DISTRIBUTED FILE SYSTEMS USING INITIATIVE DATA PREFETCHING FOR CLOUD COMPUTING

Apurva Vasudeo K[1], Amulya K[2], Kruthika P S[3], Srinivasa H P[4]

[1,2,3]*UG Students, Department of CSE, T. John Institute of Technology, Bengaluru, India.*
[4]*Associate Professor, Department of CSE, T. John Institute of Technology, Bengaluru, India.*

Abstract

In this project for cloud computing, prefetching of data concept is proposed on metadata servers in a multi-client environment. The client file system does not interfere in the process of prefetching technique. The process is carried out by analysing the history of the requests and then applying algorithms on them and fetches the relevant data which would be accessed by any of the clients in the future. The information is stored on metadata server, through which client accesses the data. This is to achieve better performance from storage servers rather than fetching the data from the cloud. In a distributed cloud environment all the clients can access the information simultaneously, hence it enables parallel execution of programs as data is split into chunks of information and stored in a hierarchical tree form of nodes.

Keywords: *Data Prefetching, Fetching the Data, Cloud Computing.*

----------***----------

1. INTRODUCTION

Distributed File System (DFS) are a system that permits several organizations to organize many files. It allows the clients to access information either to read, write, modify, delete etc. There are many ways to share files in DFS but each method is suitable only to a specific architecture depending on the complexity or simplicity of the application. In early 1980's Sun's Network File System was developed, but before that people used sneakernet method. This system had a lot of disadvantages and was unsuitable for such an environment of multi-user. Many clients started using FTP to share files but that also wasn't an efficient method as the files had to be copied from source computer to a sever an in turn from server to a destination computer. Therefore the users had to know the physical address of all computers to which file sharing must be done. Later on, as technology improved there was large number of Modern Data centres with many computers having different storage capacity. The Map reduce framework also showed great performance with computation of applications in parallel. Virtualization concepts allow multiple operating systems to co-exist on same physical server and provide dynamic resource allocation.

Since many clients can access the same files confidentiality and integrity should be maintained for security issues. Measures must be taken not to breach the important information as users can share resources from anywhere through any computer or device. The main characteristic of cloud computing is scalability and elasticity of resources. The idea of cloud computing is use storage; execute complex or simple computations without any worries of how these specially made development areas work internally.

2. EXISTING SYSTEM

In the system which is exists, the applications of the system have many problems like how to send the data to other system and where exactly to store the data. The most common distributed file systems to deal with these kinds of problems are the Hadoop file system which is a form of Google file system. But the HDFS has two possible problems. The first one is, management of almost all the operations of all the data blocks in the file system is dependent on a single node and this leads to critical resources and in turn becomes the point of failure of the system. The second problem is that, it is dependent on the transmission control protocol to send data from one system to another. TCP may not send the complete data at once even though the links have the capacity to carry the complete data. It sends packets of data at a time which leads to under-utilization of the link and the receiver takes more time to download the complete data.

3. PROPOSED SYSTEM

The proposed system uses a front end server which is made up of light weight component to connect all the requests from the client with many name nodes. This server helps to distribute the load of one node to many other name nodes. The aim of the proposed system is to use an effective protocol to send data. The protocol used in the proposed system can perform better than HDFS and GFS and it can achieve utilization of full link and faster download times. To achieve this we face a lot of complex problems like how to send the data, store the data in real-time, performance unpredictability, resizable storage, fast scaling for the workloads which vary, bottlenecks during data transfer etc.

To overcome these problems the applications which are storage intensive such as social networks and search engines

needs an effective, strong and scalable algorithms and protocols. The large distributed systems such as Face book, Yahoo and Google uses the GFS or HDFS algorithm. Only one name node is used in these file systems, keeps a list of all the files and their metadata in the cloud. This node is called as the i-node. This single i-node has to manage all the file related operations such as open, copy, move, update, delete etc. This may lead to a bottleneck of resources due to its un-scalable property. The use of this node may also lead to the point of failure of the HDFS. The file system goes offline when the name node goes down and when the name node comes back it has to perform all the operations again. This process of replaying can take half an hour for a big cluster. The data can also be accessed by unauthorized users through network interfaces which affects the security of the data.

In the proposed system we use a better distributed architecture with frontend serve which use light weight components and is much more scalable than the HDFS/GFS. We also use a protocol which efficiently transfers the data and route the data. It also leads to better link utilization than TCP used in the existing system and also achieves faster data transfer time.

In the proposed system for cloud computing, prefetching of data concept is proposed on metadata servers in a multi-client environment. In this technique, the client file systems do not interfere in the process of prefetching. But after knowing the history of I/O accesses of the disk the server used for storage can prefetch the data directly and send to the appropriate client system. The proposed system uses Random series and Sequential prediction algorithm to estimate the I/O access which occurs in future.

In this technique the client file system has to only piggyback their information of the particular I/O requests to the server used for storage. The servers used for storage have to login the I/O requests of the disk and has to classify the access patterns of the I/O requests. After this we can use the two algorithms to predict I/O access which may occur in the future and carry out the process of prefetching. Then the servers used for storage can send the data that was prefetched to the appropriate client machine and satisfy their future requests.

The access patterns are divided into two types sequential and random access. In order to estimate the future I/O requests of the disk that belongs to different access patterns as correctly as possible these algorithms are used. The proposed system yields better I/O performance.

Even though the prefetching scheme puts an extra load on the servers used for storage as they have to predict the I/O accesses required in the future by knowing the history of the I/Os of the disk. But it is a good to build a system for storage to improve the I/O performance of the system. This helps the client system which has limited hardware and software resources. This scheme can be used in multi-client file system for mobile cloud computing where many smart terminals and tablets are used. This scheme can be employed as a backend system for storage which may have few client systems that have limited resources. The file system on the client will not perform any work like logging the I/O events or predicting the I/O requests required in future. Spreading workload to different and most appropriate systems is more flexible. Extensible to add resources and software as needed.

4. SYSTEM ARCHITCTURE

This paper proposes a prefetching method in cloud computing to increase the performance of I/O. Few assumptions are made in order to propose few mechanisms of prefetching.

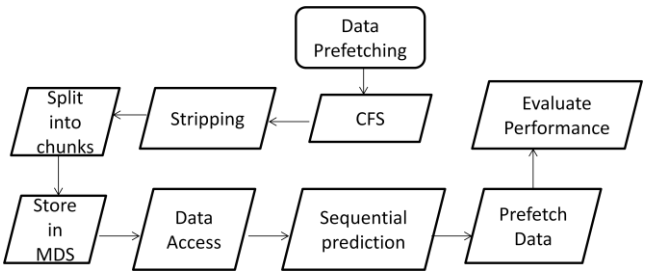

Fig 4.1 System Architecture

4.1 Assumptions

For the proposed system to work, few assumptions must be met, the proposed prefetching methods can be used only if the cloud has client systems which have limited resources and the mechanisms are good for read intensive apps because these apps will have less access patterns and the same patterns keep reoccurring.

4.2 Piggybacking

Many proposals made these days concentrated on retrieval of I/O taking place in CFS,which we can use for studying the access patterns.Without pertinent details about the retrieval of I/O it is strenuous to set up the connection between apps and DFS.Here,data will be prefetched already from memory to the server and after studying the access patterns this data will be sent to the CFS for fulfilling the app's requests. The information is piggybacked by the CFS and the server will maintain the information and will record all the events.This information is nothing but the metadata.

4.3 CFS

The main aim of this CFS is that it provides interface to the apps. It concentrates on requesting the metadata to the server which holds the metadata.It also collects data of client and piggybacks this data to I/O request.

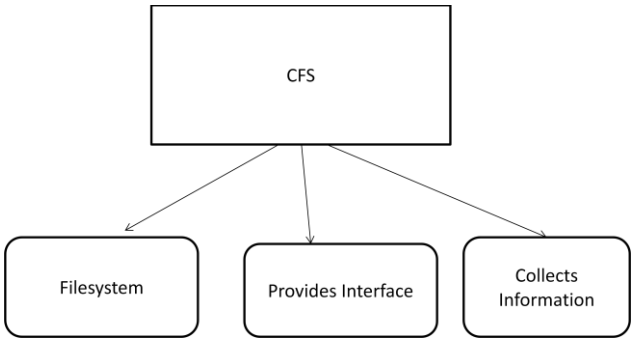

Fig 4.2 Module Description of CFS

4.4 MDS

The object's metadata and the servers used for storage are handled by and monitored by this metadata server and also the commands to create or remove the stripes on the servers are issued by this metadata server.

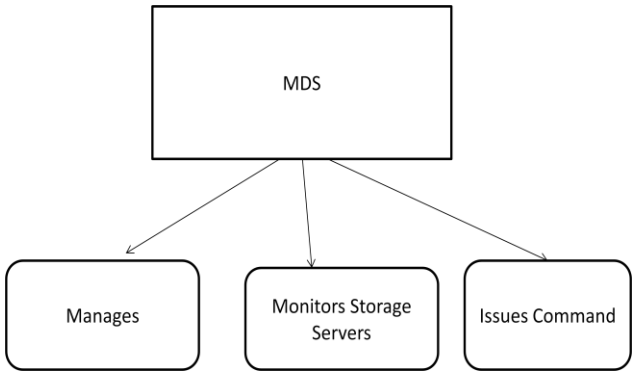

Fig 4.3 Module Description of MDS

4.5 Data Access

Data can be accessed either in sequential manner or random manner. An algorithm is used to determine how data is accessed. This algorithm has a set of addresses access which are referred within a set of time.

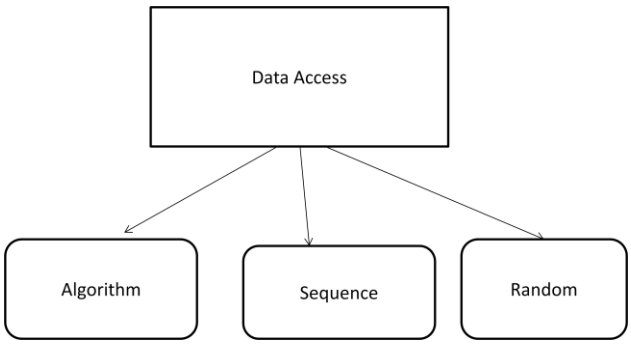

Fig 4.4 Module Description of Data Access

4.6 Sequential Prediction (SP)

This algorithm is used to make sure that the data access is sequential or not. This algorithm monitors the history of the block access and it checks for the recent events which occurred after a given setoff time. Also the access events that occurred previously can be limited by this algorithm. We define a certain threshold value and if all the access events is greater than the threshold value,then the system will verify and decide that the access request is a part of the sequential pattern.

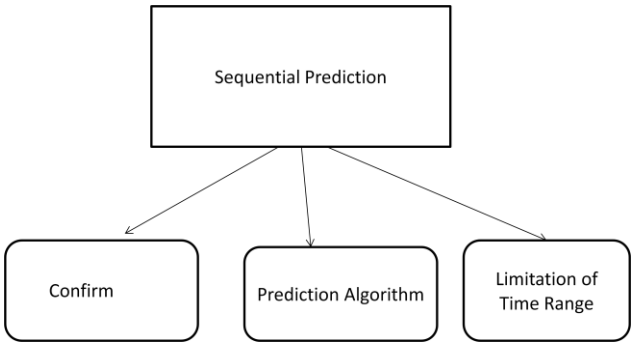

Fig 4.5 Module Description of SP

4.7 Prefetching

The read requests are maintained by the server used for storage. The server used for storage usually issues read request by studying the history of read operations in advance.This data which is prefetched is sent to the CFS.

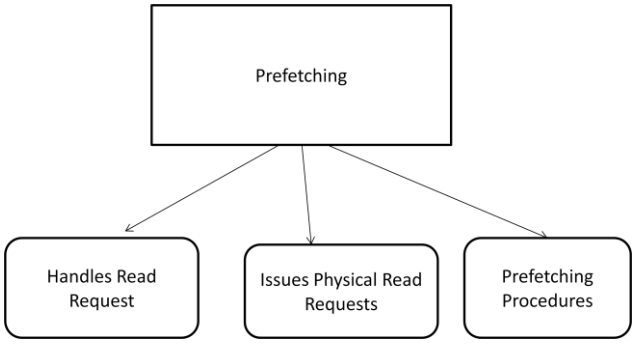

Fig 4.6 Module Description of Prefetching

4.8 Analysis of Algorithm

Error on varaying workloads is analysed in this paper.The total number of read operations and the total number of prediction hits are recorded and saved in storage server.All read operations cannot be predicted by our system.

5. CONCLUSION

In this paper, we have suggested a mechanism for prefetching of data in DFS which is evaluated and implemented in cloud envirnoment. The I/O access is analysed by the server used for storage so that the data is fetched in advance and the future application request can be satisfied by the CFS. And this can be implemented by making use of access patterns and sending the data which is prefetched to CFS.The information about CFS will be piggybacked and then sent to server nodes from client nodes which resuts in reduction of network latency and work traffic increasing the efficiency.

REFERENCES

[1] J.Gantz and D.Reinsel. The Digital Universe in 2020:Big Data, Bigger Digital Shadows, Biggest Growth in the Far East-UnitedStates.
[2] J. Kunkel and T. Ludwig, Performance Evaluation of the PVFS2Architecture, In Proceedings of 15th EUROMICRO InternationalConference on Parallel, Distributed and Network-Based Processing,
PDP ’07, 2007
[3] S. Ghemawat, H. Gobioff, S. Leung, The Google file system, InProceedings of the nineteenth ACM symposium on Operatingsystems principles (SOSP ’03), 2003.
[4] IO Signature Plus (IOSIG+) Software Suite[Accessed on Nov 2012].
[5] UMass Trace Repository: OLTP Application I/O[Accessed on Jan 2014].
[6] MobiDFS: Mobile Distributed File System[Accessed on Nov 2014]
[7] E. E. Marinelli. Hyrax: Cloud computing on mobile devices usingmapreduce. CMU, Tech. Rep., 2009.
[8] P.Sehgal, V.Tarasov, E.Zadok. Evaluating Performance and Energyin File System Server Workloads. The 8th USENIX Conferenceon File and Storage Technologies (FAST ’10) pp.253-266, 2010.
[9] V. Tarasov, S. Bhanage, E. Zadok, et al. Benchmarking file systembenchmarking: It is rocket science. In Proceedings of HotOS XIII,2011.
[10] N.Nieuwejaar and D. Kotz. The galley parallel file system and Parallel Computing.