INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

IMPLEMENTATION OF INITIATIVE DATA PREFETCHING IN DISTRIBUTED FILE SYSTEMS FOR CLOUD COMPUTING

MS. SONAL D. JIRAPURE, DR. A. D. GAWANDE, MRS. L. K. GAUTAM

Department of Computer Science & Engineering, Sipna College of Engineering & Technology, Near Nemani Godown, Badnera Road, Amravati, Maharashtra, India.

Accepted Date: 15/03/2016; Published Date: 01/05/2016

Abstract*:* This work presents an initiative data prefetching scheme on the storage servers in distributed file systems for cloud computing. In this prefetching technique, the client machines are not substantially involved in the process of data prefetching, but the storage servers can directly prefetch the data after analyzing the history of disk I/O access events, and then send the prefetched data to the relevant client machines proactively. To put this technique to work, the information about client nodes is piggybacked onto the real client I/O requests, and then forwarded to the relevant storage server. Next, a prediction algorithm has been proposed to forecast future block access operations for directing what data should be fetched on storage servers in advance. Finally, the prefetched data can be pushed to the relevant client machine from the storage server. Through a series of evaluation experiments we have obtained the result that initiative prefetching technique can benefit distributed file systems for cloud environments to achieve better I/O performance. In particular, configuration - limited client machines in the cloud are not responsible for predicting I/O access operations.

Keywords: Mobile Cloud Computing, Distributed File Systems, Time Series, Server-side Prediction, Initiative Data Prefetching.

PAPER-QR CODE

Corresponding Author: MS. SONAL D. JIRAPURE

Access Online On:

www.ijpret.com

How to Cite This Article:

Sonal D. Jirapure, IJPRET, 2016; Volume 4(9): 1676-1683

INTRODUCTION

The assimilation of distributed computing for search engines, multimedia websites, and data-intensive applications has brought about the generation of data at unprecedented speed [1]. In general, the file system deployed in a distributed computing environment is called a distributed file system, which is always used to be a backend storage system to provide I/O services for various sorts of data intensive applications in cloud computing environments. In fact, the distributed file system employs multiple distributed I/O devices by striping file data across the I/O nodes, and uses high aggregate bandwidth to meet the growing I/O requirements of distributed and parallel scientific applications [2], [3], [4], [5]. However, because distributed file systems scale both numerically and geographically, the network delay is becoming the dominant factor in remote file system access [6], [7]. With regard to this issue, numerous data prefetching mechanisms have been proposed to hide the latency in distributed file systems caused by network communication and disk operations. In these conventional prefetching mechanisms, the client file system (which is a part of the file system and runs on the client machine) is supposed to predict future access by analyzing the history of occurred I/O access without any application intervention. After that, the client file system may send relevant I/O requests to storage servers for reading the relevant data in advance [8], [5], [7]. Consequently, the applications that have intensive read workloads can automatically yield not only better use of available bandwidth, but also less file operations via batched I/O requests through prefetching [9], [10]. On the other hand, mobile devices generally have limited processing power, battery life and storage, but cloud computing offers an illusion of infinite computing resources. For combining the mobile devices and cloud computing to create a new infrastructure, the mobile cloud computing research field emerged[11][12]. Furthermore, considering only disk I/O events can reveal the disk tracks that can offer critical information to perform I/O optimization tactics[13],certain prefetching techniques have been proposed in succession to read the data on the disk in advance after analyzing disk I/O traces [14], [15]. But, this kind of prefetching only works for local file systems, and the prefetched data is cached on the local machine to fulfill the application's I/O requests passively. In brief, although block access history reveals the behaviour of disk tracks, there are no prefetching schemes on storage servers in a distributed file system for yielding better system performance. And the reason for this situation is because of the difficulties in modelling the block access history to generate block access patterns and deciding the destination client machine for driving the prefetched data from storage servers.

I. Existing System

- **Non-prefetching scheme (Non-prefetch**), which means that no prefetching scheme is enabled in the distributed file system.
- **Readahead prefetching scheme (Readahead**), which may prefetch the data in next neighbour blocks on the storage servers, and it can adjust the amount of the prefetched data based on the amount of data already requested by the application.
- **Signature-based prefetching scheme (IOSig+),** which is a typical data prefetching scheme on the client file systems by analyzing applications' access patterns, and the source code is also available [4].

II. Proposed System

- **Proposed System:** (Initiative Data Prefetching)

We have proposed a system which intends to propose a novel prefetching scheme for distributed file systems in cloud computing environments to yield better I/O performance.

This proposed system makes two contributions:

1) **Chaotic time series prediction and linear regression prediction to forecast disk I/O access.**

We have modeled the disk I/O access operations, and classified them into two kinds of access patterns, i.e. the random access pattern and the sequential access pattern.

2) **Initiative data prefetching on storage servers**.

The scheme of initiative data prefetching is a novel idea presented in this work, and scheme is demonstrated in Figure while it handles read requests (the assumed synopsis of a read operation is read(int fildes, size t size, off t off)). In the figure, the storage server can predict the future read operation by analyzing the history of disk I/Os, so that it can directly issue a physical read request to fetch data in advance. The most attractive idea in the figure is that the prefetched data will be forwarded to the relevant client file system proactively, but the client file system is not involved in both prediction and prefetching procedures. Finally, the client file system can respond to the hit read request sent by the application with the buffered data. As a result, the read latency on the application side can be reduced significantly. Furthermore, the client machine, which might

have limited computing power and energy supply, can focus on its own work rather than predicting related tasks.

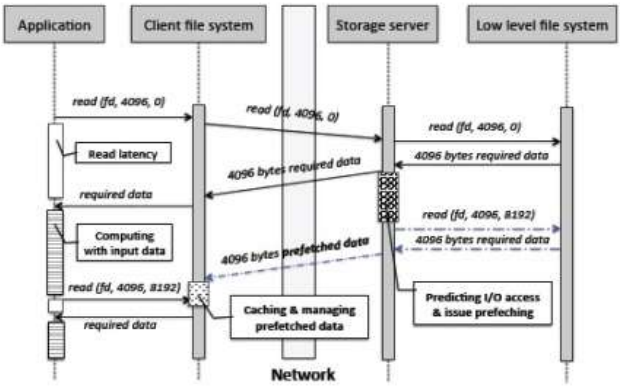

Figure.1 Initiative Prefetching Scheme

The first read request is fulfilled as a normal case in the distributed file system, so that the application on the client machine can perform computing tasks after receiving the required data. At the same time, the storage server is able to predict the future I/O access by analyzing the history of disk I/O access, therefore, it issues a physical read request to fetch the data (that will be accessed by the predicted read I/O access) in advance. At last, the prefetched data will be pushed to relevant client file systems proactively. While the prediction of I/O access made a hit, the buffered data on the client file systems can be responded to the application directly, that is why the read latency can be reduced.

A. Implementation

Our proposed application is a prototype of distributed file system. Components are

Client file system:

Client file system is responsible for collecting the information about the client file system and applications, as well as piggybacking it onto real I/O requests.

Storage Server:

It handles file management and provides the actual file I/O service for client file systems and performs initiative prefetching. It records the block access events with the piggybacked information sent by the client file systems.

The main steps of the work are as follows:

1. User Registration
2. File Upload by User on server in distributed environment.
3. Display time taken for uploading file on server.
4. Display file uploaded on server.
5. Display the way file is uploaded, whether it is on single server or multi server.
6. Display values from database of registered users and files uploaded on the server.
7. Display the files to be downloaded according to request of user.
8. File Download.

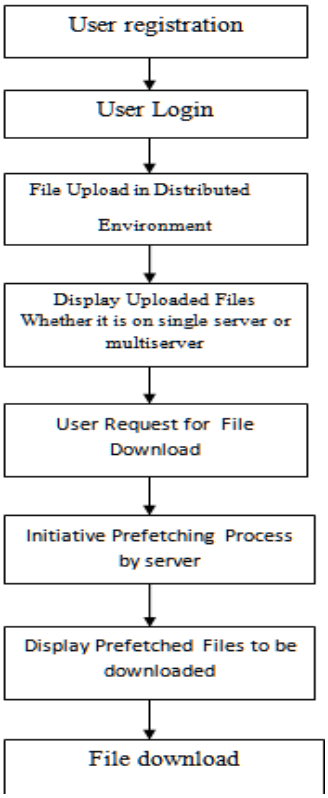

Fig.2 Working Flow of Application

This application and the prefetching mechanism is mainly designed for cloud having resource limited client machines and many Online Transaction Processing applications.

Firstly new user has to register himself. Then he can login into the application with his registered username and password.

After the login user can see a window where he can upload the file he wants to be uploaded. As we are having distributed environment, storage server will store the file on single server or multi-server by knowing the type and size of file itself. Since the prefetching application can predict and prefetch i/o access history of user, the server acts like metadata server. It receives first read request i.e to upload the file and stores the information about itself which can be used at the time of prefetching the file. This method is nothing but the piggybacking of the information with read request. Piggybacking method used in application reduce read latency on application side and user can easily work with related tasks.

Next Step is user can see the downloaded files which are uploaded. In such a way that whether file is uploaded on a single server or on multi-servers. This information about the files in distributed environment get displayed.

When User wants to download the file stored in distributed environment, again a request sent to the server. Here the initiative prefetching mechanism is applied which works according to the initiative prefetching algorithm. In this mechanism, storage server which has been considered as metadata server acts as low level file system now. After receiving the read request from user, storage server analyzes the i/o access history of user and prefetches the data which can be accessed in near future according to history. This reduces the latency and user can access the data easily in OLTP and resource limited client machines.

This prefetched data is displayed on the user side so that the required file can be made available to the user for downloading. Hence user can download the file in distributed environment easily and can work effectively even in heavy work load.

Apart from this, application allows a user to see the time taken by a file to upload and download. Also the registered users and files uploaded can also be displayed on the request of user.

CONCLUSION

We have proposed, an initiative data prefetching approach in which the storage servers are capable of predicting future disk I/O access to guide fetching data in advance after analyzing the existing logs, and then they proactively push the prefetched data to relevant client file systems for satisfying future applications' requests. For the purpose of effectively modelling disk I/O access patterns and accurately forwarding the prefetched data, the information about client file systems is piggybacked onto relevant I/O requests, then transferred from client nodes to corresponding storage server nodes. Therefore, the client file systems running on the client nodes neither log I/O events nor conduct I/O access prediction; consequently, the thin client nodes can focus on performing necessary tasks with limited computing capacity and energy endurance.

In this way the Initiative Prefetching Mechanism based application works effectively for the users working with heavy work load and data on cloud.

REFERENCES

1. J. Gantz and D. Reinsel. The Digital Universe in 2020: Big Data, Bigger Digital Shadows, Biggest Growth IN http://www.emc.com/collateral/analystreports/ idc-digital-universe-united-states. pdf [Accessed on Oct. 2013], 2013.
2. J. Kunkel and T. Ludwig, Performance Evaluation of the PVFS2 Architecture, In Proceedings of 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing, PDP '07, 2007
3. S. Ghemawat, H. Gobioff, S. Leung, The Google file system, In Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03), pp. 29–43, 2003.
4. E. Shriver, C. Small, and K. A. Smith. Why does file system prefetching work? In Proceedings of the USENIX Annual Technical Conference (ATC '99), USENIX Association, 1999.
5. J. Stribling, Y. Sovran, I. Zhang and R. Morris et al. Flexible, widearea storage for distributed systems with WheelFS. In Proceedings of the 6th USENIX symposium on Networked systems design and implementation (NSDI'09), USENIX Association, pp. 43–58, 2009. IEEE TRANSACTIONS ON CLOUD COMPUTING , Volume: PP,Year: 2015
6. J. Griffioen, and R. Appleton. Reducing file system latency using a predictive approach. In Proceedings of the 1994 USENIX Annual Technical Conference (ATC '94), pp. 197-107, 1994.
7. T. M. Kroeger and D. Long. Predicting file system actions from prior events. In Proceedings of the USENIX Annual Technical Conference. USENIX Association, 1996

8. J. He, J. Bent, A. Torres, and X. Sun et al. I/O Acceleration with Pattern Detection. In Proceedings of the 22nd International ACM Symposium on High Performance Parallel and Distributed Computing (HPDC '13), pp. 26–35, 2013.
9. S. Byna, Y. Chen, X. Sun, R. Thakur, and W. Gropp. Parallel I/O prefetching using mpi file caching and I/O signatures. In Proceedings of the 2008 ACM/IEEE conference on Supercomputing (SC 08), pp. 1-12, 2008
10. M. S. Obaidat. QoS-Guaranteed Bandwidth Shifting and Redistribution in Mobile Cloud Environment. IEEE Transactions on Cloud Computing, Vol.2(2):181–193, April-June 2014, DOI:10.1109/TCC.2013.19
11. C. Chen, M. Won, R. Stoleru and G. Xie. Energy-Efficient Fault-Tolerant Data Storage & Processing in Mobile Cloud. IEEE Transactions on Cloud Computing, DOI: 10.1109/TCC.2014.2326169,online in 2014.
12. S. Narayan, J. A. Chandy. Trace Based Analysis of File System Effects on Disk I/O. In Proceedings of 2004 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS '04), 2004.
13. X. Ding, S. Jiang, F. Chen, K. Davis, and X. Zhang. DiskSeen: Exploiting Disk Layout and Access History to Enhance I/O Prefetch.
14. S. Jiang, X. Ding, Y. Xu, and K. Davis. A Prefetching Scheme Exploiting both Data Layout and Access History on Disk. ACM Transaction on Storage Vol. 9(3), Article 10, 23 pages, 2013.