I/O Traffic Management on PC-Cluster based Cloud Storage System

Tin Tin Yee, Thinn Thu Naing
University of Computer Studies, Yangon, Myanmar
tintinyee.tty@gmail.com, thinnthu@gmail.com

Abstract

Cloud storage system architecture and design plays a vital role in the cloud computing infrastructure. Cloud storage system provides users to efficient storage space with elasticity feature. One of the challenges of cloud storage server is difficult to balance the providing huge elastic capacity of storage and investment of expensive cost for it. In order to solve this issue, we propose the low cost PC cluster based storage system architecture which can be activated to store a large amount of data and provide cost-effective for cloud storage user and present FP-growth algorithm for metadata prefetching method and Least Recently Used (LRU) replacement policy for caching to improve performance by reducing response time. According to the experimental testing, the storage can be utilized more than 90% of storage space and can significantly reduce the average response time.

1. Introduction

Nowadays, everyone with a computer spends a lot of time acquiring data and then trying to find a way to store it. For some computer owners, finding enough storage space to hold all the data they have acquired are a real challenge. Some people invest in larger hard drives. Others prefer external storage devices like thumb drives or compact discs. Desperate computer owners might delete entire folders worth of old files in order to make space for new information. But some are choosing to rely in a growing trend: ***cloud storage.***

Cloud storage is a storage service mode which the service providers supply storage capacities and data storage services through the Internet to the clients. Cloud storage has several advantages over traditional data storage. If the data store on a cloud storage system, will be able to get to that data from any location that has the Internet access. There is no need to carry around a physical storage device or use the same computer to save and retrieve the information.

In cloud storage, disk storage is one of the biggest expenses. There is strong concern that cloud service providers will drown in the expense of storing data, especially unstructured data such as documents, presentations, PDFs, VM Images, Multimedia data, etc. Cloud service providers offer huge capacity cost reductions, the elimination of labor required for storage management and maintenance and immediate provisioning of capacity at a very low cost per terabyte. Therefore, the storage and computing on massive data are major key challenge for a cloud computing infrastructure.

The rest of this paper is organized as follows. Section 2 describes related work. In section 3 discusses physical topology of PC cluster based cloud storage system. In section 4 describes I/O traffic management scheme and experimental environment presents in section 5. Finally, section 6 is the conclusion and future work.

2. Related Work

Recently, there are many cloud computing and cloud storage providers, such as IBM, Google, Microsoft, Amazon, HP, iCloud, etc. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the cloud. It gives any developer access to the same highly scalable, reliable, secure, fast,

inexpensive infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers. Yong et al. [1] proposed a storage cluster architecture CoStore using network attached storage devices. In CoStore the consistency of a unified file namespace is collaboratively maintained by all participating cluster members without any central file manager and designed the data anywhere and metadata at fixed locations file system layout for efficiency and scalability in a CoStore cluster. Qinlu et al. [3] presented the idea of P2P model based on cloud storage online backup prototype system and the prototype system architecture and topology analysis, research and development provide a theoretical basis. Jiyi et al. [5] introduced cloud storage reference model. This model designed scalable and easy to manage storage system but aren't designed to be high performance. This paper presented the key technologies, several different types of clouds services, the advantages and challenges of cloud storage. Xu et al. [6] proposed new cloud storage architecture based on P2P which provide a pure distributed data storage environment without any central entity. The cloud based on the proposed architecture is self-organized and self-managed and as better scalability and fault tolerance using Distributed Hash Table approach**.**

Bo et al. [2] presented hadoop distributed file system does not perform well for massive file since huge number of small files imposed heavy burden on NameNode of HDFS, correlations between files were not considered for data placement and no prefetching mechanism was provided to improve I/O performance. Lin et al. [4] introduced an Affinity-based Metadata Prefetching (AMP) scheme is proposed for metadata servers in large scale distributed storage system to provide aggressive metadata prefetching. AMP scheme that applies data mining techniques to discover and identify the affinities existing among metadata accesses from past history and then uses these affinities as hints to judiciously perform aggressive metadata prefetching.

In this paper, we propose the low cost PC cluster based storage system architecture that can be activated to store a large amount of data and provide cost-effective for cloud storage user and FP-growth based metadata prefetching algorithm to improve the efficient of the I/O traffic for PC cluster based storage system.

3. Physical Topology of PC Cluster based Cloud Storage System

In this section, we describe how to build physical topology of PC cluster based cloud storage system. The system framework is shown in figure 1. The overall framework of PC cluster based cloud storage system consists of three layers that are web based application services layer, Hadoop Distributed File System (HDFS) layer and PC cluster layer.

The web based application layer provides interface for the users whose can store their own applications such Virtual Machine (VM) images, dataset and multimedia data, etc. The HDFS layer supports the file system for PC cluster layer. The PC cluster layer provides to store large amount of data.

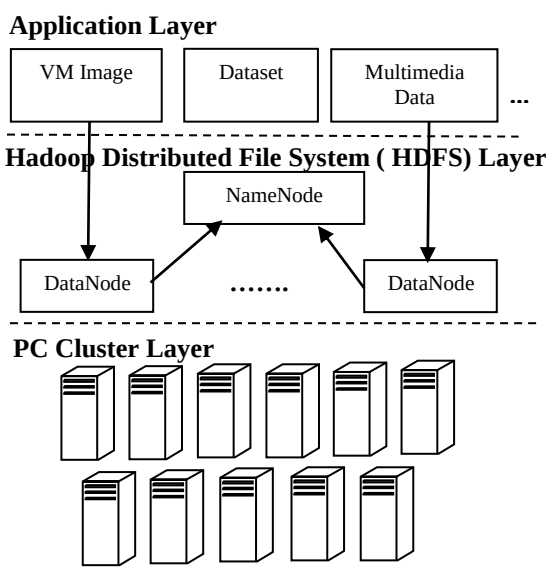

Figure 1: System Framework of PC Cluster based Cloud Storage System

3.1 Hadoop Distributed File System (HDFS)

The Hadoop Distributed File System (HDFS) is a distributed, highly fault-tolerant file system designed for storing very large files with streaming data access patterns, running on clusters consisting of low-cost commodity hardware. A HDFS is divided into two kinds of nodes operating with a master-slave pattern: a NameNode and many DataNodes. The NameNode maintains the file system namespace including the file system tree and the metadata for all the files and directories (i.e. block size, file length, replication, modification time, ownership, permission information, etc). The DataNodes store the file measured by "block" (with the default size being 64MB), which is much larger than the typical "block" of POSIX file systems. HDFS breaks a large file into blocks and stores them across DataNodes. It typically replicates every block three times on different DataNodes for fault tolerant. HDFS adopts direct client access mechanism to achieve high bandwidth. When clients read file from HDFS, they interact with NameNode for file metadata and then perform actual I/O operations directly with relevant DataNode to retrieve data.

3.2 PC Cluster

Nowadays, many organizations need terabytes storage systems but are very expensive and require higher degree of skills for their operations and maintenance. Usually a desktop PC contains more than 100 GB Hard Disk Drive (HDD), at least 256 MB or greater RAM and 2GHz or higher processor. A typically an operating system installation and other software installation do not use more than 20 GB of HDD storage. This leaves on the average about 80% of the storage space to be used. Therefore, we proposed PC cluster based storage server that is inexpensive and easy to maintain.

PC cluster is a collection of computer nodes, which is interconnected by a high-speed switching network, all nodes can be used individually or collectively as a cluster.

3.2.1 System Overview of PC Cluster based Cloud Storage System

PC cluster based storage system tries to transfer from the cluster computing to storage server. The storage system uses inexpensive PC components. The large files can be stored by striping the data across multiple nodes. In PC cluster consists of one NameNode as server and many DataNodes as clients. The cluster based storage system uses HDFS (Hadoop Distributed File System) to store data in the collection of the nodes. Each node has its own memory, I/O devices and operating system. The nodes are physically separated and connected via a LAN.

In this system consists of a NameNode and many DataNodes. NameNode manages the whole PC cluster and maintains the metadata of HDFS that contains the information of blocks, the current locations of blocks, and monitoring the states of all nodes in the cluster. NameNode is recorded any changes to the file system namespace such as opening, closing, renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes store the physical storage of the files. DataNodes also perform block creation, deletion and replication upon instruction from the NameNode. In the system, files are divided into blocks that are stored as independent units. In PC cluster, each individual machine of a cluster is referred to as a node. Our system is based on client-server architecture and consists of one server node (NameNode) and many clients nodes(DataNodes) in order to make them work as a single machine. NameNode has two networks cards and one is connected to the backbone network and the other is connected the DataNodes using gigabyte switch. All communication protocols build on the TCP/IP protocol. HDFS clients connect to a Transmission Control Protocol (TCP) port opened on the NameNode, and then communicate with the NameNode using a proprietary Remote Procedure Call (RPC)-based protocol. Data nodes talk to the NameNode using a proprietary block-based protocol.

The PC cluster is the use of multiple computers, typically PCs which was built 1.5

GHz Pentium P4 processors, 80 GB Hard disks and 512 MB RAM and running with Ubuntu OS. These PC are connected by a gigabyte switch that can create illusion of being one machine. The proposed PC cluster based storage server utilizes the existing PC machines in our university without purchasing any extra hardware and software components. Therefore, this storage server is very cost effective architecture.

4. I/O Traffic Management Scheme for PC Cluster based Cloud Storage System

In this section, we propose I/O traffic management for PC cluster based storage system.

A. Proposed Method for Metadata Prefetching and Caching approach

When cloud users read file from PC cluster based storage server, they first interact with NameNode for file metadata and then perform actual I/O operations directly with relevant DataNodes to retrieve data. In this case, response time delay caused by reading from NameNode that is the major performance bottleneck of processing user request.

To solve the performance bottleneck, we propose FP-growth algorithm for metadata prefetching method and Least Recently Used (LRU) replacement policy for caching to improve performance of PC cluster based storage server for cloud storage. Caching keeps in memory the data that are the most likely to be used again while prefetching attempts to bring data in memory before they are need. Prefetching is an effective technique for improving file access performance which can reduce response time delay for I/O system.

Prefetching for metadata file is critical for the overall system performance. In our system, metadata will be prefetched from NameNode's historical metadata access records using FP-growth algorithm. The metadata prefetching reduces response time on NameNode. The prefetching and caching process steps are described as follows:

Step 1: When client read request receives via web interface, it first look up the file metadata from cache. If it found, then directly performs read operation from correspond DataNodes. If it not found, go to the step 2.

Step 2: The client sends request to the prefetching module on the NameNode.

Step 3: Prefetching module using FP-growth algorithm fetches the request metadata file on NameNode. And then returned to the client and stored the metadata on client caches.

Step 4: When metadata file insert in cache, Least-Recently-Used (LRU) replacement policy manages the metadata cache.

B. FP-growth Algorithm for Metadata Prefetching

We introduce our data mining based FP-growth metadata prefetching algorithm. This algorithm explores deep association from metadata files. Metadata associations widely exist in storage systems. The metadata of two or more files are affined if they are linked together either spatially or temporally. For example, /tmp always has a storage spatial association with /tmp/user, /tmp/user/part1 and /tmp/user/part2. If we can find out the storage association between these metadata, we could prefetch all these metadata files into cache simultaneously. This can potentially significantly reduce the response time in storage server.

FP-growth algorithm mine frequent patterns using FP-tree algorithm to find association rules from metadata and it adopts a divide-and-conquer strategy as follows: compress the metadata representing frequent items into frequent-pattern tree or FP-tree, but retain the itemset association information, and then divide into a set of conditional metadata, each associated with one frequent item, and mine each such metadata separately.

Let $m = \{m_1, m_2, ..., m_i\}$ be a set of items in a metadata, and a transaction in metadata file $M = \{M_1, M_2, ..., M_n\}$, where n is the total number of metadata. The support (or occurrence frequency) of a pattern A, where A is a set of

items, is the number of transactions containing A in metadata. The pattern A is frequent if A's support is no less than a predefined minimum support threshold, ξ. Table 1 and 2 list the summary of notations used in the algorithms.

Table 1. Notations Used in the FP-tree Algorithm

Notation	Description
M	The set of metadata
ξ	The predefined minimum support threshold
m	The set of frequent items
L	The list of frequent items
p	The first element of sorted frequent-item list
P	The remaining requent-item list

FP-tree Algorithm(*M, ξ*)
begin
1. Scan the metadata *M*.
2. Collect m, the set of frequent items, and the support of each frequent item.
3. Sort *m* in support-descending order as *L*, the list of frequent items.
4. Create the root of an FP-tree, *T*, and label it as "null".
5. For each transaction Trans in *M* do
   begin
6. Select the frequent items in *M*
7. Sort them according to the order of *L*.
   end
8. [*p* |*P*] select from Sorted frequent-item list
9. Call **insert_tree**([*p* | *P*], *T*).

end

Figure 2. FP-tree Algorithm

Procedure insert_tree([p | P], T)
begin
if (T has a child *N* and item-name = p.item-name)
then increment N's count by 1;
else create a new node *N*, with its count initialized to 1,its parent link to T
end if

Figure 3. Insert_tree Procedure

Table 2. Notations Used in the FP-growth Algorithm

Notation	Description
P	The single prefix-path part of Tree
β	The combination of the nodes in the path P
α	The itemset in the Metadata
a_i	A frequent item in the tree

FP_growth Algorithm(Tree,α)
begin
if (Tree contains a single path P)
then for each combination of the nodes β
in the path P
generate pattern $\beta \cup \alpha$ with
support= minimum support of
nodes in β
else if for each a_i in the header of Tree{
generate pattern $\beta = a_i \cup \alpha$ with
support = a_i.support
contruct β's conditional pattern base
then $\beta's$ conditional FP_tree Tree β
if Tree $\beta \neq \emptyset$ then
call FP_tree(Tree β, β)}
end if
end if
end if
end

Figure 4. FP-growth algorithm pseudo-code

C. Least-Recently-Used Replacement Policy for Caching

Cache is a component that transparently stores data so that future requests for that data can be served faster and improve the overall system performance.

With LRU, every *metadataname* in cache has a time-stamp assigned when inserted or when found in cache. It selects candidates for removal at cache finding the oldest files in the cache using the time-stamp stored in the cache with the

metadataname. The LRU algorithm in pseudo code as shown in figure 5.

```
LRU Algorithm
begin
  if (metadataname in cache)
     then value (metadataname) = timestamp
  end if
  while (cachefree < filesize(metadataname))
     find and remove candidate in cache
     with oldest  timestamp
    cachefree=cachefree + filesize   (candidate)
     insert value(metadataname) = timestamp
   end while
end
```

Figure 5: LRU algorithm pseudo code

4.1 Theoretical Analysis

In this section, the average response time is considered to analyze in the PC cluster based storage system. The average response time in a PC cluster based storage system is defined as equation 1.

$$\text{Average response time} = H \times Access_{c} + (1-H) \times Access_{d} \quad (1)$$

Where H , $Access_c$ and $Access_d$ are the hit rate of cache, cache access time and disk access time respectively. The definition of notations is shown in table 3.

Table 3. Definition of Notations

Symbols	Definition
H	Hit rate of cache
$Access_c$	Cache access time
Access	Disk access time
N_r	Number of request which has been accessed in metadata cache
N_t	Total number of requests to the metadata cache
seek time	Disk seek time
rotational delay	Delay time for the request sector
transfer time	Time to transfer a block
control overhead	Disk controller overhead

The hit rate $H= N_r/N_t$, where N_r is the number of request which has been accessed in the metadata cache and N_t is the total number of requests to the metadata cache. The hit rate is the accuracy of the proposed algorithm. The disk access time $Access_d$= seek time+ rotational delay + transfer time+ control overhead, where seek time is the time taken for a particular track on a storage disk, rotational delay is the time needed for the requested sector to rotate under the head, transfer time is the time takes to transfer a block and control overhead is the overhead imposed by disk controller.

The average response time of proposed system is analyzed using equation 1 based on the various hit rate. In this analysis, we assume that the seek time is 12ms, the transfer time is 0.128ms, the controller overhead is 8ms rotational delay is 5.56ms and cache access time is 5ms. The results of analysis are shown in table 4.

Table 4. Analysis Results

H (%)	$Access_c$ (ms)	$Access_d$ (ms)	Average response time(ms)
10%	5	25.688	23.6192
20%	5	25.688	20.5504
30%	5	25.688	19.4816
40%	5	25.688	17.4128
50%	5	25.688	15.344
60%	5	25.688	13.2752
70%	5	25.688	11.2064
80%	5	25.688	9.1376
90%	5	25.688	7.0688
100%	5	25.688	5

According to table 4, the average response time depends on the hit rate of cache. Therefore, the proposed algorithm is more accuracy increase as well as the average response time is also fast.

5. Experimental Environment

The test platform is built on a cluster with one NameNode and five DataNodes of commodity computer. All nodes are interconnected with 1 Gbps Ethernet network. In each node, Ubuntu server 10.10 with the kernel of version 2.6.28-11-server is installed. Java

version is 1.6.0 and Hadoop version is 0.20.2. The size of HDFS blocks is 64 MB and the number of replicas is set to three. During the experiments, installation of Ubuntu operating system use 9.8125% and the remaining 90.1875% of the storage space to be used for PC cluster based storage server.

In the system configuration used the number of five DataNodes and consumed about 10 GB Hard disks storage of each PC for installation of an operating system and other software installation. The available storage capacity of these PCs is combined together, and then can provide 5 x 70 = 350 GB of storage capacity. The storage capacity remains unused and can be utilized if combined to store huge amount of data. If our system configuration used the number of 20 DataNodes, the available storage capacity can provide 20x70=1400 GB. Therefore, in the storage server, storage capacity can be increased depending on the number of PC node in PC cluster. The average storage capacity of PC cluster based storage server is shown in figure 6. Moreover, 100 MB, 100 MB and 1000 MB of data files are stored to the cluster. The results of experiment are shown in figure 7.

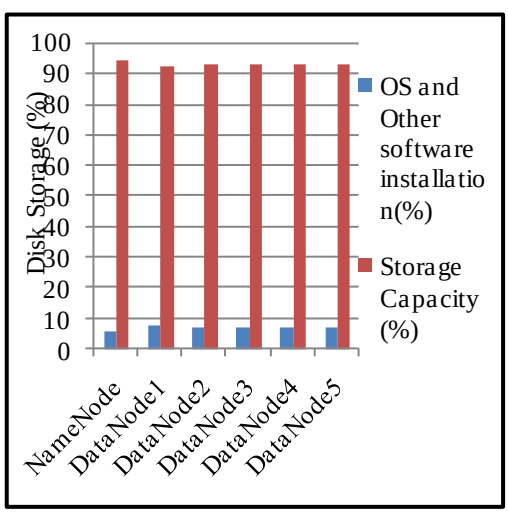

Figure 6. Average Storage Capacity of PC cluster based Storage Server

According to the figure 6, the storage space percentage of six nodes in the cluster are 93.92%, 92.46%, 92.76%, 92.87%, 92.77% and 92.60% respectively. The OS and other software installation is 6.08%, 7.54%,7.24%,7.13%,7.23% and 7.40% respectively. Therefore, the OS and other software installation used about 10% of the disk storage and 90% of the storage capacity can be used in PC cluster based storage server.

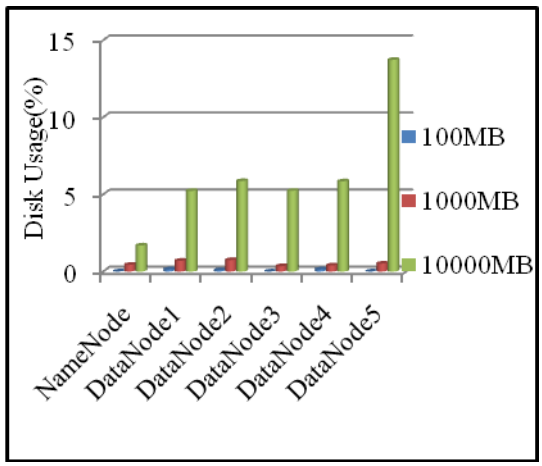

Figure 7. Average Disk Usage of PC cluster based Storage Server

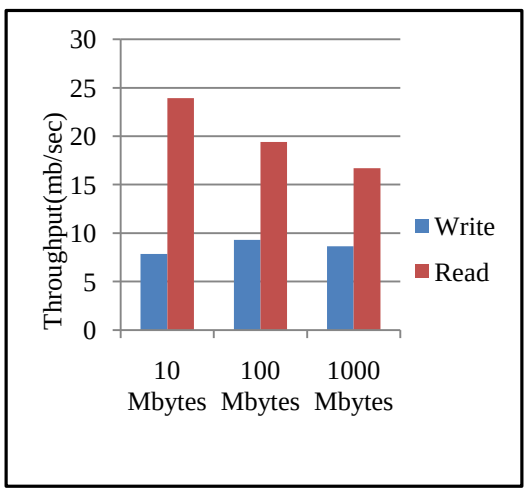

Figure 8. Throughput of the File Read and Write

According to the figure 7, the disk usage percentage of six nodes in the cluster are 0.02,0.14%, 0.09%, 0.01%, 0.14%, and 0.01% for 100MB of data files, 0.44%, 0.7%, 0.75%, 0.35%, 0.4% and 0.52% for 1000MB of data files and 1.69%, 5.23%, 5.87%, 5.24%, 5.85%

and 13.73% for 10000MB of data files respectively. In figure 8 shows the throughput when 10 Mbytes, 100 Mbytes and 1000 Mbytes of file read and write by using TestDFSIO benchmark. According to the result of figure 8, throughput of file read is higher than throughput of file write.

6. Conclusions

In this paper, two parts have been described such as (i) design and architecture of PC cluster based cloud storage system (ii) to be increased performance of I/O traffic on the target architecture. It can be used to store a large amount of data. As can be seen from experimental results, the storage can be utilized more than 90% of storage space. As future works, the greatest challenge of our system is to improve fault tolerant and security.

References

[1] C.Yong, M.Lionel and Y.Mingyao, "CoStore: A Storage Cluster Architecture Using Network Attached Storage Devices", *In Proceedings of the Ninth International Conference on Parallel and Distributed Systems*, 2002.

[2] D.Bo, Q.Jie , Z.Qinghua , Z.Xiao , L.Jingwei and L.Ying, "A Novel Approach to Improving the Efficiency of Storing and Accessing Small Files on Hadoop: a Case Study by PowerPoint Files", *In Proceedings of IEEE International Conference on Services Computing* , 2010.

[3] H.Qinlu , L. Zhanhuai, Z.Xiao, Study on Cloud Storage System based on Distributed Storage Systems", *In Proceedings of International Conference on Computational and Information Sciences*, 2010.

[4] L.Lin, L.Xuemin, J.Hong, Z.Yifeng, " AMP: An Affinity-based Metadata Prefetching Scheme in Large-Scale Distributed Storage Systems", *Technical Report*, Novermber, 2007.

[5] W. Jiyu, Z. Jianlin, L. Zhije, J. Jiehui, "Recent Advances in Cloud Storage". *In Proceedings of the Third International Symposium on Computer Science and Computational Technology*, 2010, pages 151-154.

[6] X.Ke, S.Meina, Z.Xiaoqi and S.Junde, "A Cloud Computing Platform Based on P2P". *In Proceedings of IEEE*,2009.