Negotiate Emulation Data Prefetching in Diffuse File Conformity for Cloud Enumerate

Anu D[1], Anu Priya A[2]

[1]Anu D, M.E (CSE)/ Global Institute of Engineering and Technology, Vellore India.

[1]anu91_devan@yahoo.com

[2]Anu Priya A, Asst Prof (CSE)/ Global Institute of Engineering and Technology, Vellore India.

[2]anupriya.mtech@gmail.com

ABSTRACT

It is 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, two prediction algorithms have 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 with a collection of application benchmarks, we have demonstrated that our presented 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, which can definitely contribute to preferable system performance on them.

Keywords—Prefetching, piggybacking, cloud services.

1. 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. For instance, the amount of data created, replicated, and consumed in United States may double every three years through the end of this decade, according to the EMC-IDC Digital Universe 2020 study. 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 . However, because distributed file systems scale both numerically and geographically, the network delay is becoming the dominant factor in emote file system access. 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. 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. Then, the client file system may send relevant I/O requests to storage servers for reading the relevant data in advance. 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. 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. Namely, mobile cloud computing provides mobile applications with data storage and processing services in clouds, obviating the requirement to equip a powerful hardware configuration, because all resource-intensive computing can be completed in the cloud.

Thus, conventional prefetching schemes are not the best-suited optimization strategies for distributed file systems to boost I/O performance in mobile clouds, since these schemes require the client file systems running on client machines to proactively issue prefetching requests after analyzing the occurred access events recorded by themselves, which must place negative effects to the client nodes. Furthermore, considering only disk I/O events can reveal the disk tracks that can offer critical information to perform I/O optimization tactics, certain prefetching techniques have been proposed in succession to read the data on the disk in advance after analyzing disk I/O traces. But, this kind of prefetching only works for local file systems, and the prefetched data is I/O requests passively. In brief, although block access history reveals the behavior 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 modeling the block access history to generate block access patterns and deciding the destination client machine for driving the prefetched data from storage servers. To yield attractive I/O performance in the distributed file system deployed in a mobile cloud environment or a cloud environment that has many resource-limited client machines, this paper presents an initiative data prefetching mechanism. The proposed mechanism first analyzes disk I/O tracks to predict the future disk I/O access so that the storage servers can fetch data in advance, and then forward the prefetched data to relevant client file systems for future potential usages. In short, this paper makes the following 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. Therefore, in order to predict the future I/O access that belongs to the different access patterns as accurately as possible (note that the future I/O access indicates what data will be requested in the near future), two prediction algorithms including the chaotic time series prediction algorithm and the linear regression prediction algorithm have been proposed respectively.

2.Initiative data prefetching on storage servers. Without any intervention from client file systems except for piggybacking their information onto relevant I/O requests to the storage servers. The storage servers are supposed to log disk I/O access and classify access patterns after modeling disk I/O events. Next, by properly using two proposed prediction algorithms, the storage servers can predict the future disk I/O access to guide prefetching data. Finally, the storage servers proactively forward the prefetched data to the relevant client file systems for satisfying future application's requests.

2. OVERVIEW OF EXISTING SYSTEMS

It have been proposed, implemented and evaluated an initiative data prefetching approach on the storage servers for distributed file systems, which can be employed as a backend storage system in a cloud environment that may have certain resource-limited client machines. To be specific, 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 modeling 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. Besides, the prefetched data will be proactively forwarded to the relevant client file system, and the latter does not need to issue a prefetching request. So that both network traffics and network latency can be reduced to a certain extent, which have been demonstrated in our evaluation experiments.

3. PROPOSED APPROACH

Two prediction algorithms have 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. It is proposed, implemented and evaluated an initiative data prefetching approach on the storage servers for distributed file systems, which can be employed as a backend storage system in a cloud environment that may have certain resource-limited client machines. To be specific, 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 modeling 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. Besides, the prefetched data will be proactively forwarded to the relevant client file system, and the latter does not need to issue a prefetching request. So that both network traffics and network latency can be reduced to a certain extent, which have been demonstrated in our evaluation experiments.

4. SYSTEM DESIGN

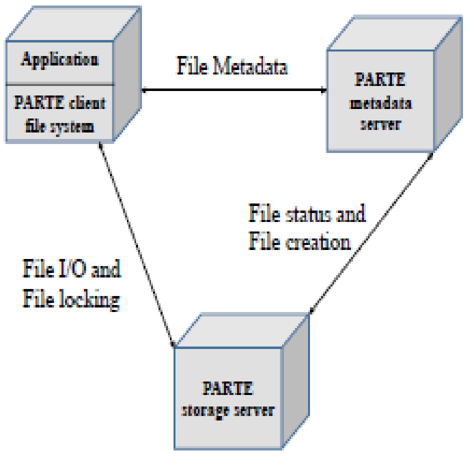

Fig-1: System Architecture

4.1 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

4.2 Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

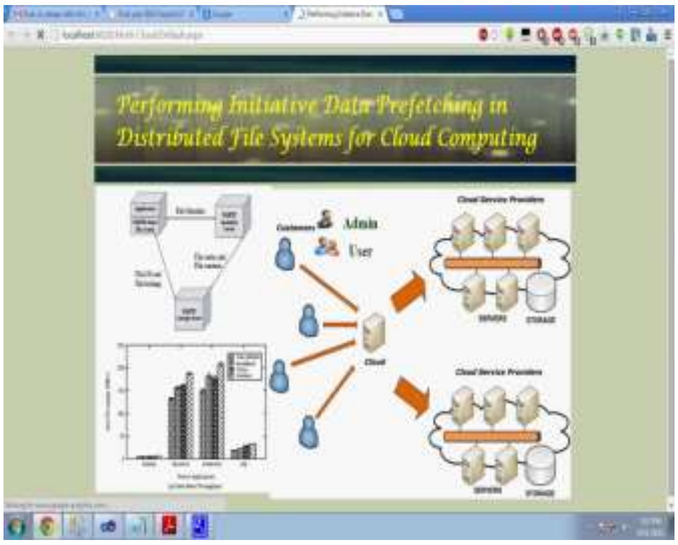
Performing Initiative Data Prefetching in
Distributed File Systems for Cloud Computing
Admin
User
Customers
Cloud
Cloud Service Providers
SERVERS
STORAGE

Fig-2: System Process in Cloud

5. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.Three key considerations involved in the feasibility analysis are economical feasibility, technical feasibility and social feasibility.

5.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

5.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

5.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

6. MODULE DESCRIPTION

6.1 Home

Distributed file systems for mobile clouds. Moreover many studies about the storage systems for cloud environments that enable mobile client devices have been published. A new mobile distributed file system called mobile. DFS has been proposed and implemented in which aims to reduce computing in mobile devices by transferring computing requirements to servers. Hyrax, which is a infrastructure derived from Hadoop support cloud computing on mobile devices. But Hadoop is designed for general distributed computing, and the client machines are assumed to be traditional computers. In short, neither of related work targets at the clouds that have certain resource-limited client machines, for yielding attractive performance enhancements.

6.2 Rank Module

This paper intends to propose a novel prefetching scheme for distributed file systems in cloud computing environments to yield better I/O performance. In this section, we first introduce the assumed application contexts to use the proposed prefetching mechanism; then the architecture and related prediction algorithms of the prefetching mechanism are discussed specifically; finally, we briefly present the implementation details of the file system used in evaluation experiments, which enables the proposed prefetching scheme.

Fig-3: Rank Module

6.3 Cryptography Module

Initiative data prefetching on storage servers:. Without any intervention from client file systems except for piggybacking their information onto relevant I/O requests to the storage servers. The storage servers are supposed to log disk I/O access, and classify access patterns after modeling disk I/O events. Next, by properly using two proposed prediction algorithms, the storage servers can predict the future disk I/O access to guide prefetching data. Finally, the storage servers proactively forward the prefetched data to the relevant client file system for satisfying future application's requests.

6.4 Encryption and Decryption Module

File bench which allows generating. A large variety of workloads to assess the performance of storage systems. Besides, File bench is quite flexible and enables to minutely specify a collection of applications, such as mail, web, file, and database servers. We chose File bench as one of benchmarks, as it has been widely used to evaluate file systems by emulating a variety of several server-like applications. I Ozone, which is a micro-benchmark that evaluates the performance of a file system by employing a collection of access workloads with regular patterns, such as sequential, random, reverse order, and strided. That is why we utilized it to measure read data throughput of the file systems with various prefetching schemes, when the workload have different access patterns.

6.5 Architecture And Implementation

This paper intends to propose a novel prefetching scheme for distributed file systems in cloud computing environments to yield better I/O performance. In this section, we first introduce the assumed application contexts to use the proposed prefetching mechanism. Then the architecture and related prediction algorithms of the prefetching mechanism are discussed specifically; finally, we briefly present the implementation details of the file system used in evaluation experiments, which enables the proposed prefetching scheme.

6.6 Assumptions in Application Contexts

This newly presented prefetching mechanism cannot work well for all workloads in the real world, and its target application contexts must meet two assumptions

Assumption 1: Resource-limited client machines. This newly proposed prefetching mechanism can be used primarily for the clouds that have many resourcelimited client machines, not for generic cloud environments. This is a reasonable assumption given that mobile cloud computing, which employs powerful cloud infrastructures to offer computing and storage services on demand, for alleviating resource utilization in mobile devices.

Assumption 2: On-Line Transaction Processing (OLTP) applications. It is true that all prefetching schemes in distributed file systems make sense for a limited number of read-intensive applications such as database-related OLTP and server-like applications. That is because these long-time running applications may have a limited number of access patterns, and the patterns may occur repetitively during the lifetime of execution, which can definitely contribute to boosting the effectiveness of perfecting.

6.7 Piggybacking Client Information

Most of the I/O tracing approaches proposed by other researchers focus on the logical I/O access events occurred on the client file systems, which might be useful for affirming application's I/O access patterns [16], [30]. Nevertheless, without relevant information about physical I/O access, it is difficult to build the connection between the applications and the distributed file system for improving the I/O performance to a great extend.

In this newly presented initiative prefetching approach, the data is prefetched by storage servers after analyzing disk I/O traces, and the data is then proactively pushed to the relevant client file system for satisfying potential application's requests. Thus, for the storage servers, it is necessary to understand the information about client file systems and applications. To this end, we leverage a piggybacking mechanism.when sending a logical I/O request to the storage server, the client file system piggybacks information about the client file systems and the application. In this way, the storage servers are able to record disk I/O events with associated client information, which plays a critical role for classifying access patterns and determining the destination client file system for the prefetched data. On the other side, the client information is piggybacked to the storage servers, so that the storage servers are possible to record the disk I/O operations accompanying with the information about relevant logical I/O events. Demonstrates the structure of each piece of logged information stored on the relevant storage server. The information about logical access includes anode information, file descriptor, offset and requested size. And the information about the relevant physical access contains storage server ID, stripe ID, block ID and requested size.

6.8 I/O Access Prediction

Many heuristic algorithms have been proposed to shepherd distributing file data on disk storage, as a result, data stripes that are expected to be used together will be located close to one another. Work flow among client who runs the application, client file system, storage server and low level file system: the client file system is responsible for generating extra information about the application, client file system (CFS) and the logical access attributes; after that, it piggybacks the extra information onto relevant I/O request, and sends them to the corresponding storage server. On the other hand, the storage server is supposed to parse the request to separate piggybacked information and the real I/O request. Apart from forwarding the I/O request to the low level file system, the storage server records the disk I/O access with the information about the corresponding logical I/O access.

7. CONCLUSION

This paper have been implemented and evaluated an initiative data prefetching approach on the storage servers for distributed file systems, which can be employed as a backend storage system in a loud environment that may have certain resource-limited client machines. Be specific, 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 modeling 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. Besides, the prefetched data will be proactively forwarded to the relevant client file system, and the latter does not need to issue a prefetching request. So that both network traffics and network latency can be reduced to a certain extent, which have been demonstrated in our evaluation experiments.

REFERENCES

[1] J. Gantz and D. Reinsel. The Digital Universe in 2020:Big Data, Bigger Digital Shadows, Biggest Growth in the Far East-United States. 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 PVFS2Architecture, 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] IO Signature Plus (IOSIG+) Software Suite[Accessed on Nov. 2012]. https://code.google.com/p/iosig/.

[5] UMass Trace Repository: OLTP Application I/O[Accessed on Jan. 2014]. http://traces.cs.umass.edu/index.php/Storage/Storage.

[6] MobiDFS: Mobile Distributed File System[Accessed on Nov. 2014]. https://code.google.com/p/mobidfs/.

[7] E. E. Marinelli. Hyrax: Cloud computing on mobile devices using mapreduce. CMU, Tech. Rep., 2009.

[8] P. Sehgal, V. Tarasov, E. Zadok. Evaluating Performance and Energy in File System Server Workloads. the 8th USENIX Conference on File and Storage Technologies (FAST '10), pp.253-266, 2010.

[9] V. Tarasov, S. Bhanage, E. Zadok, et al. Benchmarking file system benchmarking: It* is* rocket science. In Proceedings of HotOS XIII, 2011.

[10] N. Nieuwejaar and D. Kotz. The galley parallel file system. Parallel Computing, 23(4-5):447–476, 1997.

[11] IOzone Filesystem Benchmark. http://www.iozone.org.

[12] Filebench Ver. 1.4.9.1 [Accessed on Apr. 2012], http://filebench.sourceforge.net.

[13] SysBench benchmark. http://sysbench.sourceforge.net.

[14] M. Cencini, F. Cecconi and N. Vulpiani. Chaos From Simple models to complex systems. World Scientific, ISBN:9814277657, 2010.

[15] J. Liao, Y. Ishikawa. Partial Replication of Metadata to Achieve High
Metadata Availability in Parallel File Systems. In the Proceedings of 41st International Conference on Parallel Processing (ICPP '12), pp. 168–177, 2012.