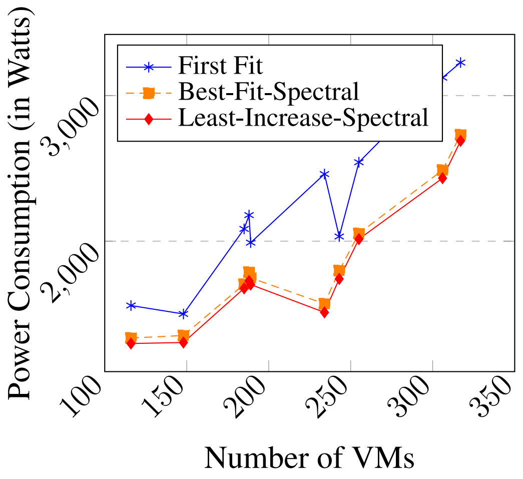


(b)

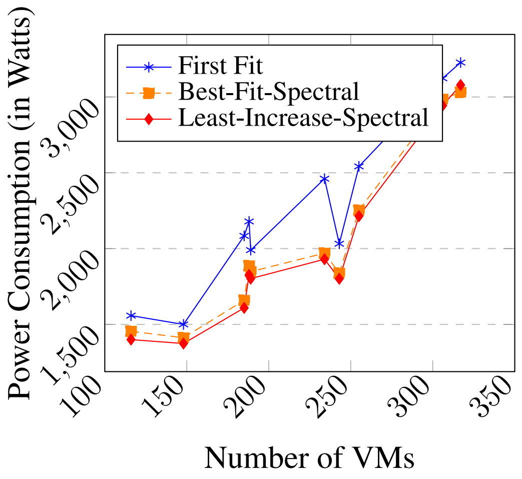


(c)

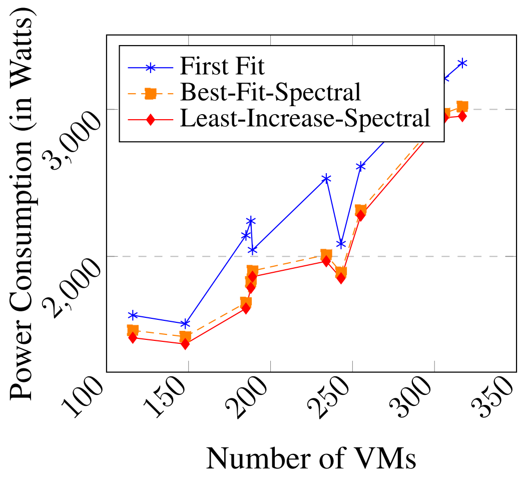
Figure 4.2: Online with Spectral: Energy Consumption for DC2 (a) 3-Tier (b)B-cube (c) Hyper-Cube



(a)



(b)



(c)

Figure 4.3: Online with Spectral: Energy Consumption for DC3 (a) 3-Tier (b)B-cube (c) Hyper-Cube

4.5 Summary

In this chapter, we addressed the online version of the problem of jointly optimizing the energy consumed by server and networking elements by an efficient VM placement. The problem is modelled as an integer programming problem. Two solutions, called best-fit-spectral and least-increase-spectral are proposed to solve the problem. The first algorithm recursively selects a set of virtual machines that has least inter and high intra communication and place them in the best-fit way, where as the second algorithm places the groups of communicating virtual machines in a way that gives least increase in energy. Spectral graph partitioning is used to find the partitions of the virtual machines graph that has the least inter communication between the two partitions. The proposed algorithms are tested for three different data center network architectures of three varying sizes and compared against standard VM placement algorithm called first fit heuristic. The proposed VM placement algorithms have achieved energy savings by 18 % and 21 % on an average respectively when compared with first-fit decreasing. The the proposed algorithms are scalable as similar percentage of energy savings achieved even when the size of the data center is increasing as the load on the data center is increasing.

Chapter 5

Intermediate Node Selection for Energy Efficient Scatter-Gather VM Migration

Live Migration [9] [10] [11] [12] is a process of migrating a virtual machine from one physical machine to another physical machine in a cloud data center without disconnecting the client or application. This is a key feature of a virtualized cloud data center as it allows seamless movement of the virtual machines from one node to another. During live migration, memory and disk data of the virtual machine are transferred across the network from the source host to the destination host.

In general, the performance metrics used for measuring the effectiveness of a virtual machine migration are total migration time and downtime. Deshpande *et al.* [13] proposed a new parameter - eviction time and developed a new VM migration mechanism - Scatter-Gather VM migration to reduce the eviction time. Eviction time is defined as the time taken to completely evict the state of one or more VMs being migrated from the source host. The following situations demand quick eviction of a virtual machine - Employ opportunistic power saving mechanisms by turning off excess server capacity [92][93][94][95][96][97][98], quick hotspot elimination for performance guarantees [99], evicting lower priority VMs immediately to accommodate other higher priority ones, perform emergency maintenance, or for handling impending failures. During normal migration, the source host resources occupied by the migrated VM are coupled with the destination host for the entire duration of the migration. Eviction time normally is equal to the total migration time. In certain situations like – destination under intense resource pressure, high congestion in the network, or destinations receiving from multiple resources, destination host might receive pages slower than they are evicted. In such a situation, the eviction time increases. In the Scatter-Gather Migration, VM migration is done through some intermediate nodes which would store the memory pages during the migration process. The source host sends (Scatter) the memory pages of the VM to multiple intermediate nodes in addition to the destination host. At the same time, the destination starts receiving (Gather) the memory pages from the source as well as the intermediate nodes concurrently. Thus the source can evict the VM with full speed even while the destination is slow in receiving. The intermediate nodes could be other hosts or network middle boxes which have some cache or memory. The essential idea in Scatter-Gather Migration is to decouple the source and destination as early as possible through the intermediates to reduce the eviction time. But the authors of the work have not addressed the selection of intermediates. Improper selection of intermediates sometimes can lead to increase in the total migration time, network congestion, and power consumption of the migration. This could be due to usage of more hosts and/or selection of network middle boxes as intermediates. Thus the intermediate nodes are to be selected optimally for effective implementation of the Scatter-Gather migration.

This chapter proposes to:

- Model the problem of selecting the intermediates for Scatter-Gather VM live migration as an integer programming problem with two optimality criterion – to reduce eviction time and to improve energy savings. The computational intractability of the problem is proved by showing it as NP-Complete.
- Two heuristic algorithms, called Max-Decrease-in-Eviction and Least-Increase-in-Energy are proposed to solve the problem of intermediate node selection in Scatter-Gather migration. The proposed solutions are tested against three naive solutions - Random Selection policy, Farthest Node First, Closest Node Next and are analysed with respect to three parameters - eviction time, total migration time and energy.

5.1 Motivation

Virtualization technology maximizes the utilization of hardware resources by sharing them among multiple users. A thin layer of software called Virtual Machine Monitor (VMM) or hypervisor running between operating system and system hardware manages and controls the virtual machines running on a platform [100][101]. In cloud computing, to improve data center efficiency many resource management techniques like load balancing, server consolidation, and power management are applied by migrating one or multiple virtual machines from one physical machine to another physical machine [92][93][94][95][96][97][98].

Live migration is the process of moving a virtual machine that is servicing an application from one host to another host with minimal disruption to the service [102].

The metrics that are used for virtual machine migration are total migration time and downtime. Total migration time is the time from when the process of migration begins from the source machine until the time the destination VM has complete control over the migrated VM [102].

$$t_m = \frac{v_m}{b} \tag{5.1}$$

where v_m is the total memory to be transferred and b is the bandwidth allocated from the source to the destination.

Downtime is the time during the migration when the application experiences service unavailability. If d, l, t_n are the page dirty rate, page size, and duration of the last pre-copy round respectively, then downtime is given by [102]

$$t_d = \frac{d * l * t_n}{b} \tag{5.2}$$

Deshpande *et al.* [13] proposed a new metric called eviction time which measures the time taken for releasing the resources of the source machine used for hosting the VM being migrated. It is the time taken to decouple the source to decouple the source from destination during migration.

5.1.1 Scatter Gather VM Live Migration

The Scatter-Gather VM migration has two phases - scatter phase and gather phase. As shown in the Figure 5.1, in the scatter phase the memory contents of the virtual machine that is being migrated are sent to some intermediate nodes along with the destination node. This facilitates the source to get decoupled quickly from the migration process. The scattering of the content is achieved through an abstracted layer Virtual Memory Device (VMD). The VMD collects all the free memory available at the intermediate nodes and presents as a block device one per the migration. The source migration manager then scatters the memory contents to multiple intermediates by writing them to the block device allocated to it and each page written by the source on the block device would be sent to an intermediate node. The page availability information is sent to the destination. In the gather phase, the destination migration manager collects the contents concurrently from the intermediate nodes at its own pace.

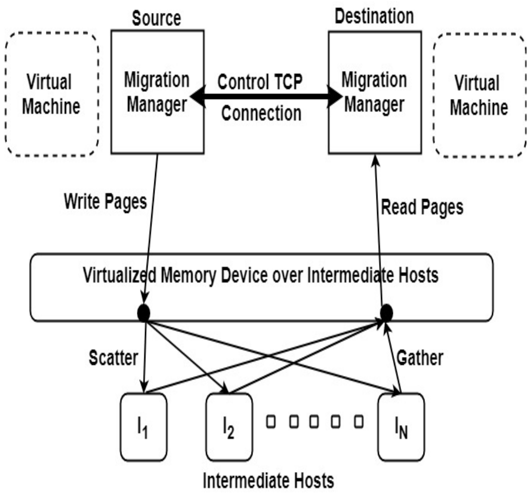


Figure 5.1: Scatter-Gather VM Migration

In Scatter-Gather VM migration, the problem of selection of the intermediate nodes and its impact on the eviction time has not been addressed . For effective implementation of the Scatter-Gather VM migration, the intermediate nodes have to be selected carefully. The proposed algorithms for intermediate node selection would be implemented in VMD over the intermediate nodes.

Table 5.1: Scatter-Gather: Notations

Notation	Meaning
R	virtual machine migration request
s	Source of migration request R
d	Destination of migration request R
m	Data of migration request R
b	Bandwidth requirement of migration request R
μ	eviction threshold of migration request R
$\mathcal{G}(\mathbb{V}, \mathbb{E})$	Data Center Graph
$\mathbb{V}$	Set of nodes of data center graph
$\mathbb{E}$	Set of edges of the data center graph
C_i	Capacity of node i
$A[j, k]$	Bandwidth capacity of the link $B[j, k]$
$\mathbf{X}_i$	Boolean variable indicating whether node i is selected as intermediate or not
E_i	Energy consumption of the node i
P_{idle}	Node idle power Consumption
P_{max}	Node maximum power consumption

5.2 System Model

In this section, the system model considered in this work is presented in detail.

5.2.1 Data Center Model

The data center network is represented as a weighted graph $\mathcal{G}(\mathbb{V}, \mathbb{E})$. The vertices in $\mathbb{V}$ represent either a server or a network middle box. $\mathbb{E}$ is the set of all the edges in the data center network. $A[j, k]$ is the bandwidth capacity of two directly connected nodes j and k in $\mathbb{E}$ and is zero otherwise. C_i is the capacity of node i in $\mathbb{V}$.

5.2.2 Migration Request Model

A virtual machine migration request R is represented as a quadruple $R(s, d, b, m, \mu)$, where s is the source host, d is the destination host, b bandwidth requirement, m is the amount of

VM data that is to be transferred to the destination, and μ is a threshold on eviction time for the request.

5.2.3 Energy Model

Middle boxes consists of network elements that do specialized tasks like load balancing, security and performance enhancement in the data center networks. Either servers or middle boxes are considered as intermediate nodes for Scatter-Gather VM migration. The power model considered for the server or middle box is utilization based energy model. The power consumed by a server or middle box at utilization u is

$$P_u = (P_{max} - P_{idle})u + P_{idle} \tag{5.3}$$

where P_{idle} is the power consumed by the node in idle state, i.e, when there is no load on it and P_{max} is the power consumed by the node at the maximum(100%) load.

The energy consumed by the intermediate node for a given duration of time is

$$E_i = \int_{t_0}^{t_1} P\left(u(t)\right) dt \tag{5.4}$$

5.3 Problem Formulation

$\mathbf{X}_i$ is a boolean variable indicating whether a node other than the source $i \in \mathbb{V} - \{s\}$ is selected as an intermediate node or not in the Scatter-Gather live migration of the virtual machine migration request R. N_i is the data chunk that needs to be transferred to the node i if it is selected as an intermediate node. $b[s, i]$ is the bandwidth to be allocated on the path from the source s of R to the node i to transfer content of the migration. The node capacity is indicated by C_i and the bandwidth capacity between two nodes i and j is given by $A[i, j]$. The distance of a node i from the source s is given by d_i. The capacities of the intermediates and the bandwidths of the paths to intermediates should not get exhausted because of the migration and only a fraction of the remaining capacities and bandwidths should be allocated to the migration. Hence two limiting factors α and β are considered to limit the memory allocated to the intermediates and the bandwidth allocated on the path to the intermediates.

Then eviction time of a migration is given by

$$e = \max_{\forall i \in \{\mathbb{V} - \{s\}\}} \left(\frac{X_i * N_i * d_i}{b\langle s, i \rangle} \right) \tag{5.5}$$

5.3.1 Minimize the Eviction Time

Problem Definition: Given a migration request $R(s, d, b, m, \mu)$ and a data center graph $G(V, E)$ with node and link capacities, the objective is to select a subset I of vertex set V as intermediate nodes to minimize the eviction time, while the cumulative capacity of the nodes of I is higher than the migration data m, and the collective bandwidth of the paths from the source of the migration to the nodes in I satisfy the bandwidth requirement of the migration request.

5.3.1.1 Objective Function

$$\mathit{min} \quad e = \max_{\forall i \in \{\mathbb{V} - \{s\}\}} \left(\frac{X_i * N_i * d_i}{b\langle s, i \rangle} \right) \tag{5.6}$$

The objective function is to minimize the eviction time of the migration. Eviction time is the maximum of the transfer times for sending the allotted migration data to the selected intermediate nodes.

5.3.1.2 Constraints

$$\sum_{\forall i \in \mathbb{V}} X_i \times b\langle s, i \rangle \geq b \tag{5.7}$$

$$\sum_{\forall i \in \mathbb{V}} X_i \times N_i = m \tag{5.8}$$

$$X_i \times N_i \leq \alpha \times C_i, \forall i \tag{5.9}$$

$$X_i \times b\langle s, i \rangle \leq \beta \times A[i, j], \forall i \tag{5.10}$$

$$X_i \in \{0,1\}, \forall i \tag{5.11}$$

$$X_d = 1 \tag{5.12}$$

Here, Eq.5.7 is a constraint to ensure that the bandwidth requirement is maintained. Eq.5.8 is to make sure that the total data that need to be sent to the intermediates is equivalent to m. Eq.5.9 is a capacity constraint that makes sure that the load on an intermediate doesn't exceed α percentage of the capacity. Likewise, Eq.5.10 is to keep the load on a link does not exceed β percentage of its capacity. Eq.5.11 is an integer constraint that indicates whether node i is selected as an intermediate or not. To ensure some of the memory contents are also to be sent to the destination directly, the last constraint i.e., Eq.5.12 is incorporated.

5.3.2 Minimize the Excess Energy Consumption

Problem Definition: Given a migration request $R(s, d, b, m, \mu)$ and a data center graph $G(V, E)$ with node and link capacities, the objective here is to select a subset I of vertex set V as intermediate nodes to minimize the excess energy consumed for the migration, while the cumulative capacity of the nodes of I is higher than the migration data m and the collective bandwidth of the paths from the source of the migration to the nodes in I satisfy the bandwidth requirement of the migration request.

5.3.2.1 Objective Function

The objective is to minimize the excess energy consumed because of the migration.

$$\texttt{min} \sum_{\forall i \in \mathbb{V}-\{s\}} X_i \times \bar{E}_i \tag{5.13}$$

where

$$\bar{E}_i = E_i^{Post} - E_i^{Pre} \tag{5.14}$$

$\bar{E}_i$ is the excess energy for using the node i as the intermediate. The pre-migration energy and post-migration energies are calculated using the utilization based energy consumption model which is usually used for physical machines that are intermediates here.

5.3.2.2 Constraints

$$\sum_{\forall i \in \mathbb{V}} X_i \times b\langle s,i \rangle \geq b \tag{5.15}$$

$$\sum_{\forall i \in \mathbb{V}} X_i \times N_i = m \tag{5.16}$$

$$X_i \times N_i \leq \alpha \times C_i, \forall i \tag{5.17}$$

$$X_i \times b\langle s,i \rangle \leq \beta \times A[i,j], \forall i \tag{5.18}$$

$$e \leq \mu \tag{5.19}$$

$$X_d = 1 \tag{5.20}$$

$$X_i \in \{0,1\}, \forall i \tag{5.21}$$

The constraints for this objective are the same as mentioned for the first objective except the constraint in eq.5.19 which ensures that the total eviction time is less than some determined value μ.

5.4 Computational Intractability of Intermediate Node Selection Problem(INS-SGM)

In this section, computational intractability of the INS-SGM problem is discussed. We establish that the problem of Intermediate Node Selection in Scatter-Gather Migration (INS-SGM) is NP hard by reducing 0-1 knapsack problem to Minimum Subset Sum Problem(MSSP) and then MSSP to INS-SGM.

The decision version of the 0-1 knapsack problem is to search for a subset $\mathbb{U}_0$ of a set $\mathbb{U}$ of objects with a weight $w(u) \in Z^+$ and a value $p(u) \in Z^+$ for each for each $u \in \mathbb{U}$ satisfying

$$\sum_{u \in \mathbb{U}_0} w(u) \leq W \quad \texttt{and} \quad \sum_{u \in \mathbb{U}_0} p(u) \geq P$$

where P is a desired value of the knapsack.

The decision version of minimum subset sum problem is to decide whether a subset $\mathbb{A}'$ exists in a given finite multi set $\mathbb{A}$ with the size of each element $s(a)$, $a \in \mathbb{A}$ such that

$$M \leq \sum_{a \in A'} s(a) \leq K$$

where K is any integer and M is the target sum.

We will now prove that INS-SGM is NP-hard in two stages.

Claim 5.4.1. *MSSP is NP-hard*

Proof. Consider an instance of the 0-1 Knapsack problem for which the profit and weight values are same for each object in U, i.e, $w(u) = p(u), \forall u \in U$. Assume that there exists a subroutine S[A, s, K, M] for solving minimum subset problem with parameters $A = \{a_1, a_2, \ldots\}$, $s : A \rightarrow Z^+$, a target sum M, and an integer $K \geq M$. Then the instance of the 0-1 knapsack problem defined above can be solved by $S[U, w, W, P]$, where W is the bin capacity and P is the profit required. It is easy to see that the 0-1 knapsack would be solved in polynomial time if subroutine S were a polynomial time algorithm for solving minimum subset sum. Hence it is established that the 0-1 knapsack problem is reducible to minimum subset sum problem. □

The decision version of the intermediate node selection for Scatter-Gather migration is to determine whether a subset $\mathbb{V}_0$ of a set $\mathbb{V}$ of vertices, with capacities $C(b) \in Z^+$, cost value(energy or eviction time) $E(b) \in Z^+$ for each $b \in V$ and positive integers L (migration request size) and P'(maximum cost), such that

$$L \leq \sum_{b \in \mathbb{V}_0} C(b) \leq P'?$$

Claim 5.4.2. *INS-SGM Problem is NP-complete*

Proof. To prove INS-SGM to be NP-Complete, first we show that it is in NP and then show that MSSP is reducible to INS-SGM. For a given set of intermediate nodes it can be easily verified in polynomial time that the total sum of the capacities of the intermediate nodes is greater than migration request memory size L and the cost for selecting the given intermediates in terms of energy or eviction time is less than P'. So it is in NP.

Assume that there exists a subroutine $S'[V, c, o, P', L]$ for solving INS-SGM with parameters $I = \{a_1, a_2, \ldots\}$, $c : I \rightarrow Z^+$,$o : I \rightarrow Z^+$, migration request size L, and maximum cost P'. Then any instance of the minimum subset sum problem can be solved by $S'[A, s, s, K, M]$, where M is the desired sum and k is an integer greater than M. Here the weight value of an element is passed as both capacity and cost of a node. It is easy to see that the minimum subset sum problem would be solved in polynomial time if subroutine S' were a polynomial time algorithm for solving intermediate node selection problem. So the minimum subset sum problem is reduced to INS-SGM problem. Hence INS-SGM problem is NP-complete. □

5.5 Heuristics for Intermediate Node Selection in Scatter-Gather Migration

As INS-SGM is NP-complete and the intermediate nodes are to be selected quickly, a heuristic solution is a viable approach to solve the problem. We now propose two heuristic algorithms for solving the problem.

5.5.1 Maximum Decrease in Eviction Time

Eviction time for a normal migration process without using any intermediate node is equal to the total migration time. To reduce the eviction time the memory contents are sent to intermediate nodes along with destination. With respect to the intermediate nodes, the eviction time depends on the distances from the source to the selected intermediate nodes and the bandwidths between the source and the intermediate nodes. The time taken to transfer the contents from the source to an intermediate node is proportional to the distance between the source and the intermediate node. The transfer time between the source and an intermediate node is inversely proportional to the bandwidth of the path between the source and the intermediate node.

A heuristic solution called maximum decrease in eviction time is proposed as described in the Algorithm 5.1. Here the idea is to select the node that has least expected transfer time first as intermediate node. Initially, the set of potential intermediate nodes is calculated by adding all the nodes present in the paths from the source to the destination of the migration request. Then for each potential intermediate node, shortest path from the source is calculated and the shortest path hop count is used as a measure of the distance between the source and the intermediate node. Then for all the potential intermediate nodes, expected transfer time to send $\alpha\%$ of the capacity of the intermediate node with $\beta\%$ of the bandwidth capacity of the link is calculated. However, if the shortest paths to any two intermediate nodes are not node disjoint, then the transfer time for the second intermediate node in the common path may be more. So, the expected transfer times are updated by checking whether the shortest paths to the intermediate nodes are node-disjoint or not. Then, based on these expected transfer times the potential intermediate nodes are ordered in non decreasing order. Then the first in the sorted order is selected as an intermediate node in order to reduce the eviction time. The total weight of the pages that are sent to the selected intermediate node would be equivalent to $\alpha\%$ of its capacity. Then this selected node is removed from the list of possible intermediate nodes and the next one in the order is selected as the intermediated node. This process of choosing intermediate nodes is continued as described above until all the memory pages are allocated to the intermediate nodes.

5.5.2 Least Increase in Energy Consumption

When compared with normal migration process, Scatter-Gather migration uses extra nodes for completing the process of migration. So the Scatter-Gather migration consumes more energy than the normal migration and the amount of energy consumed for the migration depends on the intermediate nodes selected. Since the reduction of energy consumption is a concern for cloud administrators, it is essential to implement the Scatter-Gather migration energy efficiently. Hence a heuristic solution based on 5/4 approximation algorithm for minimum subset sum [5] is proposed for intermediate node selection that minimize the excess energy consumed by the intermediates. The proposed solution is specified in Algorithm 5.2. Initially, all the nodes in the paths from the source to the destination are selected as the potential intermediate nodes indicated by the set $\mathbb{U}$. The capacities of the intermediate nodes in $\mathbb{U}$ are adjusted to of $\alpha\%$ of their maximum capacities. Then subroutine cover is called which returns the set of intermediate nodes in $\mathbb{U}$ which has capacity more than m and as close to as m. The memory content assigned to an intermediate node is equivalent to its revised capacity and $\beta\%$ of the bandwidth capacity on the path from the source to the destination is allocated to transfer the assigned data. Based on these values expected eviction time is calculated and if it is more than the eviction threshold μ then the value of α is increased and the procedure is repeated from step 2.

5.5.2.1 COVER

The procedure cover is a 5/4 approximation algorithm for minimum subset sum proposed by Grigoriu *et al.* [5]. It gives the set of subset of intermediate nodes whose sum of the capacities is at least m and as close to m as possible. The procedure is specified in the Algorithm 5.3.

First, $\mathbb{U}$ is split into five multiple subsets as $\mathbb{I}_1 = \{i \in \mathbb{I} | cap(i) < \frac{1}{4}m\}$, $\mathbb{I}_2 = \{i \in$

Algorithm 5.1 Maximum-Decrease-in-Eviction Time

```
Input: $\mathcal{G}, R, \alpha, \beta$
Output: $\mathbb{I}$                                   ▷ Intermediate node set
 1: Find all the nodes present in the paths from the source $s$ to the destination $d$ and add
    them to potential intermediate node set $\mathbb{U}$ without duplicates;
 2: for each node $i$ in $\mathbb{U}$ do
 3:     $P[i] \leftarrow \textsc{shortestPath}(s, i)$;
 4:     $shops[i] \leftarrow \textsc{shopCount}(P[i])$;
 5:     $bw[s,i] \leftarrow \textsc{spathBW}(s, i, P[i])$;
 6:     $transferTime[i] = \frac{C_i \times shops[i]}{bw[s,i]}$;
 7:     $cTime[i] = transferTime[i]$;
 8: end for
 9: $i_1 = s$;
10: $flag = 0$;
11: while $flag = 0$ do
12:     flag=1;
13:     for each node $i$ in $\mathbb{U}$ do
14:         $i_2 = i_1$;
15:         for $\{j = 1;\ j < shops[i];\ j = j + 1\}$ do
16:             $i_3 = P[i_2][j]$;
17:             if $cTime[i_3] < transferTime[i_3] + cTime[i_2]$ then
18:                 $cTime[i_3] \leftarrow transferTime[i_3] + cTime[i_2]$;
19:                 flag=0;
20:             end if
21:             $i_2 = i_3$;
22:         end for
23:     end for
24: end while
25: $\textsc{sort}(\mathbb{U}, cTime)$;   ▷ sort $\mathbb{U}$ based on Critical Time values in non decreasing order
26: while $m \neq 0$ do
27:     Select the first node $U_1$ in sorted $\mathbb{U}$ as an intermediate node;
28:     The amount of data that allocated to the node is $N_{U_1} = min(\alpha * C_1, m)$;
29:     m= $m - N_{U_1}$;
30:     $b\langle s,i\rangle = \beta \times A[s,i]$ ▷ Allocate $\beta\%$ of the bandwidth capacity for transferring data.
31:     $\mathbb{U}$=$\mathbb{U}$-$U_1$;
32:     $\mathbb{I} \leftarrow \mathbb{I} \cup U_1$
33: end while
34: while $\sum_{\forall i \in \mathbb{I}} b\langle s,i\rangle < b$ do
35:     Refine($\beta$);   ▷ iteratively increase the $\beta$ value till the condition is satisfied
36: end while
    return $\mathbb{I}$
```

$\mathbb{I} | \frac{1}{4}m \leq \textit{cap}(i) < \frac{1}{2}m\}$, $\mathbb{I}_3 = \{i \in \mathbb{I} | \frac{1}{2}m \leq \textit{cap}(i) < \frac{3}{4}m\}$, $\mathbb{I}_4 = \{i \in \mathbb{I} | \frac{3}{4}m \leq \textit{cap}(i) < m\}$, $\mathbb{I}_l = \{i \in \mathbb{I} | \textit{cap}(i) \geq m\}$.

Then, two subroutines used in the algorithm cover are described as below.

- add$I_1(\mathbb{Q}, \mathbb{I}_1, m)$: This subroutine adds nodes from $\mathbb{I}_1$ to the set of nodes $\mathbb{Q} \subseteq \mathbb{I} \backslash \mathbb{I}_1$ with $m - \sum_{i \in \mathbb{I}_1} \textit{cap}(i) \leq \sum_{i \in \mathbb{Q}} \textit{cap}(i) \leq \frac{5}{4}m$ until the total capacity of the nodes in $\mathbb{Q}$ is at least m. So the returned set $\mathbb{Q}$ has a total capacity of at least m and less than $\frac{5}{4}m$ as the capacities of all the nodes from $\mathbb{I}_1$ are less than $\frac{1}{4}m$.

- update($\mathbb{Q}$, IBest) : It updates the best solution IBest if the newly found set Q is better.

Let $min_j(\mathbb{J}_h)$ and $max_j(\mathbb{J}_h)$ indicate the set of intermediate nodes with j smallest and j largest capacities respectively from the set $\mathbb{J}_h$.

Algorithm 5.2 Least-Increase-in-Energy

```
Input: $\mathcal{G}, R, \alpha, \beta$
Output: $\mathbb{I}$                                        ▷ Intermediate node set
 1: Find all the nodes present in the paths from the source $s$ to the destination $d$ and add
    them to Potential Intermediate node set $\mathbb{U}$ without duplicates
 2: for each node $i$ in $\mathbb{U}$ do
 3:     $cap(i) = \alpha \times C_i$
 4: end for
 5: $\mathbb{I}$= COVER($\mathbb{U}, m$)
 6: for each node $i$ in $\mathbb{I}$ do
 7:     $N_i = cap(i)$
 8:     $P[i] \leftarrow$ SHORTESTPATH($s, i$);
 9:     $shops[i] \leftarrow$ SHOPCOUNT($P[i]$)
10:     $b\langle s, i\rangle = \beta \times A[s, i]$ ▷ Allocate $\beta\%$ of the bandwidth capacity for transferring data.
11: end for
12: while $\sum_{\forall i \in I} b\langle s, i\rangle < b$ do
13:     Refine($\beta$);          ▷ iteratively increase the $\beta$ value till the condition is satisfied
14: end while
15: $e = \max_{\forall i \in I} \left(\frac{N_i * shops[i]}{b\langle s, i\rangle}\right)$
16: if $e < \mu$ then
17:     Refine($\alpha$);                                   ▷ increase the $\alpha$ value
18:     Repeat from step 2
19: end if
    return $\mathbb{I}$
```

Algorithm 5.3 Cover

```
Input: $\mathbb{U}, m$
Output: $IBest$                                    ▷ Intermediate node set
1:  Partition $\mathbb{U}$ into five subsets as $\mathbb{I}_1 = \{i \in \mathbb{I} | cap(i) < \frac{1}{4}m\}$, $\mathbb{I}_2 = \{i \in \mathbb{I} | \frac{1}{4}m \leq cap(i) < \frac{1}{2}m\}$, $\mathbb{I}_3 = \{i \in \mathbb{I} | \frac{1}{2}m \leq cap(i) < \frac{3}{4}m\}$, $\mathbb{I}_4 = \{i \in \mathbb{I} | \frac{3}{4}m \leq cap(i) < m\}$, $\mathbb{I}_l = \{i \in \mathbb{I} | cap(i) \leq m\}$.
2:  Let $sumI_1 = \sum_{i \in I_1} cap(i)$;
3:  if $sumI_1 \geq m$ then then return add$I_1(\phi, \mathbb{I}_1, m)$
4:  end if
5:  if $|I_l| \geq 1$ then $Ibest = min_1(I_l)$
6:  else$Ibest = I$
7:  end if
8:  if $|I_2 \cup I_3 \cup I_4| \geq 1$ then
9:      let $q \in max_1(I_2 \cup I_3 \cup I_4)$ ;
10:     if $cap(q) + SumI_1 \geq m$ then
11:         return add$I_1(\{q\}, \mathbb{I}_1, m)$
12:     end if
13: end if
14: if $(|I_2| \geq 2 \wedge \sum_{i \in max_2(I_2)} cap(i) + SumI_1 \geq m)$ then
15:     return add$I_1(max_2(I_2), \mathbb{I}_1, m)$
16: end if
17: if $|I_2| \geq 3$ then
18:     Let $K := min_3(I_2)$ and $K' := max_3(J_2)$
19:     if $\sum_{i \in K} cap(i) \geq m$ then
20:         update($k$, IBest);
21:     else if $\sum_{i \in K} cap(i) + sumI_1 \geq m$ then
22:         return add$I_1(k, \mathbb{I}_1, m)$;
23:     else
24:         Iteratively replace the nodes in $k$ with those of $k'$ until either $\sum_{k \in K} cap(i) \geq m$ is true(then return $k$) or all the nodes in $k$ are replaced.
25:     end if
```

```
26:     if $\sum_{i \in K} cap(i) + sumI_1 \geq m$ then
27:         return add$I_1(k, \mathbb{I}_1, m)$;
28:     end if
29: end if
30: if $|I_2| \geq 4$ Update($min_4(I_2), Ibest$); then
31: end if
32: if $|I_2| \geq 2 \wedge |I_3| \geq 1$ Update($min_1(I_3) \cup min_2(I_2), Ibest$); then
33: end if
34: if $|I_3| \geq 1 \wedge |I_2| \geq 1$ then
35:     if $\sum_{i \in max_1(I_3) \cup max_1(I_2)} cap(i) + sumI1 \geq m$ then
36:         return add$I_1(max_1(I_3) \cup max_1(I_2), I_1)$;
37:     end if
38: end if
39: if $|I_3| \geq 2$ then update($min_2(I_3), Ibest$);
40: end if
41: if $|I_2 \cup I_3| \geq 1 \wedge |I_4| \geq 1$ then update($min_1(I_2 \cup I_3) \cup min_1(J_4), Ibest$)
42: end if
43: if $|I_4| \geq 2$ then update($min_2(I_4), Ibest$);
44: end if
    return Ibest
```

5.6 Performance Analysis

In this section, we present the performance analysis of the intermediate node selection policies proposed in the chapter.

5.6.1 Performance Metrics

For comparing the efficiency of the proposed heuristic solutions for intermediate node selection, several performance metrics are used. The first metric is the eviction time, it is the time taken to transfer all the memory contents of the migrating VM from the source to the intermediate nodes and the destination. When no intermediate nodes are used the eviction time is equivalent to the total migration time. But when intermediate nodes are used it will be less than the total migration time. The second metric is the increase in energy consumption. It gives the additional energy that the migration causes for using intermediate nodes when compared with migration without using any intermediate nodes. It includes the additional energy consumed by the intermediates for holding the migration data and the network switches for transferring the data to the intermediate nodes. The last performance metric considered is the total migration time. It is the time taken to completely transfer the contents of the migration data to the destination node so that it can start the virtual machine being migrated. The total migration time would be slightly higher for Scatter-Gather migration when compared with normal migration due to the overhead delays incurred by the use of intermediate nodes. So it is important to analyze the solutions with respect to total migration time.

5.6.2 Experimental Setup

The simulations are done on a system with intel core i5-4200U CPU with four cores and 6GB RAM. Java is used to implement the algorithms proposed in this chapter. The data center network considered is 3-tier data center network. A 3-tier data center network consists of three layers namely, edge, aggregation, and core. The network can be divided into k pods. Each pod in a k-ary fat-tree consists of $\frac{k^2}{2}$ severs and two layers of $\frac{k}{2}$ k port switches. Each edge switch is connected to $\frac{k}{2}$ servers and $\frac{k}{2}$ aggregate switches. Each aggregate switch connects to $\frac{k}{2}$ edge and $\frac{k}{2}$ core switches. k ary fat-tree contains $\frac{k^2}{2}$ core switches, where each core switch connects to k pods.

The virtual machines considered here are of four types, where the requirements of each type of virtual machines are as shown in Table 5.2. The physical machines are assumed to be heterogeneous. They are classified into four types of physical machines, and the capacities of each type of physical machine are given in the Table 5.3. The power consumption values of a server for each type of physical machine are of standard values [3]. The peak power consumption of a commodity switch is taken as 40 watts, which is a standard value for a typical commodity switch.

The power consumption model considered for the intermediates is a utilization-based power consumption model that is usually used for a physical machine as described in section 5.2.3.

We have considered three data centers of different sizes to evaluate the proposed algo-

Table 5.2: Scatter-Gather: VM Types

VM type	Memory
VM1	512
VM2	1024
VM3	2048
VM4	4096

Table 5.3: Scatter-Gather: PM Types

PM Type	Memory	P_{max}	P_idle
PM1	4096	79	42
PM2	8768	101	53
PM3	32768	152	78
PM4	31072	194	101

rithms as shown in the Table 5.4. The first data center contains 40 physical machines in 4 racks of a single cluster. These 40 machines are connected by fat-tree network containing 7 switches. The second data center contain 240 physical machines organized into 8 racks and one cluster. These 240 physical machines are connected by 14 switches. The third data center contains 2 clusters, where the first cluster contain 480 machines arranged into 16 racks and the second cluster contains 160 physical machines arranged into 4 racks. These 640 machines are connected by fat-tree network containing 35 switches.

Table 5.4: Scatter-Gather: Data Center sizes

Type	Clusters	racks	PMs	Switches
DC1	1	4	40	7
DC2	1	8	240	14
DC3	2	20(16, 4)	640	35

Initially a set of randomly created virtual machine requests are scheduled on the physical machines using first-fit-decreasing heuristic. At a random time between the starting and ending of the schedule of the virtual machines, a migration request is generated by selecting a virtual machine for migration with uniform probability. The network traffic generated between a pair of VMs, in a particular interval is applied on all the links and switches present in the shortest path between the corresponding PMs, where the communicating VMs are hosted. The network traffic load among all possible pairs of communicating VMs is aggregated similarly. While migrating a VM, total memory image is being transferred from source PM to destination PM, and the network traffic load is accumulated during that migration period, on the links and switches used in the migration.

The proposed algorithms for selecting the intermediates are compared against 3 naive solutions called random selection, farthest node first and closest node next. Random selection policy is selecting a node randomly and allocating $\alpha\%$ of its capacity as the memory allocated to it and repeating the process until the entire memory is allocated to the intermediates. The random machine is selected from the set of potential intermediate nodes using a uniformly distributed random variable. In farthest node first policy, the farthest node to the source with respect to the paths from the source to the destination is selected and $\alpha\%$ of the migration memory is transferred to that node. This process is repeated until the entire memory is exhausted. This policy is better suited for some scenarios. The decision to migrate a virtual machine is made sometimes because the top-of-rack switch of that physical machine is overloaded. In such situations it is desirable to select the intermediate nodes near to the destination instead of the source. In closest node next policy, the closest node to the source node is selected and $\alpha\%$ of the migration memory is transferred to that node. Again the process is repeated by selecting the next closest node till the entire memory is allocated to the intermediate nodes.

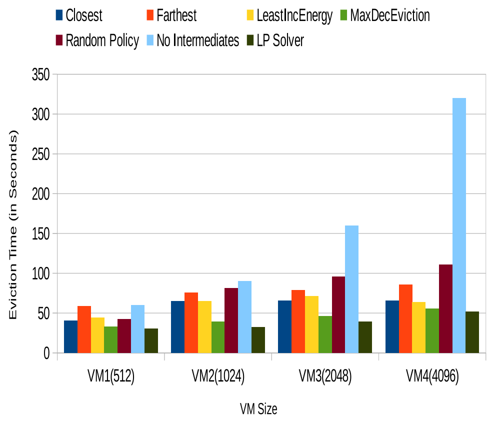
5.6.3 Performance Evaluation

For the data centers DC1 and DC2, the proposed algorithms are compared with IBM C-plex LP solver. The LP-solvers usually gives solutions which are very close to optimal values. So C-plex LP solver values are taken as the optimal values for the problem and are used to verify the effectiveness of the solutions.

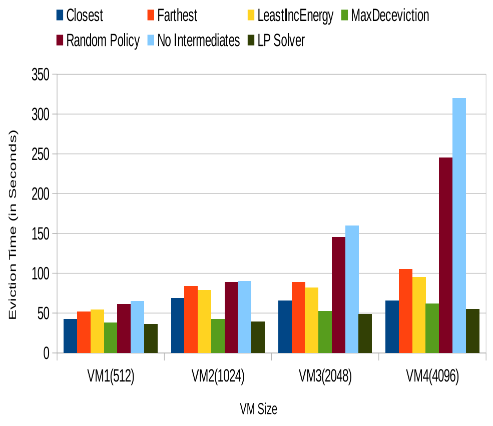
Here we present the evaluation of the proposed heuristic solutions with respect to three performance metrics.

5.6.3.1 Eviction Time

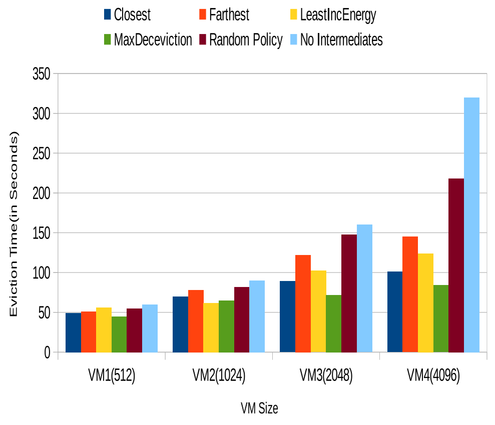
As already stated the eviction time indicates how quickly the resources used by the virtual machine being migrated are released. The eviction times for migrating four different types



(a) edc3:1

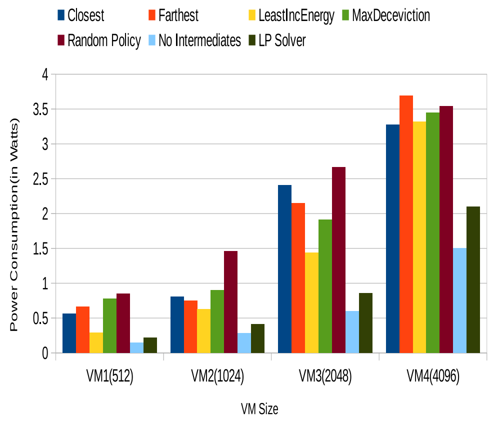


(b) edc3:2

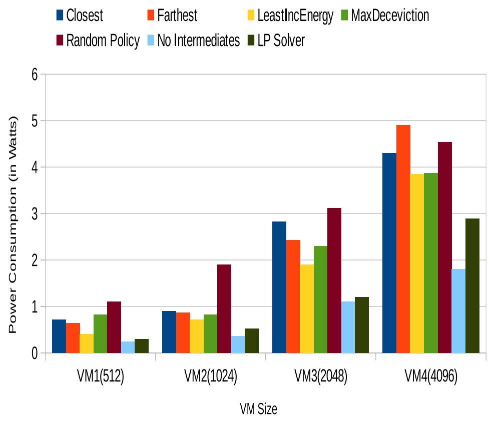


(c) edc3:3

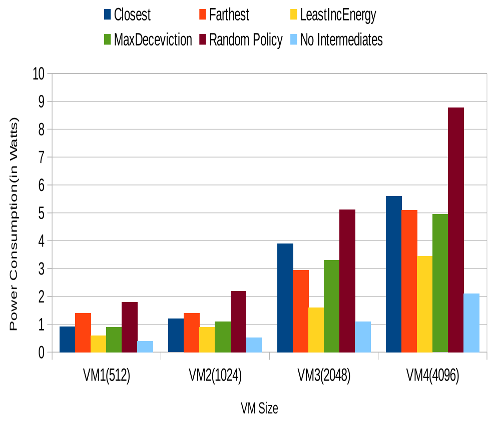
Figure 5.2: Eviction Time (a) DC1 (b) DC2 (c) DC3



(a) pdc3:1

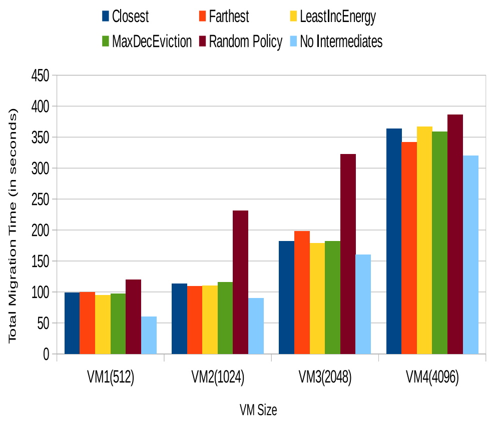


(b) pdc3:2

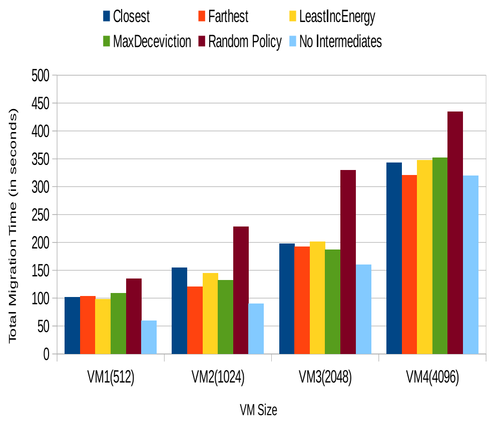


(c) pdc3:3

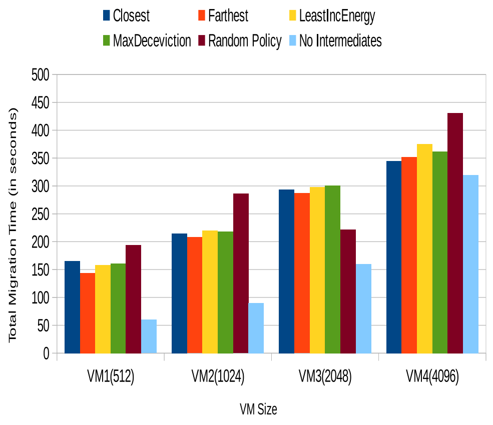
Figure 5.3: Power Consumption (a) DC1 (b) DC2 (c) DC3



(a) tmdc3:1



(b) tmdc3:2



(c) tmdc3:3

Figure 5.4: Total Migration Time (a) DC1 (b) DC2 (c) DC3

of virtual machines using the heuristic solutions as intermediate node selection policy are plotted in the Figure 5.2. As it can be observed from the graph that for all the data center configurations, Maximum-Decrease-in-Eviction-Time algorithm is giving the lowest time. For a single data center, as the VM size is increasing, the eviction times are also increasing. The closest next heuristic is also faring better when compared with the remaining heuristic solutions as the nodes that picked are close to the source. The random policy is having very high eviction times confirming the need for the heuristic solutions proposed. When no intermediate nodes are used, the eviction time is equal to the total migration time as observed in the results.

The time taken to obtain results using LP-solver for DC1 and DC2 are 1.05 and 33.41 minutes respectively, where as the proposed heuristic solution has given the result in 1.4 and 2.1 seconds respectively for DC1 and DC2. For the largest data center DC3, the LP-solver has not terminated even after running for one hour, where as the proposed heuristic solution has taken 4.3 seconds to give the result, necessitating the need for heuristic solutions to solve the problem. The eviction time values for Maximum-Decrease-in-Eviction-Time heuristic are near to the optimal values given by LP solver for both DC1 and DC2.

5.6.3.2 Energy Consumption

The plotting of the power consumption values when virtual machines of four different sizes are migrated using different policies is presented in the Figure 5.3. As it can be inferred from the graph, the proposed solution, least-increase-in-energy is performing better than all the other heuristic solutions across all the data center configurations. The random selection policy is performing poorly with respect to energy consumption, as it may select the physical machines which are in sleep mode when selecting the intermediates randomly. For smaller size virtual machines, the heuristic solutions are consuming very less power and as the size increases the power consumption for the migration is increasing.

The power consumption of a Scatter-Gather migration that uses intermediate node would be more than the power consumed by normal migration. This is validated by the results in the graph. However, the proposed solution is causing the least increase in energy consumption even by using intermediate nodes.

The time taken to obtain the results using LP-solver for DC1 and DC2 are 1.253 and 31.65 minutes respectively. The proposed heuristic solution gives the results in 1.163 and 2.9 seconds respectively for DC1 and DC2. For the data center DC3, the LP-solver has not terminated even after one hour, whereas the proposed heuristic solution has produced the solution in 5.2 seconds indicating the need for designing heuristic solutions for the problem. With respect to power consumption, the values obtained by the proposed solution are not near to the optimal values generated by the LP solver, even though they are much better than the other heuristic solutions. The reason is, the power consumed by the switches(networking nodes) while transferring the data from the source to the intermediates has not been considered to reduce the additional complexity in implementing the problem in C-plex. However, in the simulation of the proposed heuristic solution, the power consumed by the networking elements during the transfer of the memory contents to the intermediate nodes has been considered in the calculations of power consumed by the migration. The proposed algorithm is performing better than the other approaches because it selects the intermediates such that their cumulative remaining capacity is almost equal to the amount of memory that needs to be migrated, resulting in selection of a minimum number of intermediates for transferring contents of a VM migration for a given eviction time threshold. As it finds a near optimal tight packing for the VM migration data into the intermediates, the additional energy caused by the usage of intermediates would be minimal. The other approaches may not get such tight packing, resulting in higher additional energy for the usage of intermediates.

5.6.3.3 Total Migration Time

The total migration time is the duration of time from the start of the migration process to the time the virtual machine can be started and the source can be discarded. The total migration time for all the policies for the three data centers of varying sizes is depicted in Figure 5.4. In all the data centers, random policy is giving the worst performance with respect to total migration time but among the heuristic solutions proposed there is no clear pattern and the values are close to each other. In DC3, for virtual machine of size 512 MB, the farthest next is performing better, where as for the virtual machine of size 4096 MB the closest-first heuristic is doing better. In DC2, for the virtual machine of size 512 MB, least increase in energy is performing better and for other 3 virtual machines farthest is faring better. For the virtual machine of size 4096 MB almost for all the heuristic solutions, the values are similar in all the three data centers. So when the goal is to choose an intermediate node selection policy for having less total migration time, except random policy any one of heuristic can be applied as they are all performing similarly.

The total migration time for Scatter-Gather migration would be higher than that of normal migration without any intermediates. However, both the proposed solutions for minimizing the eviction time and increase in energy are not having high total migration time values as can be observed from the graphs.

5.7 Summary

Scatter-Gather migration is used to quickly evict the virtual machine from a physical machine in a virtualized data center. This work advances the implementation of the Scatter-Gather migration by optimal selection of the intermediates with respect to two goals, eviction time and energy. The problem of intermediate node selection in Scatter-Gather migration is modelled as an integer programming problem. We have also proved the intractable nature of the problem and proposed two heuristics algorithms. It is inferred from the experimental results that the maximum-decrease-in-eviction time heuristic is the best heuristic for reducing eviction time and the least-increase-in-energy heuristic is best suited when the goal is reducing the energy consumed for migration. With respect to total migration time, the performance of all the heuristics is almost similar, although they all are performing much better when compared with random selection policy.

Chapter 6

Federation Formation Mechanism for Energy Efficient Resource Sharing in Federated Cloud

Cloud federation is an effective solution for reducing the energy expenditure of the cloud providers by offering a way to make profit out of the idle resources residing in their data centers [71]. Cloud federation is collaboration of two or more cloud service providers to provide cloud services to the users. The cloud providers participating in the federation, mutually agree to co-operate and offer a pool of virtual instances to service the applications of the users. Cloud federation enables the cloud providers to scale up the resources as and when needed based on the load by getting additional resources from the other cloud providers participating in the federation. The users requests are satisfied by resources from multiple cloud providers [71][72]. Without cloud federation, the cloud provider which doesn't have the required resources would reject requests, there by losing money and reputation. Simultaneously the cloud providers with more idle resources earn profit by offering them to the federation. Cloud federation results in a great reduction of the energy costs for the cloud provider as the utilization of the data center resources increases considerably.

In a federation, a cloud provider earns income by offering services to the requests either by it's own resources or by leasing in/out resources from the other providers. It gains profit if the income from providing the services to the users is more than the operational costs of resources to provide those services like energy costs or the leasing costs. Cloud provider, being a selfish agent, always tries to maximize its profit and would join the federation if its profit is maximised or at least gets more profit than what it gets when acting alone. However, achieving the social welfare of the federation, i.e, the total sum of the profits obtained by all the providers participating in the federation, helps the cloud providers to have better sustainability in the long run. So an important challenge is how to achieve the global energy sustainability of the federation and in turn prompt the cloud providers to take part in the federation.

Majority of the existing approaches in the literature focus on maximising the individual profits of the cloud providers but not the social welfare. The ones who tries to improve the social welfare of the federation consider one request at a time and form a federation to service that request. However, since the federation is formed for each single request, resource sharing may not be efficient in terms of global energy consumed by the providers for servicing many requests in a given time. An efficient, yet practical way to achieve better global welfare in the long run, instead of trying to form a federation as soon as the request arrives, stall it till a minimum number of requests are arrived or for a very short time period of time. Then a federation is formed to this pool of requests arrived in that short period so that resource sharing and allocation decisions are taken better to achieve good social welfare.

With that motivation, we formulate the energy aware resource allocation for a set of requests in federated cloud as a co-operative game. Then the properties of the modelled co-operative game like stability and fairness are studied. To find the optional federation for a set of requests from the users, a novel federation formation mechanism is designed.

The major contributions of the chapter are as follows.

- Energy aware resource allocation for a set of requests from multiple tenants in federated cloud is modeled as a co-operative game.
- The properties of the modeled game, stability and fairness are discussed in detail.
- As a solution to the problem, a novel federation formation mechanism based on branch and bound technique is designed.

- The effectiveness of the solution is demonstrated through the experimental results obtained by comparing it against two variations of greedy best fit, one is online and the other is semi-online.

6.1 System Model

We consider a federated cloud, which consists of different cloud providers, a broker and several cloud users. The job of the broker [75] is to receive new requests from the users, getting the required resources from the cloud providers for servicing the users requests and distributing the profit to the cloud providers based on the resources they provided to service the requests. The time is divided into small epochs. The key assumption we are making here is, the resource allocation is done by the broker either at the end of each epoch or the number of users requesting the VM requests is equivalent to the number of cloud providers ready to participate in the federation, whichever is earlier. Allocation each VM request immediately would not give optimal social welfare to the federation, and at the same time completely static solution is infeasible and impractical. A reasonable approach to achieve better social welfare while the requests are serviced quickly is to hold the incoming requests for a very small epoch or till minimal number of requests are arrived at the broker, whichever is earlier.

Let the cloud providers that are present in the federation are represented by set **CP** = $\{CP_1, CP_2, \ldots, CP_m\}$. The set of physical servers owned by the cloud provider CP_i is given by P_i = $\{P_{ik} | k = 1, \ldots, l\}$. The capacity of a physical server P_{ik} is given by

P_{ik}^{core} - Computing capacity in terms of number of cores

P_{ik}^{mem} - Memory capacity

$P_{ik}^{storage}$ - Storage capacity.

A cloud provider CP_i supply it's physical resources in the form of virtual machines, that are of different types. A cloud provider in the federation offers n types of virtual machines. The configuration of virtual machine of type j is given by:

v_j^{core} - Computational requirement in terms of number of cores

v_j^{mem} - Memory requirement

$v_j^{storage}$ - Storage Requirement.

Each cloud provider reserves some of it's total capacity for it's internal users and the remaining capacity is offered to the federation. The aggregate capacity offered by the cloud provider CP_i to the federation is given by:

CP_i^{core} - Computing capacity in terms of number of cores

CP_i^{mem} - Memory capacity

$CP_i^{storage}$ - Storage capacity.

Set of users or tenants of the federated cloud is represented with a set $U = \{U_1, U_2, \dots, \}$. A user sends a request to the broker regarding the number of VM instances it requires. The set of requests to the broker by all the tenants is represented with set $R = \{R_1, R_2 \dots\}$. A VM request from an user U_a is represented as a two tuple $R_a = (V, G)$. The first variable is a set $V = \{V_{a1}, V_{a2}, \dots V_{an}\}$ indicating the number of virtual machines user requires of each type. The second variable is a weighted graph indicating the communication patterns of the virtual machines of the request. Pr_j indicates the per hour price the user pays to the federation for acquiring a VM of type j.

The cost incurred to the cloud provider to provide a virtual machine v of R_a is given by

$$\alpha_v^{cost} = Pr_i^{electricity} * P_{ik}^v + \sum_{\forall \tilde{v} \in R_a} commcost(v, \tilde{v}) \tag{6.1}$$

Here, $Pr_i^{electricity}$ is the electricity price at cloud provider CP_i location and the cost incurred to the provider CP_i because of the virtual machine v communication with the remaining virtual machines of the user is given by $commcost(v, \tilde{v})$. P_{ik}^v is the power consumed by a physical machine which is used to host the virtual machine v.

A server in the data center consumes some fixed amount of power when it is on and any additional power consumption depends on its utilization. The utilization of a server depends on how many fractions of its resources are used by the virtual machines that are placed on it. So the power consumption of a physical server in a data center can be divided into two parts: first, a baseline static consumption and second, dynamic power consumption that is load dependent. Here the dynamic power consumption is based on VM power metering that is followed in [80]. The total power consumption of a server at a time t is given by

$$\mathbf{P}_{ik}(t) = \mathbf{P}_{\texttt{baseline}} + \sum_{\hat{v} \in \mathbb{V}_k^t} \mathbf{P}(\hat{v}) \tag{6.2}$$

where $\mathbf{P}_{\texttt{baseline}}$ is the base line power consumption, $\mathbb{V}_i^t$ gives the set of virtual machines that are placed on the physical machine P_{ik} at time t, $\mathbf{P}(\hat{v})$ is the power consumption of the virtual machine $\hat{v}$ on the physical machine P_{ik}.

If the power consumed by a VM is expanded further in terms of the utilization of the VM in cpu, memory and i/o as in [81], then the power consumption of a server is expressed as

$$\begin{aligned}\mathbf{P}_{ik}(t) = \alpha \cdot \sum_{\hat{v} \in \mathbb{V}_i^t} \mathbf{U}_{\texttt{cpu}}(\hat{v}) + \beta \cdot \sum_{\hat{v} \in \mathbb{V}_i^t} \mathbf{U}_{\texttt{mem}}(\hat{v}) \\ +\gamma \cdot \sum_{\hat{v} \in \mathbb{V}_i^t} \mathbf{U}_{\texttt{io}}(\hat{v}) + n_i \cdot e + \mathbf{P}_{\texttt{baseline}}\end{aligned} \tag{6.3}$$

where α, β, γ are the weight-ages for utilization of the server by the VM in CPU, memory and IO respectively.

The profit a cloud provider i gets if he participates in the federation by offering M_i virtual machine instances to serve a request is given by the following equation.

$$Profit(M_i) = \sum_{j=1}^{n} M_{ij} * Price(Pr_j) - \alpha_{ij}^{cost} \tag{6.4}$$

6.2 Energy efficient Resource Allocation Game in Federated Cloud

In this section, we introduce Energy efficient resource allocation game, a coalition game that can represent cloud federations, and allows us to understand and evaluate the different federation structures. Energy efficient resource allocation in federated cloud is modelled as a coalition game $I(CP, v)$ with transferable utility, where each cloud provider in CP is a player in the game and v is the characteristic function, defined on $F \subseteq CP$. The characteristic function values are calculated for each request in R.

6.2.1 Characteristic Function

The characteristic function is the profit obtained when the cloud providers of federation $F \subseteq CP$ co-operate as a coalition. It is a real valued function which maps each coalition to a real value and $v(\phi) = 0$. The characteristic function is given as follows.

$$\text{Maximize} \sum_{\mathcal{CP}_i \in \mathcal{F}} \sum_{j=1}^{n} X_{ij} * Price(Pr_j) - \alpha_{ij}^{cost}, \tag{6.5}$$

subject to:

$$\sum_{j=1}^{n} c_{ij}^{core} X_{ij} \leq CP_i^{core}, \quad (\forall \mathcal{CP}_i \in \mathcal{F}), \tag{6.6}$$

$$\sum_{j=1}^{n} c_{il}^{mem} X_{ij} \leq CP_i^{mem}, \quad (\forall \mathcal{CP}_i \in \mathcal{F}), \tag{6.7}$$

$$\sum_{j=1}^{n} c_{il}^{storage} X_{ij} \leq CP_i^{storage}, \quad (\forall \mathcal{CP}_i \in \mathcal{F}), \tag{6.8}$$

$$\sum_{\mathcal{C}_i \in \mathcal{F}} M_{ij} = V_{aj}, \quad (\forall j = 1, \ldots, n), \tag{6.9}$$

$$X_{ij} \geq 0, \text{and is integer}$$
$$(\forall \mathcal{CP}_i \in \mathcal{F} \text{ and } \forall j = 1, \dots, n). \tag{6.10}$$

Here the value of a federation is the objective function value it achieves. The objective is the profit obtained by the federation for servicing the request. Equations 6.6, 6.7 and 6.8 are the capacity constraints and equation 6.9 indicate that the resources offered by all the cloud providers should be equal to the ones requested by the user.

Any co-operative game should satisfy two properties, fairness and stability.

6.2.2 Fairness

The profit a federation achieves should be distributed fairly to the cloud providers participating in the federation. As the cost incurred to the cloud provider for offering virtual machines also depends on the communication between those virtual machines to the other virtual machines of the user, the order of joining the federation matters in calculating the value of a provider to the federation. Hence the suitable index for fairly dividing the profit each federation acquires to the cloud providers is shapley value [103]. Shapley value gives the expected marginal contribution of a player to a coalition. It is the average of the marginal contributions over all orders in which the players could join the complete coalition. The shapley value of a cloud provider in a federation is defined as follows [104]:

$$\phi_i(N) = \frac{1}{|N|!} \sum_R \left(v(P_i^R \cup \{i\}) - v(P_i^R) \right) \ , \tag{6.11}$$

If it is expanded further, it can be written as:

$$\phi_i(N) = \frac{1}{|N|!} \sum_{S \subseteq N \setminus \{i\}} |S|!(|N| - |S| - 1)! \left(v(S \cup \{i\}) - v(S)\right) \tag{6.12}$$

where n is the total number of cloud providers and the sum extends over all subsets S of N not containing the provider CP_i.

6.2.3 Stability

Definition 6.2.1. *A Coalitional game (N, v) with transferable payoff is cohesive if*

$$v(N) \geq \sum_{k=1}^{K} v(S_k), \textit{for every partition } S_k \textit{ of } N \tag{6.13}$$

Another definition for a cohesive coalitional game is as follows.

Definition 6.2.2. *A coalitional game is cohesive[105] if, for every partition $S_1, ..., S_k$ of the set of all players and every combination $(a_{S_1}, ..., a_{S_k})$ of actions, one for every coalition in the partition, the grand coalition N has an action that is at least as desirable for every player i as the action a_{S_j} of the member S_j of the partition to the player i belongs.*

Energy Aware resource allocation game modeled in the above section is not cohesive. This can be illustrated with an example.

Let 3 cloud providers $CP1$, $CP2$, $CP3$ are formed as a federation. The federation offers four types of the virtual machines namely small, medium, large and extra large. The configurations of the virtual machines of different types is listed in the Table 6.1. The capacities of the cloud providers are same and given by the Table 6.2. The cost incurred to each cloud provider to offer a type of virtual machine is same and it is given as $\{0.5, 0.8, 1.5, 2.0\}$. The prices of virtual machine of different types are given as $\{1.2, 1.5, 2.2, 3\}$. Now a user requests 3 virtual machines whose types are small, medium, and large respectively. The price user pays to the federation is 4.9. Then the values of all the possible coalitions are $v(CP_1) = 0$, $v(CP_2) = 0$, $v(CP_3) = 0$, $v(CP_1, CP_2) = 1.1$, $v(CP_1, CP_3) = 1.1$, $v(CP_2, CP_3) = 1.1$, $v(CP_1, CP_2, CP_3) = 1.1$. Here, the value that a player gets in coalitions of size 2 can not be achieved by the grand coalition. So the grand coalition (CP_1, CP_2, CP_3) does not achieve outcome that is at least as desirable for every players as that of smaller coalitions. Hence it is not cohesive.

Since the game is not cohesive, the grand coalition may not be the optimal coalition for this game. The game doesn't need to converge to the grand coalition.

The core of a coalitional game, which indicate stability of a coalitional game is defined as follows.

Table 6.1: Federated Cloud: VM Types and Requirements

VM Type	Cores	Main Memory	Secondary Memory
Small	1	1.5 GB	0.10 TB
Medium	2	3 GB	0.40 TB
Large	4	5 GB	1 TB
Extra Large	8	10 GB	2 TB

Table 6.2: Federated Cloud: Cloud Provider Capacities

Cloud Provider	Cores	Main Memory	Secondary Memory
CP_i	4	8GB	1 TB

Definition 6.2.3. *A payoff vector x is in the core of a coalitional game (N,v) if and only if*

$$\forall S \subset N, \sum_{i \in S} x_i \geq v(S) \tag{6.14}$$

The core of a coalition game is the set of actions a_N of the grand coalition N that are not blocked by any coalition. If a coalition S has an action that all its members prefer to an action a_N of the grand coalition, we say that S blocks a_N. In the resource allocation game proposed above, the actions of the grand coalition can be blocked by smaller coalitions actions since it is non-cohesive. So the core of this game can be empty.

6.3 Cloud Federation Formation

As the proposed game is not cohesive, the grand coalition may not form. Hence, independent and disjoint federations would form. In this section, a mechanism for cloud federation formation is presented. Here we inspect different coalition structures possible in this game as the grand coalition may not be the desired coalition. Coalition structure is defined in the following way.

Definition 6.3.1. *A coalition structure over N is defined as a collection of coalitions $CS = \{S_1, S_2, \ldots S_k\}$ such that $\bigcup_{i=1}^{k} S_i = N$ and $S_i \cap S_j = \phi$ for any $i, j \in \{1, 2, \ldots, k\} : i \neq j$.*

Coalition structure is the partitioning the set of total agents into mutually disjoint sets. The value of a coalition structure is the sum of the values of all of it's coalitions. Formally, the set of all possible coalition structures are defined with a set Π^N and the value of a $CS \in \Pi^N$ is defined as

$$V(CS) = \sum_{i}^{k} V(S_i) \tag{6.15}$$

The coalition structure generation problem is finding the coalition structure whose value is maximal. More formally, it is finding a coalition structure $CS^* \in$ arg $\max_{CS \in \Pi^N} V(CS)$. But finding an optimal coalition structure is NP-Complete. Total number of coalition structures possible is equivalent to n^{th} bell number B_n, which is a very large number. It is not possible to do exhaustive search on all the coalition structures possible. Here a mechanism for generating an optimal coalition structure is presented.

Given a set of cloud providers $N = \{CP_1, CP_2, \ldots CP_m\}$, a set of requests $R = \{r_1, r_2, \ldots r_n\}$, and a characteristic function value $V(C, r)$ for assigning request r to coalition $C \subseteq N$, then we need to find a set of coalitions $\{C_1, C_2, \ldots C_k\}$ that maximizes $\sum_{i=1}^{k} V(C_i, r_i)$ where $C_i \subseteq N$, $S_i \cap S_j = \phi$ for any $i, j \in \{1, 2, \ldots, k\} : i \neq j$ and $\bigcup_{i=1}^{k} C_i = N$.

As a solution to the problem, we present a branch and bound based algorithm here. The steps involved in the solution are

- Partitioning of the entire search space into sub spaces: To remove the parts of the solution space which contain sub-optimal solutions, the entire search space is divided into sub spaces based on integer partitioning of a number
- Finding bounds for each subspace: For each subspace, upper bound and lower bounds are calculated.
- Searching for optimal solution: A sequential search is done on the partitions of the solution search space using branch bound technique.

6.3.1 Partitioning of the Search Space

The partitioning of the search space is done based on "integer partitioning of a number". Integer partition of a number $l \in \mathbb{N}$ is defined as a way to write the number l as sum of two integers. Now, given a set of cloud providers $N = \{CP_1, CP_2, \ldots CP_m\}$, and a set of requests R = $\{r_1, r_2, \ldots r_n\}$, the following steps are used to divide the search space into subspaces.

- First, make sets out of all the possible distinct integer partitions of the number $|N| = m$ whose addends are less than or equal to $|R| + 1$. For example, if the number of cloud providers are 5, and the requests are 3, then the possible integer partition sets are {4, 1}, {3, 2} , {3, 1, 1}, and {2, 1, 1, 1}.
- Add zeroes to each set to make it's size equivalent to $|R| + 1$. The sets generated for the example in the previous step, would become like this: {4, 1, 0, 0}, {3, 2, 0, 0} , {3, 1, 1, 0}, and {2, 1, 1, 1}
- Generate all the permutations of the multisets created in the step 2. Each such permutation is considered as a subspace. For example for the multiset {2, 1, 1, 1}, one of the possible permutation is {1, 2, 1, 1}. This is considered as a representative of one of the subspaces. The permutation {2, 1, 1, 1} corresponds to assigning 2 cloud providers to request 1, 1 cloud provider to request 2, and 1 cloud provider to request 3 and 1 cloud provider is not servicing any request. This subspace contains all the different the federation structures that has 2 cloud providers in the first federation, one provider in each of the second, third and fourth federations. For example, if there are 8 cloud providers, {[2, 4], [5], [1], [7]} is one coalition structure that is part of the {2, 1, 1, 1} subspace. {[3, 5], [4], [6], [2]} is another coalition structure that is part of the same subspace.

6.3.2 Finding Bounds for Each Subspace

Once the subspaces are formed based on the integer partioning of the number of cloud providers, bounds for each sub space is calculated as follows. To calculate bounds for each subspace, let $A_p = (X \subset A : |X| = p)$ and define two functions M and Avg as:

- $M(p, r) = \text{MAX}\{V(C, r) : C \in A_p\}$
- $\text{Avg}(p, r) = \dfrac{\sum\{V(C, r) : C \in A_p\}}{|A_p|}$

Given those two functions, the bounds for the subspace represented by a multiset permutation $P =< p_1, p_2, \ldots, P_n >$ can be calculated as follows.

- Upper Bound for the partition represented by P is given by $U_P = \sum_{i=1}^{n} M(p_i, r_i)$
- Lower bound for the partition represented by P is given by $L_P = \sum_{i=1}^{n} \text{Avg}(p_i, r_i)$

6.3.3 Searching for Optimal Solution

The entire solution space is searched by expanding one subspace after another in an ordered manner. The order of expansion depends on two simple rules. For any two subspaces P_1 and P_2, the subspace with greater upper bound is expanded first, i.e, P_1 is expanded first if $U_{P_1} > U_{P_2}$. If both the subspaces upper bounds are same, then the subspace with greater lower bound is expanded first, i.e, P_1 is expanded first if $L_{P_1} > L_{P_2}$. Using these two rules order of precedence among the subspaces is established. Based on this order each partition is searched one by one. After expanding one subspace and finding a solution, all the remaining subspace whose upper bound is less than the new solution value would be completely discarded. Only the subspaces whose upper bounds are more than the newly found value would be considered for expanding next.

6.4 Performance Evaluation

In this section, results of performance evaluation of the proposed solution is presented.

In the simulation set up, we have considered 8 independent cloud providers to participate in the federation. The capacities of these cloud providers in terms of number of servers is generated randomly. Each cloud provider offers 11 types of virtual machines. The configurations of these virtual machines are the standard values taken from Amazon EC2 instances. The power consumption model considered for the servers is the VM power consumption model. The peak power consumption value is taken at 185 kWh, which is a standard value for a typical data center server. Standard values from amazon EC2 instances are taken as the prices of each virtual machine type. As an input, we generated 32 requests from multiple tenants where each request contain many requests for all types of virtual

Algorithm 6.4 Optimal Federation Structure Generation

```
Input: $\mathbb{A}, R, V[C, R]$
Output: $\mathbb{CS}^*$                                  ▷ Optimal federation structure
 1: $SP = \phi$;
 2: $L$ = Set of all possible integer partitioning of the number $|A|$ where the number of
    addends is less than $|R| + 1$
 3: for each set $L_i \in L$ do
 4:     if $|L_i| < |R| + 1$ then
 5:         Add zeroes to the set $L_i$ till $|L_i| = |R| + 1$;
 6:     end if
 7:     $SP = SP \cup Multiset - Permutations(L_i)$;
 8: end for
 9: for $p = 1$ to $m$ do
10:     $A_p =$ Set of federations whose size is equal to $p$
11:     for $j = 1$ to $n$ do
12:         $M[p, r_j]$ = max$\{V(C, r_j), C \in A_p\}$
13:         Avg$[p, r_j]$ = Avg$\{V(C, r_j), C \in A_p\}$
14:     end for
15: end for
16: for each permutation $P_j \in SP$ do
17:     Calculate Upper Bound of it's subspace as $U_{P_j} = \sum_{i=1}^{n} M(p_i, r_i)$;
18:     Calculate Lower Bound of it's subspace as $L_{P_j} = \sum_{i=1}^{n}$ Avg$(p_i, r_i)$;
19: end for
20: $UB^* = Max_{\{P_j \in SP\}}(U_{P_j})$;
21: $LB^* = Max_{\{P_j \in SP\}}(L_{P_j})$;
22: $SP^{'}$= Prune(SP, $LB^*$);
23: $CS^*$=SearchSpace($SP^{'}$);
24: return $CS^*$;
```

Algorithm 6.5 Prune

```
Input: $SP, LB^*$
Output: $SP^{'}$                                  ▷ Reduced Search Space
1: $SP^{'} = SP$
2: for $p \in SP^{'}$ do
3:     if $UB_p < LB^*$ then
4:         $SP^{'} = SP^{'} \backslash P_p$;
5:     end if
6: end for
```

Algorithm 6.6 Search-Space($SP^{'}$)

```
Input: $SP^{'}$
Output: $OS$
1: Give ordering among the subspaces in $SP^{'}$ using their upper bounds and lower bounds;
2: Pick a subspace $P_k$ to expand based on that ordering
3: $SP^{'} = SP^{'} \backslash P_k$;    ▷ Remove that subspace from the list of subsapces that need to be explored
4: Calculate the optimal value in the search space $P_k$ and call it as $OP^{'}$
5: if desired level of optimality is achieved then
6:     return $OP^{'}$;
7: end if
8: Prune($SP^{'}$, $OP^{'}$);
9: Search-Space($SP^{'}$)
```

machines. The requests are generated such that no provider is able to service all the requests alone. The proposed federation mechanism is compared with two variations of best fit greedy approach, one that finds the federation for each incoming request immediately and the other one that waits for a small time to accumulate minimum number of requests to take decisions.

We have considered the following six parameters to evaluate the proposed solution.
Total profit: It is the sum of all the individual profits of the cloud providers in the federation. It indicates the social welfare of the federation.
Request service rate: It is the ratio of the requests serviced to the total requests arrived at the federation broker
Overall utilization: The percentage of capacity utilized for servicing the requests over all the total available resources of the federation.
Individual profit: The profit obtained by the individual cloud provider. Better individual profits indicate higher contribution in terms of resources in cloud federations to service the requests.
Average federation size: Average federation size for servicing the requests indicate the degree of participation by the cloud providers in federation formation for servicing the requests.
VM Type utilization: It indicates how much percentage of each type of VM resources is utilized out of the available resources.

6.4.1 An analysis of the Results

In Figure.6.1 the total profit obtained by the three approaches is depicted. As it can be observed from the graph, that the proposed solution achieves more than double the profits when compared with the two variations of the greedy best fit approach. The proposed solution is achieving very high social welfare as it finds the federation that is good globally for each request.

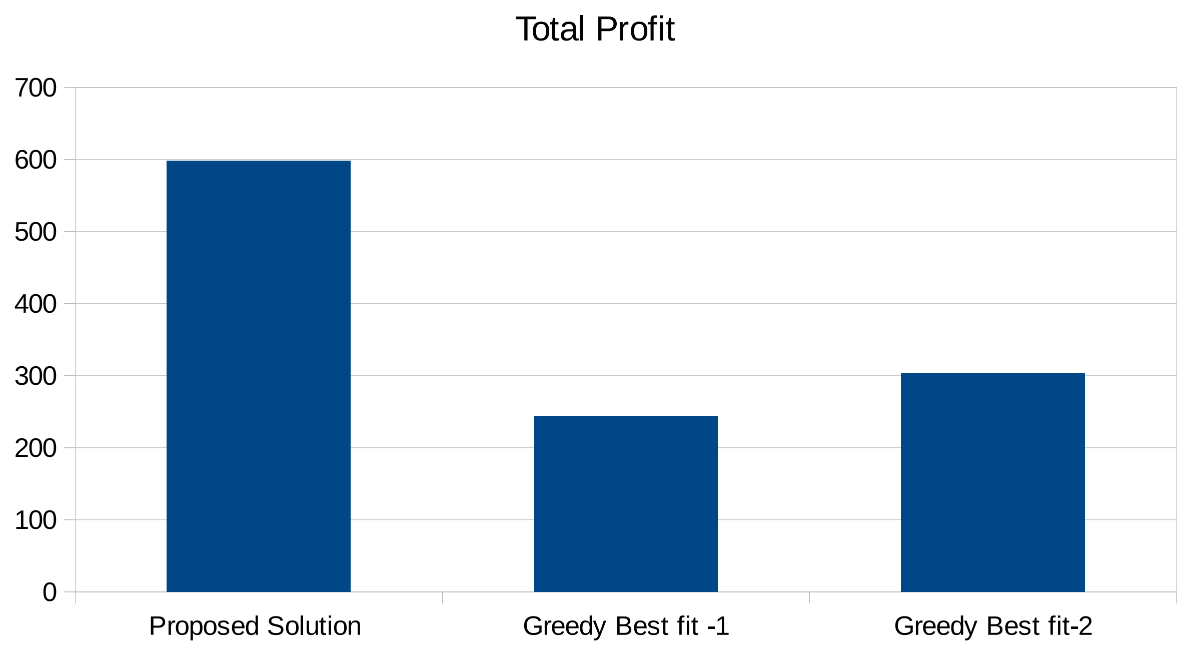


Figure 6.1: Resource Sharing in Federated Cloud: Total profit

In Figure 6.2 we plot the percentage of requests serviced by the three approaches considered in the performance evaluation. The proposed solution is servicing 90% of the requests that come to the federation and the request service percentage for greedy best fit-1 is 59 % and greedy best fit-2 is 68%. It is evident that the greedy best fit-2 is servicing more requests than the greedy first fit-1 as it stalls for a small time epoch to collect minimum number of requests, however it is achieving significantly less percentage when compared with the proposed solution.

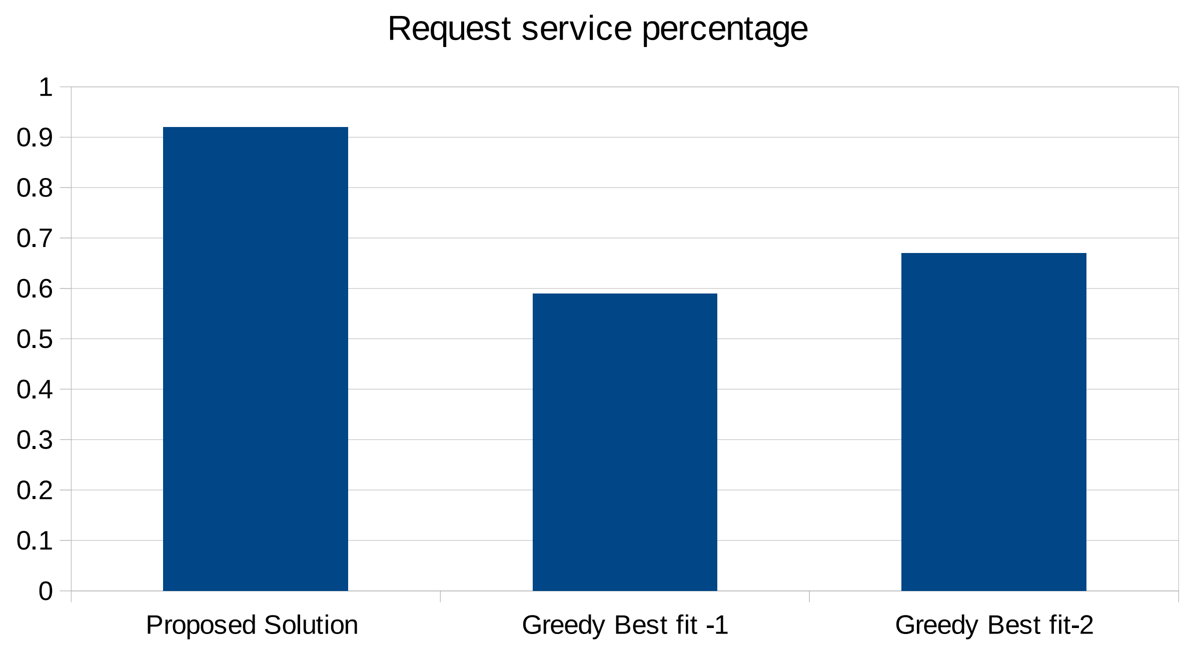


Figure 6.2: Resource Sharing in Federated Cloud: Request service percentage

Figure 6.3 gives the overall utilization of the entire federation. It indicates how optimally the available federation resources are utilised and as it can be observed from the graph that overall utilization is very low for greedy best fit-1 and greedy best fit-2 at 0.45 and 0.5 respectively. The proposed solution has achieved 0.84 overall resource utilization as it serves more requests by selecting the federations optimally.

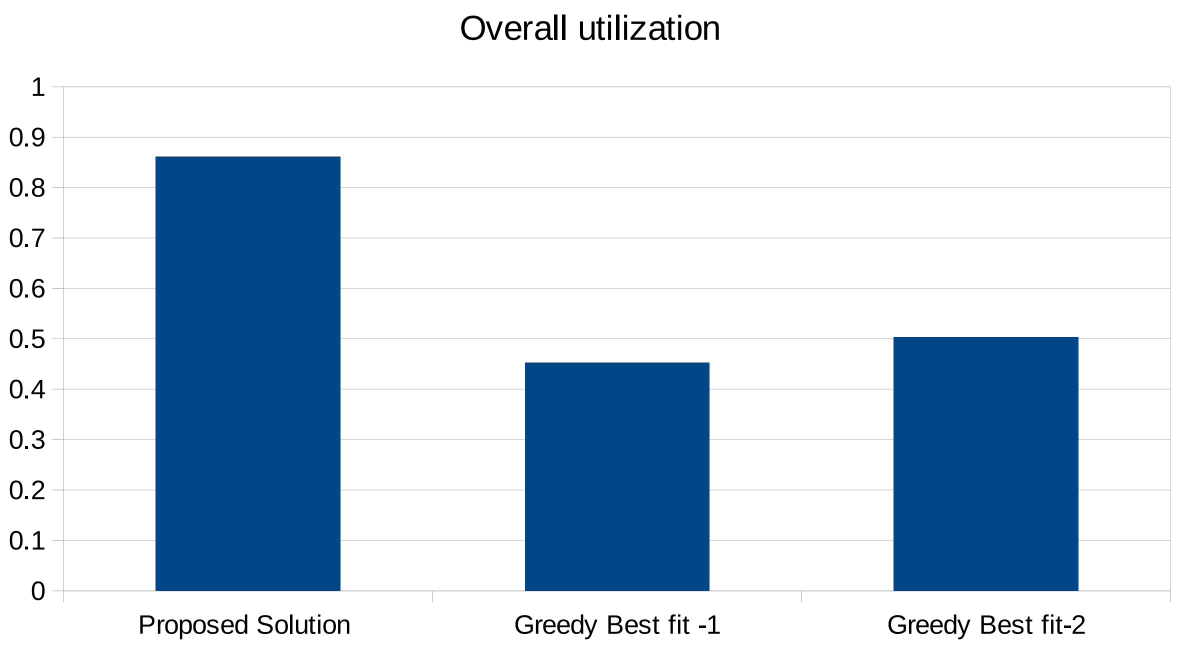


Figure 6.3: Resource Sharing in Federated Cloud: Overall Utilization

In Figure 6.4, the average federation size for servicing the requests for the three approaches is plotted. The average federation size is 2.2 for the proposed solution and the average sizes 1.6 and 1.75 for greedy first fit-1 and greedy first fit-2 respectively. Higher average federation size indicates higher participation by the cloud providers in federation formation for servicing the requests.

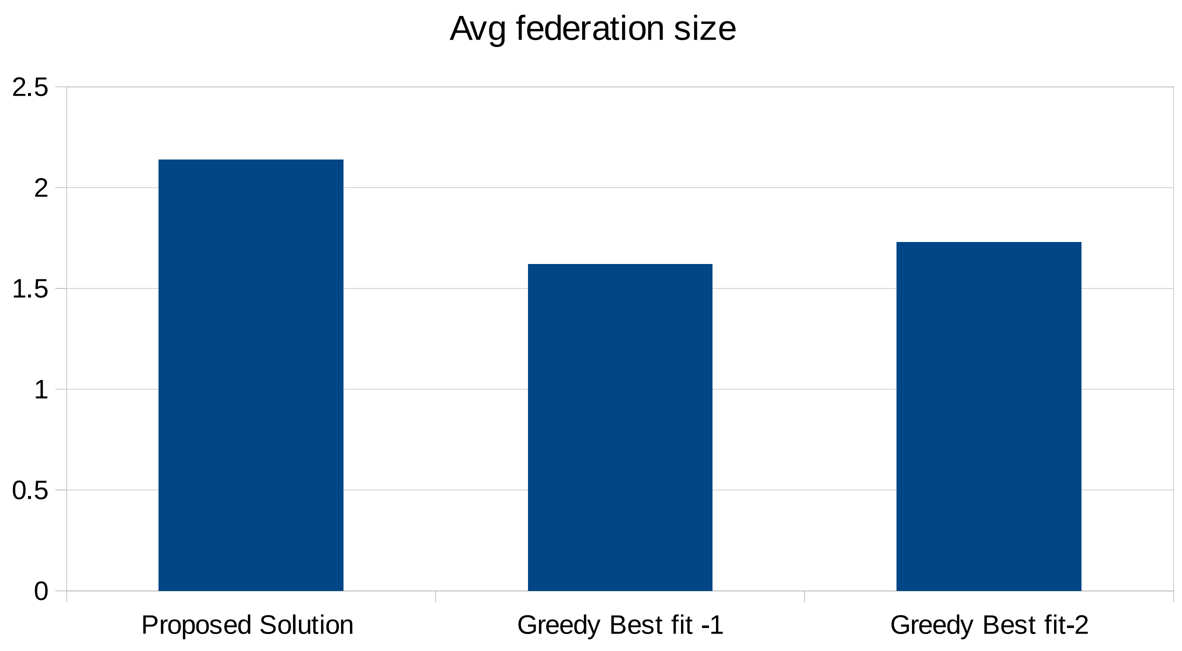


Figure 6.4: Resource Sharing in Federated Cloud: Average Federation Size

Individual profits of the 8 cloud providers considered are plotted in Figure 6.5. The individual profits obtained by all the 8 cloud providers except CP3 using the proposed solution are higher than the profits obtained by the other two approaches. Best fit-1 yielded better profits for CP3. However, for CP5 and CP7 the profit obtained using best fit-1 is zero, i.e., the best fit-1 approach has not utilized any of the resource of CP5 and CP7 to service the requests.

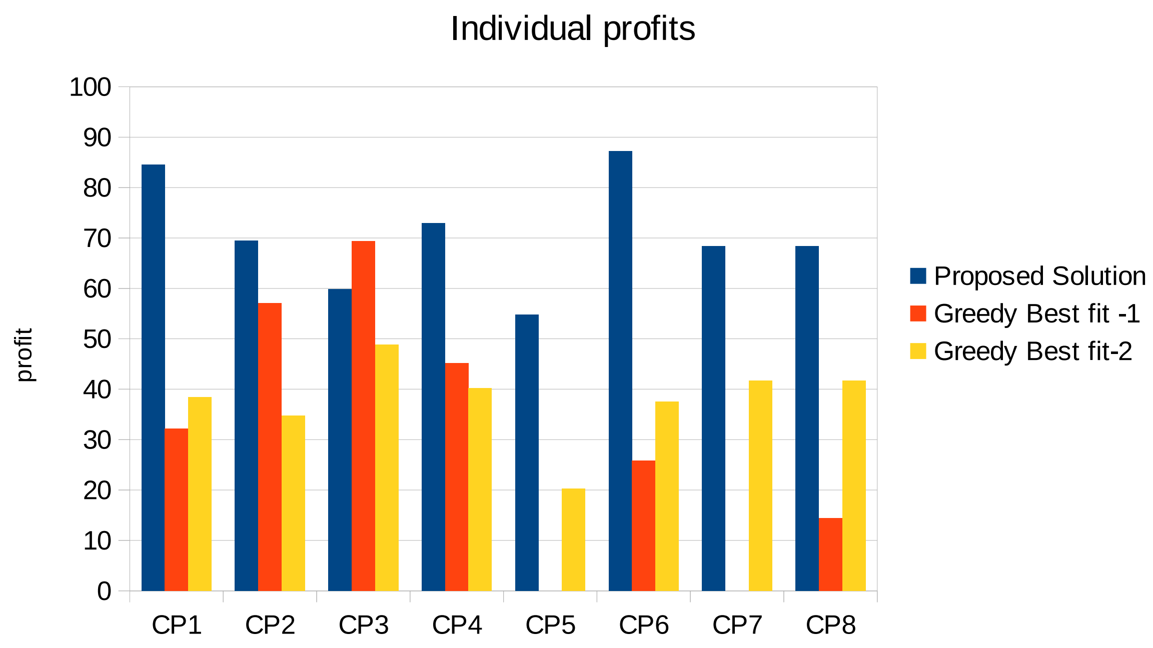


Figure 6.5: Resource Sharing in Federated Cloud: Individual Profit

We have also plotted utilization of the available virtual machines for each of their types in Figure.6.6. All the types of virtual machines are utilized at higher percentages using the proposed approach. Best fit-1 and Best fit-2 yielded less percentages due to in-effective formation of the federations for servicing the requests.

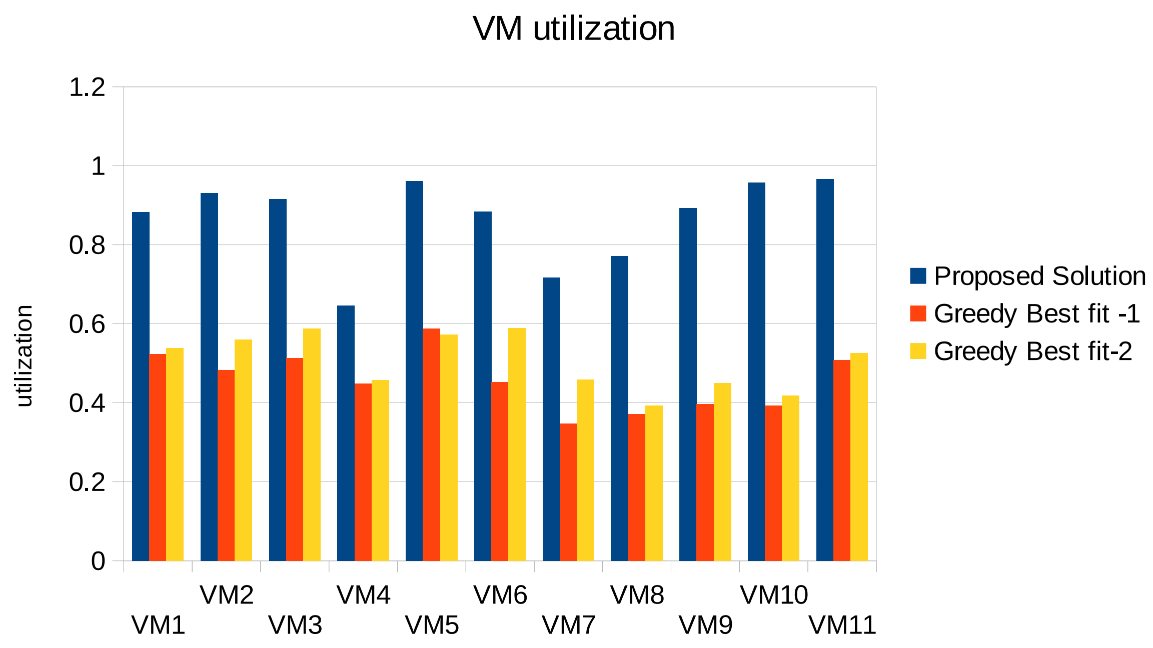


Figure 6.6: Resource Sharing in Federated Cloud: VM Utilization

6.5 Summary

In this chapter, we proposed a mechanism for enhancing the way the resources are pooled and shared in a federated cloud to service multiple virtual machine requests from the tenants. The federation formation mechanism to service requests from multiple tenants is characterized by a co-operative game. The game fairness is achieved through shapely value and it is shown that the game is not cohesive, hence grand coalition may not form. So to find out the optimal federation structure, a novel federation formation mechanism that is based upon integer partitioning of the number of cloud providers is proposed. Simulation results demonstrated that the proposed solution outperforms greedy best fit approach in terms of total profit, individual profit, total utilization, individual utilization and utilization per VM type.

Chapter 7

Conclusions and Future Research

Cloud Computing has revolutionized the IT industry with its pay-per-use model of service, ability to allocate the resources as and when the users require and alleviating them from the operational issues of the resources. It offers a cost effective and highly scalable approach for running the applications. The popularity of the cloud is ever increasing resulting in huge number of customers deploying their workloads, data on to the cloud instead of the local resources. Due to the growing popularity for cloud services expansive cloud data centers comprising vast number of servers and storage devices are being built around the world by cloud providers. However, these cloud data centers consume a massive amount of energy. Reduction of energy consumed by the data centers is a crucial problem with respect to energy costs and environmental concerns for the cloud providers. So to address this key challenge of reducing the energy consumed by cloud data centers, we have defined four objectives in chapter 1.

In chapter 2, we presented a detailed review of the literature related to energy efficient resource management. The literature review helped us in identifying research gaps, identifying the objectives and finding suitable models and solutions for the objectives.

Chapter 3 focused on offline version of the problem of joint optimization of energy consumed by server and network elements with intelligent VM and flow consolidation. First we modelled the joint server and network optimization problem as a mixed integer programming problem and then presented a two-phase solution where each phase contains a ant colony based meta-heuristic solution. A heuristic called affinity quantifying the benefit we get in terms of energy saving is proposed as part of the solution. It is demonstrated through the simulation results that the proposed solution achieves 17 % and 23 % energy savings when compared with first-fit decreasing heuristic and round robin solutions. The proposed solution is shown to be scalable with respect to number of applications that need to be deployed on the data center and it performs well for different types of data center networks, although it is better suited for 3-tier data center network among others. The proposed solution is suitable to the scenarios where several tenants send requests for reserving a set of virtual machines in advance to deploy their delay-sensitive applications.

In chapter 4, we addressed the online version of joint optimization of server and network elements energy in cloud computing, where a set of virtual machines are requested on the go and they are not reserved. The problem is modelled as an mixed integer programming problem and two algorithms called best-fit-spectral and least-increase-spectral are proposed as solutions to the problem. In both of the solutions, spectral graph partitioning is applied on the communication graph of the set of virtual machines requested from a tenant so that groups of virtual machines with high communication are placed on the physical machines recursively. The simulation results presented in this chapter, show that the proposed solutions achieve significant energy savings when compared against first-fit heuristic and are scalable with respect to both the size of the data center and the load on the data center.

Virtual machine migration is used to achieve opportunistic energy savings by server consolidation. Scatter-gather migration is a new type of migration that aims at reducing the eviction time of the migration. In chapter 5, we have defined a new problem called intermediate node selection problem for energy efficient implementation of scatter-gather migration. The hardness of the problem is discussed. We have also modelled the problem with respect to minimizing the eviction time. Then as solutions two greedy solutions called max-decrease-in-eviction and least-increase-in-energy are proposed for minimizing the eviction time and energy respectively. Through simulation results it is shown that the proposed solutions perform better when compared against solutions like random selection, Farthest Node First and Closest Node Next policies.

In chapter 6, we addressed the problem of energy efficient resource allocation in federated cloud. We formulated the federation formation for resource sharing to service the VM requests as a co-operative game and the properties of fairness and stability of the modelled game are discussed. Then, we designed a novel federation formation mechanism that is based upon integer partitioning of the number of cloud providers. The proposed solution is compared against two variations of best-fit greedy approach and the results showed that our solution achieves better performance in achieving total profit, individual profit, total utilization, average federation size and VM utilization.

7.1 Future Directions

As a future work, the research is planned in the following directions.

First, it is planned to extend the work done for the first objective by applying over commitment approach to it. In VM over commitment, the virtual machines are placed on to a PM such that cumulative requirements of the VMs placed on the PM is more than its capacity. It is based on the practical observation that the actual load of any VM would be less than what it is reserved for, and with a notion that peak loads of all VMs placed on the PM will not be at the same time.

Second, the ideas of the first work and the second work would be applied for energy efficient VM placement and communication routing in software defined data center network(SD-DCN). In SD-DCN, the control plane and data plane are separated and hence it facilitates better VM flow consolidation, resulting in improved network energy savings.

Third, it is planned to extend the work for the third objective to minimize the excess bandwidth allocation for the Scatter-Gather migration when compared with normal migration.

Fourth, we wish to model the problem of intermediate node selection for gang Scatter-Gather migration and propose heuristics to optimize the objectives energy, eviction time and excess bandwidth.

The memory contents of the migrated VM are transferred to many intermediates and then from the intermediates contents are fetched to the destination. A possible extension of the work is to propose a routing mechanism for Scatter-Gather migration that reduces the energy consumed by the networking elements while maintaining a reasonable total migration time.

Bibliography

[1] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. *National Institute of Science and Technology, Special Publication, 800*, 145, 2011.

[2] J Kaplan, W Forrest, and N Kindler. Revolutionizing Data Center Energy Efficiency. Technical report, Mckinsey and Company, July 2008.

[3] Arman Shehabi, Sarah Josephine Smith, Dale A. Sartor, Richard E. Brown, Magnus Herrlin, Jonathan G. Koomey, Eric R. Masanet, Nathaniel Horner, Inês Lima Azevedo, and William Lintner. United states data center energy usage report. Technical report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 06/2016 2016.

[4] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, Dec 2007.

[5] Liliana Grigoriu and Dirk Briskorn. Scheduling jobs and maintenance activities subject to job-dependent machine deteriorations. *Journal of Scheduling*, 20(2):183–197, Apr 2017.

[6] Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. Energy-efficient resource allocation and provisioning framework for cloud data centers. *IEEE Transactions on Network and Service Management*, 12(3):377–391, 2015.

[7] Daniel Versick and Djamshid Tavangarian. Reducing energy consumption by load aggregation with an optimized dynamic live migration of virtual machines. In *2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 164–170. IEEE, 2010.

[8] Dong Jiankang, Wang Hongbo, Li Yangyang, and Cheng Shiduan. Virtual machine scheduling for improving energy efciency in iaas cloud. *China communications*, 11(3):1–12, 2014.

[9] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, 2005.

[10] Michael Nelson, Beng-Hong Lim, Greg Hutchins, et al. Fast transparent migration for virtual machines. In *USENIX Annual technical conference, general track*, pages 391–394, 2005.

[11] Michael R Hines, Umesh Deshpande, and Kartik Gopalan. Post-copy live migration of virtual machines. *ACM SIGOPS operating systems review*, 43(3):14–26, 2009.

[12] Takahiro Hirofuchi and Isaku Yamahata. Yabusame: Postcopy live migration for qemu/kvm. In *KVM Forum*, volume 2011, 2011.

[13] U. Deshpande, D. Chan, S. Chan, K. Gopalan, and N. Bila. Scatter-gather live migration of virtual machines. *IEEE Transactions on Cloud Computing*, 6(1):196–208, Jan 2018.

[14] Kashif Bilal, Samee Ullah Khan, Joanna Kolodziej, Limin Zhang, Khizar Hayat, Sajjad Ahmad Madani, Nasro Min-Allah, Lizhe Wang, and Dan Chen. A comparative study of data center network architectures. In *ECMS 2012 Proceedings edited by: K. G. Troitzsch, M. Moehring, U. Lotzmann*, pages 526–532, 2012. Exported from https://app.dimensions.ai on 2018/07/17.

[15] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. Bcube: a high performance, server-centric network architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 63–74, 2009.

[16] Dong Lin, Yang Liu, Mounir Hamdi, and Jogesh Muppala. Hyper-bcube: A scalable data center network. In *2012 IEEE International Conference on Communications (ICC)*, pages 2918–2923. IEEE, 2012.

[17] Lin Wang, Fa Zhang, Jordi Arjona Aroca, Athanasios V Vasilakos, Kai Zheng, Chenying Hou, Dan Li, and Zhiyong Liu. Greendcn: A general framework for achieving energy efficiency in data center networks. *IEEE Journal on Selected Areas in Communications*, 32(1):4–15, 2013.

[18] C. Lefurgy, X. Wang, and M. Ware. Server-level power control. In *Fourth International Conference on Autonomic Computing (ICAC'07)*, June 2007.

[19] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[20] X Fan, W D Weber, and L A Barroso. Power provisioning for a warehouse-sized computer. *in Proc. of the 34th Annual International Symposium on Computer Architecture*, 2007.

[21] Etienne Le Sueur and Gernot Heiser. Dynamic voltage and frequency scaling: The laws of diminishing returns. In *Proceedings of the 2010 international conference on Power aware computing and systems*, pages 1–8, 2010.

[22] D Suleiman, M Ibrahim, and I Hamarash. Dynamic voltage frequency scaling (dvfs) for microprocessors power and energy reduction. In *4th International Conference on Electrical and Electronics Engineering*, volume 12, 2005.

[23] Howard David, Chris Fallin, Eugene Gorbatov, Ulf R. Hanebutte, and Onur Mutlu. Memory power management via dynamic voltage/frequency scaling. In *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ICAC '11, page 31–40, New York, NY, USA, 2011. Association for Computing Machinery.

[24] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proceedings of the 2007 international symposium on Low power electronics and design (ISLPED '07)*, pages 38–43, Aug 2007.

[25] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. Server-level power control. In *Fourth International Conference on Autonomic Computing (ICAC'07)*. IEEE, 2007.

[26] Tibor Horvath, Tarek Abdelzaher, Kevin Skadron, and Xue Liu. Dynamic voltage scaling in multitier web servers with end-to-end delay control. *IEEE Transactions on Computers*, 56(4):444–458, 2007.

[27] Rodrigo N Calheiros and Rajkumar Buyya. Energy-efficient scheduling of urgent bag-of-tasks applications in clouds through dvfs. In *2014 IEEE 6th international conference on cloud computing technology and science*, pages 342–349. IEEE, 2014.

[28] M. A. Oxley, S. Pasricha, A. A. Maciejewski, H. J. Siegel, J. Apodaca, D. Young, L. Briceño, J. Smith, S. Bahirat, B. Khemka, A. Ramirez, and Y. Zou. Makespan and energy robust stochastic static resource allocation of a bag-of-tasks to a heterogeneous computing system. *IEEE Transactions on Parallel and Distributed Systems*, 26(10):2791–2805, Oct 2015.

[29] Wei Wang Wanneng Shu and Yunji Wang. A novel energy efficient resource allocation algorithm based on immune clonal optimization for green cloud computing. *Journal on Wireless Communication and Networking*, 2014.

[30] A. Alahmadi, D. Che, M. Khaleel, M. M. Zhu, and P. Ghodous. An innovative energy-aware cloud task scheduling framework. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 493–500, June 2015.

[31] Qingyuan Deng, David Meisner, Abhishek Bhattacharjee, Thomas F. Wenisch, and Ricardo Bianchini. Coscale: Coordinating cpu and memory system dvfs in server systems. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-45, pages 143–154, Washington, DC, USA, 2012. IEEE Computer Society.

[32] S. Wang, Z. Qian, J. Yuan, and I. You. A dvfs based energy-efficient tasks scheduling in a data center. *IEEE Access*, 5:13090–13102, 2017.

[33] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu. Dynamic voltage scaling in multitier web servers with end-to-end delay control. *IEEE Transactions on Computers*, 56(4):444–458, April 2007.

[34] L. Li, J. Dong, D. Zuo, and J. Wu. Sla-aware and energy-efficient vm consolidation in cloud data centers using robust linear regression prediction model. *IEEE Access*, 7:9490–9500, 2019.

[35] F. Farahnakian, A. Ashraf, P. Liljeberg, T. Pahikkala, J. Plosila, I. Porres, and H. Tenhunen. Energy-aware dynamic vm consolidation in cloud data centers using ant colony system. In *2014 IEEE 7th International Conference on Cloud Computing*, pages 104–111, 2014.

[36] Md Hasanul Ferdaus, Manzur Murshed, Rodrigo N. Calheiros, and Rajkumar Buyya. Virtual machine consolidation in cloud data centers using aco metaheuristic. In Fernando Silva, Inês Dutra, and Vítor Santos Costa, editors, *Euro-Par 2014 Parallel Processing*, pages 306–317, Cham, 2014. Springer International Publishing.

[37] Nguyen Quang-Hung, Nam Thoai, and Nguyen Thanh Son. Epobf: energy efficient allocation of virtual machines in high performance computing cloud. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XVI*, pages 71–86. Springer, 2014.

[38] Nilesh Pachorkar and Rajesh Ingle. Multi-dimensional affinity aware vm placement algorithm in cloud computing. *International Journal of Advanced Computer Research*, 3(4):121, 2013.

[39] Wanneng Shu, Wei Wang, and Yunji Wang. A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):1–9, 2014.

[40] Parvathy S Pillai and Shrisha Rao. Resource allocation in cloud computing using the uncertainty principle of game theory. *IEEE Systems Journal*, 10(2):637–648, 2014.

[41] Xin Xu, Huiqun Yu, and Xinyu Cong. A qos-constrained resource allocation game in federated cloud. In *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 268–275. IEEE, 2013.

[42] Altino M Sampaio and Jorge G Barbosa. Dynamic power-and failure-aware cloud resources allocation for sets of independent tasks. In *2013 IEEE International Conference on Cloud Engineering (IC2E)*, pages 1–10. IEEE, 2013.

[43] Kejiang Ye, Zhaohui Wu, Chen Wang, Bing Bing Zhou, Weisheng Si, Xiaohong Jiang, and Albert Y Zomaya. Profiling-based workload consolidation and migration in virtualized data centers. *IEEE Transactions on Parallel and Distributed Systems*, 26(3):878–890, 2014.

[44] Daochao Huang, Yangyang Gao, Fei Song, Dong Yang, and Hongke Zhang. Multi-objective virtual machine migration in virtualized data center environments. In *2013 IEEE International Conference on Communications (ICC)*, pages 3699–3704. IEEE, 2013.

[45] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755 – 768, 2012. Special Section: Energy efficiency in large-scale distributed systems.

[46] N. K. Sharma and G. R. M. Reddy. Multi-objective energy efficient virtual machines allocation at the cloud data center. *IEEE Transactions on Services Computing*, 12(1):158–171, 2019.

[47] K. Ye, Z. Wu, C. Wang, B. B. Zhou, W. Si, X. Jiang, and A. Y. Zomaya. Profiling-based workload consolidation and migration in virtualized data centers. *IEEE Transactions on Parallel and Distributed Systems*, 26(3):878–890, March 2015.

[48] D. Versick and D. Tavangarian. Reducing energy consumption by load aggregation with an optimized dynamic live migration of virtual machines. In *2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 164–170, Nov 2010.

[49] Ismael Solis Moreno, Peter Garraghan, Paul Townend, and Jie Xu. Analysis, modeling and simulation of workload patterns in a large-scale utility cloud. *IEEE Transactions on Cloud Computing*, 2(2):208–221, 2014.

[50] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu. Real-time tasks oriented energy-aware scheduling in virtualized clouds. *IEEE Transactions on Cloud Computing*, 2(2):168–180, April 2014.

[51] L. Mashayekhy, M. M. Nejad, D. Grosu, Q. Zhang, and W. Shi. Energy-aware scheduling of mapreduce jobs for big data applications. *IEEE Transactions on Parallel and Distributed Systems*, 26(10):2720–2733, Oct 2015.

[52] Y. J. Chiang, Y. C. Ouyang, and C. H. (. Hsu. An efficient green control algorithm in cloud computing for cost optimization. *IEEE Transactions on Cloud Computing*, 3(2):145–155, April 2015.

[53] M. Hähnel, J. Martinovic, G. Scheithauer, A. Fischer, A. Schill, and W. Dargie. Extending the cutting stock problem for consolidating services with stochastic workloads. *IEEE Transactions on Parallel and Distributed Systems*, 29(11):2478–2488, 2018.

[54] Dara Kusic, Jeffrey O Kephart, James E Hanson, Nagarajan Kandasamy, and Guofei Jiang. Power and performance management of virtualized computing environments via lookahead control. *Cluster computing*, 12(1):1–15, 2009.

[55] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes. Energy-efficient resource allocation and provisioning framework for cloud data centers. *IEEE Transactions on Network and Service Management*, 12(3):377–391, Sept 2015.

[56] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[57] Jason L Speyer and Walter H Chung. *Stochastic processes, estimation, and control*. SIAM, 2008.

[58] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang. Power and performance management of virtualized computing environments via lookahead control. In *2008 International Conference on Autonomic Computing*, pages 3–12, June 2008.

[59] D. Jiankang, W. Hongbo, L. Yangyang, and C. Shiduan. Virtual machine scheduling for improving energy efciency in iaas cloud. *China Communications*, 11(3):1–12, March 2014.

[60] Brandon Heller, Srini Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. Elastictree: Saving energy in data center networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, Berkeley, CA, USA, 2010. USENIX Association.

[61] T. Wang, Y. Xia, J. Muppala, and M. Hamdi. Achieving energy efficiency in data centers using an artificial intelligence abstraction model. *IEEE Transactions on Cloud Computing*, 6(3):612–624, 2018.

[62] J. Son, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya. Sla-aware and energy-efficient dynamic overbooking in sdn-based cloud data centers. *IEEE Transactions on Sustainable Computing*, 2(2):76–89, 2017.

[63] Qinghui Tang, Sandeep KS Gupta, Daniel Stanzione, and Phil Cayton. Thermal-aware task scheduling to minimize energy usage of blade server based datacenters. In *2006 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 195–202. IEEE, 2006.

[64] Hong Xu, Chen Feng, and Baochun Li. Temperature aware workload managementin geo-distributed data centers. *IEEE Transactions on Parallel and Distributed Systems*, 26(6):1743–1753, 2014.

[65] J. Yao, H. Guan, J. Luo, L. Rao, and X. Liu. Adaptive power management through thermal aware workload balancing in internet data centers. *IEEE Transactions on Parallel and Distributed Systems*, 26(9):2400–2409, Sept 2015.

[66] H. Xu, C. Feng, and B. Li. Temperature aware workload managementin geo-distributed data centers. *IEEE Transactions on Parallel and Distributed Systems*, 26(6):1743–1753, June 2015.

[67] Q. Tang, S. K. S. Gupta, D. Stanzione, and P. Cayton. Thermal-aware task scheduling to minimize energy usage of blade server based datacenters. In *2006 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 195–202, Sept 2006.

[68] U. Mandal, M. F. Habib, S. Zhang, B. Mukherjee, and M. Tornatore. Greening the cloud using renewable-energy-aware service migration. *IEEE Network*, 27(6):36–43, November 2013.

[69] D. Cheng, J. Rao, C. Jiang, and X. Zhou. Elastic power-aware resource provisioning of heterogeneous workloads in self-sustainable datacenters. *IEEE Transactions on Computers*, 65(2):508–521, Feb 2016.

[70] D. Bruneo, A. Lhoas, F. Longo, and A. Puliafito. Modeling and evaluation of energy policies in green clouds. *IEEE Transactions on Parallel and Distributed Systems*, 26(11):3052–3065, Nov 2015.

[71] Benny Rochwerger, D Breitgand, D Hadas, I Llorente, R Montero, Philippe Massonet, Eliezer Levy, Alex Galis, M Villari, Y Wolfsthal, et al. An architecture for federated cloud computing. *Cloud Computing*, pages 391–411, 2010.

[72] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):4:1–4:11, 2009.

[73] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 13–31. Springer, 2010.

[74] Inigo Goiri, Jordi Guitart, and Jordi Torres. Characterizing cloud federation for enhancing providers' profit. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 123–130. IEEE, 2010.

[75] L. Mashayekhy, M. M. Nejad, and D. Grosu. Cloud federations in the sky: Formation game and mechanism. *IEEE Transactions on Cloud Computing*, 3(1):14–27, 2015.

[76] M. M. Hassan, M. Abdullah-Al-Wadud, A. Almogren, B. Song, and A. Alamri. Energy-aware resource and revenue management in federated cloud: A game-theoretic approach. *IEEE Systems Journal*, 11(2):951–961, 2017.

[77] Quang-Hung N, Thoai N, and Son N.T. Epobf: Energy efficient allocation of virtual machines in high performance computing cloud. In Hameurlain A, Küng J, Wagner R, Dang T, and Thoai N, editors, *Transactions on Large-Scale Data- and Knowledge-Centered Systems XVI: Lecture Notes in Computer Science*, volume 8960. Springer, Berlin, Heidelberg, 2014.

[78] L. Wang, F. Zhang, J. A. Aroca, A. V. Vasilakos, K. Zheng, C. Hou, D. Li, and Z. Liu. Greendcn: A general framework for achieving energy efficiency in data center networks. *IEEE Journal on Selected Areas in Communications*, 32(1):4–15, January 2014.

[79] M. Dayarathna, Y. Wen, and R. Fan. Data center energy consumption modeling: A survey. *IEEE Communications Surveys Tutorials*, 18(1):732–794, Firstquarter 2016.

[80] Y. Li, Y. Wang, B. Yin, and L. Guan. An online power metering model for cloud environment. In *2012 IEEE 11th International Symposium on Network Computing and Applications*, pages 175–180, Aug 2012.

[81] Jiang Lin, Hongzhong Zheng, Zhichun Zhu, Eugene Gorbatov, Howard David, and Zhao Zhang. Software thermal management of dram memory for multicore systems. *SIGMETRICS Perform. Eval. Rev.*, 36(1):337–348, June 2008.

[82] Priya Mahadevan, Sujata Banerjee, and Puneet Sharma. Energy proportionality of an enterprise network. In *Proceedings of the First ACM SIGCOMM Workshop on Green Networking*, Green Networking '10, pages 53–60, New York, NY, USA, 2010. ACM.

[83] H. Tian, J. Wu, and H. Shen. Efficient algorithms for vm placement in cloud data centers. In *2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pages 75–80, Dec 2017.

[84] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, Nov 2006.

[85] Weiwei Fang, Xiangmin Liang, Shengxin Li, Luca Chiaraviglio, and Naixue Xiong. Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. *Computer Networks*, 57(1):179 – 196, 2013.

[86] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[87] Julie Falkner, Franz Rendl, and Henry Wolkowicz. A computational study of graph partitioning. *Mathematical Programming*, 66(1-3):211–239, 1994.

[88] Russell Merris. Laplacian graph eigenvectors. *Linear algebra and its applications*, 278(1-3):221–236, 1998.

[89] S. Y. Charles J. Alpert. Spectral partitioning: The more eigenvectors, the better. In *32nd Design Automation Conference*, pages 195–200, 1995.

[90] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[91] Smile - Statistical Machine Intelligence and Learning Engine, url = http://haifengl.github.io/clustering.html.

[92] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012.

[93] Nilton Bila, Eric J Wright, Eyal De Lara, Kaustubh Joshi, H Andrés Lagar-Cavilla, Eunbyung Park, Ashvin Goel, Matti Hiltunen, and Mahadev Satyanarayanan. Energy-oriented partial desktop virtual machine migration. *ACM Transactions on Computer Systems (TOCS)*, 33(1):2, 2015.

[94] Norman Bobroff, Andrzej Kochut, and Kirk Beaty. Dynamic placement of virtual machines for managing sla violations. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 119–128. IEEE, 2007.

[95] Akshat Verma, Puneet Ahuja, and Anindya Neogi. pmapper: power and migration cost aware application placement in virtualized systems. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 243–264. Springer-Verlag New York, Inc., 2008.

[96] Tathagata Das, Pradeep Padala, Venkata N Padmanabhan, Ramachandran Ramjee, and Kang G Shin. Litegreen: saving energy in networked desktops using virtualization. In *USENIX annual technical conference*, pages 26–34, 2010.

[97] Niraj Tolia, Zhikui Wang, Manish Marwah, Cullen Bash, Parthasarathy Ranganathan, and Xiaoyun Zhu. Delivering energy proportionality with non energy-proportional systems-optimizing the ensemble. *HotPower*, 8:2–2, 2008.

[98] A Sudarshan Chakravarthy, Sudhakar, and T Ramesh. Energy efficient vm scheduling and routing in multi-tenant cloud data center. *Sustainable Computing: Informatics and Systems*, 22:139–151, 2019.

[99] Takahiro Hirofuchi, Hidemoto Nakada, Satoshi Itoh, and Satoshi Sekiguchi. Reactive consolidation of virtual machines enabled by postcopy live migration. In *Proceedings of the 5th international workshop on Virtualization technologies in distributed computing*, pages 11–18. ACM, 2011.

[100] Andrew J Younge, Robert Henschel, James T Brown, Gregor Von Laszewski, Judy Qiu, and Geoffrey C Fox. Analysis of virtualization technologies for high performance computing environments. In *2011 IEEE 4th International Conference on Cloud Computing*, pages 9–16. IEEE, 2011.

[101] Edouard Bugnion, Scott Devine, Mendel Rosenblum, Jeremy Sugerman, and Edward Y Wang. Bringing virtualization to the x86 architecture with the original vmware workstation. *ACM Transactions on Computer Systems (TOCS)*, 30(4):12, 2012.

[102] Anita Choudhary, Mahesh Chandra Govil, Girdhari Singh, Lalit K Awasthi, Emmanuel S Pilli, and Divya Kapil. A critical survey of live virtual machine migration techniques. *Journal of Cloud Computing*, 6(1):23, 2017.

[103] René van den Brink. An axiomatization of the shapley value using a fairness property. *International Journal of Game Theory*, 30(3):309–319, 2002.

[104] Eyal Winter et al. The shapley value. *Handbook of game theory with economic applications*, 3(2):2025–2054, 2002.

[105] Georgios Chalkiadakis, Gianluigi Greco, and Evangelos Markakis. Characteristic function games with restricted agent interactions: Core-stability and coalition structures. *Artificial Intelligence*, 232:76–113, 2016.

List of Publications

1. **A Sudarshan Chakravarthy** and Chapram Sudhakar and T. Ramesh. “Energy efficient VM scheduling and routing in multi-tenant cloud data center.” *Sustainable Computing: Informatics and Systems* 22 (2019): 139-151. https://doi.org/10.1016/j.suscom.2019.04.004

2. **A Sudarshan Chakravarthy**, Chapram Sudhakar, and T. Ramesh. “Intermediate node selection for Scatter-Gather VM migration in cloud data center.” *Engineering Science and Technology an International Journal* (2020). https://doi.org/10.1016/j.jestch.2020.01.008

3. **A Sudarshan Chakravarthy**, Chapram Sudhakar and T Ramesh,”Communication Aware VM Placement Using Spectral Graph Partitioning ”, *Springer: New Generation Computing* (Communicated)

4. **A Sudarshan Chakravarthy**, Chapram Sudhakar and T Ramesh,”Social Welfare Maximization in Federated Cloud: A Coalition Based approach”, *IEEE transactions on sustainable computing* (Communicated)