A Model to Enhance the Performance of Distributed File System for Cloud Computing

Pradheep Manisekaran* and Ashwin Dhivakar M R**
*Assistant professor, Department of Computer Science and Engineering, NIMS University, Jaipur, India
Email: pradheep45@hotmail.com
**Research Scholar Jaipur National University Jaipur
Email: ashdhiv@gmail.com

Abstract: Cloud computing is a new era of computer technology. Clouds have no borders and the data can be physically located anywhere in any data center across the network geographically distributed. Large scale distributed systems such as cloud computing applications are getting very general. These applications come with increasing challenges on how to transfer and where to store and compute data. The most current distributed file systems to deal with these challenges are the Hadoop file system (HDFS) and Google file system (GFS). But HDFS has some issues. The most factors are that it depends on one name node to handle the majority operations of every data block in the file system. As a result, it may be a bottleneck resource and one purpose of failure. The second potential problem with HDFS is that it depends on TCP to transfer data. Usually, TCP takes several rounds before it will send at the complete capability of the links in the cloud. This results in low link utilization and longer downloads times. In such file systems, nodes simultaneously serve computing and storage functions; a file is divided into a number of chunks allocated to distinct nodes so MapReduce tasks may be performed in parallel over the nodes. However, in a cloud computing, the crash is the commonplace, and nodes could also be upgraded, replaced, and added to the system. Files can even be dynamically created, deleted, and appended. This results in load imbalance in a distributed file system; that's, the file chunks aren't distributed as uniformly as potential among the nodes. Growing distributed file systems in production systems powerfully depend upon a central node for chunk reallocation. This confidence is clearly inadequate in a large-scale, failure-prone setting as a result of the central load balancer is put out vital workload that's linearly scaled with the system size therefore, it become the performance bottleneck a single purpose of failure. Suppose we tend to save the files in cloud information and a few third party accesses those files and adds some extraneous information which will damage our system. thus to boost the performance and security of cloud computing in this thesis we use a new approach called load balancing with round robin algorithm.

Keywords: Cloud computing, File system, Distributed System, Storage System, Load balancing.

Introduction

Cloud computing is a compelling technology .in cloud users can dynamically store and access their resources without sophisticated deployment and management of resources by means of internet. Cloud computing is emerging as a new paradigm of large-scale distributed computing.it has moved computing and users data away from desktop, portable devices into large data centers.it has the capability to utilize the power of internet and wide area network to access the resources that are available remotely (e.g. software, storage, data, network).cloud computing has two broad categories such as cloud and cloud technologies. The term “cloud” refers to a collection of infrastructure services such as software as a service, infrastructure as a service, and platform as a service. The term “cloud Technologies” refers to various cloud runtimes such as MapReduce framework [1], Hadoop Distributed File System (HDFS), Google File System (GFS), etc.

Cloud computing involves distributed technologies to satisfy a number of users and applications by providing functionalities like resource sharing, software, hardware, information through internet.in order to reduce the capital and operational cost, and to increase the performance in terms of response time and data processing time, maintain the system stability. Day by day the number of users, amount of data, structure of the network is increasing rapidly so that there are lot of technical challenges involves in this process such as virtual machine migration, data transfer, bottleneck performance, unpredictability, server consolidation, fault tolerance, scalable storage, high availability and major issue is the load balancing. Dealing with these challenges of large scale distributed data computer and storage intensive applications such as search engines, cloud storage applications, and social networks require robust scalable efficient algorithms and protocols.

The google File System (GFS) which is used by google and Hadoop Distributed File System (HDFS) is a most common algorithm deployed in Facebook and yahoo today. Distributed file system are key building blocks for cloud computing application. Based on the MapReduce framework in such file systems nodes simultaneously serve computing and storage functions; a file is partitioned into a number of chunks allocated to distinct nodes so that MapReduce task can be performed

in parallel over the nodes. And these file chunks are assigned to different cloud storage node known as chunk server.in such a distributed file system the load of a node is typically proportional to the number of file chunks the node possesses [4].because the files in a cloud computing can be haphazardly created, deleted, and appended in the file system[6], And nodes can be upgraded, replace and added in the file system .the file chunks cannot be distributed uniformly as possible among the nodes.in this case load, balance among storage nodes is a critical function in clouds.

However, GFS and HDFS has some potential problem. The first one is HDFS depends on a single name node to manage almost all operations of every data block in the file system. As a result, in can be a bottleneck resource and a single point of failures and once it fails to perform the action it takes a long time to recover. And the second one is it totally depends on TCP to transfer data. Usually, TCP takes several rounds to transfer data in cloud this results in low link utilization and longer download time

In this paper, we studies and address these problems with current system such as GFS and HDFS .in order to increase the system scalability we using a light weight end server to connect and share all requests with many name nodes. This makes a single name node to many name nodes. And it is stateless. If it goes down no data will be lost and we can bring it up instantly and another main feature of our system is that it uses an efficient load balancing algorithm to balance and split the load between the name node servers. Our proposed model can achieve full link utilization and also decreases download time. As a result, of this, there won't be any bottleneck failure and we can achieve lower chunk transfer times

The contribution of this paper involves:

- We propose a vertically distributed framework that defines bindings between client system and the name node servers
- We propose an approach to schedule jobs with CPU and resource requirements in shared heterogeneous cloud computing

The proposed policy is demand driven and it improves overall resource utilization. The proposed scheduling policy is studied under various system and workload parameters

Background

Cloud technologies

The cloud technologies such as MapReduce and Dryad, Google File System, Hadoop Distributed File System, Microsoft Dryad and CGL-MapReduce have created new trends [26]. Distributed file systems such as GFS and HDFS are used to access data through distributed storage system built on heterogeneous compute nodes and the Dryad and CGL-MapReduce used to read data from local disks

Cloud computing services

Cloud computing offers three major services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [8].

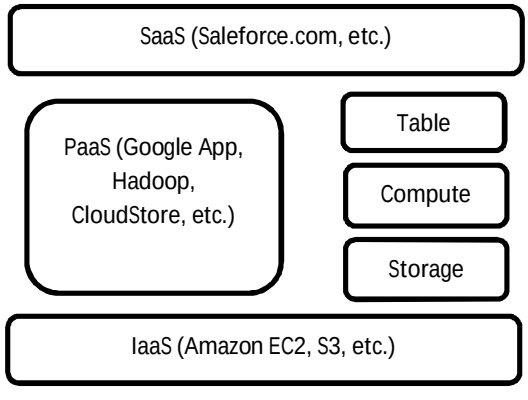

Fig 1: Cloud computing services

IaaS provides computing resources such as storage, servers, other low-level networks, and hardware resource virtually over the internet based on the demand, for example, GoGrid, Amazon's EC2 serving infrastructure to the IT industries [26]

PaaS provides application framework and a set of application programming interface that can be used for developing purposed as well as for developing applications for the cloud. The Google and Microsoft are the major companies which providing PaaS
SaaS provides fully executable software delivered over the internet. Through the internet, the users can operate and make utilize the functionality of the product

Existing Load Balancing Algorithms

Equally Spread Current Execution Algorithm(ESCE)
Here the load will be given equally to all the virtual machine connected to the data center. If any request coming from client/node .the load balancer scans the index table which contains information about the virtual machines as well as a number of jobs currently assigned to the virtual machine. If the request comes from the data center to allocate the VM [13]. It scans the index for least loader VM. The first identified VM is selected for handling the request from the client.After completing the assigned task, the load balancer will update the index table by decreasing the allocation count for identified VM. Here scanning process of index table again and again is a major issue

Round Robin Algorithm
This is very simple and lightweight algorithm that works on the concept of time quantum here time is divided into multiple slices and each node is given a particular time quantum.in this time quantum, the node has to perform all the operations allocated to it.in Round Robin scheduling algorithm time quantum play a very major role for scheduling the resources to the client. The time quantum should not be extremely smaller or extremely bigger than the RR scheduling algorithm.it is very light weight and simple algorithm but there is an additional load on the scheduler to decide the size of quantum

Throttled Load Balancing Algorithm(TLB)
In this algorithm, the load balancer maintains an index table same like ESC algorithm which contains a list of virtual machines as well as their states. The client will first pass a request to the data center to find a suitable virtual machine and then data center will query the load balancer for allocation of VM. The load balancer scans the index table from the top until finding the suitable VM and job will be allocated for that VM. If the VM is not found the load balancer will return -1 to the data center. After completion of the job the details will be updated in index table to free the VM and make it ready for next job allocation.Here also the scanning process for VM will makes lot of delays.

Proposed Work

Architecture

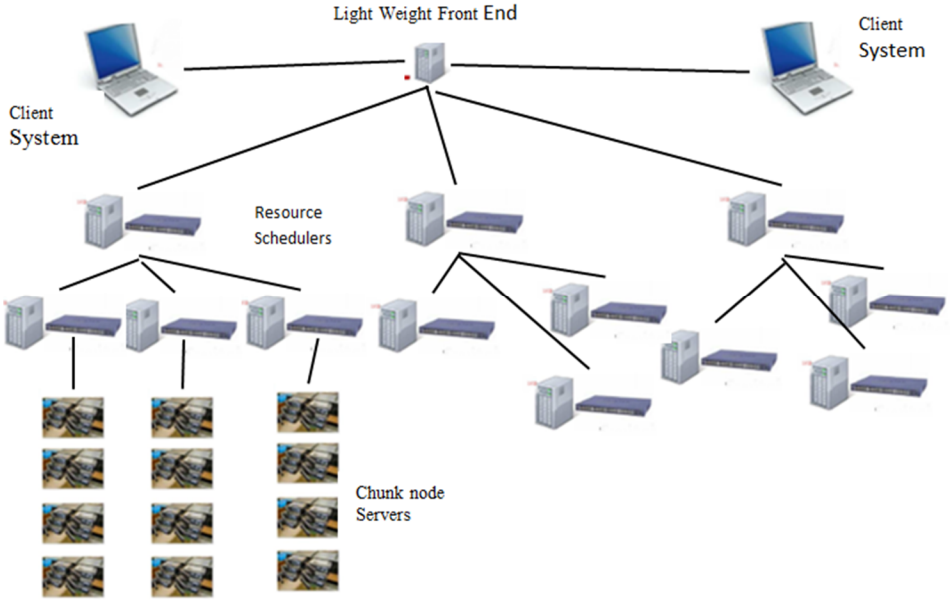

Fig 2: vertically distributed framework

Load Balancing Algorithm

In our proposed work we using two sharing approaches such as Time sharing and space sharing. Time sharing approach helps to balance the load in a number of jobs on process node servers and also helps to allocate that job to nodes can execute according to its time quantum which results in getting less weight time for the jobs. We use the parameter multiprogramming level (MPL) to control the number of tasks among which the processor is time shared at given time. Since each and every processors have different processing speed, the MPL [14] is determined as follows

$$MPL = \frac{CPU\ speed * Basic\ MPL}{slowest\ processor\ speed}$$

The space sharing technique also allows splitting the job on different processing nodes if one node is not able to full the requirements of the job then the job will be split into the different processing nodes which make the job to be executed in less time. The proposed system also uses the demand driven approach it makes the system more efficient. In the demand-driven approach if the node is in the idle state then it will demand their parent for the jobs and if the parent does not have the job then it will demand the job from its parent. it will make the system wait less for the jobs. In the workload model, all tasks of jobs have equal service demand. Job cumulative service demand is dividing into maximum jobs and each job will have a demand for minimum time. This workload shows the advantage of space sharing policy. The adaptive scheduling used for Heterogeneous Multi-cluster System can be framed using following steps:

I. **Job selection**: Job selection policy is used to select the jobs in the queue. The global scheduler consists the jobs in the queue. The aim of scheduling policy is to carry the job from the queue in some manner. So we use First Come First Serve policy. It is one of the simple policies and it has less overhead as compared to other policies. It implements just one queue which holds the tasks in the order they come in. The job is served in arrival order.it is done by our lightweight front end server. In case if it crashed also we won't face any data loss and it recovered instantly

II. **Selecting site**: The Site/Cluster is selected on the basis of where can our job perfectly runs. The Most-fit policy is used to select the cluster. The perfect policy is used to minimize the data that is divided into fragments by choosing the appropriate cluster which wasteless number of processing nodes and by taking care of the other jobs in the queue.
    - In our scenario, we define the tree structure in which the main front end server divided into three resource schedulers. After this three resource schedulers it further divided into three nodes each, our system contains total 9 name nodes. Each node connected to 8 chunk servers.
    - All the nodes find their BPU request according to the total sum of their BPUs range. In this system, all the processing nodes perform the task which is divide equally to them.
    - Multiprogramming Level (MPL) of all the node is also fixed. We have formula to calculate MPL of each processor. MPL= (Processor speed ×Basic MPL)/ (slowest processor speed). Or we can make it fixed (preferred) to simplify it. We can fix it to 2 i.e. each processing nodes can do time-sharing between two jobs. For time-sharing we use Round Robin algorithm.
    - Round Robin algorithm is used to find the waiting time and remaining time to finish the task of all nodes to allocate the jobs equally to them.

And after all scheduling and allocation of chunks the details will be stored by resource scheduler in their index table by means of the parameters such as BPU, starting time, ending time, node id .at each and every iterations this file table will be updated.

Numerical Result

Every processing node in a cluster consists of different number of BPUs (Intra cluster heterogeneity). Number of BPUs for each processor it is fixed to find the performance of our model. For example N33 = 1 BPU, N34 = 4 BPUs, N35 = 2 BPUs, N36 = 12 BPUs, N37 = 2 BPUs, N38 = 3 BPUs, N39 = 10 BPUs, N40 = 7 BPUs. In starting System will be in neutral state. There won't be any jobs in intermediate layers or any other processors. Jobs travel down in hierarchy order based on the demand from the client systems Multiprogramming Level (MPL) of all the processor is also fixed. We have formula to calculate MPL of each processor. Or we can make it fixed (preferred) to simplify it. We fixed it as 2 i.e. each processing nodes can do time-sharing between two jobs. And we got the result as follows:

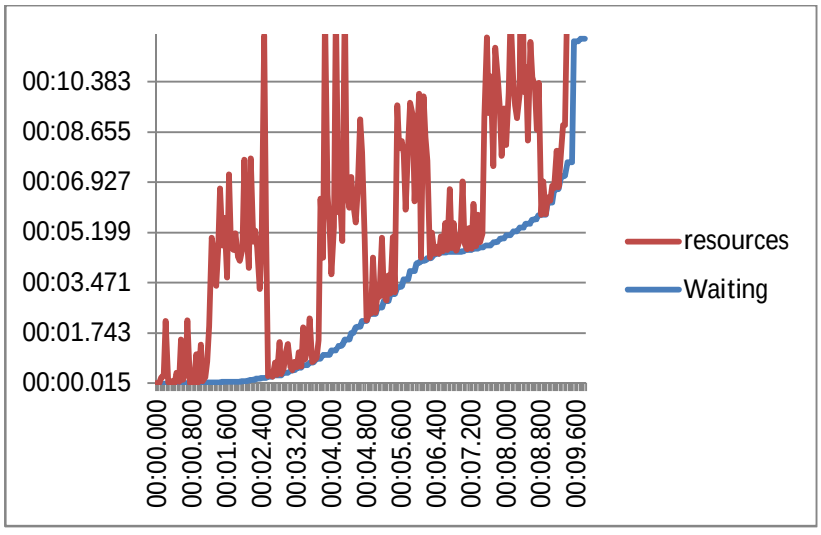

Fig 3: performance of the system graphically

Representing the values in the form of table as follows:

Table 1: Comparison of waiting times between our and existing system

resources	Average wait time new system	Average Waiting time old system
resources10% ar=6	00:02.840	00:03.117
resources 20% ar=12	00:03.112	00:03.384
resources30% ar=18	00:04.123	00:04.856
resources 40% ar=24	00:05.224	00:05.999
resources50% ar=30	00:06.111	00:06.455
resources 60% ar=36	00:07.222	00:07.888
resources70% ar=42	00:08.567	00:09.111
resources 80% ar=48	00:09.445	00:10.111
resources 90% ar=54	00:10.345	00:11.234

Conclusion

In this dissertation work, a new scalable and efficient scheduling algorithm in distributed file system is planned and then enforced in virtual cloud computing environment using Microsoft visual studio, in c# language. Our proposal is to balance the loads of nodes, to increase the processing speed in file system and also to reduce the cost as much as possible. This thesis presents design of a scalable and efficient distributed file system. The system uses a light weight front and back end server to

manage sessions and compute the storage and processing of data. This design solves the potential bottleneck scenario that the name node server of current systems by can be allocation the work load into further host. Our research work conjointly offers an adaptive and efficient resource allocation scheme which may lead to full link utilization and hence much reduced chunk transfer time. By visualizing the parameters in graphs and tables we can able to simply identify that the response time and data centre processing time is improved yet as well as cost is reduced in comparison to the existing scheduling parameters. Based on the numerical results presented, our algorithm will overcome standard existing distributed file systems .our model can be directly implemented in current distributed file systems.

Reference

[1] Dean, J.and S.Ghemawat. 2008." MapReduce:simplified data processing on large clusters.Commun.ACM 51(1):107-113".
[2] ASF.2009.Apache Hadoop Core. http://hadoop.apache.org/core
[3] Ali M. Alakeel, (2010),"A Guide to Dynamic Load Balancing in Distributed Computer Systems",International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.
[4] Hadoop Distributed File System, http://hadoop.apache.org/hdfs/, 2012
[5] Debessay Fesehaye, Rahul Malik, Klara Nahrstedt, A Scalable Distributed File System for Cloud Computing.(n.d)[online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.176.5236&rep=rep1&type=pdf
[6] Hadoop Distributed File System "Rebalancing Blocks", http://developer.yahoo.com/hadoopp/tutorial/module2.html#rebalancing, 2012
[7] Cloud computing principles, systems and applications NICK Antonopoulos (n.d.)[online].Available: http://mgitech.wordpress.com.
[8] C.Vecchiola, S.Pandey, and R.Buyya,"High-Performance Cloud Computing: A view of scientific applications,"CoRR,Vol.abs/0910.1979,2009
[9] Mladen A. Vouk, Cloud Computing Issues, Research and Implementations, Proceedings of the ITI 2008 30th International Conference on Information Technology Interfaces, 2008, June 23-26.
[10] lizhe wang, rajiv Ranjan. (n.d.). Cloud Computing Methodology, Systems and Applications. Available:http://www.unitiv.com.
[11] Luyang Dong, Bin Gong, (2012),"A Hierarchical Scheduling Policy for Large-Scale Rendering",IEEE International Conference on Systems, Man, and Cybernetics, 2012.
[12] Tejinder Sharma, Vijay Kumar Banga, (2013), "Efficient and Enhanced Algorithm in Cloud Computing", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-1.
[13] Jasmin James, Dr. Bhupendra Verma,(2012)," Efficient VM Load Balancing Algorithm For A Cloud Computing Environment", Jasmin James et al. International Journal on Computer Science and Engineering (IJCSE),Vol-4,Issue No.9.
[14] J. H. Abawajy and S. P. Dandamudi, "Parallel job scheduling on multicluster computing systems," in Proceedings of the IEEE International Conference on Cluster Computing,Hong Kong, China, 2003, pp. 11-18.
[15] Jaliya Ekanayake and Geoffrey Fox, High Performance Parallel Computing with Clouds and Cloud Technologies, Presented at Cloud Computing - First International Conference, CloudComp, Munich, Germany, 2009
[16] Jonthan Strickland. How Cloud Computing Works (n.d.) [online]. Available:http://www.howstuffworks.com/cloud-computing/cloud-computing1.html
[17] Abhisek Pan, John Paul Walters, Vijay S. Pai, Dong-In D. Kang, Stephen P. Crago, "Integrating High Performance File Systems in a Cloud Computing Environment", The International Workshop on Data-Intensive Scalable Computing Systems (DISCS), in conjunction with the 2012 ACM/IEEE Supercomputing Conference (SC'12), November 2012.
[18] Hung-Chang Hsiao, Member, Hsueh-Yi Chung, Haiying Shen,, and Yu-Chang Chao,(2013)," Load Rebalancing for Distributed File Systems in Clouds", IEEE Transactions On Parallel And Distributed Systems, Vol. 24, No. 5.
[19] Cong Wang, Qian Wang, and Kui Ren, Wenjing Lou, Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing, Presented at IEEE INFOCOM 2010.
[20] Satoshi Togawa, Kazuhide Kanenishi, Private Cloud Cooperation Framework of e-Learning Environment for Disaster Recovery , IEEE International Conference on Systems, Man, and Cybernetics, 2013.
[21] Sonal Guleria1, Dr. Sonia Vatta2, (2013) "To Enhance Multimedia Security In Cloud Computing Environment Using Crossbreed Algorithm",International Journal of Application or Innovation in Engineering and Management, Volume 2, Issue 6.
[22] CemOzdogan,(2011, Feb.14). Round robin scheduling. [Online]. Available: http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node125.html.
[23] Amandeep Kaur Sidhu, Supriya Kinger, (2013)," Analysis of Load Balancing Techniques in Cloud Computing", International Journal of Computers & Technology,Volume 4 No. 2.
[24] 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.
[25] Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. (2009, Feb. 10)," Above the clouds: A Berkeley view of cloud computing", EECS Dept., Univ. California, Berkeley, No. UCB/EECS-2009-28.
[26] Anthony T.Velte, Toby J.Velte, Robert Elsenpeter,"Cloud Computing A Practical Approach", TATA McGRAW-HILL Edition 2010.