LOAD BALANCING IN CLOUD COMPUTING SYSTEMS

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

Bachelor of Technology

in

Computer Science and Engineering

by

Ram Prasad Padhy (107CS046)

P Goutam Prasad Rao (107CS039)

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela-769 008, Orissa, India
May, 2011.

LOAD BALANCING IN CLOUD COMPUTING SYSTEMS

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

Bachelor of Technology

in

Computer Science and Engineering

by

Ram Prasad Padhy (107CS046)

P Goutam Prasad Rao (107CS039)

under the guidance of

Dr. Pabitra Mohan Khilar

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela-769 008, Orissa, India
May, 2011.

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

Certificate

This is to certify that the work in this thesis Report entitled ***"Load Balancing in Cloud computing systems"*** submitted by ***Ram Prasad Padhy(107CS046)*** and ***P Goutam Prasad Rao(107CS039)*** has been carried out under my supervision in partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science during session 2010-2011 in the department of Computer Science and Engineering, National Institute of Technology, Rourkela, and this work has not been submitted for any degree or academic award elsewhere.

Place: NIT, Rourkela
Date: 09/05/11

Dr. Pabitra Mohan Khilar
Assistant Professor
Department of CSE
NIT Rourkela

ACKNOWLEDGEMENT

First of all we would like to express our profound sense of gratitude towards our guide **Dr. Pabitra Mohan Khilar**, *Asst. Professor, Department of Computer Science and Engineering*, for his able guidance, support and encouragement throughout the period this work was carried out. His readiness for consultation at all times, his educative comments, his concern and assistance even with practical things have been invaluable.

We would also like to convey our sincerest gratitude and indebtedness to our entire faculty members and staff of the Department of Computer Science and Engineering, NIT Rourkela, who bestowed their efforts and guidance at appropriate times without which it would have been very difficult on our part to finish the project work. A vote of thanks to our fellow students for their friendly co-operation and suggestions.

Ram Prasad Padhy(107CS046)

P Goutam Prasad Rao(107CS039)

Abstract

”Cloud computing” is a term, which involves virtualization, distributed computing, networking, software and web services. A cloud consists of several elements such as clients, datacenter and distributed servers. It includes fault tolerance, high availability, scalability, flexibility, reduced overhead for users, reduced cost of ownership, on demand services etc.

Central to these issues lies the establishment of an effective load balancing algorithm. The load can be CPU load, memory capacity, delay or network load. Load balancing is the process of distributing the load among various nodes of a distributed system to improve both resource utilization and job response time while also avoiding a situation where some of the nodes are heavily loaded while other nodes are idle or doing very little work. Load balancing ensures that all the processor in the system or every node in the network does approximately the equal amount of work at any instant of time. This technique can be sender initiated, receiver initiated or symmetric type (combination of sender initiated and receiver initiated types).

Our objective is to develop an effective load balancing algorithm using Divisible load scheduling theorm to maximize or minimize different performance parameters (throughput, latency for example) for the clouds of different sizes (virtual topology depending on the application requirement).

Contents

List of Figures

Chapter 1

Introduction

Introduction

Thesis Motivation

Objectives

Thesis Organization

1 Introduction

1.1 Introduction

Cloud computing is an on demand service in which shared resources, information, software and other devices are provided according to the clients requirement at specific time. Its a term which is generally used in case of Internet. The whole Internet can be viewed as a cloud. Capital and operational costs can be cut using cloud computing.

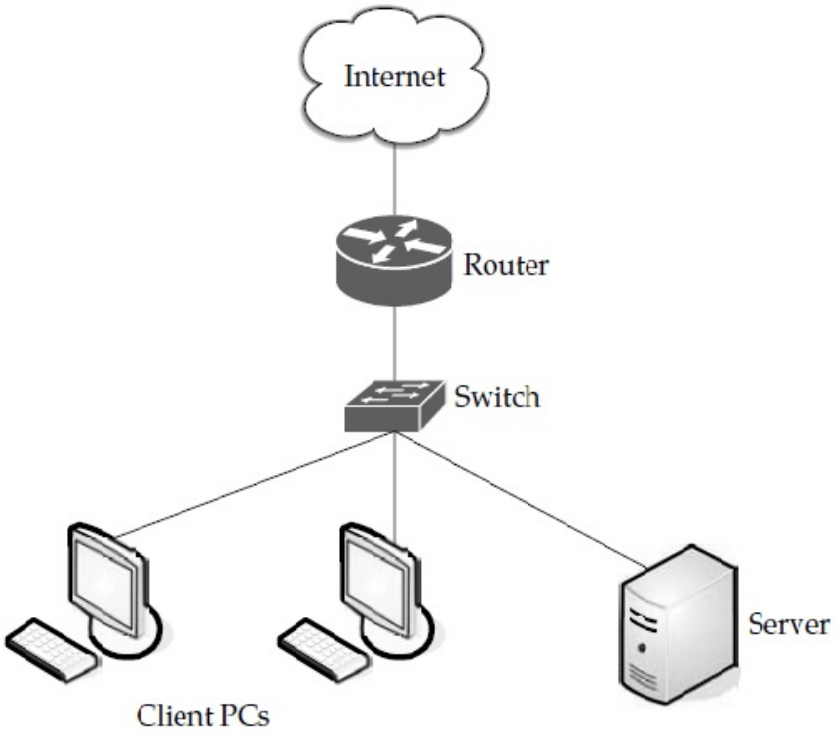

Figure 1: A cloud is used in network diagrams to depict the Internet (adopted from [1]).

Load balancing in cloud computing systems is really a challenge now. Always a distributed solution is required. Because it is not always practically feasible or cost efficient to maintain one or more idle services just as to fulfill the required demands. Jobs cant be assigned to appropriate servers and clients individually for efficient load balancing as cloud is a very complex structure and components are present throughout a wide spread area. Here some uncertainty is attached while jobs are assigned.

This paper considers some of the methods of load balancing in large scale Cloud systems. Our aim is to provide an evaluation and comparative study of these approaches,

demonstrating different distributed algorithms for load balancing and to improve the different performance parameters like throughput, latency etc. for the clouds of different sizes. As the whole Internet can be viewed as a cloud of many connection-less and connection-oriented services, thus concept of load balancing in Wireless sensor networks (WSN) proposed in [9] can also be applied to cloud computing systems as WSN is analogous to a cloud having no. of master computers (Servers) and no. of slave computers (Clients) joined in a complex structure. A comparative study of different algorithms has been carried out using divisible load scheduling theory proposed in [9].

1.2 Thesis Motivation

Cloud computing is a vast concept. Many of the algorithms for load balancing in cloud computing have been proposed. Some of those algorithms have been overviewed in this thesis. The whole Internet can be considered as a cloud of many connection less and connection oriented services. So the divisible load scheduling theory for Wire less networks described in [9] can also be applied for clouds. The performance of various algorithms have been studied and compared.

1.3 Objectives

1. To study the performance of some of the existing load balancing algorithms
2. To design and develop the concept of load balancing using Divisible Load Scheduling Theory (DLT) for the clouds of different sizes
3. To evalute the performance of the proposed scheme using analytical studies proposed in [9] and using MATLAB

1.4 Thesis Organization

The rest of the thesis is organised as follows:

Chapter 1 gives a brief introduction of the concepts used in the thesis, its contents, objectives and the related work done in the particular domain.

Chapter 2 gives an overall idea of cloud computing, its components, functionalities and services provided by the cloud.

Chapter 3 describes the concept of load balancing in distributed systems, its goals, different types and different policies used.

Chapter 4 we studied different load balancing algorithms which are applicable to clouds of different sizes.

Chapter 5 presents our proposed work i.e the divisible load scheduling theorm for clouds of different sizes.

Chapter 6 gives the system model description, overview of divisible load scheduling theory(DLT), and different approaches under DLT that can be applied to clouds.

Chapter 7 describes the simulation results of different approaches and their comparison.

Chapter 8 presents the conclusion about the work done in this thesis

Chapter 9 gives the summary of the whole thesis

1.5 Conclusion

This chapter gives a brief idea about Cloud Computing and load balancing. It also gives an overall idea about tha objectives, motivation and organisation of this thesis.

Chapter 2

Cloud Computing

Cloud Components

Type of Clouds

Virtualization

Services provided by Cloud computing

2 Cloud Computing

2.1 Introduction

In case of Cloud computing services can be used from diverse and widespread resources, rather than remote servers or local machines. There is no standard definition of Cloud computing. Generally it consists of a bunch of distributed servers known as masters, providing demanded services and resources to different clients known as clients in a network with scalability and reliability of datacenter. The distributed computers provide on-demand services. Services may be of software resources (e.g. Software as a Service, SaaS) or physical resources (e.g. Platform as a Service, PaaS) or hardware/infrastructure (e.g. Hardware as a Service, HaaS or Infrastructure as a Service, IaaS). Amazon EC2 (Amazon Elastic Compute Cloud) is an example of cloud computing services [2]..

2.2 Cloud Components

A Cloud system consists of 3 major components such as clients, datacenter, and distributed servers. Each element has a definite purpose and plays a specific role.

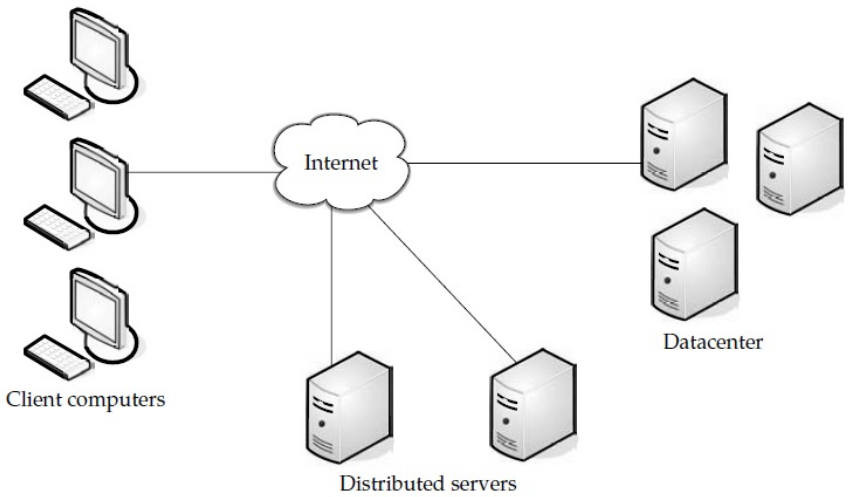

Figure 2: Three components make up a cloud computing solution(adopted from [1]).

2.2.1 Clients

End users interact with the clients to manage information related to the cloud. Clients generally fall into three categories as given in [1]:

- **Mobile**: Windows Mobile Smartphone, smartphones, like a Blackberry, or an iPhone.
- **Thin**: They don't do any computation work. They only dispaly the information. Servers do all the works for them. Thin clients don't have any internal memory.
- **Thick**: These use different browsers like *IE* or mozilla Firefox or Google Chrome to connect to the Internet cloud.

Now-a-days thin clients are more popular as compared to other clients because of their low price, security, low consuption of power, less noise, easily replaceble and repairable etc.

2.2.2 Datacenter

Datacenter is nothing but a collection of servers hosting different applications. A end user connects to the datacenter to subscribe different applications. A datacenter may exist at a large distance from the clients.

Now-a-days a concept called *virtualisation* is used to install a software that allow multiple instances of virtual server applications.

2.2.3 Distributed Servers

Distributed servers are the parts of a cloud which are present throughout the Internet hosting different applications. But while using the application from the cloud, the user will feel that he is using this application from its own machine.

2.3 Type of Clouds

Based on the domain or environment in which clouds are used, clouds can be divided into 3 catagories :

- **Public Clouds**
- **Private Clouds**
- **Hybrid Clouds (combination of bothe private and public clouds)**

2.4 Virtualization

It is a very usefull concept in context of cloud systems. Virtualisation means ”something which isn’t real”, but gives all the facilities of a real. It is the software implementation of a computer which will execute different programs like a real machine.

Virtualisation is related to cloud, because using virtualisation an end user can use different services of a cloud. The remote datacenter will provide different services in a fully or partial virtualised manner.

2 types of virtualization are found in case of clouds as given in [1] :

- Full virtualization
- Paravirtualization

2.4.1 Full Virtualization

In case of full virtualisation a complete installation of one machine is done on the another machine. It will result in a virtual machine which which will have all the softwares that are present in the actual server.

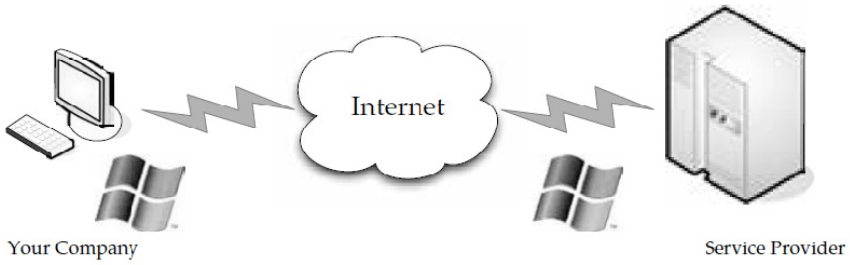

Figure 3: Full Virtualization (adopted from [1]).

Here the remote datacenter delivers the services in a fully virtualised manner. Full virtualization has been successful for several purposes as pointed out in [1]:

- Sharing a computer system among multiple users
- Isolating users from each other and from the control program
- Emulating hardware on another machine

2.4.2 Paravirtualization

In paravitualisation, the hardware allows multiple operating systems to run on single machine by efficient use of system resources such as memory and processor. e.g. VMware software. Here all the services are not fully available, rather the services are provided partially.

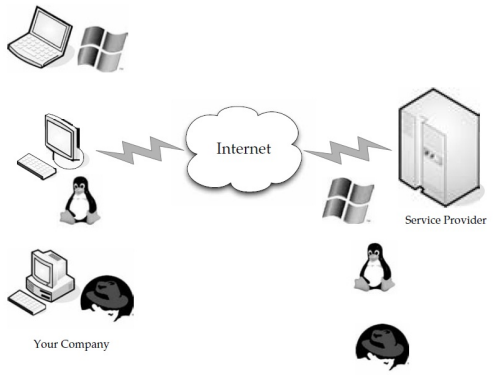

Figure 4: Paravirtualization (adopted from [1]).

Paravirtualization has the following advantages as given in [1]:

- **Disaster recovery**: In the event of a system failure, guest instances are moved to another hardware until the machine is repaired or replaced.
- **Migration**: As the hardware can be replaced easily, hence migrating or moving the different parts of a new machine is faster and eiasier.
- **Capacity management**: In a virtalised environment, it is easier and faster to add more hard drive capacity and processing power. As the system parts or hardwares can be moved or replaced or repaired easily, capacity management is simple and eaiser.

2.5 Services provided by Cloud computing

Service means different types of applications provided by different servers across the cloud. It is generally given as ”as a service”. Services in a cloud are of 3 types as given in [1] :

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Hardware as a Service (HaaS) or Infrastructure as a Service (IaaS)

2.5.1 Software as a Service (SaaS)

In SaaS, the user uses different software applications from different servers through the Internet. The user uses the software as it is without any change and do not need to make lots of changes or doen't require integration to other systems. The provider does all the upgrades and patching while keeping the infrastructure running [2].

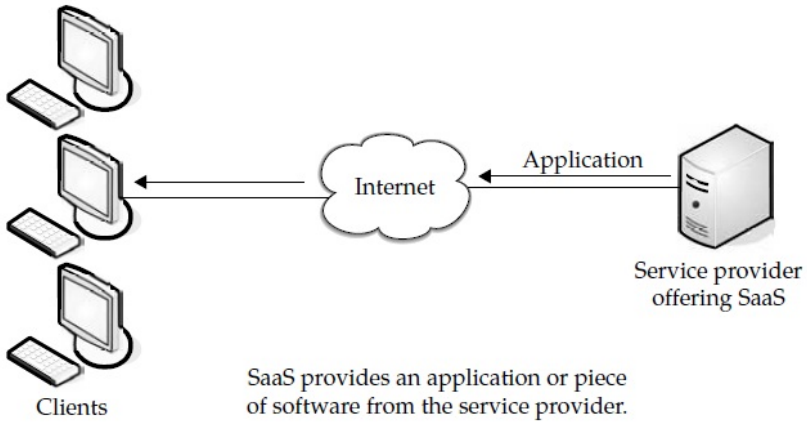

Figure 5: Software as a service (SaaS) (adopted from [1])

The client will have to pay for the time he uses the software. The software that does a simple task without any need to interact with other systems makes it an ideal candidate for Software as a Service. Customer who isn't inclined to perform software development but needs high-powered applications can also be benefitted from SaaS.

Some of these applications include (taken from [1]):

- Customer resource management (CRM)
- Video conferencing
- IT service management
- Accounting
- Web analytics
- Web content management

Benefits:The biggest benefit of SaaS is costing less money than buying the whole application. The service provider generally offers cheaper and more reliable applications as compared to the organisation [1]. Some other benefits include (given in [1]): Familiarity with the Internet, Better marketing, Smaller staff, reliability of the Internet, data Security, More bandwidth etc.

Obstacles:

- SaaS isn't of any help when the organisation has a very specific computational need that doesn't match to the SaaS services
- While making the contract with a new vendor, there may be a problem. Because the old vendor may charge the moving fee. Thus it will increase the unnecessary costs.
- SaaS faces challenges from the availability of cheaper hardwares and open source applications.

2.5.2 Platform as a Service (PaaS)

PaaS provides all the resources that are required for building applications and services completely from the Internet, without downloading or installing a software [1].

PaaS services are software design, development, testing, deployment, and hosting. Other services can be team collaboration, database integration, web service integration, data security, storage and versioning etc.

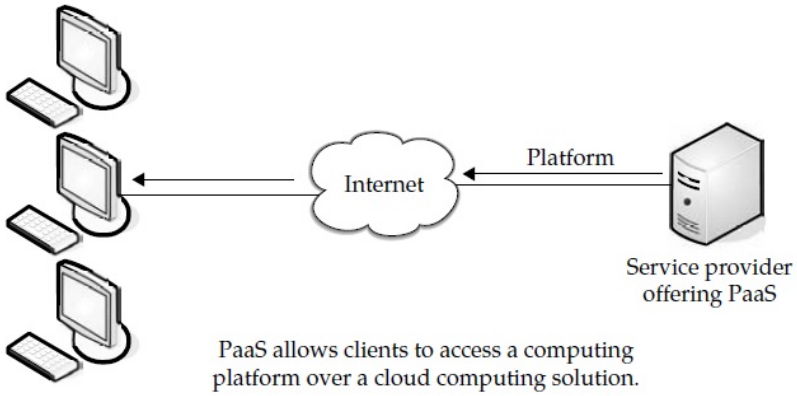

Figure 6: Platform as a service (PaaS) (adopted from [1])

Downfall (taken from [1]):

- lack of portability among different providers.
- if the service provider is out of business, the user's applications, data will be lost.

2.5.3 Hardware as a Service (HaaS)

It is also known as Infrastructure as a Service (IaaS). It offers the hardware as a service to a organisation so that it can put anything into the hardware according to its will [1].

HaaS allows the user to "rent" resources (taken from [1]) as

- Server space
- Network equipment
- Memory
- CPU cycles
- Storage space

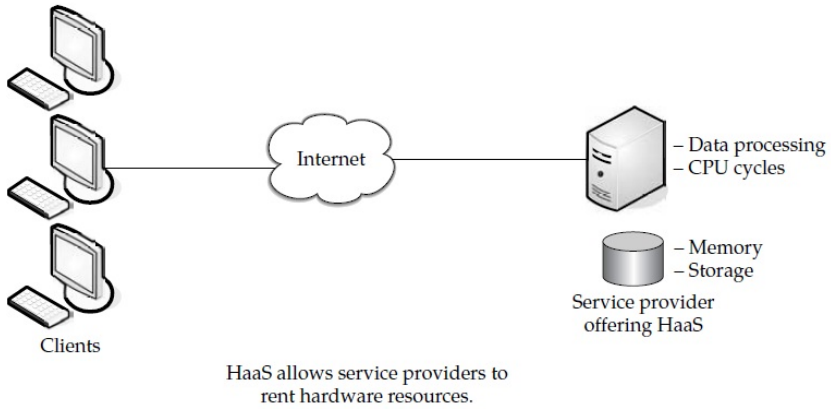

Figure 7: Hardware as a service (HaaS) (adopted from [1])

Cloud computing provides a Service Oriented Architecture (SOA) and Internet of Services (IoS) type applications, including fault tolerance, high scalability, availability, flexibility, reduced information technology overhead for the user, reduced cost of ownership, on demand services etc. Central to these issues lies the establishment of an effective load balancing algorithm.

2.6 Conclusion

This chapter gives a general idea about the basic concepts of cloud computing along with the services provided by cloud computing systems.

Chapter 3

Load Balancing

Goals of Load balancing

Types of Load balancing algorithms

Dynamic Load balancing algorithm

Policies or Strategies in dynamic load balancing

3 Load Balancing

3.1 Introduction

It is a process of reassigning the total load to the individual nodes of the collective system to make resource utilization effective and to improve the response time of the job, simultaneously removing a condition in which some of the nodes are over loaded while some others are under loaded. A load balancing algorithm which is dynamic in nature does not consider the previous state or behavior of the system, that is, it depends on the present behavior of the system. The important things to consider while developing such algorithm are : estimation of load, comparison of load, stability of different system, performance of system, interaction between the nodes, nature of work to be transferred, selecting of nodes and many other ones [4] . This load considered can be in terms of CPU load, amount of memory used, delay or Network load.

3.2 Goals of Load balancing

As given in [4], the goals of load balancing are :

- To improve the performance substantially
- To have a backup plan in case the system fails even partially
- To maintain the system stability
- To accommodate future modification in the system

3.3 Types of Load balancing algorithms

Depending on who initiated the process, load balancing algorithms can be of three catagories as given in [4]:

- **Sender Initiated**: If the load balancing algorithm is initialised by the sender
- **Receiver Initiated**: If the load balancing algorithm is initiated by the receiver

- **Symmetric**: It is the combination of both sender initiated and receiver initiated

Depending on the current state of the system, load balancing algorithms can be divided into 2 catagories as given in [4]:

- **Static**: It doesnt depend on the current state of the system. Prior knowledge of the system is needed
- **Dynamic**: Decisions on load balancing are based on current state of the system. No prior knowledge is needed. So it is better than static approach.

Here we will discuss on various dynamic load balancing algorithms for the clouds of different sizes.

3.4 Dynamic Load balancing algorithm

In a **distributed system**, dynamic load balancing can be done in two different ways: distributed and non-distributed. In the distributed one, the dynamic load balancing algorithm is executed by all nodes present in the system and the task of load balancing is shared among them. The interaction among nodes to achieve load balancing can take two forms: cooperative and non-cooperative [4]. In the first one, the nodes work side-by-side to achieve a common objective, for example, to improve the overall response time, etc. In the second form, each node works independently toward a goal local to it, for example, to improve the response time of a local task. Dynamic load balancing algorithms of distributed nature, usually generate more messages than the non-distributed ones because, each of the nodes in the system needs to interact with every other node. A benefit, of this is that even if one or more nodes in the system fail, it will not cause the total load balancing process to halt, it instead would effect the system performance to some extent. Distributed dynamic load balancing can introduce immense stress on a system in which each node needs to interchange status information with every other node in the system. It is more advantageous when most of the nodes act individually with very few interactions with others.

In **non-distributed** type, either one node or a group of nodes do the task of load balancing. Non-distributed dynamic load balancing algorithms can take two forms: centralized and semi-distributed. In the first form, the load balancing algorithm is executed only by a single node in the whole system: the central node. This node is solely responsible for load balancing of the whole system. The other nodes interact only with the central node. In semi-distributed form, nodes of the system are partitioned into clusters, where the load balancing in each cluster is of centralized form. A central node is elected in each cluster by appropriate election technique which takes care of load balancing within that cluster. Hence, the load balancing of the whole system is done via the central nodes of each cluster [4].

Centralized dynamic load balancing takes fewer messages to reach a decision, as the number of overall interactions in the system decreases drastically as compared to the semi-distributed case. However, centralized algorithms can cause a bottleneck in the system at the central node and also the load balancing process is rendered useless once the central node crashes. Therefore, this algorithm is most suited for networks with small size.

3.5 Policies or Strategies in dynamic load balancing

There are 4 policies [4]:

- **Transfer Policy**: The part of the dynamic load balancing algorithm which selects a job for transferring from a local node to a remote node is reffered to as Transfer policy or Transfer strategy.
- **Selection Policy**: It specifies the processors involved in the load exchange (processor matching)
- **Location Policy**: The part of the load balancing algorithm which selects a destination node for a transferred task is reffered to as location policy or Location strategy.
- **Information Policy**: The part of the dynamic load balancing algorithm responsible for collecting information about the nodes in the system is reffered to as Information policy or Information strategy.

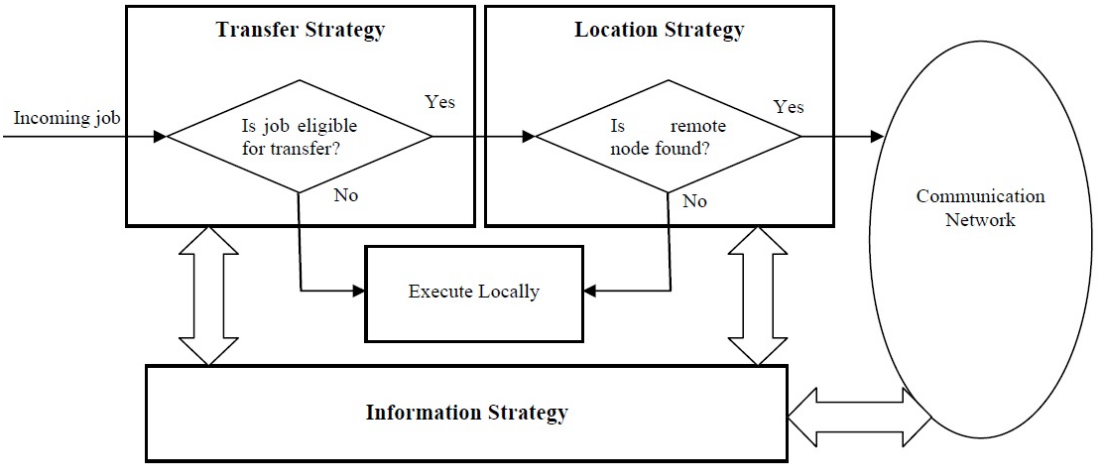

Figure 8: Interaction among components of a dynamic load balancing algorithm (adopted from [4])

3.6 Conclusion

This chapter explains the concept of load balancing, types of load balancing algorithms, general idea about dynamic load balancing algorithms and the different policies that can be used in it.

Chapter 4

Distributed Load Balancing for the Clouds

Honeybee Foraging Algorithm

Biased Random Sampling

Active Clustering

4 Distributed Load Balancing for the Clouds

4.1 Introduction

In complex and large systems, there is a tremendous need for load balancing. For simplifying load balancing globally (e.g. in a cloud), one thing which can be done is, employing techniques would act at the components of the clouds in such a way that the load of the whole cloud is balanced. For this purpose, we are discussing three types of solutions which can be applied to a distributed system [7]: honeybee foraging algorithm, a biased random sampling on a random walk procedure and Active Clustering.

4.2 Honeybee Foraging Algorithm

This algorithm is derived from the behavior of honey bees for finding and reaping food. There is a class of bees called the forager bees which forage for food sources, upon finding one, they come back to the beehive to advertise this using a dance called waggle dance. The display of this dance, gives the idea of the quality or quantity of food and also its distance from the beehive. Scout bees then follow the foragers to the location of food and then began to reap it. They then return to the beehive and do a waggle dance, which gives an idea of how much food is left and hence results in more exploitation or abandonment of the food source.

In case of load balancing, as the webservers demand increases or decreases, the services are assigned dynamically to regulate the changing demands of the user. The servers are grouped under virtual servers (VS), each VS having its own virtual service queues. Each server processing a request from its queue calculates a profit or reward, which is analogous to the quality that the bees show in their waggle dance. One measure of this reward can be the amount of time that the CPU spends on the processing of a request. The dance floor in case of honey bees is analogous to an advert board here. This board is also used to advertise the profit of the entire colony.

Each of the servers takes the role of either a forager or a scout. The server after processing a request can post their profit on the advert boards with a probability of pr. A

server can choose a queue of a VS by a probability of px showing forage/explore behavior, or it can check for advertisements (see dance) and serve it, thus showing scout behavior. A server serving a request, calculates its profit and compare it with the colony profit and then sets its px. If this profit was high, then the server stays at the current virtual server; posting an advertisement for it by probability pr. If it was low, then the server returns to the forage or scout behavior.

```
[A] Initialisation– si in Vj serving Qj, Revenue rate interval Tpr,
                    Advert: posting prob p, reading prob r, read interval Tr
  [B] forever
    [C] while Qj Not Empty do // service queue
          serve request;
          if Tpr expired then
            compute revenue rate;
            adjust r from lookup table;
          if Flip(p) == TRUE then Post Advert;
          if Tr expired && Read(r) == TRUE then
            if forager then Select/Read advert id Vk // randomly select
              else virtual server id Vk // randomly select
            if Vk Not.Eq Vj then Switch(Vk) // migrate to virtual server Vk
      endwhile
    endforever
```

Figure 9: Algorithm used in Honey bee technique (adopted from [7])

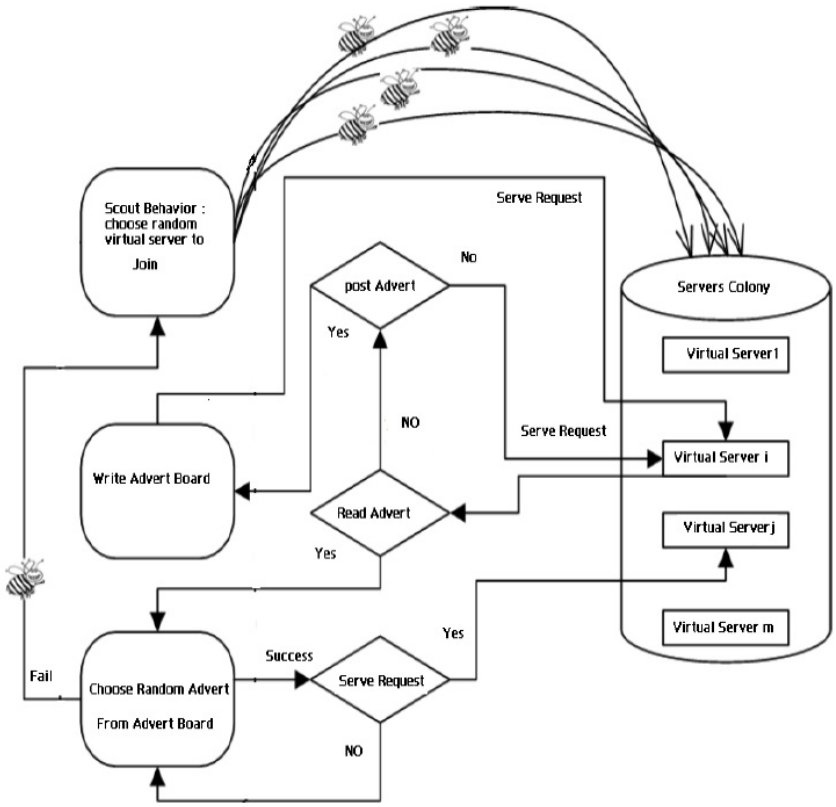

Figure 10: Server Allocations by Foraging in Honey bee technique (adopted from [7])

4.3 Biased Random Sampling

Here a virtual graph is constructed, with the connectivity of each node (a server is treated as a node) representing the load on the server. Each server is symbolized as a node in the graph, with each indegree directed to the free resources of the server.

Regarding job execution and completion,

- Whenever a node does or executes a job, it deletes an incoming edge, which indicates reduction in the availability of free resource.
- After completion of a job, the node creates an incoming edge, which indicates an increase in the availability of free resource.

The addition and deletion of processes is done by the process of random sampling. The walk starts at any one node and at every step a neighbor is chosen randomly. The last node is selected for allocation for load. Alternatively, another method can be used for selection of a node for load allocation, that being selecting a node based on certain criteria like computing efficiency, etc. Yet another method can be selecting that node for load allocation which is underloaded i.e. having highest in degree. If b is the walk length, then, as b increases, the efficiency of load allocation increases. We define a threshold value of b, which is generally equal to log n experimentally.

A node upon receiving a job, will execute it only if its current walk length is equal to or greater than the threshold value. Else, the walk length of the job under consideration is incremented and another neighbor node is selected randomly. When, a job is executed by a node then in the graph, an incoming edge of that node is deleted. After completion of the job, an edge is created from the node initiating the load allocation process to the node which was executing the job.

Finally what we get is a directed graph. The load balancing scheme used here is fully decentralized, thus making it apt for large network systems like that in a cloud.

4.4 Active Clustering

Active Clustering works on the principle of grouping similar nodes together and working on these groups. The process involved is:

- A node initiates the process and selects another node called the matchmaker node from its neighbors satisfying the criteria that it should be of a different type than the former one.
- The so called matchmaker node then forms a connection between a neighbor of it which is of the same type as the initial node.
- The matchmaker node then detaches the connection between itself and the initial node.

The above set of processes is followed iteratively.

4.5 Conclusion

This chapter gives an overall description of various distributed load balancing algorithms that can be used in case of clouds.

Chapter 5

Proposed Work

5 Proposed Work

5.1 Introduction

The time required for completing a task with in one process is very high. So the task is divided into no. of sub-tasks and each sub-task is given one one job. Let the task S is divided into no. of sub-tasks S1,S2,S3...Sn. Out of these some are executed sequentially and some are executed parallely. So the total time period for completing the task decreases and hence the performance increases. These sub-tasks can be represented in a graph structure known as state diagram. An example is given below.

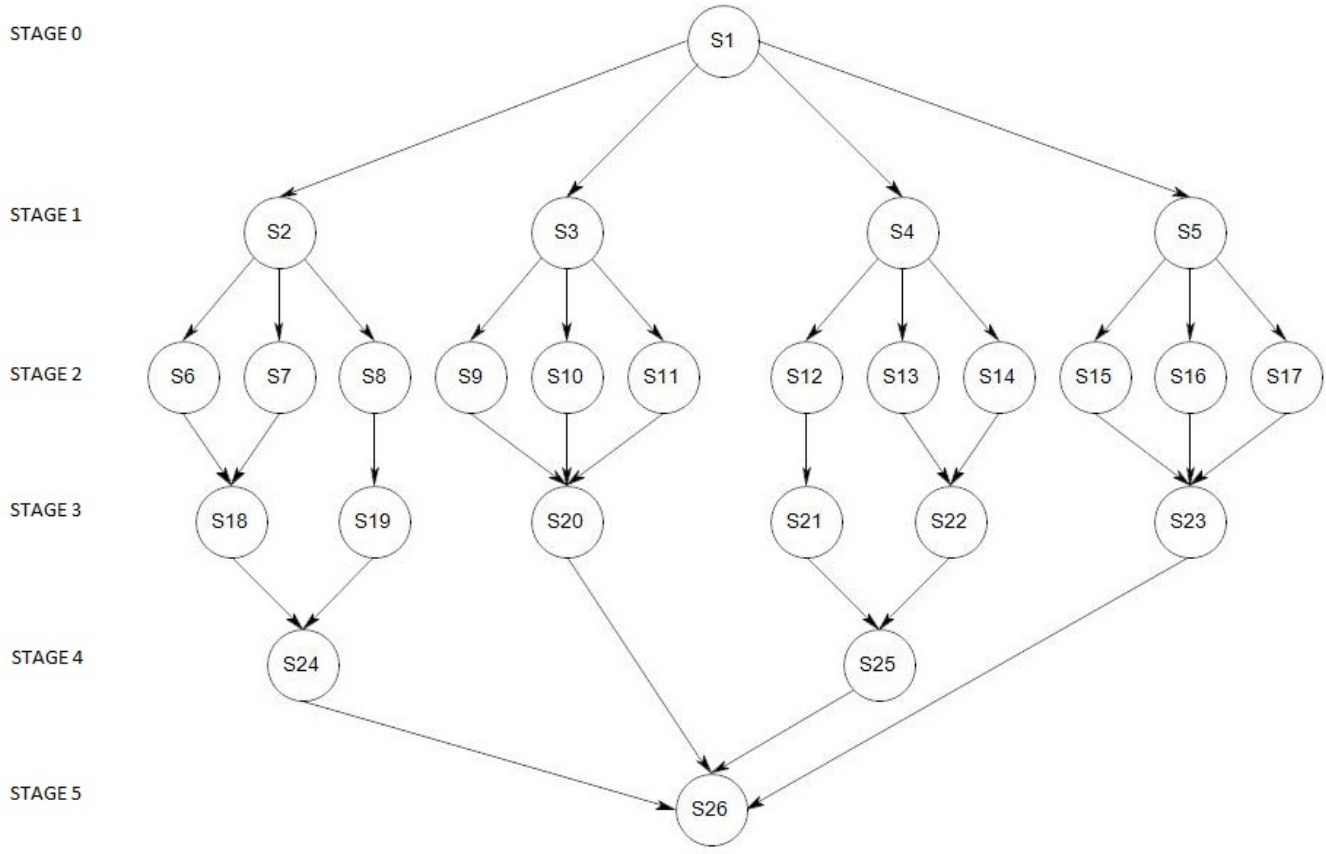

Figure 11: State Diagram

S1 is executed first. S2,S3,S4 and S5 can be executed parallely during the same time slice. S18 requires the execution of S6 and S7 both, but S19 requires the execution of S8 and so on for all the sub tasks as shown in the state diagram. Our aim is to execute these tasks in different nodes of a distributed network so that the performance can be enhanced.

5.2 Description

The distributed network may follow different topologies. The tasks are distributed over the whole network. One topological network connects with the other through a gateway. One of the physiacal topologies forming a cloud is shown in the diagram12.

This distributed network is a cloud, because some of the nodes are Mobile clients, some of them are Thin and some are Thick clients. Some of them are treated as masters and some are treated as slaves. There are one or more datacenters distributed among the various nodes, which keeps track of various computational details. Our aim is to apply the Divisible Load Scheduling Theory(DLT) proposed in [9] for the clouds of different sizes and analyze different performance parameters for different algorithms under DLT and compare them.

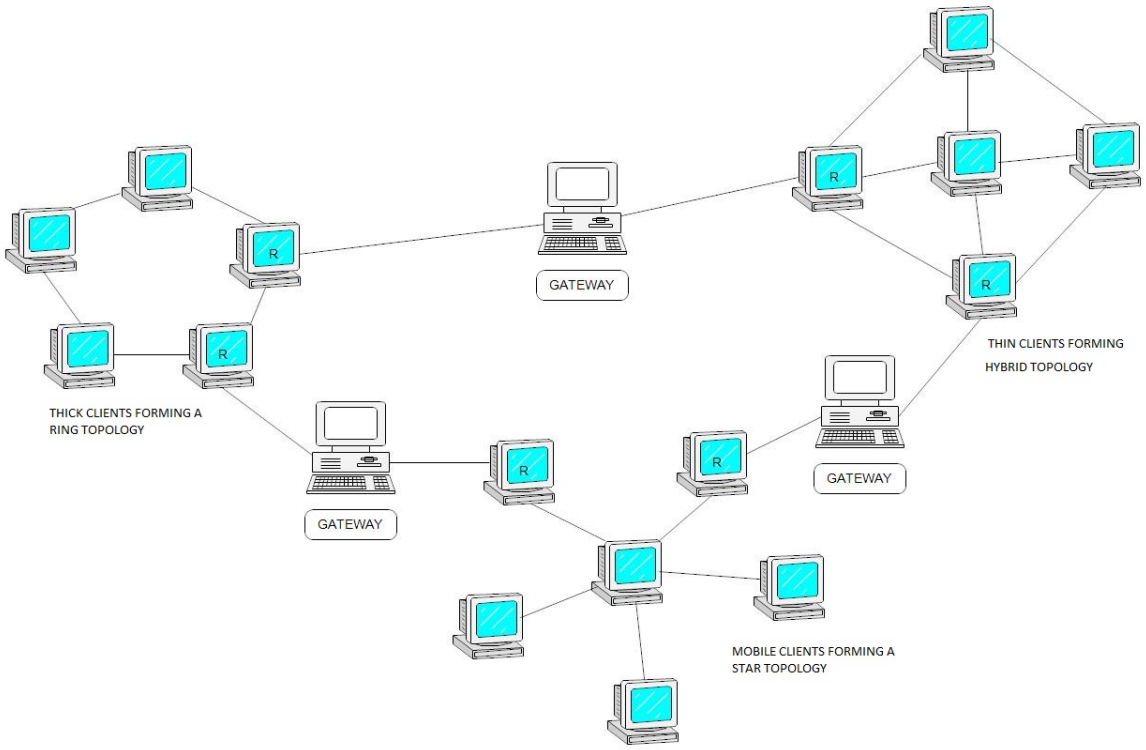

Figure 12: A cloud showing different topologies

5.3 Conclusion

This chapter explains our proposed work.

Chapter 6

Divisible Load Scheduling Theory in Clouds

System Model

Measurement and Reporting Time

6 Divisible Load Scheduling Theory in Clouds

6.1 Introduction

Divisible load scheduling theory (DLT) in case of clouds is an optimal division of loads among a number of master computers, slave computers and their communication links. Our objective is to obtain a mimimal partition of the processing load of a cloud connected via different communication links such that the entire load can be distributed and processed in the shortest possible amount of time [9].

The whole Internet can be viewed as a cloud of many connection-less and connection-oriented services. The concept of load balancing in Wireless sensor netwoks (WSN) proposed in [9] can also be applied to clouds as WSN is analogous to a cloud having no. of master computers (Servers) and no. of slave computers(Clients).

The slave computers are assumed to have a certain measurement capacity. We assume that computation will be done by the master computers, once all the measured data is gathered from corresponding slave computers. Only the measurement and communication times of the slave computers are considered and the computation time of the slave computers is neglected. Here we consider both heterogeneous and homogeneous clouds. That is the cloud elements may possess different measurement capacities, and communication link speeds or the same measurement capacities, and communication link speeds. One slave computer may be connected to one or more master computers at a certain instant of time.

In DLT in case of clouds, an arbitrarily divisible load without having any previous relations is divided and first distributed among the various master computers (for simplicity here the load is divided equally betwwen the master computers) and the each master computer distributes the load among the corresponding slave computers so that the entire load can be processed in shortest possible amount of time. An important reason for using DLT is its flexibility, tractability, data parallelism, computational difficulties[9].

6.2 System Model

The cloud that we have considered here is a single level tree (star) topology consisting of K no. of master computers and each communicating *N* no. of slave computers as shown in Fig 13.

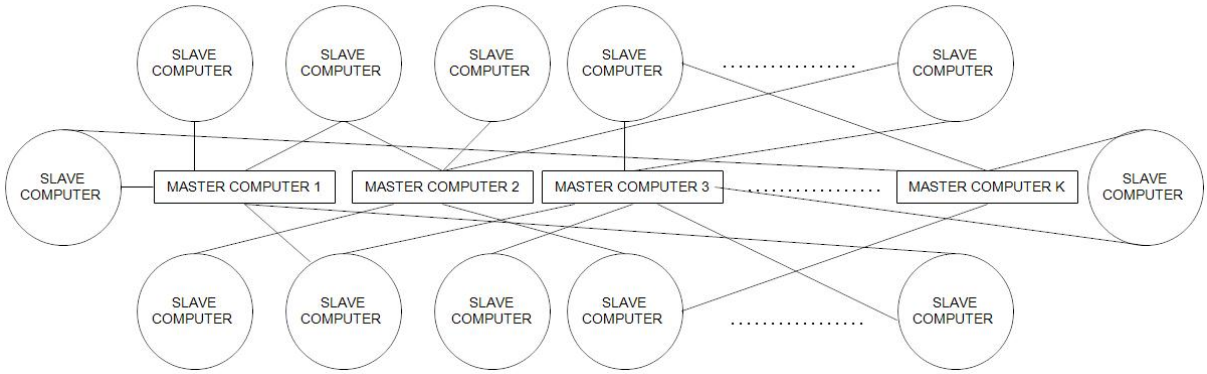

Figure 13: K no. of master computers each joing N no. of slave computers in single level Tree network (STAR Topology)

It is assumed that the total load considered here is of the arbitrarily divisible kind that can be partitioned into fractions of loads to be assigned to all the master and slave computers in the cloud. In this case each master computer first assigns a load share to be measured to each of the corresponding *N* slave computers and then receives the measured data from each slave. Each slave then begins to measure its share of the load once the measurement instructions from the respective master have been completely received by each slave. We also assume that computation time is negligible compared with communication and measurement time.

6.2.1 Parameters, Notation and Definitions

β_{k_i} The fraction of load that is assigned to a slave *i* by master *k*.

a_{k_i} A constant that is inversely proportional to the measuring speed of slave *i* in the cloud.

b_{k_i} A constant that is inversely proportional to the communication speed of link *i* in the cloud.

T_{ms} Measurement intensity constant. This is the time it takes the ith slave to measure the entire load when $a_{k_i} = 1$. The entire assigned measurement load can be measured on the

ith slave in time $a_{k_i}T_{ms}$.

T_{cm} Communication intensity constant. This is the time it takes to transmit all of the measurement load over a link when b_{k_i} = 1. The entire load can be transmitted over the ith link in time $b_{k_i}T_{cm}$.

T_{k_i} The total time that elapses between the beginning of the scheduling process at t =0 and the time when slaver i completes its reporting to the maste k, i =0,1,... ,N. This includes, in addition to measurement time, reporting time and idle time.

T_{f_k} This is the time when the last slave of the master k finishes reporting (finish time or make-span).

$$T_{f_k} = max(T_{k_1}, T_{k_2}, T_{k_3}, ..., T_{k_N}).$$

T_f This is the time when the last master receives the measurement from its last slave.

$$T_f = max(T_{f_1}, T_{f_2}, T_{f_3}, ..., T_{f_N}).$$

Some of the above used parameters and notations are taken from [9]. These parameters were already used for finding closed form equations for load balancing for Wireless Sensor Networks(WSN) in [9].

6.3 Measurement and Reporting Time

6.3.1 When Measurement starts Simultaneously and Reporting is done sequentially

Initially when time t = 0, all the slaves are idle and the master computers start to communicate with the first slave of the corresponding slaves in the cloud. By time t = t_1, each slave will receive its instructions for measurement from the corresponding master as shown in fig 14. It is assumed that after measurements are made, only one slave will report back to the root master at a time (or we can say only a single link exists between them).

The slaves here receive a fraction of load from their corresponding master sequentially and the computation will start after each slave completely receives its load share.

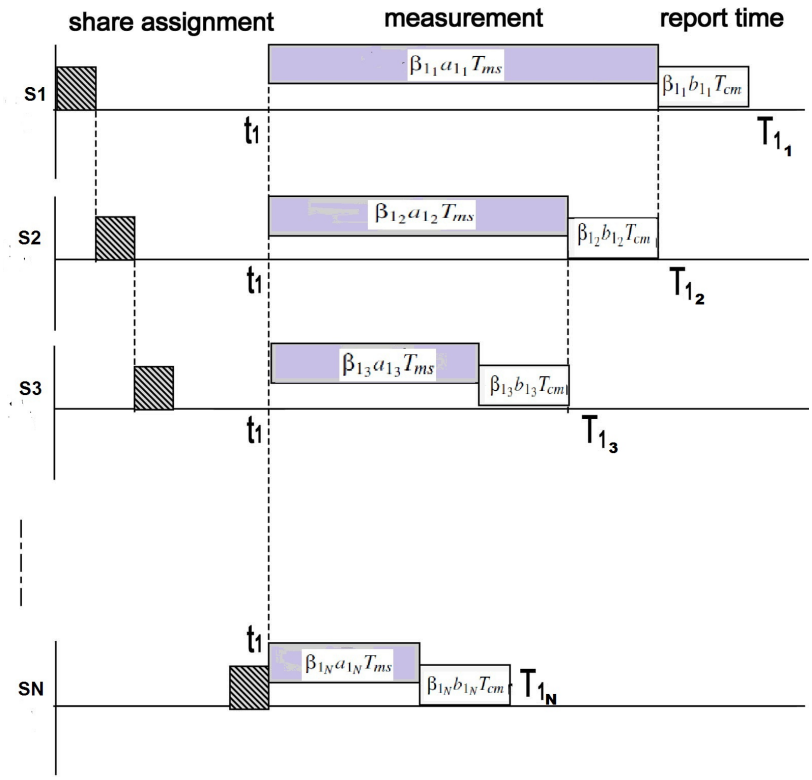

Figure 14: Timing diagram for single level tree network with a master computer and N slaves which report sequentially(adopted from [9])

Let us consider the first master computer and its corresponding group of slaves. From the definition of T_{k_i}, we can write

$$T_{1_1} = t_1 + \beta_{1_1} a_{1_1} T_{ms} + \beta_{1_1} b_{1_1} T_{cm} \tag{1}$$

$$T_{1_2} = t_1 + \beta_{1_2} a_{1_2} T_{ms} + \beta_{1_2} b_{1_2} T_{cm} \tag{2}$$

.

.

.

$$T_{1_N} = t_1 + \beta_{1_N} a_{1_N} T_{ms} + \beta_{1_N} b_{1_N} T_{cm} \tag{3}$$

The total measurement load originating at all the master computers is assumed to be normalized to a unit load. Thus each master computer will handle (1/K) load. So

$$\beta_{1_1} + \beta_{1_2} + \beta_{1_3} + ... + \beta_{1_{N-1}} + \beta_{1_N} = 1/K \tag{4}$$

Based on the timing diagram, we can write

$$\beta_{1_1} a_{1_1} T_{ms} = \beta_{1_2} a_{1_2} T_{ms} + \beta_{1_2} b_{1_2} T_{cm} \tag{5}$$

$$\beta_{1_2} a_{1_2} T_{ms} = \beta_{1_3} a_{1_3} T_{ms} + \beta_{1_3} b_{1_3} T_{cm} \tag{6}$$

.

.

.

$$\beta_{1_{N-2}} a_{1_{N-2}} T_{ms} = \beta_{1_{N-1}} a_{1_{N-1}} T_{ms} + \beta_{1_{N-1}} b_{1_{N-1}} T_{cm} \tag{7}$$

$$\beta_{1_{N-1}} a_{1_{N-1}} T_{ms} = \beta_{1_N} a_{1_N} T_{ms} + \beta_{1_N} b_{1_N} T_{cm} \tag{8}$$

A general expression for the above set of equations is

$$\beta_{1_i} = s_{1_i} \beta_{1_{i-1}} \tag{9}$$

where $s_{1_i} = a_{1_{i-1}} T_{ms} / (a_{1_i} T_{ms} + b_{1_i} T_{cm}) and i = 2, 3, ..., N$. The above recursive equation for β_{1_i} can be rewritten in terms of β_{1_1} only as

$$\beta_{1_i} = \prod_{j=2}^{i} s_{1_j} \beta_{1_1} \tag{10}$$

Now using the above sets of equations and the normalization equation, one can solve for β_{1_1} as

$$\beta_{1_1} + \sum_{i=2}^{N} \prod_{j=2}^{i} s_{1_j} \beta_{1_1} = 1/K \tag{11}$$

So β_{1_1} can be written as

$$\beta_{1_1} = \frac{1}{K(1+\sum_{i=2}^{N}\prod_{j=2}^{i} s_{1_j})} \tag{12}$$

Putting in eq-(10),

$$\beta_{1_i} = \frac{\prod_{j=2}^{i} s_{1_j}}{K(1+\sum_{i=2}^{N}\prod_{j=2}^{i} s_{1_j})} \tag{13}$$

where i =2,3,4,... ,*N*.

The minimum measuring and reporting time of the network will then be given as

$$T_{f_1} = t_1 + \frac{(a_{1_1}T_{ms} + b_{1_1}T_{cm})}{K(1+\sum_{i=2}^{N}\prod_{j=2}^{i} s_{1_j})} \tag{14}$$

Similarly we can obtain the generalised equation for master computer *r* as

$$T_{f_r} = t_1 + \frac{(a_{r_1}T_{ms} + b_{r_1}T_{cm})}{K(1+\sum_{i=2}^{N}\prod_{j=2}^{i} s_{r_j})} \tag{15}$$

In case of homogeneous networks (same measurement capacities and link speeds), we can write

$s_{1_1} = s_{1_2} = s_{1_3} = ... = s_{1_{N-1}} = s_1$

$a_{1_1} = a_{1_2} = a_{1_3} = ... = a_{1_N} = a_1$

$b_{1_1} = b_{1_2} = b_{1_3} = ... = b_{1_N} = b_1$

So, eq-(5) becomes

$$\beta_{1_1}(1 + s_1 + s_1^2 + ... + s_1^{N-2} + s_1^{N-1}) = 1/K \tag{16}$$

where $s_1 = a_1T_{ms}/(a_1T_{ms} + b_1T_{cm})$.

Simplifying the above equation,

$$\beta_{1_1} = \frac{1 - s_1}{K(1 - s_1^N)} \tag{17}$$

The master computer 1 will use the value of β_{1_1} to obtain the amount of data that has to be measured by the rest of the N-1 slaves corresponding to it by using the following

equation :

$$\beta_{1_i} = \beta_{1_1} s_1^{i-1} \tag{18}$$

where i =2,3,4,... ,N.

The minimum measuring and reporting time of the homogeneous network will then be given as

$$T_{f_1} = t_1 + \frac{(a_1 T_{ms} + b_1 T_{cm})(1 - s_1)}{K(1 - s_1^N)}. \tag{19}$$

This measurement and reporting time of the network approaches $t_1 + (b_1 T_{cm})/K$ as N approaches infinity. So the reporting time supresses the measurement time when the no. of slaves to a corresponding master approaches infinty. Similarly we can obtain the above expression for rest of the master computers.

6.3.2 When the Measurement starts Simultaneously and Reporting ends Simultaneously

Here each of N slave computers corresponding to a master computer in the cloud finish reporting at the same time. The cloud will have the same report finishing time for each slave corresponding to a master. That is each slave has a separate channel to its master as shown in the timing diagram of the network.

In this case the slaves receive their share of load from the master concurrently and start computation after completely receiving their share of load. Each slave begins to measure its share of the load at the moment when all finish receiving their measurement instructions from the corresponding master. From the definition of T_{k_i}, we can write

$$T_{1_1} = t_1 + \beta_{1_1} a_{1_1} T_{ms} + \beta_{1_1} b_{1_1} T_{cm} \tag{20}$$

$$T_{1_2} = t_1 + \beta_{1_2} a_{1_2} T_{ms} + \beta_{1_2} b_{1_2} T_{cm} \tag{21}$$

.

.

.

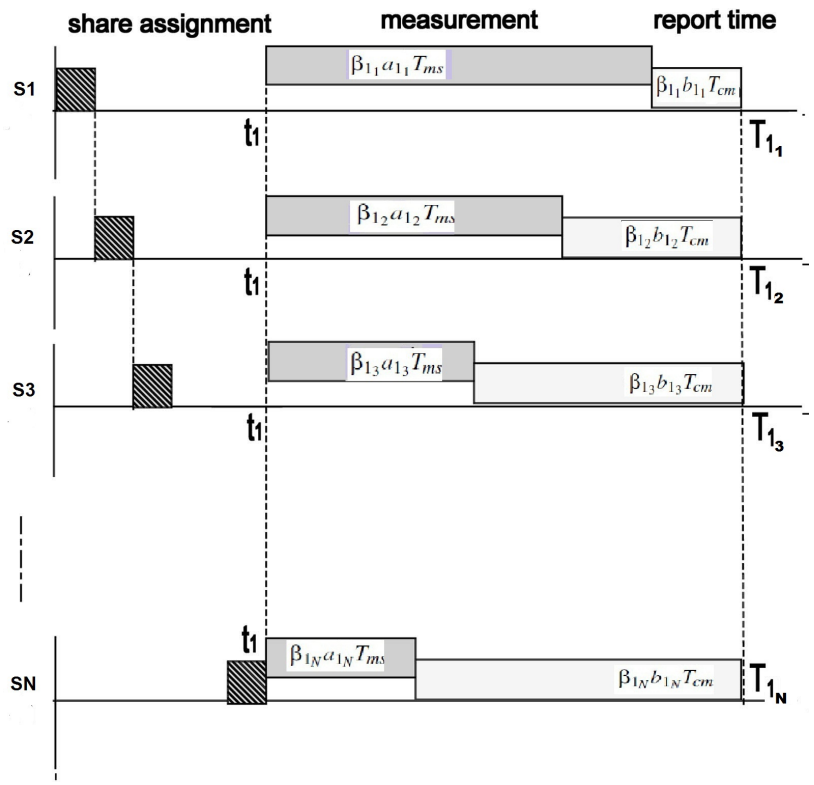

Figure 15: Timing diagram for a master computer and N slaves with simultaneous reporting termination (adopted from [9])

$$T_{1_N} = t_1 + \beta_{1_N} a_{1_N} T_{ms} + \beta_{1_N} b_{1_N} T_{cm} \tag{22}$$

The total measurement load originating at all the master computers is assumed to be normalized to a unit load. Thus each master computer will handle (1/K) load. So

$$\beta_{1_1} + \beta_{1_2} + \beta_{1_3} + ... + \beta_{1_{N-1}} + \beta_{1_N} = 1/K \tag{23}$$

In this case since all processors stop reporting at the same time, we have $T_{1_1} = T_{1_2} = T_{1_3} = ... = T_{1_N}$.

Based on the timing diagram, we can write for master computer 1 and its slaves,

$$\beta_{1_1} r_{1_1} = \beta_{1_2} r_{1_2} \tag{24}$$

$$\beta_{1_2} r_{1_2} = \beta_{1_3} r_{1_3} \tag{25}$$

.

.

.

$$\beta_{1_{N-2}} r_{1_{N-2}} = \beta_{1_{N-1}} r_{1_{N-1}} \tag{26}$$

$$\beta_{1_{N-1}} r_{1_{N-1}} = \beta_{1_N} r_{1_N} \tag{27}$$

where $r_{1_i} = a_{1_i} T_{ms} + b_{1_i} T_{cm}$, i =1,2,... ,N.

Puuting the above equations in eq.-(24),

$$\beta_{1_1} = \frac{1}{K(1 + r_{1_1} \sum_{i=2}^{N} \frac{1}{r_{1_i}})} \tag{28}$$

So we can write β_{1_i} as

$$\beta_{1_i} = \frac{\frac{1}{r_{1_i}}}{K(\sum_{i=1}^{N} \frac{1}{r_{1_i}})} \tag{29}$$

From the above expression, it can be easily seen that the share of each slave corresponding to its master will entirely depend on the combined speed of the measurement and communication of that slave. The minimum measurement and reporting time of the network will then be given as

$$T_{f_1} = T_{1_1} = t_1 + \frac{(a_{1_1} T_{ms} + b_{1_1} T_{cm}) \frac{1}{r_{1_1}}}{K(\sum_{i=1}^{N} \frac{1}{r_{1_i}})} \tag{30}$$

Similarly for the master computer p, the generalised equation will be

$$T_{f_p} = T_{1_p} = t_1 + \frac{(a_{p_1} T_{ms} + b_{p_1} T_{cm}) \frac{1}{r_{p_1}}}{K(\sum_{i=1}^{N} \frac{1}{r_{p_i}})} \tag{31}$$

For the case of a homogeneous network, each slave corresponding to a master in the network shares the load equally. That is, β_{1_i} =1/(KN), for i =1,2,3,... ,N. So, the minimum

measuring and reporting time of the network will be

$$T_{f_1} = t_1 + \frac{a_1 T_{ms} + b_1 T_{cm}}{KN} \tag{32}$$

Similarly we can obtain the above expression for rest of the master computers.

6.4 Conclusion

This chapter describes the concept of divisible load scheduling theory and how it can be applied in case of clouds. It also explains the proposed system model, the various notations used and analysis of measurement and reporting time for the two cases that we have considered.

Chapter 7

Performance Evaluation

When Measurement starts Simultaneously and Reporting is done sequentially

When the Measurement starts Simultaneously and Reporting ends Simultaneously

7 Performance Evaluation

7.1 Introduction

Here we consider the following two cases. In the first case the measurement and reporting time is plotted against the number of slaves corresponding to a master, where the link speed b is varied and measurement speed a is fixed. In the second case, the measurement and reporting time is plotted against the number of slaves corresponding to master, where link speed b is fixed and measurement speed a is varied.

7.2 When Measurement starts Simultaneously and Reporting is done sequentially

In Fig. 16, the measurement/report time is plotted against the number of homogeneous slaves corresponding to a master when the value of the communication speed b is varied from 0 to 1 at an interval of 0.3 and the value of measurement speed a is fixed to be 1.5. In all cases T_{cm} =1 and T_{ms} = 1. From the figure we can infer that the faster the communication speed, the smaller the measurement/report time and the measurement/report time levels off after a certain number of slaves for each performance curve. No. of master computers in the cloud doesn't have significant contribution to the measurement/report time of a single master.

Fig. 17 shows for the case when the inverse measuring speed a is varied from 1 to 2 at an interval of 0.3 and the inverse link speed b is fixed to be 0.2. The result confirms that the measurement time approaches $b_1 T_{cm}$, which in this case is 0.2, as N approaches infinity.

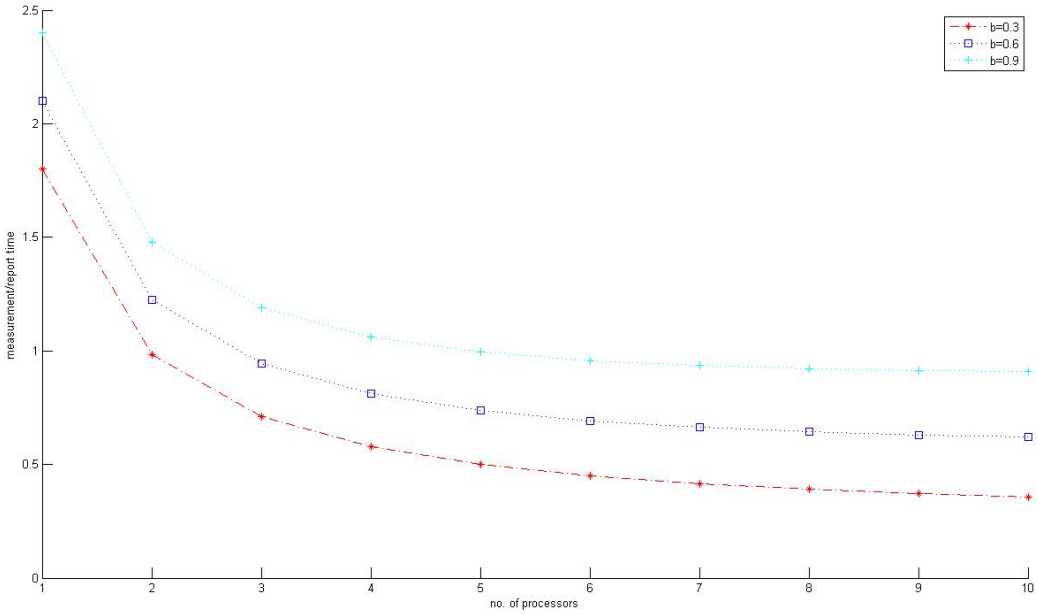

Figure 16: Measurement/report time versus number of slaves corresponding to master and variable inverse link speed b for single level tree network with master and sequential reporting time.

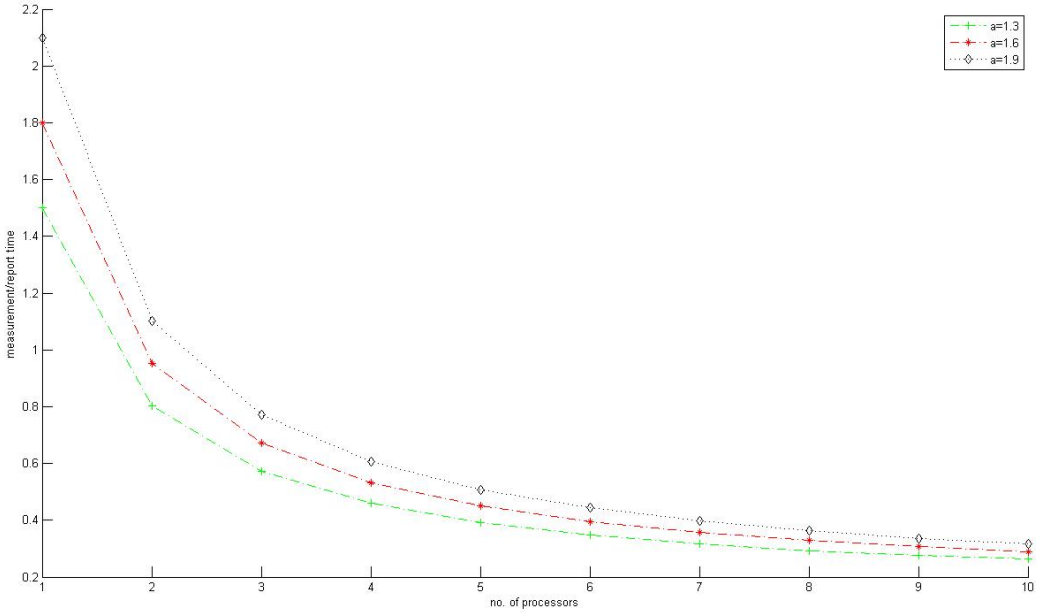

Figure 17: Measurement/report time versus number of slaves corresponding to master and variable inverse measuring speed a for single level tree network with master and sequential reporting time.

7.3 When the Measurement starts Simultaneously and Reporting ends Simultaneously

In Fig. 18, the measurement/report time is plotted against the number of slaves corresponding to a master for the simultaneous measurement start simultaneous reporting termination case. The value the inverse link speed b is varied from 0 to 1 at an interval of 0.3 while the inverse measuring speed a is fixed to be 1.5. In this case the minimum finish time decreases as the number of slaves under a master in the network is increased. This assumes that the communication speed is fast enough to distribute the load to all the slaves under a master.

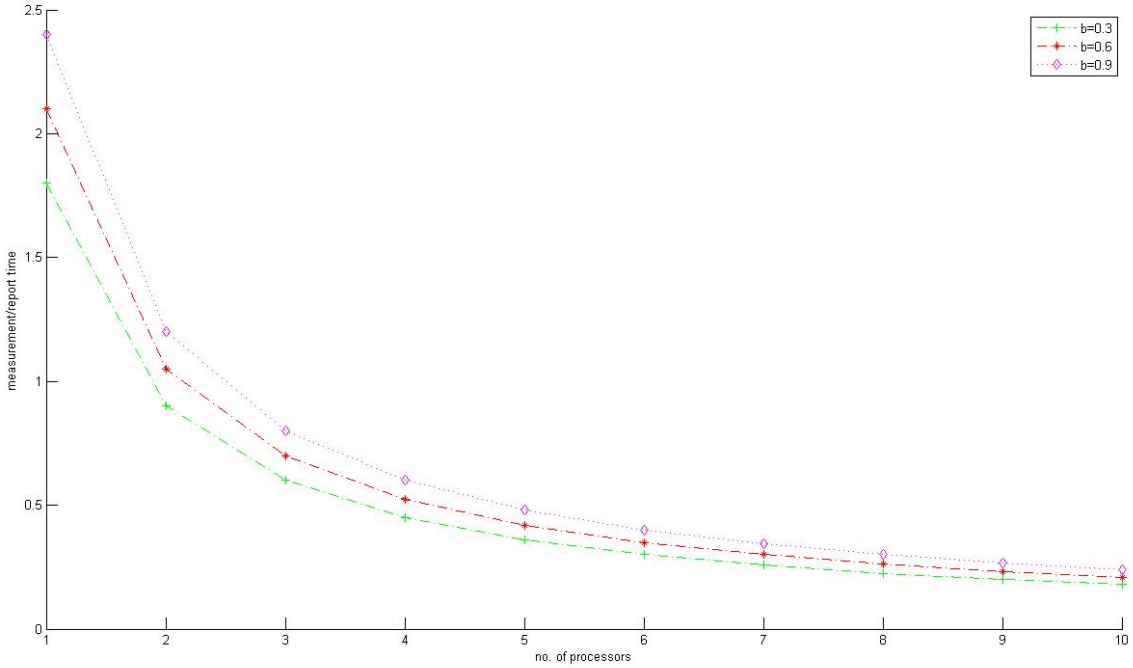

Figure 18: Measurement/report time versus number of slaves under a master and variable inverse link speed b for single level tree network with master

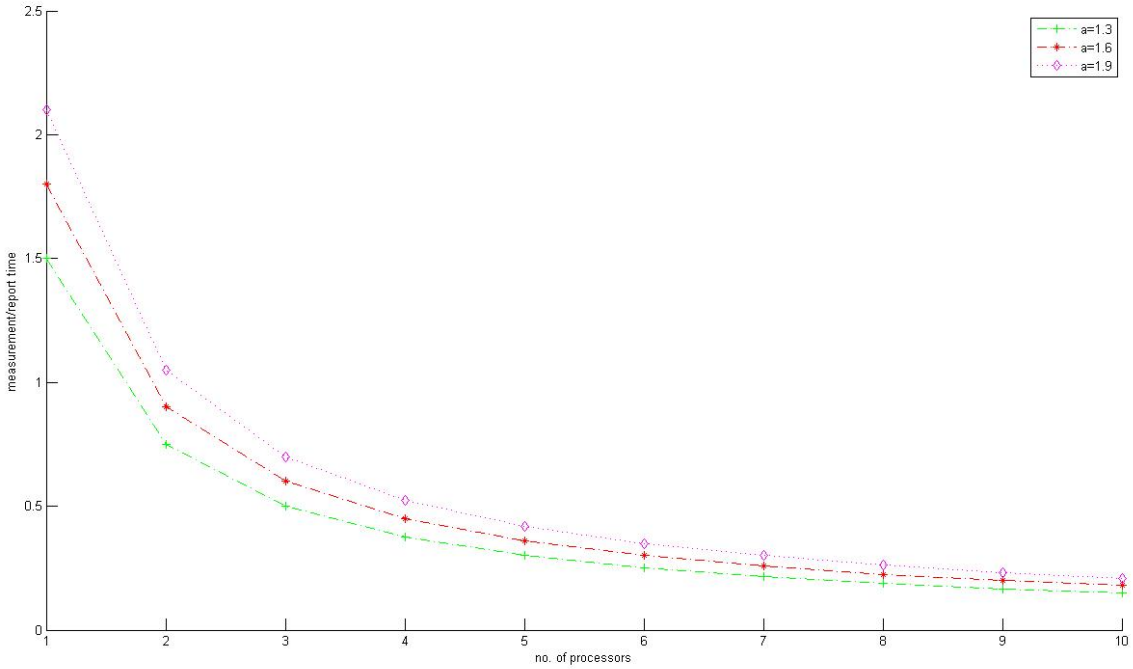

Figure 19: Measurement/report time versus number of slaves under a master and variable inverse measuring speed a for single level tree network with master

Fig. 19 shows for the case when the inverse measuring speed a is varied from 1 to 2 at an interval of 0.3 and the inverse link speed b is fixed to be 0.2.

7.4 Conclusion

This chapter evalutes the performance of the two cases that we have considered in this paper. It also shows the simulation results that we have got.

Chapter 8

Conclusion and Future Work

Conclusion

Future Work

8 Conclusion and Future Work

8.1 Conclusion

Figure 20 shows the comparision between the measurement/reporting time of both the approaches for the same no. of slave computers corresponding to the same master. Here the inverse link speed b is taken as 1 and the inverse measurement speed a is 0.5 for both the cases. Number of master computers is taken to be constant equal to 50. The plot shows that the measurement/reporting time is smaller in case of simultaneous reporting as compared to sequential reporting. It is because in case of sequential reporing, some of the slaves recieve almost zero load from its master. Number of effective slaves in this case is less as compared to the simultaneous reporting case. Hence with increase in no. of slaves with respect to a master, the finishing time remains almost same in case of sequential reprting whereas in case of simultaneous reporting, the finishing time decreases for the increase in no. of slaves corresponding to a single master. The grpah shows that the finishing time can be improved by increasing the number of slaves under a master computer in a cloud only to some extent before saturation in case of sequential measurement and sequential reporting strategy. But finishing time can be decreased significantly in case of simultaneous measurement start and simultaneous reporting termination by increasing the no. of slaves under a single master computer.

Till now we have discussed on basic concepts of Cloud Computing and Load balancing and studied some existing load balancing algorithms, which can be applied to clouds. In addition to that, the closed-form solutions for minimum measurement and reporting time for single level tree networks with different load balancing strategies were also studied. The performance of these strategies with respect to the timing and the effect of link and measurement speed was studied. A comparison is also made between different strategies.

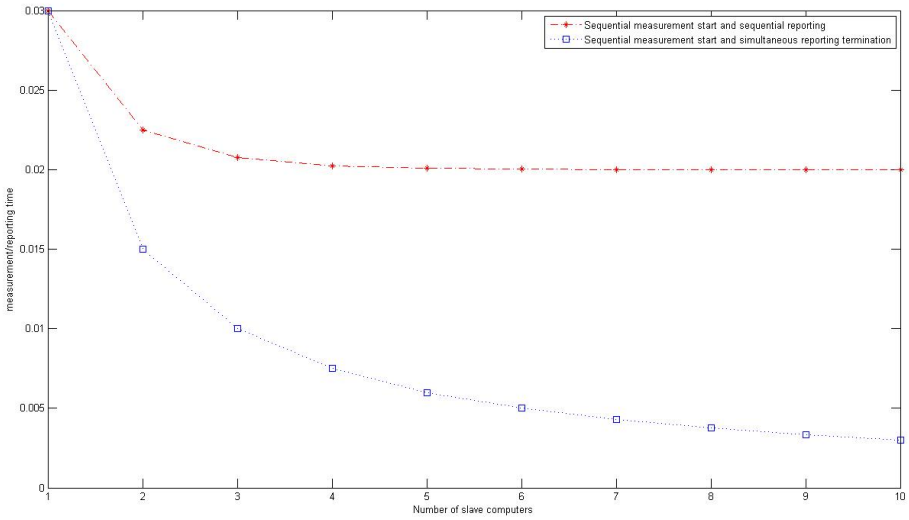

Figure 20: Comparison of Measurement/report time versus number of slaves under a single master under the same conditions of link speed and measurement speed for both cases of reporting

8.2 Future Work

Cloud Computing is a vast concept and load balancing plays a very important role in case of Clouds. There is a huge scope of improvement in this area. We have discussed only two divisible load scheduling algorithms that can be applied to clouds, but there are still other approaches that can be applied to balance the load in clouds. The performance of the given algorithms can also be increased by varying different parameters.

References

[1] Anthony T.Velte, Toby J.Velte, Robert Elsenpeter, Cloud Computing A Practical Approach, TATA McGRAW-HILL Edition 2010.

[2] Martin Randles, David Lamb, A. Taleb-Bendiab, A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing, 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops.

[3] Mladen A. Vouk, Cloud Computing Issues, Research and Implementations, Proceedings of the ITI 2008 30th Int. Conf. on Information Technology Interfaces, 2008, June 23-26.

[4] Ali M. Alakeel, A Guide to Dynamic Load Balancing in Distributed Computer Systems, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.

[5] http://www-03.ibm.com/press/us/en/pressrelease/22613.wss

[6] http://www.amazon.com/gp/browse.html?node=201590011

[7] Martin Randles, Enas Odat, David Lamb, Osama Abu- Rahmeh and A. Taleb-Bendiab, ”A Comparative Experiment in Distributed Load Balancing”, 2009 Second International Conference on Developments in eSystems Engineering.

[8] Peter S. Pacheco, ”Parallel Programming with MPI”, Morgan Kaufmann Publishers Edition 2008

[9] Mequanint Moges, Thomas G.Robertazzi, ”Wireless Sensor Networks: Scheduling for Measurement and Data Reporting”, August 31, 2005