Cloud Computing Service Selection Algorithm

Rohan Patra

Shashwat Shikshu

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela-769 008, Odisha, India

Cloud Computing Service Selection Algorithm

Thesis submitted in partial fulfilment
of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Rohan Patra
(Roll: 110CS0140)

Shashwat Shikshu
(Roll: 110CS0483)

with the supervision of

Prof. Manmath Narayan Sahoo
NIT Rourkela

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India

May 2014

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India.

May 12, 2014

Certificate

This is to certify that the work in the thesis entitled ***Cloud Computing Service Selection Algorithm*** by ***Rohan Patra*** and ***Shashwat Shikshu*** is a record of an original research work carried out with my supervision and guidance in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Prof. M. N. Sahoo
Assistant Professor
Department of CSE, NIT Rourkela

Acknowledgment

"We would like to express our deepest and earnest gratitude to our project guide, Prof. M. N. Sahoo for believing in our ability. His profound insights have helped us complete our research work. The flexibility of work and invaluable suggestions that he has offered us has deeply encouraged us and helped us in the successful completion of our project. We are in debt to all our professors, batch mates and friends at National Institute of Technology Rourkela for their cooperation and support as well. Our full dedication to the work would have not been possible without their blessings and moral support."

Rohan Patra

Shashwat Shikshu

Authors' Declaration

I hereby declare that all the work contained in this report is my own work unless otherwise acknowledged. Also, all of my work has not been previously submitted for any academic degree. All sources of quoted information have been acknowledged by means of appropriate references.

Rohan Patra

Shashwat Shikshu
National Institute of Technology, Rourkela

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India
May 2014

Abstract

In modern world Cloud Computing is one of the most promising and evolving areas of computer science. As time passes by more and more cloud devices are being setup. Similarly more companies and industries are opting for cloud services, etc. Cloud has made up a virtual reality of the practical world. It offers online storage space, online infrastructure, online platforms, etc to make our everyday computing experience easier and cheaper. One of the aspects of cloud computing is provision of servers to execute our programs which comes under Infrastructure as a Service (IaaS). In this project we have focused on devising an algorithm to schedule jobs and allocate servers in cloud systems. The algorithm is efficient as it provides optimal allocation. It maximizes the number of job requests that can be processed in unit time while conserving energy and keeping the costs low. The said optimal allocation is achieved by reducing the idle time of nodes of active servers and reducing the total number of servers used. We implemented our algorithm using random data sets of job requests with different attributes and generated simulations in forms of graphs. The graphs prove the efficiency of job scheduling algorithm and the server allocation for which we used Best Fit algorithm of the Bin Packing problem. Finally a detailed analysis is given and future works are stated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cloud Computing, in a simple words, means Internet based Computing. Since the Internet can be thought of as clouds, and therefore the term cloud computing is used. Process execution and computation is done through the Internet. The users who use Cloud can have access to any resource and database with proper authority through the Internet anytime from anyplace and for as long as they need it, without having to worry about any management or maintenance of actual resources. Besides, resources and databases in cloud are usually very scalable and dynamic. Pondering over a brief history we find that:

McCarthy, was the first to propose in the year 1957 that the time sharing of computing resources might allow industries to sell excess computation facilities for maximum and efficient usage of the resource[1]. Licklider, an ARPA programmer, mentioned that sending pieces of a single complex program to a network of individual logically connected systems could share in and pool their individual resources to solve that problem and of course a common language for those computers to be able to communicate with each other needed to be established[2]. Parkhill, in 1966, came out with a paper which identified most of the problems that cloud computing faces, such as scalability and large bandwidth requirement.

Cloud computing isn't like utility computing or grid computing. In fact, when it comes to process execution and data storage it is very independent.

1.1 Grid Computing

Grid computing is defined as the usage of available and linked computer resources from several administrative areas to solve a common problem. It can be thought of as a distributed system with un-interactive workloads consisting of numerous files, but they are more loosely coupled, heterogeneous, and not necessarily geographically constrained when contrasted with cluster computing.

In simple terms, grid computing can be portrayed as a super virtual computer consisting of a network of loosely coupled computers cooperating to perform big tasks.

Grid Computing is often regarded as the precursor of Cloud computing (Grid Computing actually evolved into the cloud). Grid is more useful when allocating resources on-demand (dynamic resource provisioning).

Grid Computing needs a software to diversify and distribute parts of a program (as a single large system image) to a large number of computers. A major drawback of the Grid is that if one part of software on a node fails, other parts of the software on the other nodes may fail. Installing a failover component on other nodes can be use to correct this, but when a few components depend on some other components the same problem may surface. Also, huge capital and operational expenses are incurred to construct and run a grid.

In the late 1980s with the grid computing concept started a paradigm shift towards the cloud. For the first time, several systems were used to solve a single problem, which were scientific and required exceptionally high levels of parallel computing[4].

1.2 Utility Computing

Though utility computing often needs infrastructure like a cloud system, its prime aim is on the business structure on which providing the computing services are based. In simple words, in utility computing service the users receive computational resources, software or hardware, from a service provider and pay as per the basis of usage.

The main advantage of Utility computing is its economical benefits. Corporate data centers are not utilized as much as they are supposed to, with important resources like servers remain idle almost 85 percent of the time[5]. Purchasing more hardware

than is generally required just to handle peak time spans such as the opening of the Wall Street trading day or the holiday shopping season, to handle anticipated future loads and to prepare for unexpected rises in demand[5]. Utility computing allows companies to only shell out money for the computing resources they require, when they require them.

Utility computing is responsible to the business models where application infrastructure resources like hardware and/or software are delivered. Whereas cloud computing is the way we design, create, implement and execute applications that run in a virtualized surrounding. Here they share the existing resources and have the freedom to dynamically expand, shrink and self recover.

1.3 Cloud Computing

Cloud Computing is a generic term that involves delivering hosted services over the Internet. The Internet is often represented by the symbol cloud due to its dynamic structure and hence Cloud computing was inspired by the same[6].

The concept of cloud computing is broader than that of utility computing and relates to the underlying architecture in which the services are designed[5]. Cloud computing can be used to imply internal corporate data centers and utility services. Cloud computing is comparable to software as a service, on-demand services, or the Internet as platform. Cloud computing delivers application or software services from widespread geographic locations instead of a single location. The transfer from locally installed infrastructure and programs to cloud computing is just getting under way in an intense manner[7]. Franco Travostino pointed out one difference between clouds and grids that the cloud was a brain child of the Web 2.0 mindset. Grid was a work of super-computing teams who have an almost religious mindset for dealing with things, which are very complex by nature, without the fear of the complexity, whereas on the other hand clouds came out of a simple mindset[8].

Although it is mind wracking to come up with an accurate and all-inclusive definition of cloud computing, at the core of it is the notion that applications run in some virtual place on the cloud (may be the Internet or an internal corporate net-

work) which the end user doesn't know or care about. But that is not big news: End Users have been using web applications for years without any concern as to where the applications actually run.

The main aspects which helped cloud computing to be adopted were the illusion of infinite computing resources, thereby getting rid of the need to plan ahead for provisioning for Cloud Computing users; getting rid of an upfront commitment by Cloud users. This in turn allows companies to be able to start small and use the required number of hardware resources depending on solely on their real time needs and the capability to pay as they require for use of computing resources on a short term basis and release them as needed[9].

Done right, cloud computing gives the freedom to IT operations and application developers to develop, implement and execute applications that can easily expand or reduce in capacity (scalability), execute faster (performance), and very rarely or never fail (reliability), all without any geographic related concern as to the nature and location of the underlying infrastructure.

Taking it to the next step, this implies that cloud computing architectures, and particularly their middleware and application platforms, should preferably have these properties:

- Self-healing

  A hot backup instance of the application should be ready to take over without disruption (called failover) in case of failure in the cloud. This also means that if a policy, which says everything should always have a backup, is set and a fail occurs so that the backup becomes the primary, the system should start a new backup, maintaining the reliability policies.

- SLA-driven

  The system is dynamically maintained by SLAs that formalize policies such as how quickly responses to requests need to be sent. If the system is experiencing very high load values, it will create extra instances of the software on more nodal servers so that it complies with the agreed service levels even at the expense of a low-priority request[10].

- Multi-tenancy

  The whole system is made in such a way that it allows several users to share infrastructure, without the users being aware of it's internal specifications and without conciliating the security and privacy of every individual user's data.

- Service-oriented

  The software allows building applications from several discrete services that are coupled loosely (basically meaning that they are independent of each other). Changes made to or failure of even one service will not disturb or influence other services. It also means that services can be used again and again.

- Virtualized

  Applications are separated from the underlying hardware. Several applications can be executed on a single computer or several computers can be used to execute a single application.

- Linearly Scalable

  The system will be predictable and very efficient in developing the application. If one server can process 10,000 transactions per second, two servers should be able to process 20,000 transactions per second, and so forth.

- Data

  The main solution to many of these problems is the management of the data: its partitioning, synchronization, distribution and security.

The US National Institute of Standards and Technology (NIST) has developed a working definition that covers the commonly agreed aspects of cloud computing. It summarizes cloud computing as: 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.[11, 12].

This definition describes cloud computing as having five essential characteristics, and two distinct sets of models.

The essential characteristics[12] are:

- On Demand self-service

  Cloud services are on-demand; that is, consumers can automatically request the service based on their needs, without human interaction with the service provider[13].

- Broad Network access

  Accessing cloud services is made easy via the network using generalized interfaces and different access mechanisms. This does not imply that the service must be always available via the internet as this will be based on the operational model used. But it should be possible to access the services as per the policy used[13].

- Resource pooling

  Resources are shared between multiple tenants, and generally assigned exclusively at run time to one consumer at a time based on the needs of the tenant. Pooling resources helps increase utilization, and thus is instrumental in decreasing the operation costs[13].

- Rapid elasticity

  Cloud Computing provides mechanisms to allow quick provisioning and releasing of resources[13]. This combined with a enormous pool of resources in the data centre gives the image of unlimited resources to the users and provides flexibility in their provisioning.

- Measured service

  Cloud Computing provides Mechanisms to measure service usage and health of the system. This enables optimization of resources and provides transparency for both users and providers allowing better utilization of the service[13].

Cloud computing has several applications such as Microsoft's Windows Azure Cloud services is being used by Infosys, which includes SQL Data services, in order to develop

Cloud-based software applications so that sharing of data on inventories can be done by vehicle dealers. Google App Engine is used by Best Buy's Giftag applet to let users make and share wish lists from different pages they surf on the Internet. IBM Cloud services is used by a retailer in China, Wang Fu Jing Department Store, including supply chain management software for its network of retail stores. Virtually limitless resources is offered by Cloud, on-demand, at a comparatively lower expense.

There are distinctively 3 components of the cloud:

Clients

Clients refer to the devices that the end users utilize to interface with the cloud when they require the services of the cloud. They can be personal computers, laptops, smart mobile phones etc. Thin clients are the computers that do not contain internal hard drives and simply display the data from the server. Thick client is a normal computer that connects to the cloud using web browsers like Internet Explorer, Mozilla Firefox etc. Thin clients have emerged as a popular solution because of their lessened price and enhanced information security. Information security is better in case of thin clients as the processing of data and storage takes place directly on the server without involving the client.

Data Center

It is an agglomeration of servers where the application to which the users have subscribed is placed. It can be stored anywhere and can be accessed via the internet. A superior solution is to use virtual servers through a single physical server. A software can be installed that permits multiple instances of virtual servers to run whenever the physical server is accessed.

Databases

The information or data is stored at these places in the cloud. The storage units can consists of several servers stored in a single place like the Facebook's data storage or it can extend over a widespread area with several servers around the world connected with each other.

The two distinct sets of models cloud computing can be separated into are:

- Deployment Models[12]

  Based on the placement and management of the cloud infrastructure there are models which decide the protocol via which the cloud services are deployed to its users.

- Service/Delivery Models[12]

  A cloud can give access to software applications such as email or office productivity tools (the Software as a Service, or SaaS, service model), or can provide an environment for customers to use to create and deploy their own software (the Platform as a Service, or PaaS, service model), or can provide network access to traditional resources used for computing such as storage and processing power (the Infrastructure as a Service, or IaaS, service model)[14].

Chapter 2

Literature Review

2.1 Deployment Models

Companies provide the cloud features with the same attributes over different mediums depending on factors like business profile, operational and technical requirements, etc. But the key factor is whether the provider is an external party or an internal IT department. There are 4 deployment methods:

- Private Cloud[12]

  It is a proprietary computing model that restricts the individuals with access to the hosted services behind a firewall. Due to security, privacy and governance control some companies prefer private clouds. In this the infrastructure is managed and operated exclusively for one company. This restricts a third party intervention.

- Public Cloud[12]

  A public cloud allows resources such as applications and storage to be accessible to the public over the internet. They are an extension of the private cloud with further value profit. They are simpler and cheaper to set up and waste fewer resources as customer only acquires what he needs. They are maintained over a private or shared server but its services are extended over to the public without any restrictions.

- Hybrid Cloud[12]

A hybrid cloud, as the name implies is composed of a minimum of one private cloud and a minimum of one public cloud. Ideally, this model allows a business to get the advantages of measurability and cost-effectiveness supplied by the public cloud model while retaining the privacy, security and policy of the private cloud model.

- Community Cloud[12]

  A community cloud is a multiple tenant infrastructure that is shared among multiple organizations from a particular group with similar computing requirements. Such requirements may be related to regulatory compliance, such as audit requirements, or might be related to performance requirements, such as hosting applications that need fast response time.

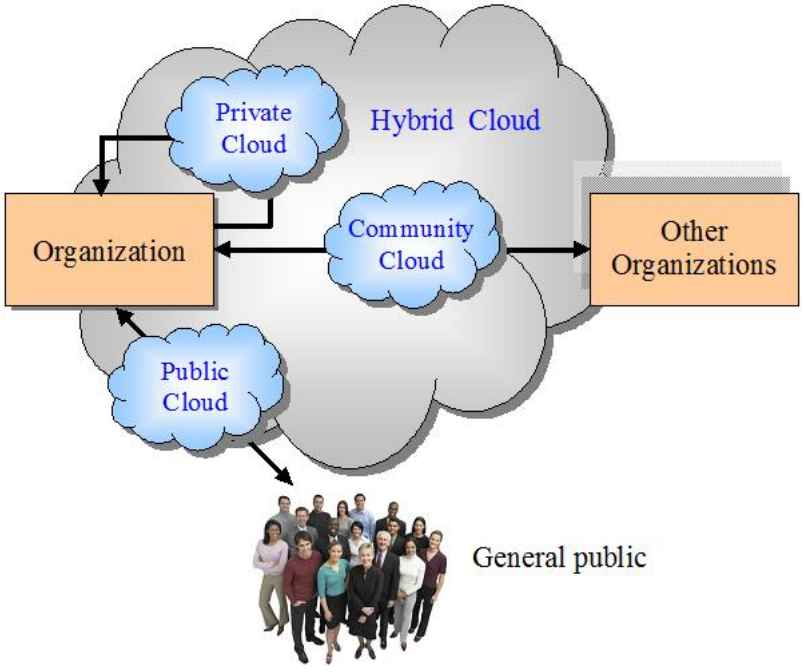

Figure 2.1: Cloud Deployment Models

Attributes	Public Cloud	Private Cloud	Hybrid Cloud	Community Cloud
Cost of building data-center on the consumer	None	High	Medium	Varies depending on number of cooperatives
Operation and maintenance cost on provider	Lowest	Highest	Less than private clouds	Similar to private clouds
Data-center size	Very large	Large	Smaller than private cloud	Larger than private cloud but smaller than public clouds
Infrastructure controllability and flexibility	Limited	Full control	Full control over public cloud and limited control over public cloud	High but limited by community policies
Level of trust	Lowest	Highest	Medium	High
Infrastructure location	Off premise	On premise	Both on and off premise	Within the cooperative facility
Infrastructure owner	IaaS vendor	Customer	IaaS vendor owns public part and customer owns on-premise part	Shared between cooperatives

Table 2.1: Cloud Deployment Models

2.2 Service Models

- Software as a Service (SaaS)[12]

  SaaS, or Software as a Service, allows users to rent or borrow software online instead of actually buying it and installing it on their own systems. It provides a whole environment with pre-installed applications. Users can access their tools and files using a web browser, in effect, doing the entire processing work and file saving on the Internet. The major advantage of Software as a Service is lowered cost for every involved party. Software vendors simply maintain and repair a single central copy of the product online instead of spending valuable resources supporting users over the phone. On the other hand, users do not have to pay the enormous up front costs of fully purchasing end user products like word processing and spreadsheet that they require. They instead pay a minimal rental fees to access the large central copy of the product whenever they need it.

- Platform as a Service (PaaS)[12]

  Platform as a Service provides a framework for the developers to create and deploy their own applications on a hosted infrastructure. It typically provides computing platforms which may include operating systems, programming language execution environments, databases, web servers etc.

- Infrastructure as a Service (IaaS)[12]

  In the case of Infrastructure as a Service, the computing resources provided are virtualized hardware, or a computing infrastructure. It is sometimes called Hardware as a Service (HaaS) because the customer only borrows the basic hardware resource for building their own framework. They have to customize their entire framework using virtual machines, memory etc. These virtual resources can be managed programmatically.

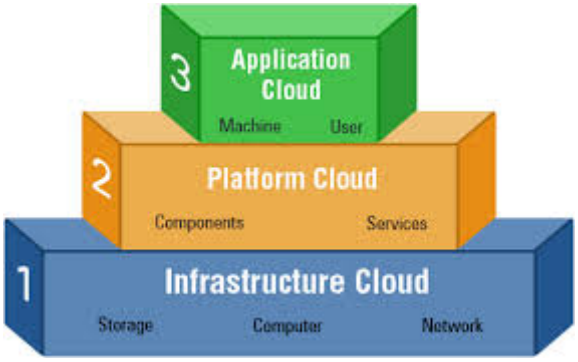

Figure 2.2: Cloud Service Models

Sl. No.	MODEL	DEPLOYMENT METHOD
1.	IaaS	Private cloud, Hybrid cloud
2.	PaaS	Community cloud, Public cloud
3.	SaaS	Community cloud, Hybrid cloud
4.	DaaS	Private cloud, Hybrid cloud

Table 2.2: Cloud Service Deployment

2.3 Job Scheduling

In cloud computing, an Infrastructure as a Service (IaaS) provider divides its physical resources (such as CPU, memory disk) into different types of virtual machines (VMs). These Virtual Machines types may have distinct sizes and features, and are offered as services to the general public[15]. A Virtual Machine is an efficient and independent substitute for a real machine. A particular service is chosen and the client issues its job requests to that service. The service is selected on the basis of unit price, distance, response time, traffic volume, storage space, processor of nodes/VMs (VM stands for Virtual Machine), etc. The VM we refer to acts as a system virtual machine providing a complete system platform which supports the execution of a complete operating system.

Now once a service is selected, the service implements an algorithm to allocate servers and VMs to the jobs requested by its users. Now this algorithm has to efficiently manage the allocation so that it ultimately aims for fastest execution time

along with proper usage/management of resources. This leads to maximum profits as it can execute several job requests in minimum time. In addition to that when some servers are idle, their power supply can be cut off in order to save energy and hence cut the costs in that department again leading to maximized profits.

Most scheduling algorithms can be classified into two types:

Static Scheduling: All information about available resources and the jobs are known before the scheduling and a job is assigned only once to a resource, so adapting is easier as it is based on schedulers perspective[16, 17].

Dynamic Scheduling: This involves dynamically scheduling the jobs over time[18]. It is more flexible than static scheduling as it allows the determination of run time in advance. To obtain stable, accurate and efficient scheduler algorithm, it is important to include load balance as a major factor[16, 17].

2.4 Virtualization

Virtualization is the major enabling technology for cloud computing. Virtualization helps by generalizing the physical infrastructure, which is the most rigid component, and makes it accessible as a soft component that is easier to use and manage. By doing this, virtualization provides the dexterity needed to accelerate IT operations, and reduces cost by increasing infrastructure utilization. By minimizing user involvement, accelerates the process and diminishes the likelihood of human mistake.

Well-defined interfaces have some limitations. Subsystems and components designed to specifications for one interface will not work with those designed for another. For example, application programs, when distributed as compiled binaries, are bound to a particular ISA and depend on a particular operating system interface. This lack of inter-operability can be restricting, especially in a world of interconnected computers where it is invaluable to move software as uninhibitedly as data[19]. Virtualization provides a method for getting around such constraints. Virtualizing a system or component (such as a processor, memory, or an I/O device) at a given abstraction level maps its interface and visible resources onto the interface and resources of an under-

lying, conceivably diverse, real system. Consequently, the real system shows up as a different virtual system or even as multiple virtual systems[19].

Virtualization is simulating a temporary or extended version of computing resources such as processors, operating systems, storage, and network resources. This virtual machine will act like a real resource as far as the consumer is concerned. Thus it creates an abstraction between resources used for computing and the consumers that use them.

The goals of virtualization are

- Increase utilization of shared resources
- Centralize resource management
- Improve data-center agility
- Improve testing
- Improve portability of applications
- Provide isolation
- Enable server consolidation
- Provide foundation for self management frameworks

2.5 Bin Packing Algortihms

The bin packing algorithm packs a list of items into the minimum number of possible bins[20]. There are many instances of this problem such as two dimensional packing, linear packing, packing by cost, packing by weight, and so on. In relation to our project, the servers can be considered as bins and the jobs as the list of items.

Bin packing is an NP hard computational complexity problem but solutions to very large instances of the bin packing can be produced with optimized algorithms. NP-hard (Non-deterministic Polynomial-time hard) is a set of problems which are as hard as the most difficult problems in NP. Many heuristics have also been created to provide fast solutions but not necessarily optimized. We are going to use to allocate

jobs, with their node requirements as weights, to the different servers. There are two types of packing online and offline (prior knowledge of all the weights is required). Let us consider that an object i requires l_i unit space and there are n objects. Number of bins present are N and each bin is of the size L.

Algorithms for bin packing:

i First Fit (FF)

Packing an arriving object to the first bin (i.e. that was opened earliest) in which it fits. If there is no such bin, open a new one and place it there.

ii Best Fit (BF)

Same as First Fit, except that when object i is to be packed, it uses the bin which after fitting object i will have the least amount of free space left.

iii Next Fit (NF)

Packing the object arriving first in the first bin, second object in second bin, and so on till all the objects are exhausted. If the bins are exhausted, the next object is packed in the first if it fits else we proceed to the next bin.

iv First Fit Decreasing (FFD)

For this the objects must be known earlier and hence it is an offline packing system. The objects are ordered in increasing order of weight and then the FF algorithm is implemented.

Ordering should be in a way such that:

$$l_i >= l_{i+1}$$

$$1 < i < n$$

v Best Fit Decreasing (BFD)

For this the objects must be known earlier and hence it is an offline packing system. The objects are ordered in increasing order of weight and then the BF algorithm is implemented.

Ordering should be in a way such that:

$$l_i >= l_{i+1}$$

$$1 < i < n$$

Packing generated by either First Fit or Best Fit uses no more than $\frac{17}{10}OPT + 2$ bins[20].

That by either First Fit Decreasing or Best Fit Decreasing uses no more than $\frac{11}{9}OPT + 4$ bins[20].

OPT refers to the number of bins given by the optimal solution.

Although FFD and BFD use lesser number of bins than plain FF or BF, they can not be used here as they are offline bin packing algorithms.

2.6 Motivation

Cloud computing as we know by now is still evolving and growing. More and more users are using the cloud services and hence the number of job requests is also increasing sometimes linearly and sometimes exponentially. While several algorithms have been proposed to manage the scheduling of jobs and allocation of different servers, in separate papers, we intend to combine a job scheduling algorithm using a priority based selection and then use an appropriate algorithm to allocate the same to different servers. Also the increasing number of jobs result in usage of multiple servers for faster processing but there comes times when there is a lesser load on servers. This leads to servers entering into idle time and hence wastage of energy. So the main challenges to overcome would be resource over provisioning, costly heat dissipation and energy conservation. The energy conservation is done by optimal allocation. Optimal allocation means maximizing the number of requests that can be processed , and minimizing the power consumed by the requests[21].

2.7 Problem Statement

The aims of this project are to devise and implement an algorithm to schedule jobs according to their priority list and another algorithm to assign different jobs to dif-

ferent servers efficiently. The later should also ensure efficient energy management of the servers.

Chapter 3

Proposed Work

3.1 User Attributes

We have assumed that the user along with the job request sends several attributes of the job. The attributes include

i. User

   user is the user id of the person who requested the job.

ii. Time

   Time is the burst time of the job (time take to execute the job)

iii. Node

   Node is the number of nodes/VMs required by the job for its execution.

iv. Sub

   Sub stores the time at which the job is submitted for execution

v. Type

   Type defines if the job is serial or parallel processing.

vi. Cust

   Cust contains the year from which the particular user started using this particular service provider.

These attributes are assigned appropriate weights for the calculation of the priority list. These weights are assumptions taken by us. The attributes are read from text files and stored in respective arrays. Each array element contains the job number and the attribute value, i.e., each array represents different attributes.

Example -

```
1  user = 12
2  time = 139
3  node = 14
4  sub = 14:06
5  type = s
6  cust = 2000
7
```

Figure 3.1: Sample Data

The user id is 12, burst time is 139 minutes. It requires 14 nodes for its execution. It was submitted at 1406 hours. It is a serial processing job. And the user who submitted this job request has been a customer since the year 2000.

3.2 Priority List

A priority list is devised in order to queue up the jobs that have to wait when there are not enough available resources to run the same. Considering each weighted attribute listed in section 4.1, the priority values are calculated. The weights assigned to these jobs are linear assumptions taken by us[22].

The job requiring the least execution time is given the highest priority value while the job requiring the most is given the least. The job requiring the least number of nodes is given the highest priority value while the job requiring the most is given the least. Depending on the time for which the user, with the job request, has been using the service, a separate priority value is assigned which is more if the time is more. Sequential jobs are given half the priority value as that given to parallel processing jobs. All these priority values are calculated and aggregated to compile the priority list. Finally the time of submission gives the whole model a real time scenario. Using the submission we check which job comes after which job and at what time.

Algorithm

To compute the priority list of job requests made by several users to the service providers for allocation of VMs.

i Reading and insertion of the attribute values of all the job requests made to the service provider into respective arrays.

$time[0][k] \leftarrow$ value for time (attribute 1)

$node[0][k] \leftarrow$ value for nodes required (attribute 2)

$type[0][k] \leftarrow$ value for type of job (attribute 4)

$cust[0][k] \leftarrow$ value for customer loyalty (attribute 5)

$*[1][k]$ is used for storing the job number. This helps in synchronizing the attribute values.

k is the k^{th} job.

ii The attribute arrays mentioned above are sorted in the required order except for the type values.

$time[][]$ is sorted in increasing order.

$node[][]$ is sorted in increasing order.

$cust[][]$ is sorted in increasing order.

iii The priority values are assigned to each job according to their attributes and the final value is stored into another array. The maximum priority value is assumed to be n where n is the total number of jobs and the least priority value is 1.

$$priority[0][k] \leftarrow priority[0][k] + pv_a$$

$$1 <= pv <= n$$

pv_a (priority value for a attribute) of k^{th} job depends on the sorted list of the attribute. In case of time, node and cust attributes the job in the first element (having least value) gets a priority value of n, the next gets n-1 and so on till the last job gets 1. This is repeated for each job with respect to each attribute.

iv Stop

priority[0][k] ← *priority value of* k^{th} *job.*

```
304     for(i=0;i<n;i++) {
305             priority[0][time[1][i]-1] += pv;
306             pv--;
307
308     }
309
310     pv=n;
311
312     for(i=0;i<n;i++) {
313             priority[0][node[1][i]-1] += pv;
314             pv--;
315     }
316
317     pv=n;
318
319     for(i=0;i<n;i++) {
320             priority[0][cust[1][i]-1] += pv;
321             pv--;
322     }
```

Figure 3.2: Priority Value Calculation Code Snippet

3.3 Parallel Scheduling - I

Algorithm

To allocate nodes to a job request using a single server.

i Repeat steps ii, iii, iv and v till there are no more jobs left.

ii Read the number of nodes required by the incoming job request.

iii Check if required number of nodes is available in the server.

iv If available allocate the nodes if not queue it in the wait list according to decreasing order of priority values.

v If a few nodes of the server are freed (when a job completes its execution), then take a job out of the wait list and jump to step ii. If node requirement is not met then wait till more nodes are available.

3.4 Parallel Scheduling - II

Algorithm

To allocate nodes to a job request using a single server.

i Repeat steps ii, iii, iv and v till there are no more jobs left.

ii Read the number of nodes required by the incoming job request.

iii Check if required number of nodes is available in the server.

iv If available allocate the nodes if not queue it in the wait list according to decreasing order of priority values and then sub-sort it according to required nodes of individual jobs.

v If a few nodes of the server are freed (when a job completes its execution), then take a job out of the wait list and jump to step ii. If node requirement is not met then proceed to the next queued job.

3.5 Server Allocation

Allocation of incoming jobs is done by Bin Packing's Best Fit algorithm. The individual servers are considered as bins which are filled by jobs (with nodes/VMs considered to be the weights).

Best Fit is used instead of the rest three types. First Fit Decreasing (FFD) and Best Fit Decreasing (BFD) cannot be used because they require prior knowledge of the job requests coming in. But since this is real time the prior information cannot be provided. Best Fit (BF) is used over Next Fit (NF), although NF has a better time complexity ($O_{nf}(n) < O_{bf}(nlogn)$), because BF utilizes the resources in an efficient manner without letting any nodes sit idle for a long time. Best Fit is used over First Fit (FF), although both have similar time complexities ($O_{ff}(nlogn) = O_{bf}(nlogn)$), because it uses the servers efficiently by allocating the job to the server which will have the least amount of nodes left. This in turn keeps a significant amount of servers idle and hence saves energy.

```
398
399         while(t_diff[k]==i && k<n) {
400
401             min = 15;
402             y = -1;
403             for(nd=0;nd<4;nd++) {
404
405                 x = 15 - N[nd] - node[0][sub[2][k]-1];
406                 if(x<min && x>=0) {
407                     min = x;
408                     y = nd;
409                 }
410             }
411
```

Figure 3.3: Server Allocation Code Snippet

Algorithm

i Read the number of nodes required by the incoming job request.

ii Find the server to which if the job is assigned for execution will leave the least number of idle nodes.

iii If all servers are busy queue the job in the wait list.

iv If not assign the job to the selected server.

v If a few nodes of a server are freed then take a job out of the wait list and jump to step ii.

vi Stop if there are no more queued jobs and no jobs being executed.

Figure 4.3 shows the code used to implement the selection of the server according to Best Fit algorithm. min stores the least number of nodes left after allocation of job and y stores the corresponding server number (step 2).

3.6 Wait List

Any job which is not allotted any server is queued into a wait list sorted in decreasing order of priority value and then again sub-sorted in increasing order of node requirement. When a job completes its execution, it frees up the VMs that it had earlier occupied. Now the algorithm checks for jobs in the waiting list. If the node requirement matches then that job is taken out of the wait list and a server is allocated to it according to Best Fit algorithm. This helps in eliminating starvation as it considers nodes with lower priority but with lower node requirements than that of the higher priority jobs as well[23].

```
if(y == -1) {                          //true if no servers are available

    printf("\n---- %d cost: %d time: %d\n",sub[2][k],cost[0][sub[2][k]-1], node[0][sub[2][k]-1]);
    insert(cost[0][sub[2][k]-1],node[0][sub[2][k]-1], sub[2][k]-1);   //insertion into wait list
    printf("\n|||| %d cost: %d time: %d\n",sub[2][k],now->cst, now->nn);
}
```

Figure 3.4: Code snippet for insertion of jobs into wait list

Fig 4.4 shows the code snippet for insertion of jobs into wait list when there are no servers available. The lines of code above and below the insert() function are used to maintain logs.

```
if(y+1){      //server 'nd+1'is available for job pointed by ptr in wait list
    if(ptr==start) {    //if job is at the start node
        comp[0][ptr -> num] = 2; //status of job changed to running
        comp[1][ptr -> num] = nd + 1;
        N[nd]+=ptr -> nn;
        printf("\nstarts here job: %d at i: %d with NODES: %d and cost: %d in
        remove_start(); //removal from wait list
        ptr = ptr ->link;
    }
    else {
        comp[0][ptr -> num] = 2; //status of job changed to running
        comp[1][ptr -> num] = nd + 1;
        N[nd]+=ptr -> nn;
        printf("\nstarts here job: %d at i: %d with NODES: %d nd cost: %d in s
        remove_node(tmp);  //removal from wait list
        ptr = tmp -> link;
    }
'
```

Figure 3.5: Code snippet for removal of a job in the wait list

Fig 4.5 is the code snippet which shows the removal of a job in the wait list which is pointed by the pointer ptr when a server (server number: $nd + 1$) is available. comp[0][k] represents the status of a job. If its value is 1 then its execution is completed else if its 2 then it is being executed. The printf() statements are used to maintain logs.

Chapter 4

Results and Analysis

4.1 Job scheduling into a single server

First we took a data set of 15 jobs and their attributes. The jobs had different attributes with the same submission time and same attributes with different submission times to test the boundaries of the proposed algorithms. A single server was considered to execute the jobs with a total 15 nodes/VMs.

4.1.1 Static job scheduling

15 jobs of different attribute sets were taken but all the jobs had the same submission time.

This was done to simulate the serial processing of the jobs just for the comparison between serial and parallel processing.

4.1.2 Parallel scheduling - I

15 jobs of different attribute sets were taken with different submission times. The jobs were scheduled according to their arrival times. If sufficient nodes/VMs were not available the jobs were added to the waiting list sorted in decreasing order of priority values. When a running job finished its execution and freed the occupied VMs/nodes, the jobs in the waiting list were considered based only according to their priority values. The demerits of this scheduling include starvation of lower priority jobs and wastage of resources.

When jobs in the wait list are considered, only the job with the highest priority

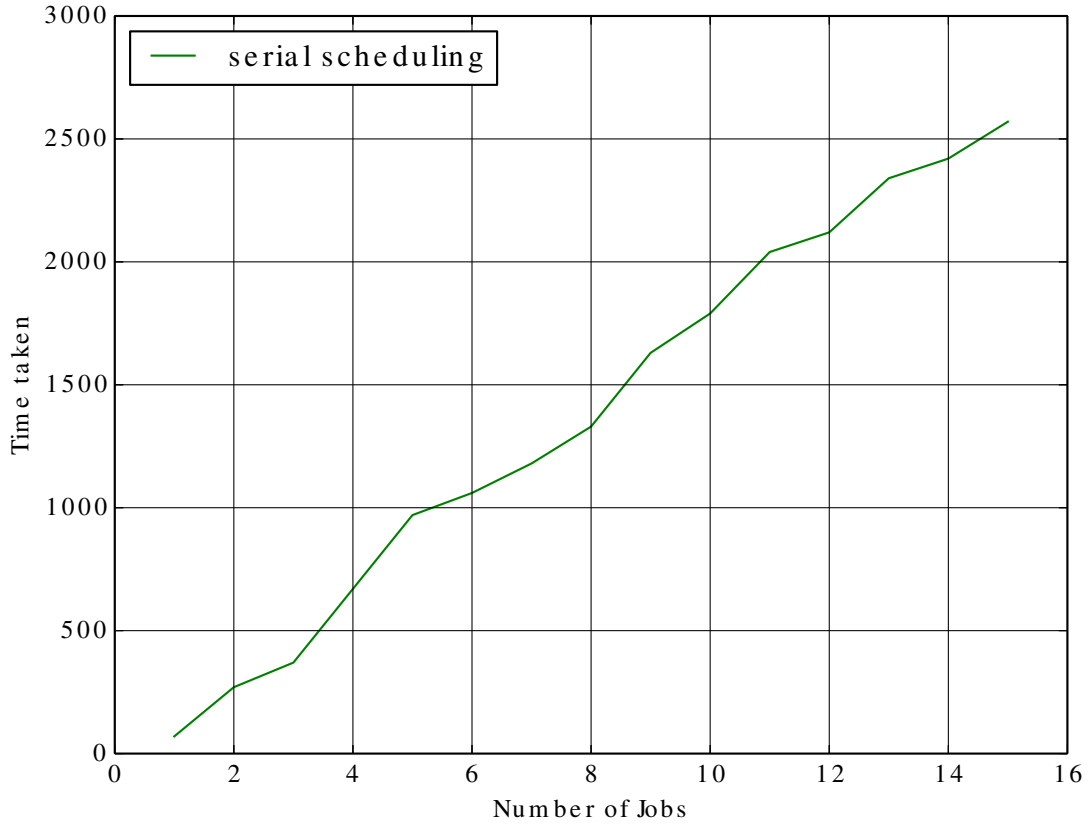

Figure 4.1: Serial Scheduling

was given importance. If that job's node requirement was not fulfilled then the process moved on. Now there may be jobs with lower priority value whose node requirement could be fulfilled but those jobs were not taken out from the wait list. This not only causes the lower priority jobs to starve but also keeps lots of nodes/VMs idle for a long time.

4.1.3 Parallel Scheduling - II

15 jobs of different attribute sets were taken with different submission times. The jobs were scheduled according to their arrival times. If sufficient nodes/VMs were not available the jobs were added to the waiting list sorted in decreasing order of priority values and then again sub-sorted in increasing order of node requirement. When a running job finished its execution and freed the occupied VMs/nodes, the jobs in the

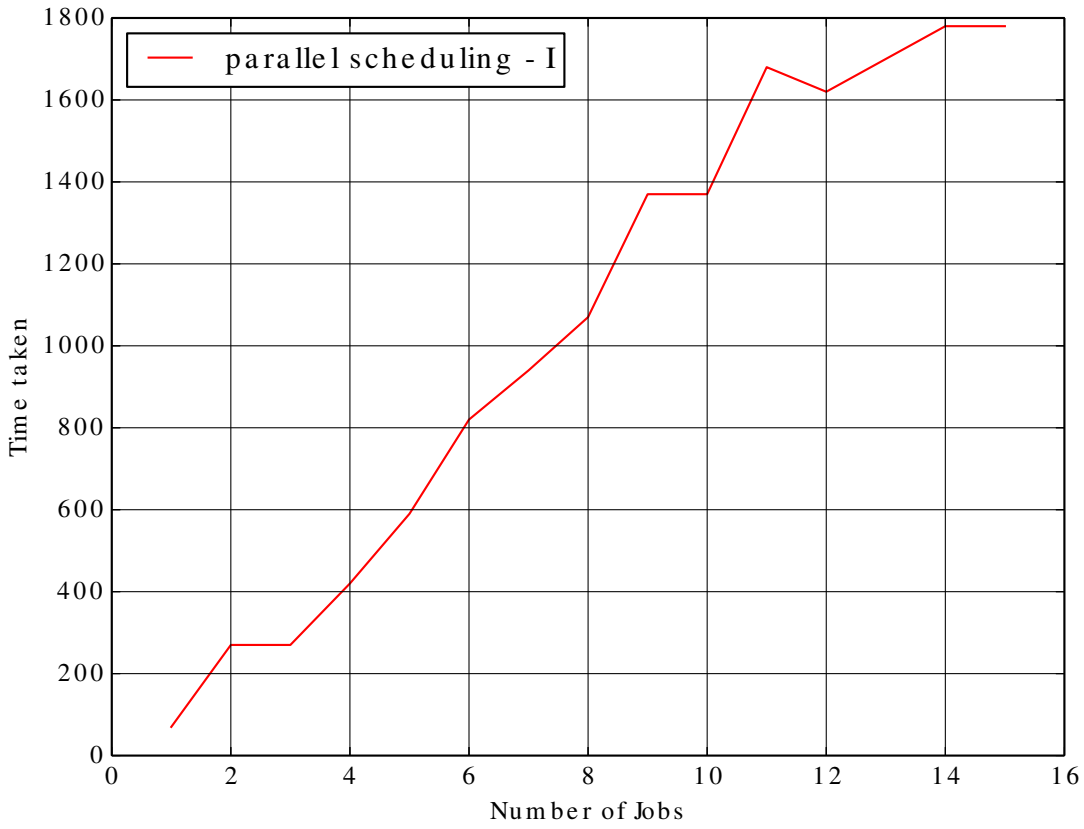

Figure 4.2: Parallel Scheduling - I

waiting list were considered based according to their priority values and then the node requirement.

This procedure solved the problem of starvation as well as decreased the idle time of the nodes/VMs. Fig. 5.4 shows that the this procedure results in a faster execution time than the Parallel Scheduling - I.

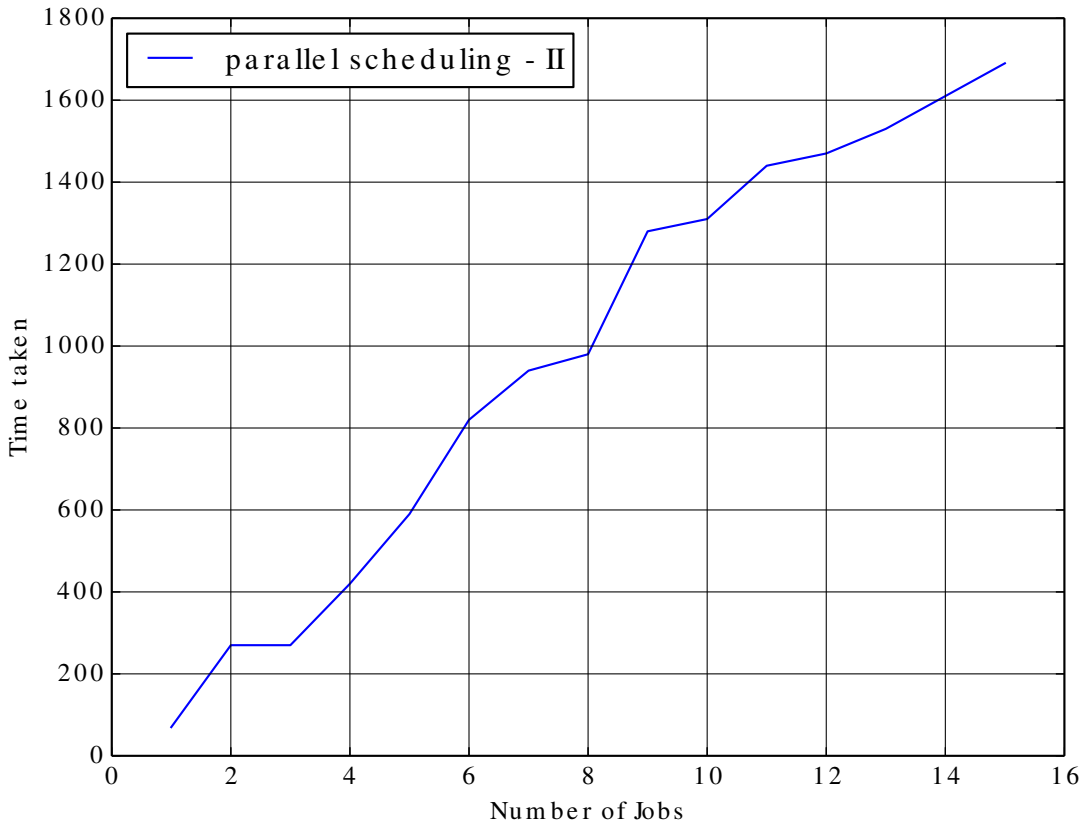

Figure 4.3: Parallel Scheduling - II

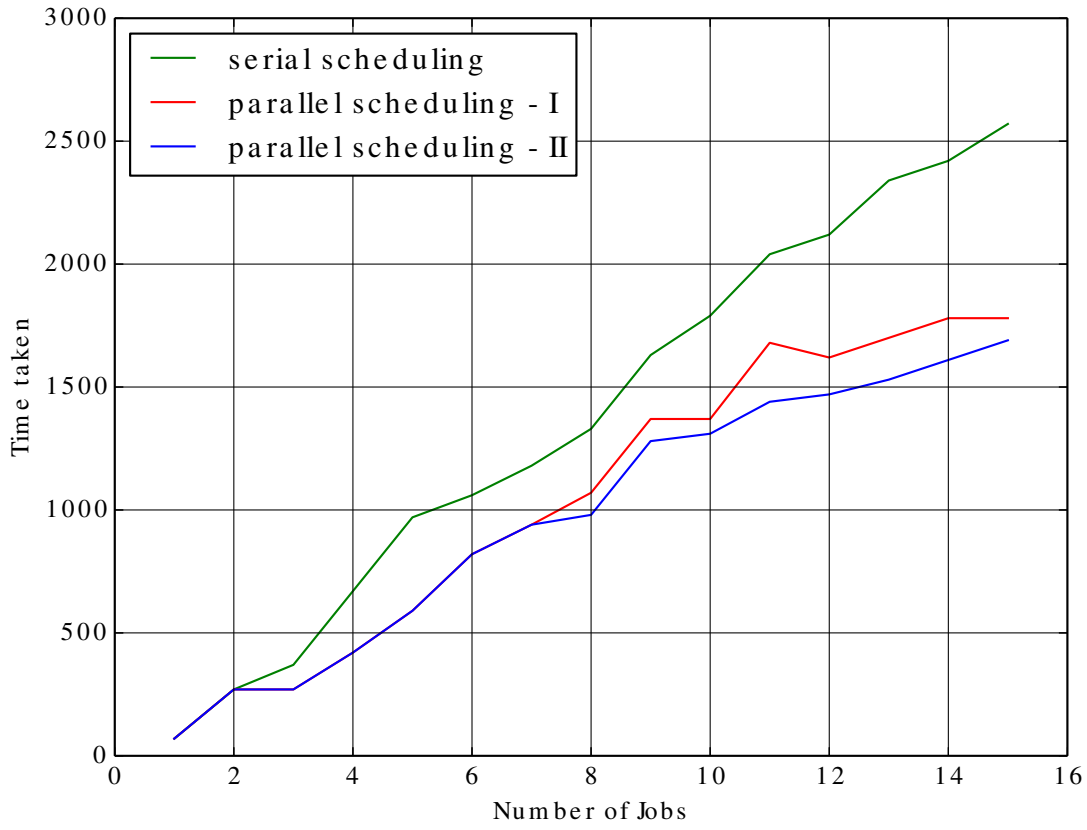

Figure 4.4: Comparing the different Scheduling processes

4.2 Server Allocation

We took 4 different servers with 15 nodes/VMs in each server. The Best Fit Bin Packing algorithm was used to allocate servers to different jobs. A dataset of 95 different jobs were considered with different attribute values and submission times.

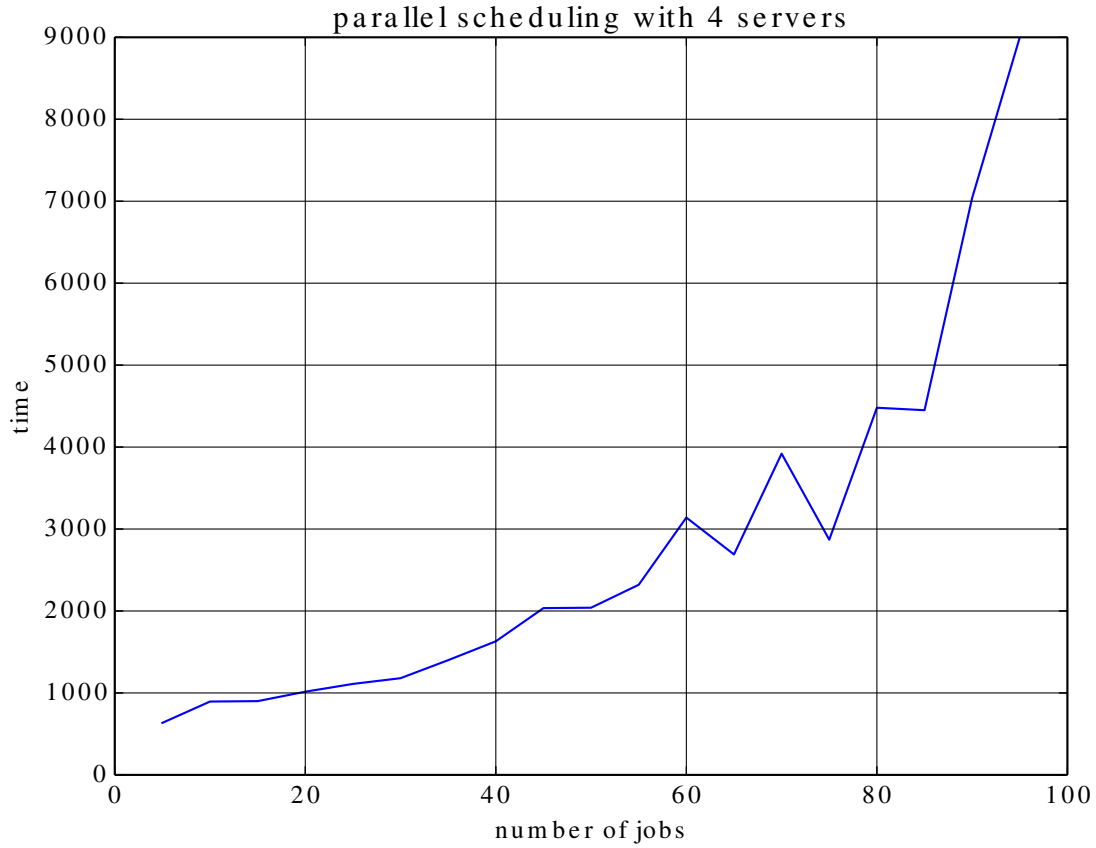

Figure 4.5: Parallel Scheduling on 4 servers

Fig. 5.5 shows the time taken (y axis) for corresponding x number of jobs (x axis).

Fig. 5.6 shows the time for which each server is free plotted against the number of jobs.

Fig. 5.7 shows the percentage time for which each server is free plotted against the number of jobs. The percentage is calculated over the total time of execution of the said number of jobs.

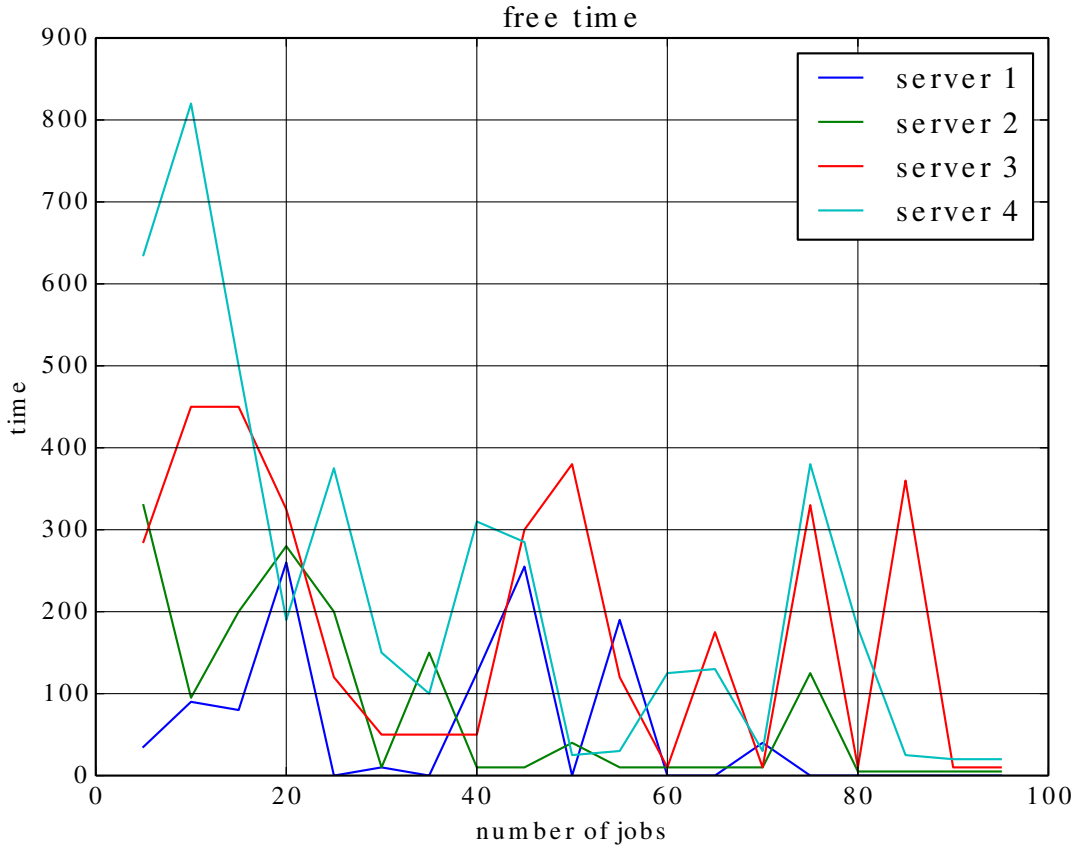

Figure 4.6: Total Free Time on Parallel Scheduling on 4 servers

As we can see the servers are free for a longer period of time when the job load is less. Hence the Best Fit Bin Packing algorithm frees the servers for a longer time hence minimizing the energy consumption and maximizing the profit.

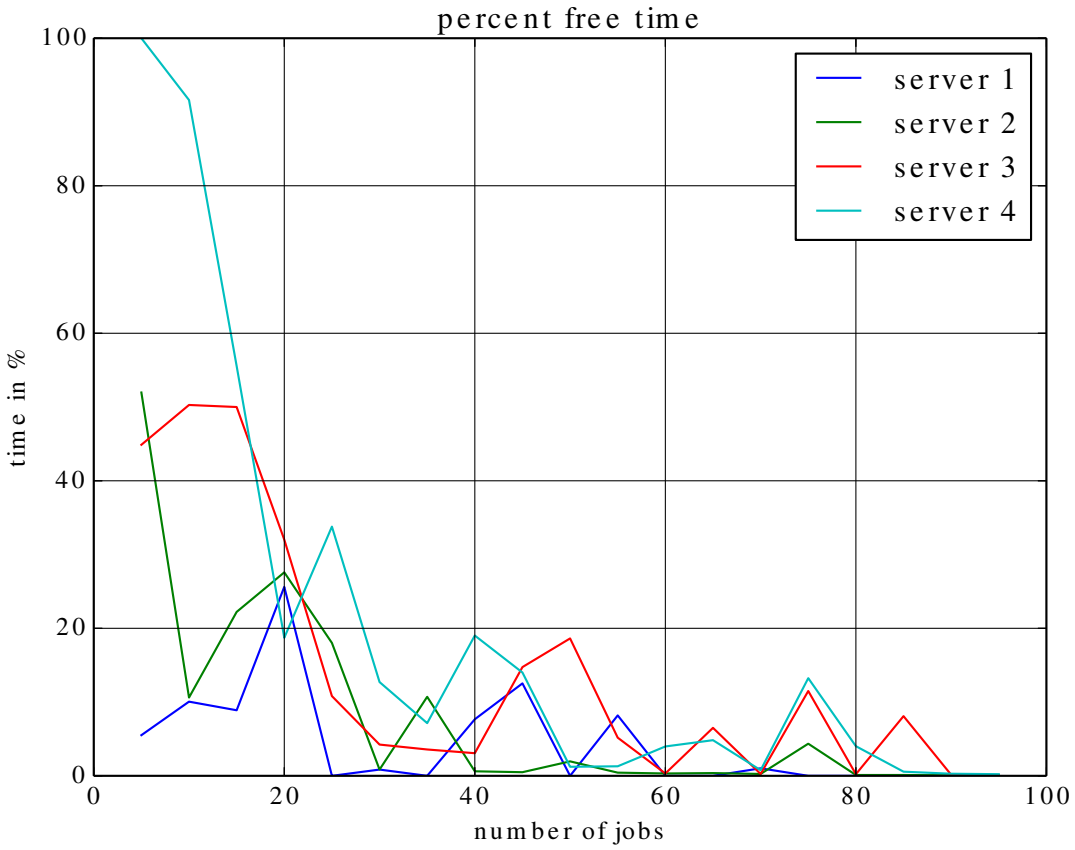

Figure 4.7: Percentage Free Time on Parallel Scheduling on 4 servers

Chapter 5

Conclusions and Future Work

5.1 Conclusion

We proposed an algorithm for scheduling jobs in Cloud Services according to a priority list and an algorithm to allocate servers to the incoming job requests. The former algorithm used weighted attributes of the incoming jobs, created individual priority values and used a double sorted wait list for queuing impending jobs in case of unavailability of servers. This reduced starvation and efficiently utilized the nodes/VMs of the individual servers. The later algorithm was based on the Best Fit algorithm of Bin Packing. It efficiently utilized the individual servers and their nodes/VMs. It also helped in utilizing as less number of servers as possible for maximum energy conservation. The proposed algorithms will show maximized profits for service providers.

5.2 Future Work

In future we intend to optimize the scheduling algorithm by including more attributes. Then implement a complete service selection algorithm [24] which would select from a number of service providers taking into the considerations of the user choices and other factors such as response time, distance etc.

Bibliography

[1] John McCarthy. A Time Sharing Operator Program for our Projected IBM 709, 1 Jan 1959.

[2] J. C. R. Licklider. Memorandum For Members and Affiliates of the Intergalactic Computer Network, April 1963.

[3] D.F. Parkhill. *The challenge of the computer utility*. Number p. 246 in The Challenge of the Computer Utility. Addison-Wesley Pub. Co., 1966.

[4] Seeding the cloud: Key infrastructure elements for cloud computing, Feb 2009.

[5] Geva Perry. How cloud and utility computing are different, Feb 2008.

[6] Zaigham Mahmood and Richard Hill. *Cloud Computing for enterprise architectures*. Springer, 2011.

[7] B T OGRAPH and Y RICHARD MORGENS. Cloud computing. *Communications of the ACM*, 51(7), 2008.

[8] Dejan Milojicic. Cloud computing: Interview with russ daniels and franco travostino. *IEEE Internet Computing*, 12(5):7–9, 2008.

[9] Armando Fox, Rean Griffith, A Joseph, R Katz, A Konwinski, G Lee, D Patterson, A Rabkin, and I Stoica. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28:13, 2009.

[10] Victor Ion Munteanu, T Fortis, and Viorel Negru. An evolutionary approach for sla-based cloud resource provisioning. In *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, pages 506–513. IEEE, 2013.

[11] Mohammad Hamdaqa and Ladan Tahvildari. Cloud computing uncovered: a research landscape. *Advances in Computers*, 86:41–85, 2012.

[12] Peter Mell and Tim Grance. The nist definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009.

[13] R Madhubala. An illustrative study on cloud computing. *International journal of soft computing and engineering*, 1(6):286–290, 2012.

[14] Lee Badger, Tim Grance, Robert Patt-Corner, and Jeff Voas. Cloud computing synopsis and recommendations. *NIST special publication*, 800:146, 2012.

[15] Qi Zhang, Quanyan Zhu, and Raouf Boutaba. Dynamic resource allocation for spot markets in cloud computing environments. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 178–185. IEEE, 2011.

[16] Swachil Patel and Upendra Bhoi. Priority based job scheduling techniques in cloud computing: A systematic review.

[17] Thomas A Henzinger, Anmol V Singh, Vasu Singh, Thomas Wies, and Damien Zufferey. Static scheduling in clouds. *memory*, 200(o1):i1, 2011.

[18] R Bhaskar, SR Deepu, and BS Shylaja. Dynamic allocation method for efficient load balancing in virtual machines for cloud computing environment. *Advanced Computing*, 3(5), 2012.

[19] James E Smith and Ravi Nair. The architecture of virtual machines. *Computer*, 38(5):32–38, 2005.

[20] Chan C Lee and Der-Tsai Lee. A simple on-line bin-packing algorithm. *Journal of the ACM (JACM)*, 32(3):562–572, 1985.

[21] Kazuki Mochizuki and S-i Kuribayashi. Evaluation of optimal resource allocation method for cloud computing environments with limited electric power capacity. In *Network-Based Information Systems (NBiS), 2011 14th International Conference on*, pages 1–5. IEEE, 2011.

[22] KC Gouda, TV Radhika, and M Akshatha. Priority based resource allocation model for cloud computing. *International Journal of Science, Engineering and Technology Research (IJSETR)*, 2(1):215–219, 2013.

[23] Chandrashekhar S Pawar and Rajnikant B Wagh. Priority based dynamic resource allocation in cloud computing with modified waiting queue. In *Intelligent Systems and Signal Processing (ISSP), 2013 International Conference on*, pages 311–316. IEEE, 2013.

[24] Wenying Zeng, Yuelong Zhao, and Junwei Zeng. Cloud service and service selection algorithm research. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 1045–1048. ACM, 2009.