Improving the Performance of Load Balancing in Cloud Environment using SJF in MapReduce

Prof. Priya.V, Prof. Subha S
School of Information Technology and Engineering, VIT University, Vellore
School of Information Technology and Engineering, VIT University, Vellore

Abstract – Today Cloud computing is a fast growing area in computing research and industry. Through virtualization many applications can be developed and various services can be offered to the end users. Cloud service providers are provided many services in a very flexible manner so that the users can scale up or scale down as they wish. In this paper, a new load balancing algorithm has been proposed using Hadoop-MapReduce and SJF Preemptive scheduling algorithm between Mapper and Reducer.

Keywords- Cloudsim, DataCenter, Virtualization, Virtual Machine, Load Balancing, MapReduce.

I.INTRODUCTION

Today Cloud computing is a fast growing area in computing research and industry. The term “cloud” and “cloud computing” everywhere means the same. The third party service providers are providing hardware, software and services for the end users over the internet. Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) are the three basic types of services in cloud computing. Virtualized servers, grids or clusters, storage, networks, memory and systems software are provided as a service. Amazon’s Elastic Compute Cloud (EC2) and Simple Storage Service (S3) are the leading service providers, it also provides access to computational resources that is CPUs, and it also provides manageable and scalable resources as a service to the end user. Load balancing mechanisms can be generally classified as dynamic or static, centralized or decentralized and periodic or non-periodic. Virtual Machines allow single computer to take the role of multiple computers by splitting the underlying physical resources into many logical resources and many workstations or servers can be created in a single system. In the existing works all load balancing methodologies are deciding which virtual machine for which cloudlet. In this paper we introduce a new load balancing algorithm using hadoop MapReduce [2].

II.CLOUD MAP REDUCE

MapReduce takes a set of input data and produces a set of output data. This MapReduce consists of two functions Map and Reduce. The primary objective of this function is to split the input data set into independent chunks that are processes in a parallel fashion. Map function takes the input data and produces a set of intermediate data. MapReduce library groups the intermediate data and passes them to the Reduce function. The Reduce function accepts the intermediate data and merges to form a smaller set of values. 0 or 1 output value is produced for one Reduce function invocation. Both the input data and output data are stored in a file system. This method of splitting and reducing the data handles large set of values to be stored in memory. 

Cloud MapReduce primarily has four advantages:

- When compared with other implementations it is highly faster
- By comparing with other systems it is more scalable
- It is more failure resistant because it has no single point of failback
- It has simple lines of code

Cloud MapReduce also has several highly desirable properties:

- The processing power is increased by adding servers. It is highly scalable.
- Only authorized and approved users can use the data in the system. It is secure.
- Data locality and server resources are employed to determine best computing operations.
- Jobs are completed according to the priority. It has optimized scheduling
- Codes can be written in any programming language. It is highly flexible.
- Several jobs make certain that jobs fail independently and restart automatically. It has resiliency and high availability
- There are no master or slave nodes. It is symmetric and decentralized
It is cost effective
At the time of failure of one node the system redirects the task to another location without loss of data. It is fault tolerant.

III. CLUSTER SETUP

Utilization of the node defines that the ratio of the used space of the node to the total capacity of the node. Utilization of the cluster defines that the ratio of the used space at the cluster to the total capacity of the cluster. A cluster is measured balanced for each data node when Utilization of the node differs from the Utilization of the cluster by a threshold value.

This module moves the highly utilized data nodes to the weakly used data nodes in a repeated manner. All data nodes moves or receives within the threshold value not more than that and also each repetition runs only within 20 minutes not more than that.

In this proposed execution, nodes are classified as

- more-utilized
- normal-utilized and
- less-utilized.

Depends on the rate of utilization load will be transferred from highly used nodes to weakly used nodes aso that the cluster will be balanced.

Understanding the nature of the nodes:

Figure 1: Framework of MapReduce

There are three major categories in a Hadoop deployment. They are client machines, master nodes and slave nodes. Master nodes takes care of two functions: HDFS and MapReduce. HDFS stores lots of data, MapReduce runs parallel computation on all that data.

NameNode monitors and coordinates the Data Storage Function i.e., HDFS. JobTracker monitors and coordinates the parallel processing of data using MapReduce. The TaskTracker is slave to JobTracker and DataNode is slave to NameNode. Each slave runs both TaskTracker and DataNode.

The client machine loads the data into the cluster and submits the jobs to MapReduce to process it and then retrieves or view the final result when it completes the job.

Working principles of the nodes in the module:

- Getting the neighbors details by the NameNode:

  NameNode contains load level information for each nearest neighbor DataNode. When there is an increase in the level of load more than the threshold value, it sends a request to the NameNode. NameNode compares all the loads of the DataNode and finds the less loaded neighbor nodes and sends the details to the specific DataNode.

- Next work starts with the DataNode:

  The total load of its nearest neighbors is compared with each DataNode’s load. If the total load of its nearest neighbor is lesser than the DataNode’s load then indiscriminately the destination node will be selected, then the load request is sent to the destination nodes.

- Lastly receiving the request:

  Message Passing Interface(MPI) manages buffer for each node to get the load request. A main thread listens the buffered queue and service the received request. Finally the node enters into the load balancing phase.

IV. CLUSTER EXECUTION

First installed Java, eclipse and then installed cygwin. Set the environmental variable, configured SSH daemon, set up the authorization keys, installed and configured hadoop, created Hadoop distributed file system and started the local hadoop cluster.

Commands to be entered in Cygwin Terminal:
Namenode:
   cd hadoop-0.19.1
   bin/hadoop namenode
Second Namenode:
   cd hadoop-0.19.1
   bin/hadoop secondarynamenode
JobTracker Node:
   cd hadoop-0.19.1
   bin/hadoop jobtracker
Datanode:
   cd hadoop-0.19.1
   bin/hadoop datanode
TaskTracker node:
   cd hadoop-0.19.1
   bin/hadoop tasktrackernode

Figure 2: Shows the cluster running successfully in cygwin terminal

Figure 3: Proposed architecture of VM load balancing algorithm

- When the datacenter allows VM to allocate in to it or when VM wants to allocate into the datacenter, the requested VM id’s with its start time and finish time (generated by CloudSim) is send as an input to MapReduce.
- Then MR will find and produce the number of requests that sent by a VM at a time. Based on the number of requests, VM id’s along with start and finish time is partitioned as blocks.
- When the blocks are given as input to the MR, apply SJF Pre-emptive scheduling algorithm in between Mapper and Reducer. Hence, as a result it produces the response and turnaround time for each VM ids.
- Response Time = FTime - ArTime + TransDelay
  
  Where, ArrTime is the user request arrival time and FinTime is the user request finish time.

  TransDelay = NwLatency + TTtransfer [1]

Where, TransDelay is the transmission delay NwLatency is the network latency and TTtransfer is the time taken to transfer the size of data of a single request (R) from source location to destination.

TTtransfer = R / BWuser [1]

BWuser = ToTBW / NUR [1]

Where, ToTBW is the sum of available bandwidth and NUR is the number of user requests communicating at present.

- With respect to the response and turnaround time, MapReduce will sort the VirtualMachines with less response and turnaround time. The sorted VirtualMachine will be allocated in the datacenter. After completing its job the allocated VirtualMachine will be de-allocated from the datacenter and the next VirtualMachine is allowed to enter into the datacenter to execute its job.

V. PROPOSED VM LOAD BALANCING ALGORITHM:
VI. EXPERIMENTAL SETUP

The proposed algorithm implemented through simulation packages like CloudSim and CloudSim based tool. Java language is used for implementing VM load balancing algorithm. Assuming, the application is deployed in one data center having 50 virtual machines (with 1024Mb of memory in each VM running on, physical processors capable of speeds of 100 MIPS) and Parameter Values area as under:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataCenter OS</td>
<td>Windows7</td>
</tr>
<tr>
<td>Virtual Machine Memory</td>
<td>1024 mb</td>
</tr>
<tr>
<td>DataCenter Architecture</td>
<td>X86</td>
</tr>
<tr>
<td>VM Bandwidth</td>
<td>1000</td>
</tr>
</tbody>
</table>

VM’s id, start time, and burst time are generated using Cloudsim tool.

Figure 4 : Output snapshot

VII. REFERENCES


IX. CONCLUSION

In this paper, a new Virtual Machine load balancing algorithm was proposed using MapReduce concepts and SJF pre-emptive scheduling algorithm is implemented between Mapper and Reducer and then implemented the code using CloudSim, it is a simulation tool to conceptualize cloud computing environment using java language. The proposed algorithm finds the expected response time of each resource (VM) using MapReduce and sends the ID of virtual machine having minimum response time to the data center controller for allocation to the new request, according to the proposed work if we find a efficient virtual machine then the overall performance in load balancing will improve in the cloud environment.