AN ENERGY EFFICIENT GENETIC ALGORITHM BASED RESOURCE ALLOCATION MODEL FOR CLOUD COMPUTING

Dissertation submitted to Jawaharlal Nehru University
in partial fulfillment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE AND TECHNOLOGY

WALIULLAH

ENROLLMENT NO. 13/10/MT/035

SCHOOL OF COMPUTER & SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI-110067
INDIA
2015

SCHOOL OF COMPUTER & SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI, 110067 (INDIA)

CERTIFICATE

This is to certify that the dissertation entitled **"An Energy Efficient Genetic Algorithm Based Resource Allocation Model For Cloud Computing"** is being submitted by **Mr. Waliullah** to **School of Computer and Systems Sciences, Jawaharlal Nehru University New Delhi-110067, India** in the partial fulfillment of the requirements for the award of the degree of **Master of Technology in Computer Science and Technology**. This work has been carried out by him in the School of Computer and Systems Sciences under the supervision of Dr. Zahid Raza. The matter personified in the dissertation has not been submitted for the award of any other degree or diploma.

DR. ZAHID RAZA
(SUPERVISOR)

DEAN, SC&SS
JNU, NEW DELHI

डीन/ Dean
कंप्यूटर और प्रणाली विज्ञान संकायन
School of Computer and Systems Sciences

DECLARATION

I hereby declare that the dissertation work entitled **"An Energy Efficient Genetic Algorithm Based Resource Allocation Model For Cloud Computing"** in partial fulfillment for the requirements for the degree of **"Master of Technology in Computer Science and Technology"** and submitted to School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi-110067, India, is the authentic record of my own work carried out during the time of Master of Technology under the supervision of Dr. Zahid Raza. This dissertation comprises only my original work. This dissertation is less than 100,000 words in length, exclusive tables, figures and bibliographies.

The matter personified in the dissertation has not been submitted for the award of any other degree or diploma.

Waliullah
Enrollment No. 13/10/MT/035
M. Tech (2013-15)
SC&SS, JNU
New Delhi India -110067

ACKNOWLEDGEMENT

I am very glad to express my sincere gratitude and thanks to my supervisor **Dr. Zahid Raza** for his guidance. I would like to express special thanks to Dr. Zahid Raza for many helpful discussions and his intellectual input to make dissertation work worthy. His extensive and invaluable research experiences were very helpful in my dissertation and the most important thing was the helping nature of him that contributes an important share in fulfillment of this work. The mythology, philosophy and problem solving methods learned by him have been very beneficial in this work and would be afterward.

I would like to express my thanks to Dean SC&SS JNU, Prof. C P katti in support to pursue my work in the School. Also my thanks go to School administration and librarian of software library and main library for supporting me, in whatever way they can, to make dissertation a success. Their support has been a real emphasize in completing this dissertation.

I would like to accord my sincere thanks to Mohammad Shahid, Mohammad Sajid, Taj Alam, Sumit Kumar, Krishan Veer Singh, Shahadat Hussain, Hedayat Ullah and Nadiya Hashim, for their valuable suggestions for my dissertation work.

My parents and family members have been my strength through the long hours of study and research. I especially thank my father and mother for their patience, unconditional love and moral support for completing this dissertation. Finally I would like to express thanks to each person & thing which is directly or indirectly related to my dissertation work.

Waliullah

DEDICATED TO ALLAH, HIS BELOVED PROPHET, MUHAMMAD (PBUH) AND MY LOVELY PARENTS

Table of Contents

Abstract

With advancement in science and technology every day the computational power of a system needs to be up to the mark to incorporate these advancements and changes. Since science is based on analysis, visualization and collaboration of available data so that useful information can be extracted for which high computation power is needed. Also, since scientific and engineering problems are getting more and more complex, user needs them to be solved precisely and accurately within the limited time. As a result of the increasing need of high computational power, the term parallel computing comes in to the picture.

In parallel computing, multiple computer or processors work together to solve a single problem or to achieve a goal. This meets the requirement of improved performance and also the need of memory is satisfied. Parallel computing is of two types, parallel processing and distributed computing. In parallel processing, several no. of processors work together to solve a problem with each processor handling a section of code and is allowed to exchange the information between them. In distributed computing system there are multiple computers with multiple software components that are working together to achieve a single goal.

In distributed system the computers can be at same physical location or globally distributed and connected via high speed network. These distributed systems include cluster computer, super computer and storage systems etc. Cloud computing is defined as a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications. Cloud computing is comparable to grid computing, a type of computing where unused processing cycles of all computers in a network are harnessed to solve problems too intensive for any stand-alone machine.

In cloud computing, the word cloud is used as a metaphor for "the Internet" so the phrase cloud computing means "a type of Internet-based computing" where different services — such as servers, storage and applications — are delivered to an organization's computers and devices through the Internet. The goal of cloud computing is to apply traditional supercomputing or high-performance computing power normally used by military and research facilities to perform tens of trillions of computations per second in

consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive online computer games.

Resource allocation to different processes and application is known as scheduling which has been proven to be NP-hard for cloud environment. NP-hard is a class of problems where it is very difficult to find an exact solution. In this category, exact solution cannot be determined but approximate solution can be obtained which is acceptable and considered as good as the exact solution. Such type of problem cannot be solved by traditional method because mathematical modeling is not easy. To handle NP-hard problems soft computing techniques are used which include Neural Network, Fuzzy Systems, Probabilistic Reasoning and Evolutionary Computing. Among all soft computing techniques, evolutionary computing is considered as a better option since it is closely related to the nature. Evolutionary computing, a global search paradigm, includes Evolutionary Strategies, Evolutionary Programs, Genetic Algorithms and Genetic programming. Genetic Algorithm was inspired by the Darwin theory of evolution i.e. "fittest of the survival".

Genetic Algorithms (GA) are most common in all evolutionary paradigms. GA mimics the process of natural evolution and finds its use in solving computing and optimization problems. In GA, a population of chromosomes, generally a sequence of bits is randomly selected. This population is then transformed into some new population by the use of some methods which are similar to the natural selection by the use of operators which are inspired by the natural genetic operators like crossover, mutation and inversion operator.

Fitness function is the deciding criteria for the natural selection of a population. According to that, the chromosomes having optimum fitness value can survive and are allowed to reproduce offspring. Among all chromosomes that survive the fittest chromosomes can reproduce to produce new offspring than the less fit chromosomes. Then the crossover operator performs crossover on the selected chromosomes based on certain features like bit location in the parent chromosomes to produce new offspring having same size. The mutation operator flips/replaces the bits at selected locations by a certain value. The inversion operator reverses the order of a subsequence in a chromosome.

When the new generation of a population is completed, then it is checked for the stopping criteria. If stopping criteria is met then the algorithm is stopped otherwise the

fitness value is again evaluated for the chromosomes of this generation and the whole process is repeated till the stopping criteria is not met. For any job execution the minimization of energy consumption has become an important issue since energy is a precious resource. In this work we have tried to develop a energy aware scheduling model based on Genetic Algorithm. GA is an established soft computing tool for such kind of combinatorial problems. The model analyses the performance of the scheduling scheme on basis of number of tasks. The simulation study reveals the effectiveness of the model.

List of Acronyms

API	Application Programming Interface
CPU	Central Processing Unit
DAG	Directed Acyclic Graph
DNA	Deoxyribonucleic Acid
GA	Genetic Algorithm
GHz	Gigahertz
GP	Genetic Programming
HPC	High Performance Computing
IaaS	Infrastructure as a Service
IBM	International Business Machines Corporation
IMC	Inter Module Communication
LAN	Local Area Network
MATLAB	Matrix Laboratory
MCT	Minimum Completion Time
MET	Minimum Execution Time
NEC	Node Execution Cost
NP	Non-deterministic Polynomial time
OLB	Opportunistic Load Balancing
PaaS	Platform as a Service
RS	Rank Selection
RW	Roulette wheel Selection
SaaS	Software as a Service
SDK	Software Development Kit
SETI	Search for Extraterrestrial Intelligence
TAT	Turnaround Time
TSP	Travelling Salesman Problem
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a service

List of Figures

List of Tables

Chapter 1

Cloud Computing

1 Introduction

One of the most emerging technologies which has reshaped the IT industry in the recent past is Cloud computing. It has been the hottest topic of research in the last few years. Cloud computing is a word which refers to different things to different group of people. For a group of people it means computing services provided over the internet seamlessly, for another it's just a way of describing "IT Outsourcing" and for some others it's a bought-in service that resides outside one's premises [2].

We often do cloud computing throughout the day without actually realizing it. When we type a query into Google and it gives a prompt reply within fraction of a second, it's not our desktop or laptop which is doing most of the work, in fact it just acts as a messenger. When we type the words, those are quickly transferred over the Internet to hundreds of thousands of servers of Google. These servers search for the information we requested for and prompt us back with all the relevant information.

There are two entities involved in a cloud computing environment, the user and the service provider. User is the entity which is entitled to use the services of a cloud over the internet. Service provider is the entity which builds its own infrastructure to provide services. The user is charged for the services it uses on the "pay as you go" basis. There exists a service level agreement between the user and the service provider to ensure the Quality of Service (QoS) [3].

Cloud computing has evolved over a period of time through different phases such as mainframe computing, parallel computing, distributed computing, utility computing and grid computing. The following sections discuss the standard definitions of cloud computing, its characteristics, evolution, service models, deployment models, issues and challenges etc .

1.1 Cloud Evolution

Different phases are involved in the evolution of cloud computing, which are shown in Figure 1.1. Each phase has been separately explained in the following sections [6].

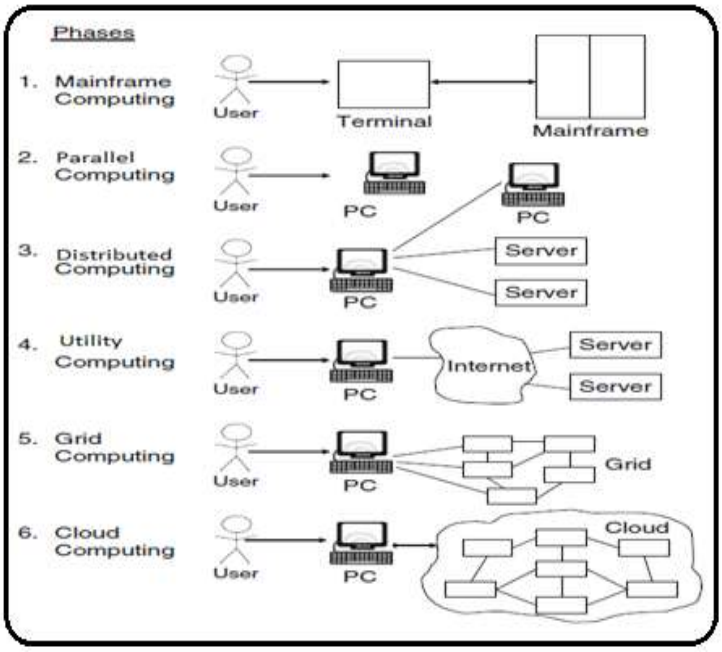

Figure 1.1: Evolution of Cloud Computing [5]

1.1.1 Mainframe Computing

To cater the demands of large organizations which requires very huge amount of computations, mainframe computers were used. Mainframe computers are very large computers with high computing capacity. These are centralized computing system with dummy terminals. This concept was prevalent when the personnel computers were not popular. After the PCs gained popularity parallel computing came into the picture. Some examples of mainframe computers are IBM 704 (1965), IBM RAMAC 305 (1956). IBM RAMAC 305 used 50 iron coated revolving disks those could accept magnetically coded data which significantly improved the state of data processing.

1.1.2 Parallel Computing

Parallel computing refers to the concept of using multiple processors on the same machine and solve a computationally big problem by dividing it into small problems. Each instance of the problem could run simultaneously on the multiple processors thus giving high

performance. Personal computers are far economical than the mainframes so they gained popularity. Parallel computing suffered from the drawback of being centralized. Example of parallel computing include processors with multiple cores which could work independently e.g. Intel Core2 duo, Intel Quad core and Intel Octa core processors. The basic philosophy in parallel computing is that a big task can be solved by dividing it into smaller tasks with these smaller units running on different cores.

1.1.3 Distributed Computing

Distributed computing can be referred to the collection of many autonomous computer system connected through a network where each computer has its own processor and memory. These computers are used to solve a computationally big problem by running different instances of the problem on each single machine. Later on the solutions of each machine are combined to reach the final solution of the problem. Distributed system carries out parallel processing but it's not centralized as in the case of parallel computing. Some examples of distributed computing can be telephone networks, cellular networks, Computer networks such as the Internet too is an example of distributed computing.

1.1.4 Utility Computing

Utility computing is a model where a service provider owns, manages and operates the resources and computing infrastructure and the user accesses these whenever required on a pay as you go basis. The resources can be accessed over the Internet or any virtual private network. Utility computing includes virtual storage, virtual servers, virtual software, etc. Example of utility computing includes renting out the computing facility of a supercomputer over a network and charge the users on the "pay-per-use" basis.

1.1.5 Grid Computing

Grid computing can be defined as the collection of multiple computing resources over various locations to attain a common goal of solving a single task. Grid computing is different from the conventional high performance computing in a way that each node in a grid performs different task or application. The nodes in a grid environment can be heterogeneous and geographically dispersed. Cloud computing is basically an extension of grid computing. Typically applications like weather forecasting, protein folding and earthquake simulation are

prime candidates for a grid infrastructure. Grids have also been used to render large-scale animation projects, like movies.

1.2 The Cloud

What cloud computing actually is? Numerous formal definitions have been proposed in the industry and the academia. The definition provided by U.S. NIST (National Institute of Standards and Technology) is one of the most appropriate which includes key common elements that are mostly used in the cloud computing community:

"*Cloud computing is a model for enabling 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"* [3]

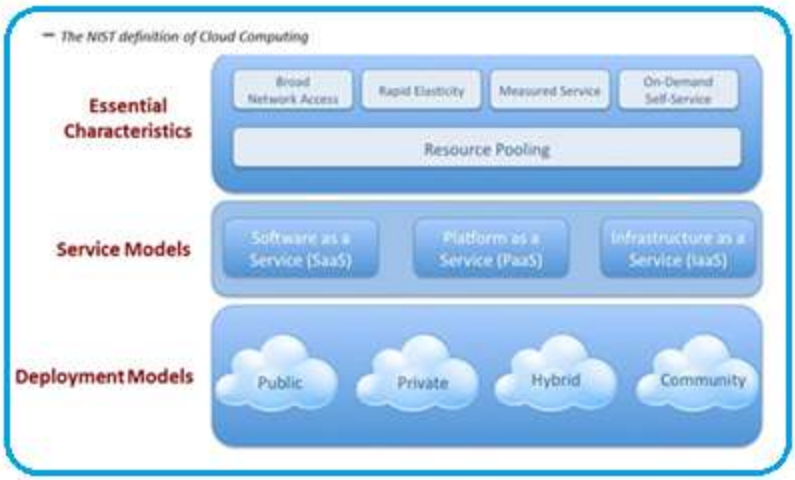

Figure 1.2: A generic view of Cloud Computing [10]

Another definition of cloud computing is as follows:

"A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet"[8]

Above two definitions point that cloud is an arrangement where resources are provided over the internet to a user in a way that the user is charged for the service. Accordingly some of the important characteristics of cloud should be [1, 3]:

1.2.1 On Demand Self Service

A user should be able to avail cloud services such as software use, network storage, CPU time etc as and when he wants without any human involvement. It is the user who decides what type of services he wants and at which period of time. The basic characteristics of a cloud is that it should be able to provide the required service at a particular timeslot.

1.2.2 Resource Pooling

The resources of a cloud service provider are pooled together in order to serve multiple users either using multi-tenancy or virtualization "with different physical and virtual resources dynamically assigned and reassigned according to consumer demand"[ ]. The benefit of having a pool-based computing system leads to two important factors: *economies of scale* and *specialization.*

The pool-based arrangement of computing resources results in a system where the resources become “invisible” to the users. The users potentially do not know or control the location, originalities and formation or the resources and are completely unaware that where the data is going to be stored on the cloud.

1.2.3 Rapid Elasticity

For users, the demand of computing resources may increase and decrease with time. There might be certain slot of time when the requirement of the resources rise rapidly or fall very quickly. The cloud should be able to adjust to these changes. It should appear to the consumer that the cloud has infinite resources and can cater its demand during the peak hours of service as well.

1.2.4 Broad Network Access

The resources are provided over the Internet and used by different client applications on various platforms such as smart phones, desktop, laptop, PDA’s etc. located at the user’s site.

1.2.5 Measured Service

Even though the resources are shared and pooled by multiple users, the cloud infrastructure must be equipped with appropriate tools to measure the consumption by each user

individually. Each consumer should be charged for what it has used with a proper account maintenance.

1.3 Cloud Service Models

Cloud provides a wide range of services. In order to provide these services the cloud environment has divided it into three major service models namely Software As a Service (SaaS), Platform As a Service (PaaS) and Infrastructure As a Service (IaaS) [hbc].

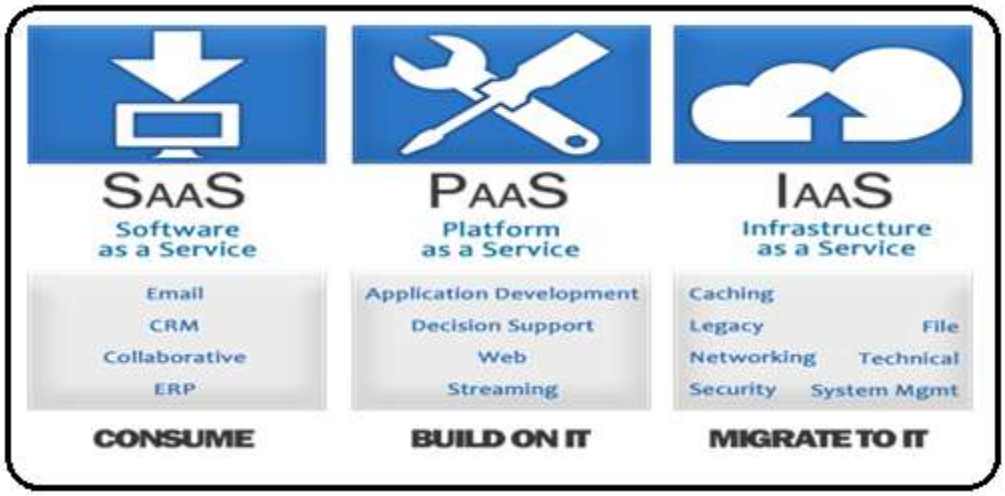

Figure 1.3: Service Models of a Cloud [10]

1.3.1 Software as a Service (SaaS)

Cloud users can release their applications on the cloud environment that can be accessed by the application users through network on laptops, smart phones, PDAs, etc. The cloud user does not possess any control over the cloud infrastructure which generally uses virtualization to achieve optimization in terms of maintenance, performance, reliability, availability, speed and security. Some examples of SaaS include Google mail, Google Docs, Salesforce.com to name a few.

1.3.2 Platform as a Service (PaaS)

PaaS constitutes of a development platform which supports the full "Software Lifecycle" that enables cloud consumers develop cloud applications and services e.g. SaaS directly over the PaaS cloud environment. So the point of difference between PaaS and SaaS is that SaaS can only host developed cloud applications while PaaS can host developed cloud applications as well as

those cloud applications which are in-progress. For this the PaaS needs to be equipped with programming environment, configuration management, tools, etc. besides having an environment that hosts applications. Google AppEngine is an example of PaaS.

1.3.3 Infrastructure as a Service (IaaS)

Cloud users use IT infrastructure such as storage, network, processing units and other computing resources available in the IaaS cloud model. This is extensively done by using virtualization which allows to combine/decompose resources in a manner to meet the increasing or decreasing demand of a cloud user. The concept behind virtualization is to build independent virtual machines (VM) which are separate from both the hardware and other virtual machines (VMs). Amazon's EC2 is an example of IaaS cloud.

1.4 Cloud Deployment Models

There are four deployment models which are proposed for cloud computing. They are namely private cloud, public cloud, community cloud and hybrid cloud [1, 3].

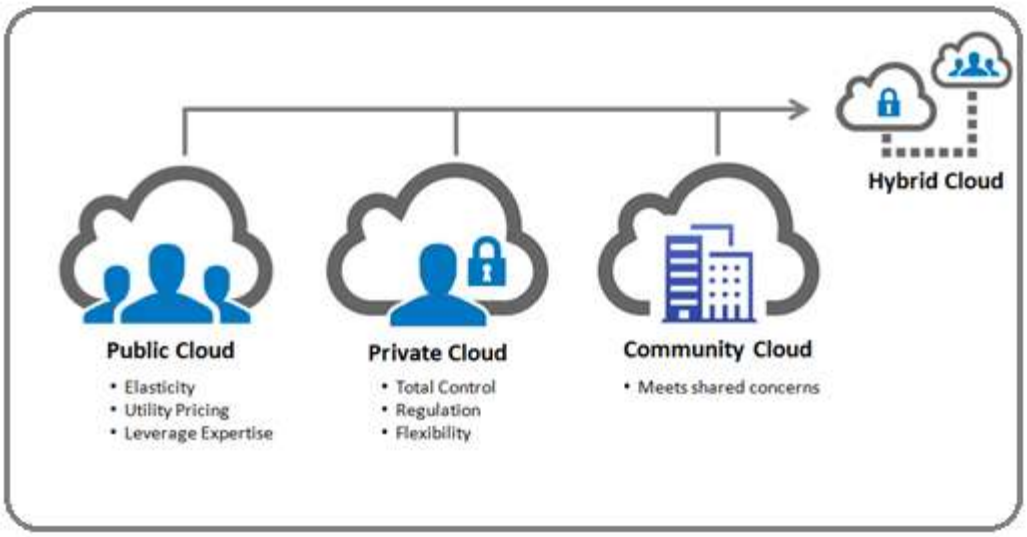

Figure 1.4: Deployment Models of a Cloud [13]

1.4.1 Private Cloud

This cloud is completely operated within an organization, which could be managed by the organization or a third party independent of the fact if it is located on premise or off premise. There could be several reasons behind developing one's own cloud. First, to optimize and

maximize the utilization of already existing resources. Second, due to security reasons such as data privacy, trust and reliability an organization may opt for a private cloud of its own. Third, the cost of data transfer from local infrastructure to public cloud is still considerable. Fourth, academics generally develop their own private cloud for teaching and research purposes. Meghdoot by CDAC, Baadal by IIT Delhi are some examples of private cloud.

1.4.2 Public Cloud

This is one of the most popular cloud deployment models. A public cloud is meant to be used by the general public cloud users where the cloud service provider has complete ownership with its own value, policy, profit, costing and charging model. Some of the popular public clouds include Google AppEngine, S3, Force.com and Amazon EC2.

1.4.3 Community Cloud

When multiple organizations collectively develop and share the cloud infrastructure, policies, concerns, values and requirements, the deployment model is known as community cloud. The cloud could be operated and managed by a third party or one of the participating organizations. Examples may include any collaboration of clouds between two or more organizations.

1.4.4 Hybrid Cloud

Hybrid cloud deployment model the cloud infrastructure is a combination of two or more clouds (public, private or community) staying as unique entities with connected standard technology which enables application and data portability. For example, to provide load balancing it can use cloud-bursting. The motive behind using a hybrid cloud among the organizations is to optimize the utilization of their resources through hybrid cloud deployment. But this model certainly raises various issues such as cloud interoperability and standardization. Combination of any two or more of the above mentioned clouds can be named as a hybrid cloud.

1.5 Issues and Challenges of Cloud Computing

Cloud computing is a technology which has evolved recently. Since its not completely established, there are certain issued and challenges which needs to be addressed to utilize the

cloud computing paradigm to its fullest. Some of the issues and challenges based on a survey conducted by IDC are presented in following sections [1, 2, 3].

1.5.1 Security and Privacy

Security and privacy are the most dominant issues which have stopped the cloud from being adopted widely. Security has been the most debated issue in the cloud computing environment. Obviously moving ones data to the cloud through network, running its software on someone else's processor are quite a big thing for an organization to think about. Based on the data of the survey security and privacy stands on the top of the list which is quite obvious as it could be daunting for any person or organization to move its data over the network.

1.5.2 Availability

Availability can be defined as the extent to which the cloud services are available as and when they are required. Even though if high availability constraints are implemented, the cloud can be affected by performance slowdown, denial of service attacks, natural disasters and equipment outages. Based on the data we can say that some of the current service providers have faced equipment outages, which recently happened with Amazon's EC2. So, availability is an important factor to consider for an organization which plans to move to cloud.

1.5.3 Costing Model

When an organization decides to adapt to cloud, it can reduce a lot of investment on the infrastructure but at the same time some costs tend to increase. The data communication cost to and from a cloud will increase and as the amount of data increases the cost increases. Same happens with the computation, the cost of per unit of computation is higher which will increase the overall computation cost. Thus, before moving on to a cloud an organization has to analyze properly whether to move or not to move to cloud computing.

1.5.4 Charging Model

Unlike a data center which charges its users for the underlying server, cloud computing charges its users for instantiated virtual machines (VM). Moreover the usage of the consumers change with time and the number of VMs too keep on changing due to elastic nature of the

cloud. Hence, the cloud needs a sound charging model which is equipped to handle the mentioned complexities.

1.5.5 Service Level Agreement

Although a cloud user does not have any control over the cloud resources as it is completely owned by the service provider but the user needs to ensure the reliability, availability and performance of the resources after the it has moved on to the cloud. A service level agreement (SLA) exists ensuring certain QoS which has been negotiated between the user and the service provider and this ensures the above mentioned concerns of a cloud user. The issue with SLA is that the provider should make sure that the negotiated level of services is offered to the user, failing to which can lead to severe consequences.

1.5.6 What to Migrate?

According to the survey conducted by IDC various organizations which have moved to cloud are: Applications Development and Deployment (16.8%), Personal Applications (25%), IT Management Applications (26.2%),Server Capacity(15.6%),Collaborative Applications (25.4%),Storage Capacity (15.5%) and Business Applications (23.4%). This data shows that organizations are still suspicious about moving their business to cloud.

1.5.7 Cloud Interoperability Issues

Interoperability refers to the efficiency with which applications and tools could be used on various cloud platforms. It could be defined at many levels such as data, management, service, application interoperability. Cloud consumers must be able to move in and out of a cloud infrastructure without any locking period of the vendor whenever they want. Vendor lock-in period is one of the barriers in the cloud adoption process. The dominant factors are open standards, VMs lack standard interface formats, open APIs and service deployment interfaces.

1.6 Scheduling in Cloud

A cloud is supposed to provide services to its users. The services could be of various types but ultimately it turns out to be in terms of processes and these processes require allocation of some resources. This process of resource allocation is known as scheduling. The main

objective of scheduling is to optimize the resource utilization along with the optimization of some parameters such as turnaround time, completion time, energy consumption, etc.

The software which is responsible for scheduling is known as the scheduler. The nature of scheduler can vary across various clouds as it depends upon the cloud service provider which type of scheduling should be carried out on the basis of the parameters to be optimized. Scheduling has been broadly classified into two categories viz. static scheduling and dynamic scheduling [32, 33].

1.6.1 Static Scheduling

When the characteristics of a the process such as task processing time requirement, synchronization requirements, communication and data dependencies are known a prior, the communication can be done at compile time. This type of scheduling is known as static scheduling. A job can be represented using a weighted node-edge graph also known as directed acyclic graph (DAG) where each node represents a task in the job and edges represent the dependencies among the tasks. The weight of a node represents the processing time requirement of the task and the edge weight represents the communication as well as the data dependencies.

1.6.2 Dynamic scheduling

When the scheduling decisions are made "on-the-fly" the type of scheduling is referred to as dynamic scheduling. A few assumptions are made about the job before scheduling it. The objectives of the dynamic scheduling not include the minimization of completion time but optimizing the scheduling overheads also. Scheduling overheads comprise of a significant part of the total cost incurred for executing the scheduler.

1.6.3 Cloud Scheduling: An NP Hard problem

Task scheduling on cloud has been proved to be an NP complete problem. Though there are some special cases which are an exception to this but in general, its considered to be an NP complete problem. Problems have been classified into many categories depending upon the amount of time it takes to solve a problem. Following are the various class of problems discussed [13, 14].

P-class of Problems

A problem is said to be P-class problem if a deterministic polynomial time algorithm exists for it. For example, sorting n numbers can be solved in O (n^2) where n is the input size.

NP- class of Problems

A class of problem for which the correctness can be verified for a given solution in polynomial time is known as NP class of problems. NP stands for non deterministic polynomial time. The NP class of problem is further classified as NP Hard and NP Complete. An example of NP problem is verifying Hamiltonian cycle problem. If the nodes are given then it can be verified in polynomial time whether it's a Hamiltonian cycle or not but its not possible to find a Hamiltonian cycle in polynomial time [16].

NP Hard class of Problems

There are some problems such that every single problem in NP can be translated to those problems and a polynomial time solution to those problems will give a polynomial time solution to every problem in NP. Such problems are known as NP hard problems. Finding the Hamiltonian cycle has been proven to be an NP hard problem.

NP Complete class of Problems

There are some NP hard problems which are not under the circumference of NP class of problems. The problems which are in both NP hard and NP class of problems are known as NP complete class of problems. All NP complete problems are NP hard but vice versa is however not true. NP complete class of problems is known to be the hardest problems. Since, we have seen that verifying a Hamiltonian cycle is an NP problem and finding the Hamiltonian cycle is an NP hard problem so it is an NP complete problem too [15].

1.6.4 Approaches to Scheduling Problem

After explaining the NP complete class of problems it is now very obvious that obtaining an exact solution to this class of problems is very difficult. Many approaches have been suggested which give as best results as possible to the scheduling problem in polynomial time. Some approaches have been discussed as follows [16, 26].

Approximation Algorithm

A way of dealing with NP hard class of problems is to use approximation algorithm. These algorithms do not guarantee the best solution. Approximation algorithms aim at giving close to optimum solution in a reasonable amount of time which is polynomial time in the worst case.

Heuristic Algorithm

This approach is used when the classic algorithms fail to find the exact solution or when they are too slow. The heuristic algorithms find a feasible solution in reasonable time by performing the algorithm many times. The solution thus found is not guaranteed to be optimal. Heuristic algorithms are effective and simple to apply. Some examples of heuristic algorithm are Genetic algorithm, simulated annealing, Tabu search, etc.

Chapter 2

Genetic Algorithm

2.1 Introduction

Genetic Algorithm (GA) is a general purpose search algorithm based on the process of evolution observed in the nature. It is a heuristic search algorithm, a part of evolutionary computing and a booming area of Artificial Intelligence, inspired by the Darwinian theory of evolution "survival of the fittest" for getting optimum solution of a problem which is not solved by traditional methods. In other words, GA is an adaptive heuristic search algorithm which uses the idea of natural selection and genetics [21].

Genetic Algorithm directs the search to the region of the sample space where better results can be obtained. Genetic Algorithm can be applied to wide variety of applications for e.g. computer games, scheduling, transportation problem, TSP, medical, adaptive control and stock market trading. In general, GA can be classified into two categories viz. Single optimization GA and Multi-objective GA. In the case of single objective optimization GA, one tries to obtain the best solution among all alternative solution by optimizing the single objective function whereas in the other case, there are multiple conflicting objectives that are to be optimized to produce a set of optimal solutions using the Pareto optimality theory and the optimal set of solutions must satisfy all the objectives as best as possible [22].

Some of the advantages of GA are as follows [21, 22]:

- Easy to understand and covers the large search space.
- Supports multi-objective optimization.
- Good for noisy environment.
- GA gives the solution of a problem through evolution process.
- It can solve any optimization problem, which can be encoded using chromosomes.
- Easy to exploit the result.
- GA is good as good as the objective function is.

- Runs in parallel.
- The fitness function can be changed at each iteration if needed to increase the
- performance.

The important disadvantages of GA could be [21, 22]:

- GA is very slow.
- We have to choose mutation rate, population size wisely; if the mutation rate is too high then they never converge to an optimal solution and if the population size is too small then search space is very small to find the solution.
- Designing the fitness function for any problem is very crucial.
- Convergence is always there, but time taken is uncertain depending on the objective function chosen.
- It does not always give the exact solution.

2.1.1 History

The term evolutionary computing was first coined by I. Rechenberg in 1960‟s in his work "Evolutions strategy". His idea was further used and developed by other researchers. The term Genetic Algorithm (GA) was first introduced by John Holland while studying cellular automata, developed by him, his students and colleagues at the University of Michigan. Holland incorporated his work in a book named "Adaption in Natural and Artificial Systems" published in 1975. Actually, Holland's goal was not to design algorithm for any specific problem but to understand the phenomenon of adaptation that occurs in nature and how to incorporate this way of adaptation in computer science. John Koza, in 1992 used the GA to evolve programs to perform certain task. This was named as Genetic Programming (GP) [23].

2.1.2 Biological Background

Cell is the basic building block of living organisms. Each cell has same set of chromosomes, which consists of genes, the basic unit of DNA. A string of DNA in a gene forms a particular trait. Each gene has its position in the chromosomes, called locus. The solution to a problem in Genetic Algorithm is called chromosomes, the parameter that is to be optimized.

Evolution is a method of searching the best among the huge number of possibilities. In biological terminology the huge possibility is the availability of set of possible genetic sequences

and the desired solution is the fittest organism among all available possibilities. In other words, solving a problem using GA can be seen as looking for a solution which is the best among the others [21].

2.2 Structure of Genetic Algorithm

The structure of Genetic Algorithm is explained as follows [22, 24, 25]:

2.2.1 Fitness Function

To solve any problem using GA, we first require formulating its mathematical model in terms of a function. Then parameters that optimize the function of the model are determined. This is known as fitness function also known as the Objective function. Fitness function basically determines or analyzes the genes holding the data and returns the fitness value to quantify the fitness or suitability of the chromosomes. This results in selection of chromosomes having higher fitness value for producing the next generation. The better is the fitness, more chances are there of selecting those chromosomes to survive. The chromosomes having poor fitness value are discarded. The fitness function varies from problem to problem. The effectiveness of the fitness function determines how well a problem can be solved.

2.2.2 Search Space

If we are looking for solution to a problem which will be best among the other, then search space comes into picture. Search space is the set of all possible solutions that can exist for a given problem. Search space is defined by the domain where all feasible/possible solution (or the object among those desired solutions) are present, also known as state space. Each solution or point in the search space is known as the feasible solution and is marked by its fitness value. Search space changes at each step of evolution.

2.2.3 Termination Condition

There could be various terminating conditions for the GA to stop. A programmer can use either one or multiple terminating conditions as per the domain requirements. Some of these are listed as follows:

- **Generation number:** A solution is obtained when maximum numbers of iterations have been run.
- **Evolution time:** when the specified time limit exceeds the process can be terminated.
- **Fitness threshold:** when the best fitness in the current population is less than the user specified fitness threshold value, when the aim is to minimize the fitness value. This also stops when the best fitness value in the current population becomes more than the user specified threshold value with the objective being maximization of the fitness.
- **Fitness convergence:** The evolution process stops when fitness converges.
- **Population convergence:** The evolution process also stops when population converges in next generations.
- **Gene convergence:** A termination method that stops the evolution process when a user specified percentage of the gene of a chromosome is greater than the percentage of genes in a chromosome of current population.
- **Maximum iteration without termination:** when further specified number of iteration does not improve the specified result the evolution process is terminated.

2.2.4 Steps of Genetic Algorithm

GA is a method that produces or evolves to a better population or children at each step by choosing the best parent chromosomes from the available set of chromosomes. The newly generated chromosomes have better rate of survival than the previous generation as they have come from the fittest parents of the previous generation. This can be done by using methods of natural selection and other genetic inspired methods like cross-over, mutation and inversion etc. Genetic Algorithm evolves until a certain termination condition is met.

The step by step process of GA can be represented as shown in Figure 2.1. The method presented can be summarized into various steps as shown below.

1. Start with the randomly generated population of n, l bit chromosomes.

2. Calculate the fitness f(x) of each chromosomes x in the population.

3. Repeat the following steps until population of size n is created.

a) Select a pair of parent chromosomes from the current population. Selection is made on the basis of fitness i.e. higher the fitness more is the probability of selecting the chromosomes. Same chromosomes can be selected more than once.

b) With a crossover probability (p_c) mate the parents to form new children. If no crossover was performed exact copies of the children is obtained. Crossover can be single point or it can be a multi-point.

c) Mutate the two offspring at each locus with probability p_m, known as mutation probability or mutation rate and place the mutated children in new population. If n is odd, a randomly selected offspring is deleted or discarded.

4. Replace the new population with the current one.

5. If the termination condition is obtained then stop, otherwise repeat steps 2-4.

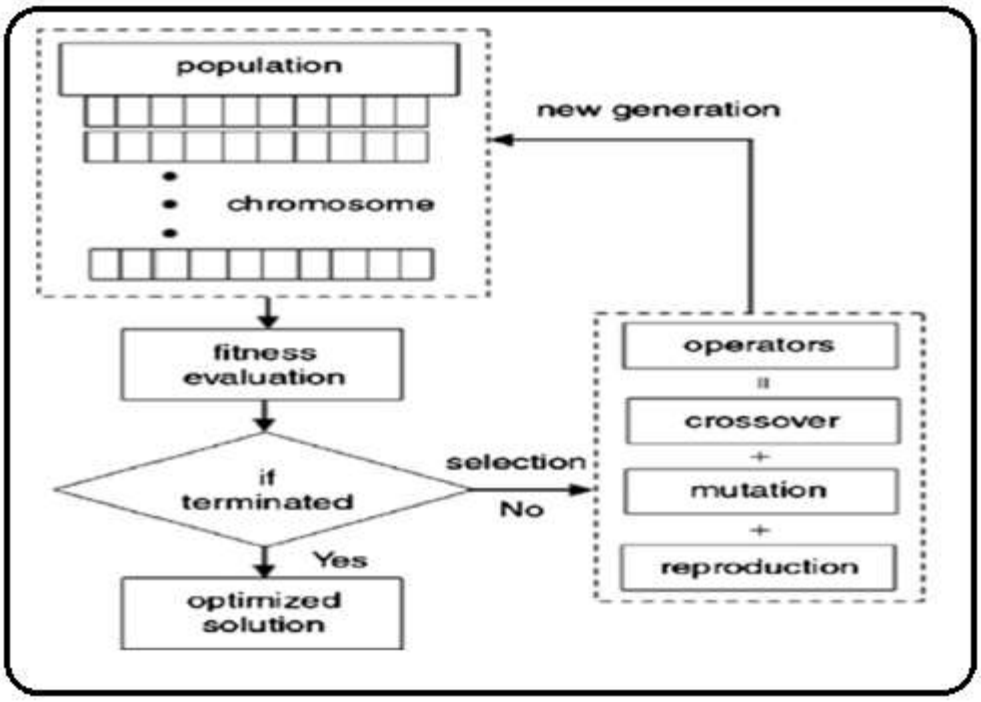

Figure 2.1: Steps for Genetic Algorithm [21]

2.3 Operators of Genetic Algorithm

In order to solve a problem by GA sometimes it is not necessary to always encode the variable. But some problems can be only solved by encoding the variables. The most common method used for encoding is binary encoding. The length of string is decided by the accuracy needed. Encoding is done either by 0 and 1, real number or integer, depending on the problem. For e.g. after binary encoding chromosomes may look like [21, 24, 27]:

Chromosomes1	1100100101100001
Chromosomes2	1000010110001011

2.3.1 Selection Operator

Selection determines which chromosome is to be chosen and how many offspring are to be generated. Selection of chromosomes can be done, first by assigning fitness value to each chromosome. Then on the basis of their fitness value they are selected with both assigning fitness value to each chromosome and selection being done on the basis of certain algorithm [gahb].

2.3.2 Crossover operator

After encoding certain operation is performed on the chromosomes so that new offspring is generated/produced. One of them is crossover, in which two chromosomes are selected for producing offspring. To accomplish, this crossover point is selected randomly in each at same locus point. After this, in new offspring first part of chromosomes is selected from the first part of first parent and second part from second parent and so on as shown below:

Chromosome1	**11010|11000001110**
Chromosome2	10010|00101010100
Offspring1	**11010**|00101010100
Offspring2	10010|**11000001110**

Here, crossover is performed at a single point. It can even be multi-point depending on the encoding of chromosomes in the problem. Specific crossover operator is used for specific type of problem. This increases the performance of the GA. Generally, single point crossover is used when size of the string is small and multipoint crossover is used when the size is large.

2.3.3 Mutation operator

After crossover is performed, mutation can be done. It is used to avoid the process of getting trapped in a local optimum. Using mutation, new offspring is randomly changed at any location. For binary encoding, in mutation, bits are randomly flipped i.e. at some position 1 flips to 0 and vice-versa. Mutation randomly changes the genetic information. When operated at bit level it is possible that mutation occurs at each bit but this has very lower probability as defined by mutation probability (Pm).

Offspring1 before mutation	**01**011010010**1**1011
Offspring1 after mutation with P_m=25%	**10**010**0**1001010**0**1011

On the basis of the encoding as well as the crossover, mutation is performed. For example in permutation encoding bits are exchanged. Mutation maintains genetic diversity and at the same time inhibits premature convergence. During evolution the mutation occurs according to the mutation probability, usually set to a fairly low value (0.01 is a good choice). If it is set to very high then the search will become a primitive random search.

2.4 Parameters of Genetic Algorithm

The basic parameters of GA are crossover and mutation probability [21, 23, 24].

2.4.1 Crossover Probability

It shows how frequent the crossover occurs. If there is no crossover then children are the exact copy of their parents i.e. crossover probability is 0% whereas if crossover probability is 100% then all offspring are formed after crossover. Crossover is done so that new generation has better fitness value to survive. Crossover rate should be high about 80%-95% but sometimes it appears that 60% of crossover rate can also serve well.

2.4.2 Mutation Probability

This shows how often the part of chromosome is mutated. If no mutation is there, the chromosome remains exact copy of their parent. If mutation is there, part of the chromosome is

mutated. If mutation is 100% entire chromosome is changed, 0% means nothing is changed. Mutation rate should be very low, 0.5%- 1% is considered as best. Other parameters of GA are as follows:

2.4.3 Population Size

It shows that how many chromosomes are there in the search space. If the population size is very small then after performing a certain operation, we have a very small search space. If the population size is very large then GA slows down. Good size of population is 20-30 whereas sometimes 50-100 gives better result. Research shows that appropriate population size depend on the encoding and the size of encoding string.

2.5 Encoding

Encoding of chromosomes greatly depends on the problem. Some of the popular encoding schemes are presented below [22, 25]:

2.5.1 Binary Encoding

Most commonly used encoding techniques uses strings of 0 and 1. For small number of alleles it gives a large number of chromosomes. This coding is not good as it faces many problems and sometime correction is needed after crossover and/or mutation. For e.g. Knapsack Problem uses Binary Encoding, where each bit of a chromosomes represent whether the particular thing is present in the knapsack or not.

Chromosome1	1100100101100001
Chromosome2	1000010110001011

2.5.2 Permutation Encoding

When focus is on ordering in a problem then Permutation Encoding is used. For example in Travelling Salesman Problem ordering of cities is the key thing. In Permutation Encoding the fitness of chromosomes is decided by the position of genes. In this encoding scheme some type of correction in crossover and/or mutation is needed to make the chromosomes consistent

Chromosome1	1 5 3 2 6 4 7 9 8
Chromosome2	8 5 6 7 2 3 1 4 9

2.5.3 Value Encoding

This encoding is used where some complicated values such as real no. character or letter or words, is used in the problems. This can be used for developing some specific crossover and/or mutation depending on the problem.

Chromosomes	1.2324 5.8243 0.4556 2.7293
Chromosomes	ABDJEIFJDHDIERHFNCJKNJW
Chromosomes	HI, WHAT DO YOU WANT?
Chromosomes	1 0 1 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 1

An example of the use of Value Encoding method is for finding weights for neural network where real value in chromosomes represents weights in the neural network.

2.5.4 Tree Encoding

This encoding is mainly used for evolving programs where the gene represents programming language commands, mathematical operations and other components of the program. Programming language LISP is used since in this the program can be easily parsed which makes the crossover and/or mutation relatively easier.

2.6 Selection

Chromosomes are selected from the population for producing children. Selection of chromosomes is done keeping in mind “survival of the fittest” i.e. the chromosomes having high fitness value are selected to produce new offspring with expectedly high fitness value. Selected chromosomes then undergoes crossover and/or mutation to produce offspring for new generation. This selection operation is governed by various methods; some of them are shown below [23,27]:

2.6.1 Roulette Wheel Selection

Chromosomes with higher fitness value is selected. In roulette wheel all population is placed according to their fitness and then a chromosome is selected at random. Chromosomes having higher fitness value are selected quite often. In this case, the other chromosomes will get very few chances to get selected. This method is also known as stochastic sampling with replacement.

2.6.2 Rank Selection

In the rank selection method, first the population is ranked, then the fitness value is decided by the rank associated with each chromosomes. The worst chromosomes have fitness 1; second worst have fitness 2 and so on. The best chromosomes have fitness N. After this all chromosomes have a chance to get selected but this can lead to slower convergence since the best chromosomes have very little difference between them.

2.6.3 Steady State Selection

In this method the chromosomes with high fitness value are selected to produce offspring. These newly generated offspring replaces chromosomes having smaller fitness value to form a new generation. In this selection method the main focus is on surviving a large part of chromosomes for next generation.

2.6.4 Tournament Selection

Tournament selection is a technique for selecting an individual for crossover in a genetic algorithm. In this method, various tournaments are held between the individuals chosen at random from the given population. The winner of each tournament is selected for crossover. The chosen individual should be removed from the population so that it is not selected again.

2.6.5 Elitism

In this method we preserve the best chromosomes i.e. chromosomes having high fitness are copied to the next generation, as they may lose during crossover and/or mutation. The rest is

done in usual way. Since Elitism preserved the best chromosomes, this greatly increases the performance of GA.

2.6.6 Linear Ranking Selection

In this method the individuals are ranked according to their fitness value. Individuals having higher fitness value have higher rank and individuals with lower fitness have lower rank. Then the probability of selection of individuals is linearly dependent on their rank.

2.7 Crossover and Mutation

Crossover and Mutation are two operators of Genetic Algorithm. The type of operator used in a particular problem is decided by the encoding scheme used and the problem itself. There are a number of operator, few of them are discussed below based on the encoding scheme used [21, 22, 24].

2.7.1 Single point crossover

A crossover point is selected in chromosomes, from first chromosomes first part is taken and the rest is taken from the second chromosomes.

Chromosomes1	Chromosomes2	Offspring
10010|110	01010|011	**10010**011

2.7.2 Two point crossover

In this kind of crossover, two crossover points are selected. Till first crossover point, bits are copied from first chromosomes, from first to second crossover point bits are copied from second chromosomes again first chromosome is used for coping the bits after the second chromosomes.

Chromosomes1	Chromosomes2	offspring
10|010|**110**	01|011|011	**10**|011|**110**

2.7.3 Uniform crossover

It can be considered as the generalization case of single point and two point crossover. In this case bits are randomly copied from the two chromosomes to produce an offspring since each bit has an equal chance of being chosen from either parent. Uniform crossover generates a random value between 0 and 1 for each gene. If the value exceeds the locus crossover probability then only the genes are exchanged otherwise they are just copied from their parents.

2.7.4 Heuristic crossover

This uses the fitness of the two parent chromosomes to determine the search direction. The offspring's are generated according to the following equation.

Offspring1 = Best parent + r*(Best parent – Worst parent)

where r is a number randomly chosen between 0 and 1.

Offspring2 = Best parent

2.7.5 Mutation

After crossover is performed, mutation can be done. It is used to avoid the process of getting trapped in a local optimum. Using mutation new offspring is randomly changed at any location. For binary encoding, in mutation, bits are randomly flipped i.e. at some position e.g. 1 flips to 0 and vice-versa. Mutation randomly changes the genetic information. When operated at bit level, it is possible that mutation occurs at each bit but this has very lower probability as defined by mutation probability (P_m).

Some of the types of mutation operator are as follows [gappr5]:

- **Flip bit:** Simply selected bits are inverted. This mutation operator is used in Binary encoding scheme.
- **Boundary:** In this the randomly selected genes are replaced by the lower or upper bound for that gene.
- **Non uniform:** This mutation operator increases the probability that the amount of
- mutation will be close to 0 (zero), which is a good choice. This also keeps the
- population from stagnating in early stage of evolution and therefore gives the fine
- solution at later stage therefore gives the fine solution at later stage.

- **Uniform:** In this, the selected genes are replaced by the uniform random value chosen between user defined upper and lower bound for that genes.
- **Gaussian:** In this a unit Gaussian distributed random value is added to the chosen genes. If the newly generated gene falls outside the upper and lower bound for those selected genes, it is clipped.

It is to be noted that all the above mutation operators are used for integer or float genes except the Flip bit mutation operator.

Chapter 3

The Proposed Model

3.1 Introduction

Cloud is a distributed computing model with heterogeneous resources to satisfy the user's demands. Since cloud is a business model, the resources are required to be optimally utilized to provide a good economy of scale. There are a number of resources which are used in the cloud computing model such as servers, CPUs, software, applications etc. For running any of these resources the cloud needs a power supply. In other words, the power supplied to a cloud is also a resource [29, 30].

Power consumed over a period of time is termed as energy. One of the rising concerns in the field of modern computing is minimizing the energy consumption as it is a costly resource and is limited too (non-renewable sources). The question arises how can we minimize the energy consumption? There are many factors which affect the level of energy consumption one of which is scheduling [29, 31]

Scheduling is at the core of the processes which are responsible for utilizing the deployed resources to a reasonable level. If the resources consuming energy are utilized properly, the energy consumption will also be minimized to a certain extent. In other words, energy consumption can be directly linked to the scheduling strategy of a cloud computing model.

Most of the times GA is preferred for such kind of highly complex problem, since it is based on natural selection and evolution process due to which it does not need to know the rules about the problem but uses its own methods to give a sub-optimal solution closer to the exact one.

This chapter focuses on Energy Efficient Genetic Algorithm based Scheduler for Cloud Computing. The strategy suggested in this chapter focuses on allocating the resources on the cloud in such a fashion which minimizes the energy consumption using Genetic Algorithm.

The submitted job is divided in to sub-jobs (task) which are dependent on each other. Since different resources in the cloud would have different energy specifications and each task will take different amount of time on various resources with different energy consumption. In

scheduling various tasks are mapped on to the available cloud resources considering the precedence of the job module as indicated by its directed acyclic graph (DAG) [30, 31].
The model uses a centralized scheduler for scheduling the jobs using Genetic Algorithm. The chapter starts with the presentation of the scheduler, the data structures used and the notation considered for design of the model. Later, the scheduling algorithm is presented followed by the simulation results and their analysis.

3.2 Proposed Scheduling Strategy Using Genetic Algorithm

The proposed scheduling model uses Genetic Algorithm to minimize the energy consumption of a job on certain set of processors. To solve any problem using GA, we first need to formulate its mathematical model in terms of a function. Then parameters that optimize the function of the model are determined. This is known as fitness function also known as the Objective function. Fitness function basically determines or analyzes the genes holding the data and returns the fitness value to quantify the fitness or suitability of the chromosomes. This results in selection of chromosomes having higher fitness value for producing the next generation. The better is the fitness, more chances are there of selecting those chromosomes to survive. The chromosomes having poor fitness value are discarded. The fitness function varies from problem to problem [30].

A cloud is a heterogeneous computing paradigm consisting of several hundred or thousands of processing nodes. These resources are divided into various groups or clusters with the help of virtualization. The proposed model considers a cluster of processors on which a job is to be scheduled. The processors in the cluster may have different attributes such as voltage, frequency, processing speed etc. [32].

The job is submitted in the form of a DAG where the nodes represent tasks (dependent sub-jobs) and the edges between the nodes represent the communication between the nodes. The weight of the edge represents the communication cost. The scheduling starts with random generation of a chromosome whose entries contain the node number or the task number and indices contain the processor number meaning that the tasks are allotted to the processors randomly. The energy for this allocation is calculated and the remaining procedure of GA as discussed above is applied. The energy model, data structures used, notation used and fitness function are discussed in the following section.

3.2.1 Energy Model

The total energy consumption of any heterogeneous computing system depends on the factors such as processors, networks, disks, memory, cooling system, fans and other various components. A lot of research work is being done on system energy consumption, only processor's energy is considered in this work since processors consume a major part of the total energy of system. The power consumption of a processor comprises of two parts namely static part and the dynamic part. The static part is the leakage power of a circuit and dynamic power is the switching power of a circuit [31, 32].

The power consumptionof a processor ***p*** under execution is given by [32]:

$$P_{total} = P_{static} + P_{dynamic} \quad (1)$$

where $P_{total}$is the total power of the processor, P_{static} is the static power consumption which is a constant, and $P_{dynamic}$ is the dynamic power consumption. When processor ***p***is in the idle mode $P_{total} = P_{static}$. When processor ***p*** is operated at a voltage level ***v***and frequency level ***f*** the dynamic power$P_{dynamic}$ can be calculated using the equation

$$P_{dynamic} = c * v^2 * f \quad (2)$$

where c_i is a constant known as physical capacitance of the processor p_i.. Power consumed over a period of time is known as energy which can be calculated as

$$Energy = P_{total} * T \quad (3)$$

where ***T*** is the time for which a task is executed on processor ***p***.

3.2.2 Data Structures Used

- ***DAG[no_task][no_task]*:** This matrix represents sparse graph having values 0 and 1, where 1 means there is a connection between the nodes and 0 means there is no connection between the corresponding nodes.
- ***ITC[no_task][no_task]:***It holds the inter-task communication cost between the different tasks in terms of time units.

- ***Chromosomes*:** The value in a chromosome is a mapping of a task to a processor. This kind of mapping makes possible to form a large set of chromosomes known as population.
- ***Population[no_chromosome][no_task]:***This matrix is a population matrix having rows equal to the no. of chromosome and column equal to the number of tasks in the job.
- ***Prospecs[no_processors][3]:***This matrix contains the specifications of the processors namely physical capacitance, voltage and frequency respectively.
- ***Power[1][no_processors]:***This matrix stores the power of each processor calculated from the prospecs matrix.
- ***Sequence[1][no_task]:***This matrix stores the order in which the tasks are to be executed since the job submitted consists of dependent tasks.
- ***Time[no_task][no_processors]:***This matrix stores the time taken by each task on each processor.
- ***MinEnergy[no_generations][1]:***This matrix stores the minimumenergy of each generation. Each row contains only one value which is the minimum energy of that generation.
- ***MinEnergySchedule[no_generations][no_task]:***This matrix contains the schedule corresponding to minimum energy for each generation.

3.2.3 Notations Used

- ***no_task:***Number of tasks in the directed acyclic graph (DAG)
- ***no_chromosome:*** Number of chromosomes in the population
- ***no_processors:***Number of processors in the cluster
- ***no_generations:***Number of generations for which GA works
- ***no_ins:*** Number of instructions
- ***size_chromosome:***size of chromosomes which is equal to the number of tasks
- ***TAT:*** Turnaround time
- $P_{dynamic,i}$**:** dynamic power of processor i
- $P_{static,i}$**:** static power of processor i
- p_i **:**Processor i
- c_{ij} **:** physical capacitance of processor i while executing task j

- v_{ij} :Voltage of processor i while executing task j
- f_{ij} **:** Frequency of processor i while executing task j
- t_{ij} **:**Execution time of task j on processor i
- tk_j**:**Represents task j
- C_i**:**Represents the communication cost of the tasks which are predecessors of the task tk_j

3.2.4 Fitness Function

The model considers a task to be allocated on a processor which results in minimum energy consumption. Since we have calculated the time for each task on each processor, the tasks should be allocated in such a way that they are scheduled on a processor which takes least time to execute it. The dynamic power of a processor p_i is given by equation (2). The static power of any processor is constant which need not be calculated. Considering these factors, the energy for a chromosome (schedule) can be calculated as:

$$Energy = \sum c_{ij} v^2_{ij} f_{ij} * (t_{ij} + max(C_i)) + (\sum P_{static,i}) * TAT \quad (4)$$

Here the first summation in equation (4) calculates the dynamic energy and the second summation calculates the static energy. The dynamic power is multiplied by the time for which a tasktk_j executes on the processor $p_i$which gives the energy and finding its summation gives the total dynamic energy for all the tasks. Static power is constant so its summation is calculated and is multiplied by *TAT* the turnaround time of the task.The above equation acts as the fitness function or the objective function as we need to minimize this energy using GA over generations. The energy is calculated for each chromosome in the population, the minimum energy is recorded and the schedule (chromosome) corresponding to that minimum energy is recorded, this is repeated for all the generations until the termination condition is met.

3.3 The Proposed Algorithm

The proposed algorithm schedules the job in the cloudso that the energy consumption can be minimized. The proposedstrategy being GA (Genetic Algorithm) is based on using Roulette wheel selection for selecting the chromosomes followed by performing the operations like crossover and mutation, if needed, on the population of chromosomes generated randomly. After performing the above mentioned operations the chromosome with minimum *Energy* value is

obtained, which offers the way, scheduling is done so as to optimize the resources. The algorithm is described in detail as below.

- The cloud is considered to be having many clusters out of which one or a few clusters may be selected for evaluation of job execution.
- For any job submitted the cluster is evaluated for *Energy* of the all tasks in thejob. The scheduler finally schedules the job on the cluster according to the schedule which results in minimum *Energy*.
- For the job submitted a population of chromosomes is generated.
- For the population of chromosomes generated,*Energy*is calculated which gives the energyfor each chromosome of the population.
- Randomly select two chromosomes from the parent population using roulette wheel selection. If the selected chromosomes are same then repeat the selection process till chromosomes obtained remain the same. Crossover operation is performed next on the selected chromosomes to produces two child chromosomes. Here, crossover performed is single point crossover. These children are then stored in a new variable named population1. If child chromosomes generated after crossover are already there in the population1 then do not store these child chromosomes but repeat the same process again. This gives a new population of child chromosomes. Now calculate the *Energy*of each chromosome. Next, pool the parent and child chromosomes together and out of them select the best chromosomes to form the next generation population having same no. of chromosomes as that of parent chromosomes.
- Perform mutation on the population if it satisfies the necessary condition. Mutation is performed after a fixed no. of generations. In this experiment the scheduling strategy is evaluated for mutation being performed after 5^{th}, 10^{th} and 15^{th} generation respectively. Not all population is selected for mutation. The no. of chromosomes being mutated is varied from 25%, 50% and 100% population respectively. This means for a generation when mutation is being performed
- Every 5^{th} generation, at first 25% population is mutated and the mutated chromosomes replaces its parent chromosomes. Similarly, it is done using 50% and 100% of the population. If child generated after mutation is already there in the new population then

repeat that step again until different chromosome is not obtained. This step is repeated for all generations and *Energy*is repeatedly calculated for the new population.

- The population generated in this generation is used as parent population for the next generation and so on.
- Repeat the above steps for the no. of generations needed to optimize the obtained result, till the termination condition is reached.

```
Schedule(job)
{
Submit the job in the form of a DAG
Select the virtual cluster(s) for evaluation
For each cluster
  {
Calculate the execution time of each task on every processor
    // Generate ECT matrix
Generate a population of chromosomes randomly
   // No. of chromosomes = 50-100
While (terminating condition is not met)
{
Calculate energy for each chromosome in the population set
       // Using the fitness function using the equation (4)
Perform selection
       // Roulette wheel selection
Perform crossover
      // Single point crossover
Perform mutation
// Every 5th, 10th and 15th generation for 25%, 50% and 100% of population
// No. of genes mutated is equal to 10% of the size of the chromosomes
}
Record the chromosome which corresponds to minimum energy
// Estimate the turnaround time for the chromosome offering minimum energy
}
Select the cluster with chromosome offering the minimum energy consumption
Schedule the job according to the chromosome recorded above
}
```

3.4 Illustrative Example

An illustrative example is presented here to make the model easy to understand. This shows the basic working of the model in terms of calculating the *Energy* for each chromosome for available cloud resources. The same method is used to calculate *Energy*for all the other chromosomes in the population and finally the minimum energy. In this illustrative example, the parameters have been scaled down for better understanding of the working of the model. The illustration explains the analysis of the suitability of the jobs on one such cluster. The same method can be adopted to find the *Energy* offered to the job by other clusters.

Let us consider that initially the cloud comprises of certain set of resources. Let's say we consider a cluster with four processors. The job submitted at any time at any cluster can berepresented in the form of a DAG as shown below. A typical job for execution in theform of DAG is represented in Figure 3.1.

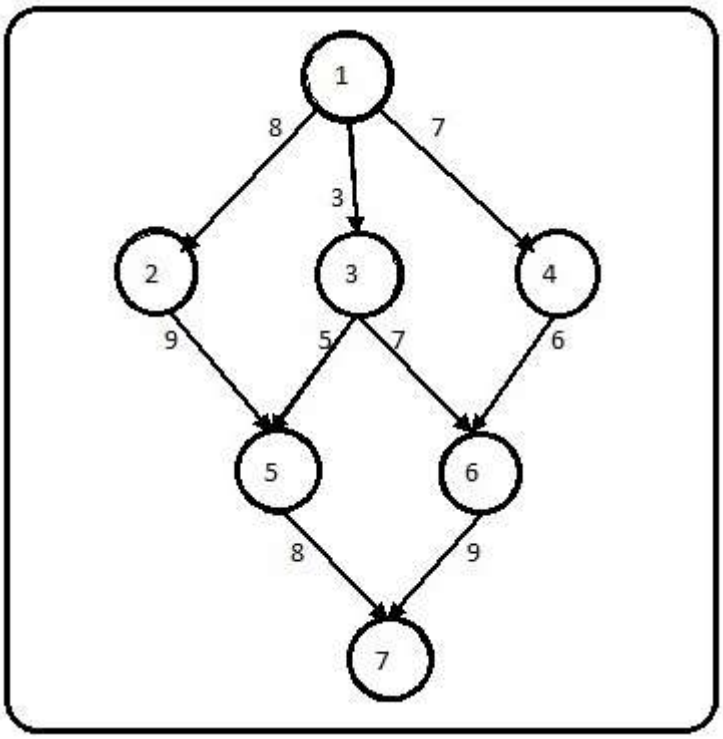

Figure 3.1 Directed Acyclic Graph

The nodes of this DAG are the tasks represented by the number within the circle. The edges represent the inter-task communication. The details of the job is given in Table 3.1.

Table 3.1

Job Specifications

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
No. of instructions (million)	52	48	41	35	47	38	42

The inter-task communication (ITC) can be represented in the form of a matrix. The matrix is represented in the Table 3.2.

Table 3.2

Inter-task Communication matrix

Task	t_1	t_2	t_3	t_4	t_5	t_6	t_7
t_1	0	8	3	7	0	0	0
t_2	0	0	0	0	9	0	0
t_3	0	0	0	0	5	7	0
t_4	0	0	0	0	0	6	0
t_5	0	0	0	0	0	0	8
t_6	0	0	0	0	0	0	9
t_7	0	0	0	0	0	0	0

The specifications of the processors such as physical capacitance (c), voltage (v) and frequency (f) are stored in the matrix shown in table 3.3.

Table 3.3

Parameters of the Processors

Processors	***Capacitance (c) * 10^{-8}***	***Voltage (v)***	***Frequency (f) * 10^{9}***
P1	3.80	1.12	1.80
P2	4.08	1.34	2.40
P3	3.59	1.68	3.00
P4	2.96	1.57	2.80

Since each processor has different frequency, every task will take different time on each processor to execute. The execution time can be calculated by dividing the number of instructions in the task by the clock frequency of that particular processor. The time foreach task on every processor has been calculated using this method and stored in the *time*matrix shown in Table 3.4. The time is in milliseconds.

Table 3.4

Expected Computational Time Matrix

Tasks	***P1***	***P2***	***P3***	***P4***
t1	28.88	21.66	17.33	18.57
t2	26.66	20.00	16.00	17.14
t3	22.77	17.08	13.66	14.64
t4	19.44	14.58	15.66	16.78
t5	26.11	19.58	15.66	16.78
t6	21.11	15.83	12.66	13.57
t7	23.33	17.50	14.00	15.00

The dynamic power of a processor can be calculated using equation (2). After calculating the dynamic power it is stored in the matrix which is shown in Table 3.5.

Table 3.5

Dynamic power matrix

	P1	***P2***	***P3***	***P4***
Power(dyn)	85.80	175.82	303.97	204.00

The static power of a processor is constant though it is different for every processor. The static powers of the processors are stored in the matrix which I shown in Table 3.6.

Table 3.6

Static power matrix

	P1	***P2***	***P3***	***P4***
Power (static)	37.2	48.7	65.2	54.7

The chromosome is an array which contains the information about how to allocate the jobs to resources. The structure of a chromosome is like that of an array. The entries in the array contain the processor id in which a task is to be scheduled. The indices of the array correspond to the task id. For example the entry in the 3rdindex is 4 which means that the 3rd task is to be scheduled 4th processor.

Let's say we have a chromosome in the population as shown below:

3	2	4	1	2	1	4
1	***2***	***3***	***4***	***5***	***6***	***7***

Since the 1st task is scheduled on the 4th processor we need to calculate the time for which the task executes on this processor. This is the starting node so its communication time is 0. The data is shown in Table 3.7.

Table 3.7

Calculated values	
Task1	***Processor3***
cv^2f	303.97
t_{ij}	17.33
$P_{static,i}$	65.2

From the data in Table 3.7 the calculated value of dynamic energy comes out to be 5.26. It is to be noted that the time is in milliseconds.Similarly the dynamic energy of task 2 can be calculated from the data given in Table 3.8. The static power would be calculated in the end when all the tasks have completed their execution and it is known that for how long the task was scheduled on the processor.

Table 3.8

Calculated values	
Task2	***Processor2***
cv^2f	175.82
t_{ij}	(20.00+8)
$P_{static,i}$	48.70

The calculated valued for dynamic energy for task 2 comes out be 4.90. Similarly the calculated values of dynamic energy for remaining tasks can be calculated. The value for task 3 comes out to be 3.59. For task 4 it is 2.26. For task 5 it comes out to be 5.02 and for task 6 it is 2.32. Finally for task 7 it comes out to be 5.10. The total dynamic energy comes out to be 28.45 joules.

The turnaround time comes out to be 99.91. The total static power $\boldsymbol{P_{static}}$of all the processors is calculated to be 205.8. Thus the total static power comes out to be 20.56 joules. Adding the

dynamic and static energies for this chromosome gives the total value of energy which comes out to be 49.01 joules. Similarly the values of energies of all the other chromosomes can be calculated. This process is repeated for the entire population until the termination condition is met.

3.5 Simulation Study

Simulation experiments were conducted to observe the scheduling of the job on the selected cluster. The experiment is conducted on Intel Core-i3 @1.8 GHz using MATLAB 7.6.0 (R2008a). The data values taken in the experiment are generated dynamically during execution.

Table 3.9

Parameters Used			
S.No.	***Parameter***	***Notation Used***	***Range***
1	No. of Resources/Processor	no_processor	5-20
2	Clock frequency of processor	f	10-20
3	No. of tasks in a job	no_task	5-50
4	No. of instructions in a module	no_ins	100-500
5	Inter-task communication	ITC	2-8
6	Population Size	no_pop	50-200
7	Size of chromosome	size_chromosomes	5-50
8	No. of generation	no_generation	100,200
9	Crossover considered during experiment	Single point	--
10	Mutation performed after no. of generations	Mu	5,15,20
11	% of Population selected for Mutation	--	25%,50%,100%
12	Rank selection method	Roulette wheel selection	--

The experiment is performed for 50, 100, 150, and 200 tasks in job. The number of generations for which the algorithm runs is 200. The size of population is taken to be 100. The variations of energy and turnaround time for different experiments are plotted below.

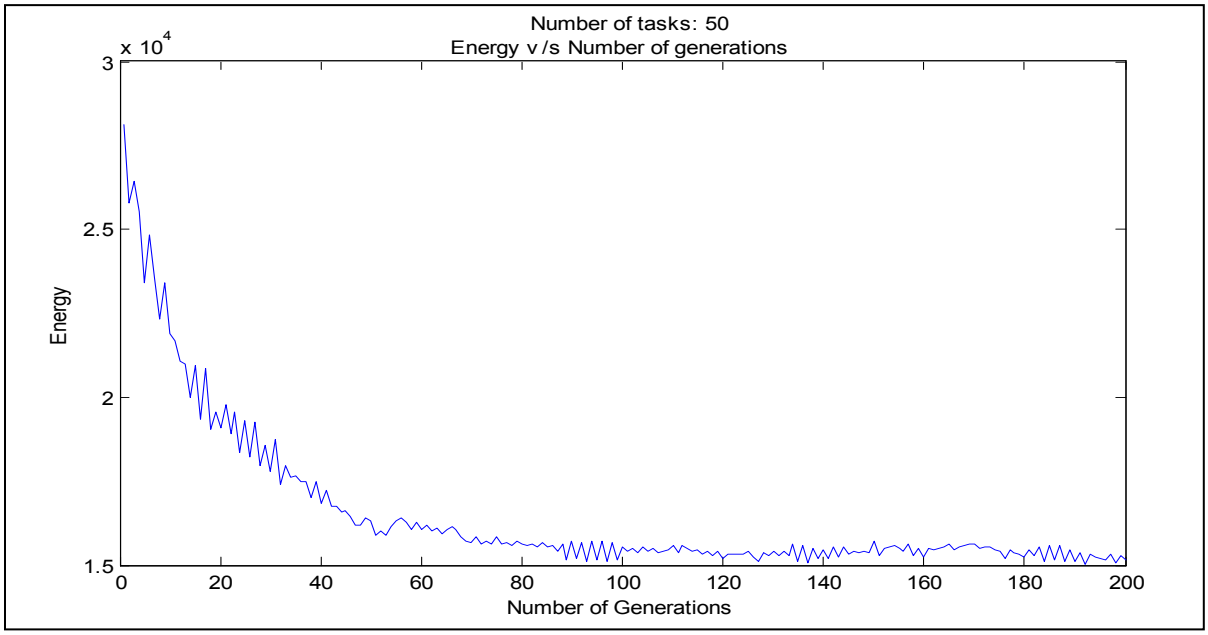

Figure 3.2: Energy Consumed v/s Number of Generations for 50 tasks

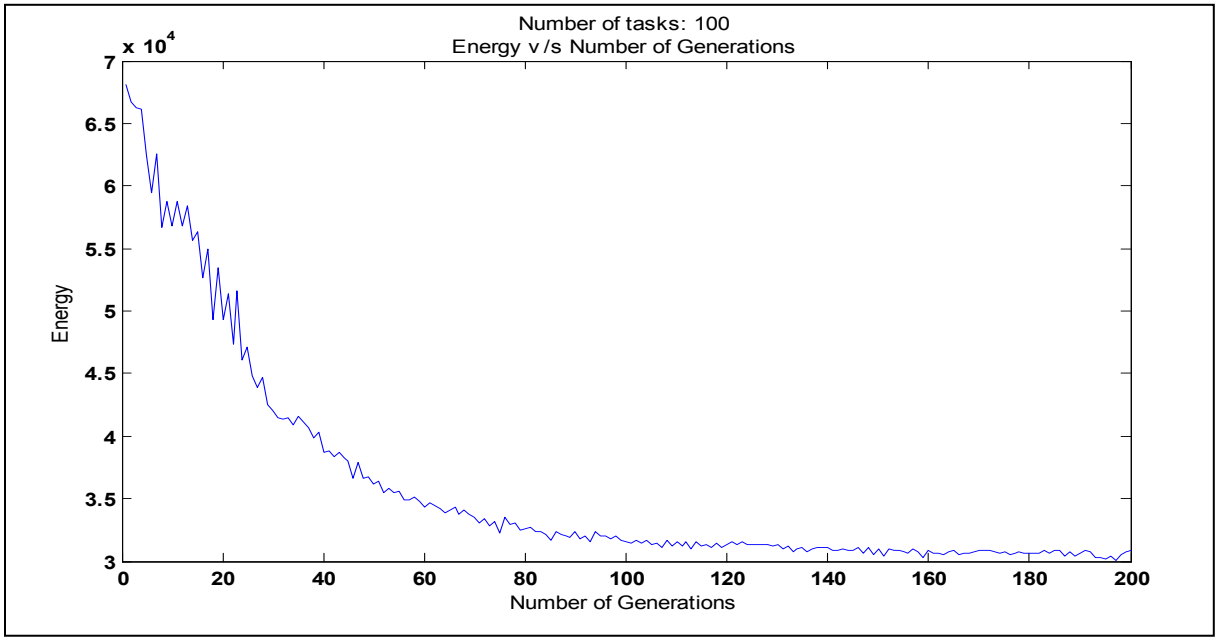

Figure 3.3: Energy Consumed v/s Number of Generations for 100 tasks

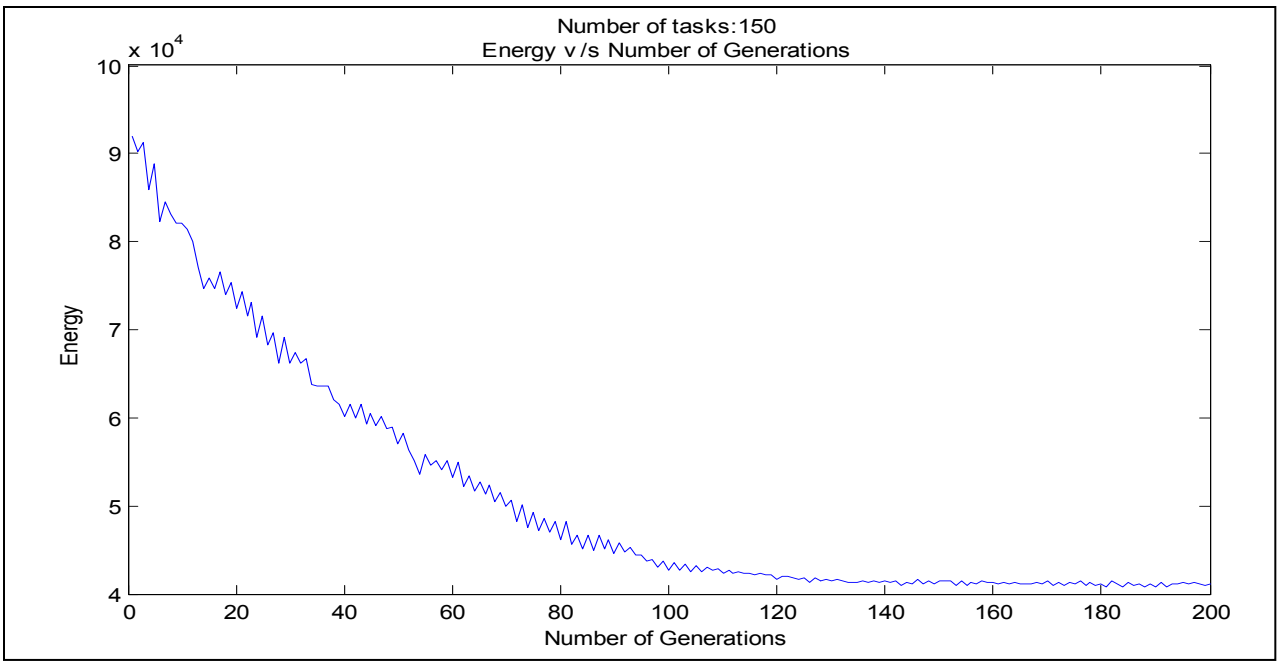

Figure 3.4: Energy Consumed v/s Number of Generations for 150 tasks

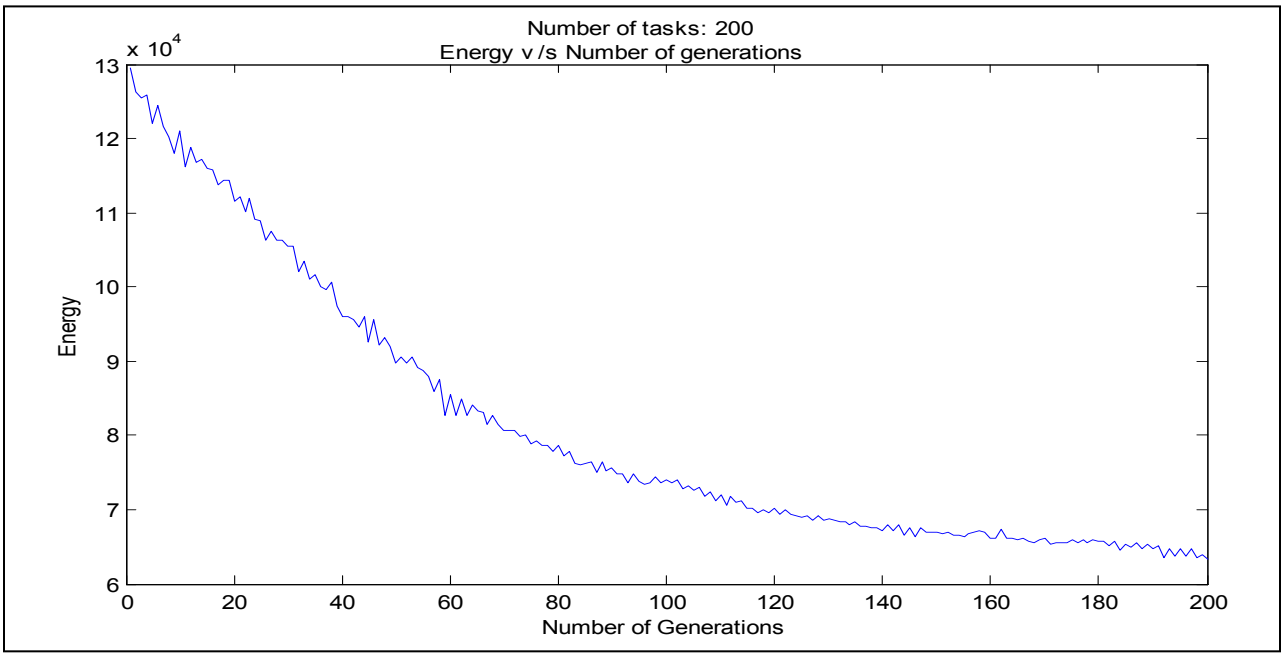

Figure 3.5: Energy Consumed v/s Number of Generations for 200 tasks

3.6 Observations

After the experiments were carried out, the results were recorded and are plotted above. This section deals with the analysis and observations of the results.

From Figure 3.2 which shows the energy graph for 50 tasks, it can be observed that the energy values decrease quickly up to 50 generations and then gradually up to 100 generations. After 100 generations the energy value decreases at very slow rate. The rate of convergence in the beginning is highest which reduces in the middle and goes very low towards the end.

From Figure 3.3 which shows the energy graph for 100 tasks, it can be observed that the energy values decrease quickly up to 70 generations and then gradually up to 120 generations. After 120 generations the rate at which the energy value decreases is very low. The rate of convergence for 100 tasks is lower as compared to that of 50 tasks.

From Figure 3.4 which shows the energy graph for 150 tasks. It can be noticed that the energy values decrease at a high rate until 100 generations and then it decreases gradually up to 140 generations. After 140 generations energy value decreases very slowly which almost becomes constant after a point. It can be observed that the rate of convergence for 150 tasks is less than that for 100 tasks.

From Figure 3.4 which shows the energy graph for 200 tasks. It can observed that the energy values decrease quickly until 100 generations and then decreases slowly between 140 and 180 generations. It can be noticed that the rate of convergence of 200 tasks is the slowest among the set owing to the larger batch size.

Therefore, it can be concluded that the model tries to converge to the allocation which results in a better energy consumption over the generations. Further, as the number of tasks increases the rate of convergence decrease. The results have been found to work satisfactorily converging within 200 generations.

Chapter 4

Conclusion and Future Scope

Cloud computing is a dynamic environment in which resources can leave or join the system at any time. This behavior makes the scheduling a tough job. Job scheduling in cloud system is an NP-Hard combinatorial optimization problem. Therefore, traditional methods are not suitable for such kind of complex problems as they have certain constraints such as they do not exploit the tolerance for imprecision and may not give the solution within the time constraints. Therefore, soft computing techniques like genetic algorithm, fuzzy logic and artificial neural network finds a wise use for such type of problem.

The proposed model uses genetic algorithm (GA) for scheduling a job on the cloud system so that the energy of the jobs that need to be executed on the system can be minimized. Here, the study is based on roulette wheel selection method for a single point crossover and mutation implemented after certain number of generations for certain percentage of the population. From the obtained results it can be said that allocations of tasks is done in such a way that the energy consumption is minimized. Tasks are allotted to those processors frequently which have such specifications that result in minimal energy consumption. Over the generations, energy value for the entire job reduces considerably. Further, simulation study reveals that the rate of convergence of the model depends on the number of tasks submitted for execution. As the number of tasks in the job increases, the rate of convergence decrease.

Since, scheduling is an NP-hard problem, the results obtained cannot be treated as the most optimized results. Accordingly, the proposed model can be tested for different selection, crossover and mutation schemes and compared with established peers for performance evaluation. Further, it also opens the possibility of the use of some other soft computing approaches like particle swarm optimization or ant colony optimization to be used and compared for performance evaluation. Multi objective functions or parameters can also be explored by optimizing more than one parameter.

References:

1. M. Sajid and Z. Raza, "*Cloud Computing: Issues And Challenges*", International Conference on Cloud, Big Data and Trust RGPV, pp. 35-41, Nov 13-15, 2013.
2. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "*Cloud Computing And Emerging IT Platforms: Vision, Hype, And Reality For Delivering Computing As The 5th Utility",* Future Generation Computer Systems, vol. 25, no. 6, pp. 599–616, June 2009.
3. T. Dillon, C. Wu and E. Chang, "*Cloud computing: Issues and challenges*", 24th IEEE International Conference on Advanced Information Networking and Applications. pp. 27-33, 2010.
4. D. A. Menasce and P. Ngo, *"Understanding Cloud Computing: Experimentation And Capacity Planning*," in Proc. Of Computer Measurement Group Conf., pp. 1-11, December 2009.
5. *http://www.cloudscaling.com/blog/cloud-computing/the-evolution-of-it-towards-cloud-computing-vmworld/*
6. *http://www.thoughtsoncloud.com/2014/03/a-brief-history-of-cloud-computing/*
7. I. Foster, Y. Zhao, I. Raicu, S. Lu, *"Cloud Computing and Grid Computing 360-Degree Compared*".
8. M. A. Vouk, "Cloud computing – Issues, research and implementations", Journal of computing and information technology, Vol. 16, no. 4, pp. 235-246, June- 2008.
9. *http://networkenhancers.blogspot.in/2012/08/how-does-cloud-computing-work.html*
10. *http://www.cloudcontrols.org/cloud-standard-information/cloud-definitions/*
11. M. N. Huhns and M. P. Singh, "*Service-Oriented Computing: Key Concepts and Principles*", IEEE Internet Computing, vol. 09, pp. 75 - 81, 2005.
12. *https://www.linkedin.com/pulse/20140921193928-23699310-cloud-computing-benefits-and-challenges.*
13. *http://drona.csa.iisc.ernet.in/~gsat/Course/DAA/lecture_notes/jeff_nphard.pdf.*
14. *http://www.cse.unl.edu/~goddard/Courses/CSCE310J/Lectures/Lecture10/NPcomplete.pdf.*
15. "Cloud Computing from Wikipedia, the free encyclopedia", *https ://en.wikipedia.org/ wiki/ Cloud_computing/, 2015.*

16. Wigderson, A., "*P, NP and Mathematics – a Computational Complexity Perspective*", December 21, 2006.
17. M. Nelson, "*Building an Open Cloud*," Science, vol. 324, p. 1656, 2009.
18. Y. Chen, V. Paxson, and R. Katz, "*What's New About Cloud Computing Security?*," 2010.
19. I. Foster and C. Kesselman, "*The Grid: Blueprint for a New Computing Infrastructure*" Morgan Kaufmann, 1998.
20. T. Harmer, P. Wright, C. Cunningham, and R. Perrott, "*Provider Independent Use of the Cloud,*" in The 15th International European Conference on Parallel and Distributed Computing, 2009, p. 465.
21. M. Melanie "*An Introduction to Genetic Algorithms*", First MIT Press Paperback Edition, 1998.
22. D.E. Goldberg, "*Genetic Algorithms in Search Optimization and Machine Learning*", Addison-Wesley, Reading, MA, 1989.
23. D.E. Goldberg, "*Sizing Populations For Serial And Parallel Genetic Algorithms*", in: J.D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms, 1989, pp. 70-79.
24. J. N. Amaral, K. Tumer, and J. Ghosh, "*Designing Genetic Algorithms For The State Assignment Problem*",IEEE Trans. Syst., Man, Cybern., vol. 25, no. 4, Apr. 1995.
25. C. M. Fonseca and P. J. Fleming, "*Genetic Algorithms For Multi Objecitve Optimization: Formulation, Discussion, And Generalization*", in Genetic Algorithms: Proc. Fifth Int. Con$, S. Forrest, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 416-423.
26. M. R. Garey and D. S. Johnson, "*Computers and Intractability: A Guide to the Theory of NP-Completeness",* San Francisco, CA: Freeman 1979.
27. W. M. Spears and K. DeJong, "*An Analysis Of Multi-Point Crossover*", in Foundations of Genetic Algorithms, G. J. E. Rawlins, Ed. 1991, pp. 301-315.
28. *http://www.webopedia.com/TERM/C/cloud_computing.html*
29. D. Li and J. Wu, ''*Energy-Aware Scheduling For Frame-Based Tasks On Heterogeneous Multiprocessor Platforms*'', in Proc. 41st ICPP, pp. 430-439.

30. R. Kaur and S. Singer, "*Enhanced Genetic Algorithm Based Task Scheduling In Cloud Computing*", International Journal of Computer Applications (0975 – 8887) Volume 101– No.14, September 2014.
31. S. Sindhu and S. Mukherjee, "*A Genetic Algorithm Based Scheduler For Cloud Environment*", 4th International Conference on Computer and Communication Technology (ICCCT), pp. 23-27, 2013.
32. K. Li, X. Tang, and K. Li, "*Energy-Efficient Stochastic Task Scheduling on Heterogeneous Computing Systems*", IEEE Transactions On Parallel And Distributed Systems, Vol. 25, No. 11, pp. 2867-2876, November 2014.
33. Y.C. Lee and A.Y. Zomaya, ''*Energy Conscious Scheduling for Distributed Computing Systems Under Different Operating Conditions*'', IEEE Trans. Parallel Distrib. Systems, vol. 22, no. 8, pp. 1374-1381, Aug. 2011.
34. I. Takouna, W. Dawoud, and C. Meinel, ''*Energy Efficient Scheduling of HPC-Jobs on Virtualized Clusters Using Host and VM Dynamic Configuration*,'' Proc. ACM SIGOPS Oper. Syst. Rev., vol. 46, no. 2, pp. 19-27, July 2012.
35. L. Wang and Y. Lu, ''*Efficient Power Management of Heterogeneous Soft Real-Time Clusters*'', in Proc. Real-Time Syst. Symp. , Barcelona, Spain, pp. 323-332, 2008.