A Review on Distributed File System for Cloud Computing

[1]Kumar pal, [2]Anil chauhan, [3]Surbhi Madan

College of Computing Science &Information Technology

Teerthanker Mahaveer University

(Moradabad), Uttar Pradesh

kumar.paltmu@gmail.com

talk2anilchauhan@gmail.com

sumeriyanaitik@gmail.com

Abstract-**Extensive distribution systems, such as cloud computing software, are becoming commonplace. These applications are brought up with more challenges for how to transfer and where to store and calculate data. The most distributed file system to address these challenges is the Hadoop file system, a variant of the Google Docs system. However, the Hadoop file system has twoPossible problems. First of all, this depends on the name node to handle almost all operations on each block in the file system. As a result, it can be a resource with a level of ability and a point of failure. The second potential issue with the Hadoop distribution system is that it relies on TCP for data transfer. As cited in many studies, TCP needs many circles before it can transmit full completeness. This leads to lower usage rates and longer download times. To overcome these issues on the Hadoop distribution system, we present a new distribution.File system. Our scheme uses a light front server to connect all queries with nodes with names. It helps to divide the nodes with names in nodes with names. Our second entry is to use data transfer and efficiency protocols. Our protocols can be used in a comprehensive way and thus reduce the download time.**

I. Introduction

The rapid development of communication technologies, and especially the Internet, has changed the way we live and work. It has a lot of potential to change.The IT industry makes the program more attractive as a service and creates the way hardware IT is designed and purchased. There are many companies that have been tampered with, such as Amazon EC2, Google AppEngine, Microsoft Azure, Salesforce, and others. Prospects show the future of cloud computing. Although the estimate is highly variable, research firm IDC predicts that in 2012, the cloud system will reach $ 42 billion.This huge investment shows more interest in this new technology. However, this progress is taking place with the emerging and complex challenges for how to transfer and calculate reliable and real-time data. Some of the challenges include data that can not be predicted, scalable loads, scalability, workload scales, etc. Addressing these issues of major data distribution, software and storage, such as social networks and search engines, requires an efficient and stable algorithm. The Hadoop distribution system, the Hadoop distribution system, is the most common distribution system in large distribution systems such as Facebook, Google, and Yahoo. These file systems use a name node to keep track of all clouds and corresponding Kono meta data. In addition to the node name, it should work in almost all file-related operations, such as copying, copying changes, deleting updates. Etc. This can not be scaled, and potentially cause the node name to be a constraint. Another limit is that the node name is the only point of failure to install the Hadoop file system [2]. If the node names down, the file system will be offline. When a backup arises the name node must do the same operation. The same process can take more than half an hour for a cluster. This article discusses these issues with current systems, such as the Google filesystem and Hadoop distribution filesystem. To make the system scalable, our scheme uses the latest low-end server to attach naming multiple queries. It helps to divide the nodes with names in nodes with names. Our front ends are only on the course, so it is not an impediment to resources. Also, our UI is no country, so if it is downloaded, no data is lost and confirmed

quickly. Another feature of our system is that it uses an efficient one.

II. Related work :

Another popular network file system for network computers is a network file system. This is a way to share files between hosts on a network if the files are on the client's hard disk. One of the disadvantages of NFS is that it is trying to create a remote file system that appears as a local file system, but it is risky in terms of simplicity. There are many situations where using NFS (compared to local file system) is either inappropriate or reliable. Andrew File System (AFS) is a distributed network file system that uses a trusted set of servers to display file names that are the same and transparent for all clients. AFS has many advantages over traditional network file systems, especially in security and scalability. It is not uncommon for AFS to exceed 250,000. AFS uses Kerberos for authentication and runs a list to control the users and group access. Each client hides the files in the local file system for higher speeds with subsequent queries for the same file. AFS may not be as easy for large file systems as when run by GFS. Other examples of file system distribution are GPFS Frangipani and InterMezzo. Frangipani is a manageable distributed file system that manages multiple disk collections on a single package. The machine is under general control and can communicate securely. There is a simple internal structure that allows them to easily handle recovery, restoration, restoration and storage systems. GPFS is a consistent file system for computing. GPFS uses central control.That may have a scalable problem. In InterMezzo, the key design solution is to use local file systems such as server storage and client cache and wrapping kernel filesystem files around the local file systems. However, they rely on existing protocols such as TCP. In addition, these systems do not have good resource allocation in terms of capacity, connectivity, storage and dynamic processing.

.

III. our protocol:

The main component of our protocol is the client client, the front-end server, the lightweight server, the node name, the namespace, the resource blocking resources, and the resource monitor. As seen by the user on our front-end server (FES) by calling the customer. The client client connects the user to the file system connection. Synchronize server manages sessions with hosts, and then transfers client requests to node-names servers. The node engine named the metadata of the user filesystem and the reference to the BS in the file's storage session. Resource Distributor specifies the name of the node server where the BS and the host blocking path are used to store BS data, based on the value of the resource monitor that it receives from each resource monitor. The RM associated with each BS monitors the resources on its blog server and regularly sends the resource distributor.

3.1 The Algorithm

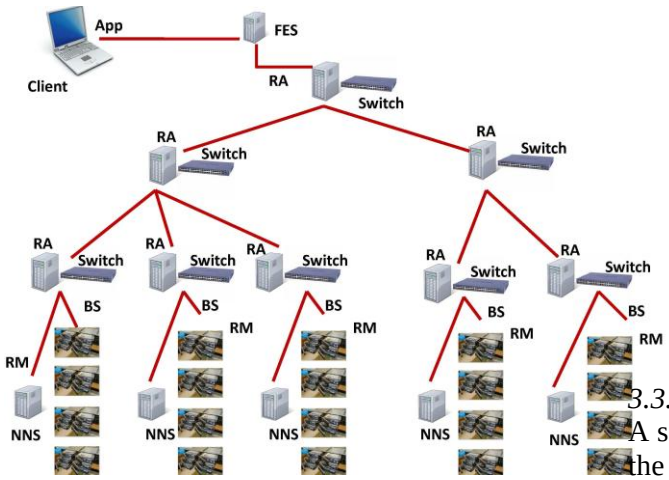
1. User program starts session with FES using UCL

2. . Fes Verify the NNS user's query is appropriate, for example, by acknowledging the identification number of the request and transmitting the name or identification number of the NNS (with the NNS password) back to the application you use.
3. The NNS session wants rheumatoid arthritis associated with local changes, getting the appropriate BS and time to BS, in which the user's program (or other nodes in the cloud) can save a block of data or you can download a saved block of data. RA uses the speed indicators that it receives from each RM from itself and from other RAs to allocate resources. RA is like a software router. RA and network switches can act as a router. More information on how to find the appropriate BS is discussed in one section.

4. The NNS sends name or ID of the BS to the user application and request ID and password to the BS.
5. User program requires BS to use the information it receives from NNS to store or retrieve block data.

6. BS verifies the user's request using information obtained from NNS and transfers data to the user or storing user data.
7. The resources associated with BS regularly send the percentages that serve as a summary resource monitor.

Upon receiving the request from FES, RA found
Highly capable BS. Here, capacity refers to the ability to connect to and from BS, which has a high capacity storage and processing capacity in the cloud. In order to find more storage capacity and capacity BS, our protocol uses indicators to track RM resources with BS.



(Overview of Our Protocol)

3.2 The RM Algorithm

The cloud-based network infrastructure is the same as shown in Figure 1. The client program first connects to the front server. Server Sync Server Select the appropriate node server named for the client. A node server named Submissions allocates for the best server block. The best distributor of BS detection is based on the monitor monitors that it receives from a resource monitor on a server, block, and other resource allocation. To search for cloud-based BS (for example, to update files), we associate the program resource monitor with each cloud block. Resource Monitor Then, apply the following rules to help retailers find the best nodes on the server block to save or calculate the data.

3.3 Simplified Examples to Clarify our Protocol

3.3.1 Cloud customers

If the client wants to keep large data in the cloud, it's here

Steps to follow

1. Subscribers are associated with the FES as mentioned in the picture.
2. The FES grants client a certificate of customer support to NNS.
3. The node server name connects to the RA to the local switch to find the BS that is best for data storage.
4. RA has a table with BS in the area with much more storage space, less busy than connecting it to the first level change that first customers join in the cloud. The reseller offers BS to the client and sends it to the NNS address, IP address or BS card he has selected, and the speed at which the client can transfer the data to the server.
5. NNS stores the location of the client data block and other metadata of the client file and sends the client BS IP address client can store its data and the speed at which the client can send the data to BS.
6. The client then sends the BS data to the NNS-level limit if the network is out of the cloud

3.3.2 A Cloud BS Replicating its Data

A second example is when a server with a node name in the cloud decides to copy data from a block server on your local changes. This distribution may be based on a policy that has a file system below. If a distributed file system wants to keep an article in the drain up to the highest level then the distribution of server resources changes in the same area the node has the best server name for the remote device using the above procedure. This is a global project for distribution. Distributions of resources that can be used by local K-based local distribution implemented by distributed file system rules. In these cases, the nodes they transfer, the copied data is set to the entity that meets the policies of the file system and server, the path from the server, to provide the highest speed lock.

CONCLUSION

This paper shows the distribution of scalable, efficient distribution systems. Light foreman server system to manage sessions and send requests to multiple nodes with names. This design solves a potential scenario that can be a node server of the current system.

This file also provides a flexible and efficient resource allocation scheme that can lead to full use of the links, and so there is little time to divide. Based on the digital results shown, our protocols may be highly effective
Existing existing distribution filesystems such as GFS and HDFS. Our protocols can be embedded directly in new distribution systems like cloud computing. We are currently conducting more detailed experimental tests. We will also implement our system and try it using the Cloud Firm test foil in Illinois.

REFERENCE

[1] *IDC Cloud Computing*. http://blogs.idc.com/ie/?p=224.

[2] *Improve Namenode Performance*. http: //issues.apache.org/jira/browse/HADOOP-3248.

[3] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. Above the clouds: A berkeley view of cloud computing. *Technical Report: Electrical Engineering and Computer Sciences University of California at Berkeley UCB/EECS-2009-28* (Feb 2009), 1–23.

[4] Borthakur, D. *The Hadoop Distributed File System: Architecture and Design*. The Apache Software Foundation, 2007.

[5] Braam, P., Callahan, M., and Schwan, P. The intermezzo file system. In *Proceedings of the 3rd of the Perl Conference, O¨i£¡Reilly Open Source Convention*, Citeseer

.[6] Campbell, R., Gupta, I., Heath, M., Ko, S., Kozuch, M., Kunze, M., Kwan, T., Lai, K., Lee, H., Lyons, M., et al. Open CirrusTM Cloud Computing Testbed: Federated Data Centers for Open Source Systems and Services Research. In *Proceedings of the USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)* (2009).

[7] Cerbelaud, D., Garg, S., and Huylebroeck, J. Opening the clouds: qualitative overview of the state-of-the-art open source vm-based cloud management platforms. In *Middleware '09: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware* (New York, NY, USA, 2009), Springer-Verlag New York, Inc., pp. 1–8.

[8] Ciurana, E. *Developing with Google App Engine*. Apress, Berkely, CA, USA, 2009.

[9] Downey, A. B. The structural cause of file size distributions. *SIGMETRICS Perform. Eval. Rev. 29*, 1 (2001), 328–329.

[10] Fouquet, M., Niedermayer, H., and Carle, G. Cloud computing for the masses. In *U-NET '09: Proceedings of the 1st ACM workshop on User-provided networking: challenges and opportunities* (New York, NY, USA, 2009), ACM, pp. 31–36.

[11] Ghemawat, S., Gobioff, H., and Leung, S.-T. The google file system. *SIGOPS Oper. Syst. Rev. 37*, 5 (2003).

[12] Howard, J., Kazar, M., Menees, S., Nichols, D., Satyanarayanan, M., Sidebotham, R., and West, M. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems (TOCS) 6*, 1 (1988), 51–81.

[13] Katabi, D., Handley, M., and Rohrs, C. Congestion control for high bandwidth-delay product networks. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications* (2002), ACM New York, NY, USA, pp. 89–102.

[14] Kohler, E., Handley, M., and Floyd, S. Designing dccp: congestion control without reliability. *SIGCOMM Comput. Commun. Rev. 36*, 4 (2006), 27–38.

[15] Padhye, J., Firoiu, V., Towsley, D., and Kurose, J. Modeling tcp throughput: a simple model and its empirical validation. *SIGCOMM Comput. Commun.Rev.28*,4(1998),303–314.