Efficient Load Balancing Techniques for Cloud Computing

Omkumari Soni¹, Shalini Khare², Rama Soni³

¹Lecturer- IT
Dr.B.R.A.Polytechnic,Gwalior(M.P.)

²Lecturer-E&T
S.R.Govt.Polytechnic,Sagar(M.P.)

³Lecturer-IT
Govt.Polytechnic,Panna(M.P.)

Abstract

Cloud computing mean is storing and accessing data and programs over the internet instead of computer's hard drive. The cloud is just a metaphor for the Internet. A cloud consists of several elements such as clients, data centre and distributed servers and also includes fault tolerance, high availability, scalability, flexibility, reduced overhead for users, reduced cost of ownership, and on demand services etc. Central to these issues lays the establishment of an effective load balancing algorithm. Load balancing ensures that all the processor in the system or every node in the network does nearly equal amount of work at any instant of time. This technique can be sender initiated, receiver initiated or symmetric. In this paper we developed an effective load balancing algorithm using load scheduling theorem with different parameters for the clouds of different sizes for higher performance.

Keywords: Cloud Computing, Load Balancing, Load Scheduling Theorem

1. Introduction

The 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. Normally, it consists of a bunch of distributed servers known as masters, providing services and resources to clients known as clients in a network with scalability and reliability of data centre. Distributed computers provide on-demand services [1]. Cloud computing is a great extent concept. There are many algorithms for load balancing in cloud computing have been proposed. The whole Internet can be considered as a cloud of many connections less and connection oriented services on the divisible load scheduling theory for Wireless networks [2]. The load balancing in cloud computing systems is really a challenge now. Distributed solution is required but, it is not always practically feasible or cost efficient to maintain one or more idle services just as to fulfil the required requirements. Jobs can’t 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 [3] and [4].

In this paper we designed and developed the concept of load balancing using Divisible Load Scheduling Theory (DLT) for the clouds of different sizes

2. Background Techniques

Cloud system

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

Based on the domain or environment in which clouds are used, clouds can be divided into 3 categories
- Public Clouds
- Private Clouds
- Hybrid Clouds (combination of both private and public clouds)

Virtualization

It is a very useful 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 data enter will provide different services in a full or partial virtualised manner.

Two types of virtualization are found in case of clouds as given:
- Full virtualization
- Para virtualization

Full Virtualization

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

Para virtualization has the following advantages as given:
- **Disaster recovery:** In the event of a system failure, guest instances are moved to 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 easier.
- **Capacity management:** In a virtualised environment, it is easier and faster to add more hard drive capacity and processing power. As the system parts or hardwires can be moved or replaced or repaired easily, capacity management is simple and easier [1] and [3] and [5].

3. 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:
- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Hardware as a Service (HaaS) or Infrastructure as a Service (IaaS)

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 does not need to make lots of changes or doesn’t require integration to other systems. The provider does all the upgrades and patching while keeping the infrastructure running. 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
- 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. Some other benefits include. Familiarity with the Internet should 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, it may charge the moving fee. Thus it will increase the unnecessary costs.

SaaS faces challenges from the availability of cheaper hardwire and open source applications.

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 software.
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 [2] and [4] and [6].

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 HaaS allows the user to “rent” resources as

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

4. Load Balancing

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 behaviour of the system. That is it depends on the present behaviour 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. This load considered can be in terms of CPU load, amount of memory used, delay or Network load.

Goals of Load balancing

The goals of load balancing are as follows:

- 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

Types of Load balancing algorithms

Depending on who initiated the process, load balancing algorithms can be of three categories as given:

Sender Initiated: If the load balancing algorithm is initialised by the sender

Receiver Initiated: If the load balancing algorithm is initiated by the receiver [3] and [7] and [8].

Dynamic Load balancing algorithm

Symmetric: It is the combination of both sender initiated and receiver initiated and depending on the current state of the system, load balancing algorithms can be divided into 2 categories.

Static: It doesn’t 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 discuss on various dynamic load balancing algorithms for the clouds of different sizes [9] and [10].
5. Proposed Techniques

Introduction

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 of combination of sender initiated and receiver initiated types. In this paper we developed the effective load balancing algorithm using divisible load scheduling theorem to maximize or minimize different performance parameters for the clouds of different sizes virtual topology depending on the application requirement.

Description of Proposed Techniques

The time required for completing a task within one process is very high. So the task is divided into no. of sub-tasks and each sub-task is given one job. Let the task \( S \) is divided into no. of sub-tasks \( S_1, S_2, S_3, ..., S_n \). Out of these some are executed sequentially and some are executed parallel. 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:

\( S_1 \) is executed first. \( S_2, S_3, S_4 \) and \( S_5 \) can be executed parallel during the same time slice. \( S_9 \) requires the execution of \( S_6 \) and \( S_7 \) both, but \( S_{19} \) requires the execution of \( S_8 \) 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.

![State Diagram](image)

Figure 1 State Diagram

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 physical topologies forming a cloud is shown in the diagram. 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 is one or more data enters distributed among the various nodes, which keeps track of various computational details. Our aim is to apply the Divisible Load Scheduling Theory (DLT) proposed for the clouds of different sizes and analyze different performance parameters for different algorithms under DLT and compare them.

Proposed 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 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.

Parameters, Notation, and Definitions

- \( β ki \): The fraction of load that is assigned to a slave \( i \) by master \( k \).
- \( a ki \): A constant that is inversely proportional to the measuring speed of slave \( i \) in the cloud.
- \( b ki \): A constant that is inversely proportional to the communication speed of link \( i \) in the cloud.
- \( Tms \): Measurement intensity constant. This is the time it takes the \( i \)-th slave to measure the entire load when \( a ki = 1 \).
- \( Tcm \): Communication intensity constant. This is the time it takes to transmit the entire measurement load over a link when \( b ki = 1 \).
- \( T ki \): The total time that elapses between the beginning of the scheduling process at \( t = 0 \) and the time when slave \( i \) completes its reporting to the master \( k \). \( i = 0, 1, ..., N \). This includes, in addition to measurement time, reporting time and idle time.
- \( Tfk \): This is the time when the last slave of the master \( k \) finishes reporting (finish time or make-span).

Measurement and Reporting Time

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 \( t = t1 \), each slave will receive its instructions for measurement from the corresponding master. 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.

Measurement and Reporting Time

Timing diagram for single level tree network with a master computer and \( N \) slaves which report sequentially:

Let us consider the first master computer and its corresponding group of slaves. From the definition of \( T ki \), we can write

\[
T11 = t1 + b11a11Tms + b11b11Tcm \\
T12 = t1 + b12a12Tms + b12b12Tcm \\
\vdots \\
T1N = t1 + b1Na1NTms + b1Nb1NTcm
\]

Measurement and Reporting Time

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

\[
b11 + b12 + b13 + \cdots + b1N = 1 = K
\]

Based on the timing diagram, we can write

\[
b11a11Tms = b12a12Tms + b12b12Tcm \\
b12a12Tms = b13a13Tms + b13b13Tcm \\
\vdots \\
b1N-2a1NTms = b1N-1a1N-1Tms + b1N-1b1N-1Tcm \\
b1N-1a1N-1Tms = b1Na1NTms + b1Nb1NTcm
\]

A general expression for the above set of equations is

\[
b1i = s1ib1i-1
\]
where \( s_{1i} = a_{1i} - 1Tms = (a_{1i} Tms + b_{1i} Tcm) \) and \( d_i = 2; 3; \ldots; N \).

Now using the above sets of equations and the normalization equation Measurement and Reporting Time So \( b_{11} \) can be written as

\[
\begin{align*}
  b_{11} &= 1K(1+\alpha N_i = 2\bar{O}_{ij} = 2 \ s_{1j}) \quad (12) \\
  b_{li} &= \bar{O}_{ij} = 2 \ s_{1j}K(1+\alpha N_i = 2\bar{O}_{ij} = 2 \ s_{1j}) \quad (13)
\end{align*}
\]

where \( i = 2, 3, 4, \ldots, N \). The minimum measuring and reporting time of the network will then be given as

\[
T_{fl1} = t_{1} + (a_{11} Tms + b_{11} Tcm) K(1 + \alpha N_i = 2 \bar{O}_{ij} = 2 \ s_{1j}) \quad (14)
\]

Similarly we can obtain the generalised equation for master computer \( r \) as

\[
T_{fr} = t_{1} + (a_{r1} Tms + b_{r1} Tcm) K(1 + \alpha N_i = 2 \bar{O}_{ij} = 2 \ s_{r j}) \quad (15)
\]

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

\[
s_{11} = s_{12} = s_{13} = \ldots = s_{1N-1} = s_{1} \\
a_{11} = a_{12} = a_{13} = \ldots = a_{1N} = a_{1} \\
b_{11} = b_{12} = b_{13} = \ldots = b_{1N} = b_{1}
\]

So, eq-(5) becomes

\[
b_{11}(1 + s_{1} + s_{21} + \ldots + s_{N-1} 21 + s_{N-1} 11) = 1 = K \quad (16)
\]

where \( s_{1} = a_{1} Tms = (a_{1} Tms + b_{1Tcm}) \).

Simplifying the above equation,

\[
b_{11} = 1 - s_{1} K(1 - s_{N-1}) \quad (17)
\]

The master computer 1 will use the value of \( b_{11} \) 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 Measurement and Reporting Time equation :

\[
b_{li} = b_{11} s_{i-11} \quad (18)
\]

where \( i = 2, 3, 4, \ldots, N \).

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

\[
T_{fl} = t_{1} + (a_{1} Tms + b_{1} Tcm)(1 - s_{1}) K(1 - s_{N-1}) \quad (19)
\]

This measurement and reporting time of the network approaches \( t_{1} + (b_{1} Tcm) = K \) as \( N \) approaches infinity. So the reporting time suppresses the measurement time when the no. of slaves to a corresponding master approaches infinity. Similarly we can obtain the above expression for rest of the master computers.

The proposed techniques for divisible load scheduling theory and it also explains the proposed system model, the various notations used and analysis of measurement and reporting in cloud computing.

6. Results Analysis

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.
When Measurement starts simultaneously and Reporting is done sequentially in Fig. 2, 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 when the Measurement starts simultaneously and Reporting ends simultaneously.

In Fig. 3, the measurement/report time is plotted against the number of slaves corresponding to a master for the simultaneous measurement start simultaneous reporting termination case.

7. Conclusion and Future Work

The 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 were studied. A comparison is also made between different strategies. 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. We developed an effective load balancing algorithm using Divisible load scheduling theorem to maximize or minimize different performance parameters. 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


