Cluster-based Distributed File System on Private Cloud Data Center

Cho Cho Khaing and Thinn Thu Naing
University of Computer Studies, Yangon
chokhaing.cho@gmail.com

Abstract

The widespread popularity of Cloud computing as a preferred platform for the deployment of web applications has resulted in an enormous number of applications moving to the cloud, and the huge success of cloud service providers. The data center storage management plays a vital role in cloud computing environments. This system presents a development framework for cluster-based distributed file system is also called (CDFS) which facilitates massive data processing on cluster storage of low cost computers. Moreover this system introduces the enhancement on MapReduce model to improve the system throughput and the scalability to keep on working with the amount of existing physical storage capacity when the number of files increase. It reduces the storage space of the cloud server using Huffman Compression method for text-based files. This paper also evaluates the system reliability on a cluster that is deployed for cloud based data storage.

.

Keywords: CDFS; MapReduce; PC cluster-based data storage; Huffman coding

1. Introduction

Information Technology (IT) organizations worldwide are dealing with the tremendous growth of data. With the growth of capacity comes the complexity of managing the storage for that data. The data growth is coming from a wealth of data-intensive applications (e.g., business analytics), expanding use of high-performance computing (e.g., financial services and life sciences), collaboration and Web 2.0 applications, and content-rich data (e.g., digital images or video). In this data-intensive environment, IT managers need to optimize the capacity and performance of storage systems while working to reduce complexity and lower costs.

In addition to the continued growth in capacity, the accelerated use of virtual servers and desktops is rapidly altering the storage landscape. IT organizations worldwide are turning to virtualized environments to improve data center flexibility and scalability. This in turn drives implementation of networked storage solutions, which can create new pressures on storage performance as I/Os that were previously more distributed are aggregated into a smaller number of host interconnects. There are also implications for organizations' data protection processes and architectures to ensure that every virtual server is protected and that the storage has the same flexibility and resiliency as the virtualized server environment.

When a dataset outgrows the storage capacity of a single physical machine, it becomes necessary to partition it across a number of separate machines. File systems that manage the storage across a network of machines are called distributed file systems. Since they are network-based, all the complications of network programming kick in, thus making distributed file systems more complex than regular disk file systems. For example, one of the biggest challenges is making the file system tolerate node failure without suffering data loss. Hadoop comes with a distributed file system called HDFS, which stands for Hadoop Distributed File system. Hadoop is a very powerful platform for processing and coordinating the movement of data across various architectural components. Its only drawback is that the primary computing model is MapReduce.

This paper presents the efficient storage system on private cloud. This approach has been designed to use over virtualized storage system. The idea of the proposed system is to exploit the use of virtualization technology to avoid unnecessary storage purchases and reduce storage space for addressing large volumes of data handling problem.

The rest of this paper is organized as follows: Section 2 describes the related work. In section 3, cloud computing is described. Section 4 presents System deployed architecture and detailed architecture of CDFS. Experimental results are described in section 5. Finally section 6 concludes the paper.

2. Related Work

Several implementations of distribute file system already existed. For example, Hadoop

Distributed File System is an open source distributed parallel fault tolerant file system. It is designed to reliably store very large files across a large-scale cluster (HDFS) [1]. Google File System is a proprietary DFS for its own network level search engine. It is designed to provide efficient, reliable access to data using large clusters of commodity hardware (GFS) [8]. RedHat Global File System is an open-standard based system with great modularity and compatibility with interconnects, networking components and storage hardware. (RGFS) [7]. Amazon S3 also is one of the commercial services for it.

Google has developed its infrastructure technologies for cloud computing in recent years, including GoogleFile System (GFS) [4], MapReduce [5] and Bigtable [2]. GFS is a scalable distributed file system, which emphasizes fault tolerance since it is designed to run on economically scalable but inevitably unreliable (due to its sheer scale) commodity hardware, and delivers high performance service to a large number of clients. Bigtable is a distributed storage system based on GFS for structured data management. It provides a huge three-dimensional mapping abstraction to applications, and has been successfully deployed in many Google products. MapReduce is a programming model with associated implementation for massive data processing. MapReduce provides an abstraction by defining a "mapper" and a "reducer". The "mapper" is applied to every input key/value pair to generate an arbitrary number of intermediate key/value pairs. The "reducer" is applied to all values associated with the same intermediate key to generate output key/value pairs. MapReduce is an easy-to-use programming model, and has sufficient expression capability to support many real world algorithms and tasks. Apache Hadoop Core is a software platform for processing massive amounts of data efficiently across large clusters of commodity hardware [6]. Hadoop Core consists of the Hadoop Distributed File System (HDFS) and an implementation of the MapReduce model.

3. Cloud Computing

The Cloud has become a new vehicle for delivering resources such as computing and storage to customers on demand. Cloud computing has been introduced by computing vendors influenced by business requirements and environment to compute enterprise applications and support information processing and data mining. And recently, there is a growing interest in and move of clouds into science. Although so far cloud computing still has no uniform definition, its idea is exactly what distributed systems means. By the means of virtualization, standardization and automation, cloud architecture integrates software and hardware resources in cloud and deliveries the cloud services to users through network. The typical cloud architecture is divided into three basic levels, including IaaS (infrastructure as a service), PaaS (platform as a service) and SaaS (Software as a service).

As a new raised computing model, cloud computing has gained widespread concerns. Especially in industry, many companies have heavily invested to deploy their own cloud computing systems. Currently, IaaS providers include Amazon's EC2, PaaS providers are Google's Google App Engine, MicroSoft's Azure and Apache Hadoop, and SaaS providers' representative is Salesforce. Recently, with the mature of cloud, it has been using for academic research projects, and the most famous projects are Scientific Cloud and OpenNebula project [3].

Cloud storage is one of the services which provide storage resource and service based on the remote servers on cloud computing. Cloud storage will be able to provide storage service at a lower cost and more reliability. The advantage of cloud storage is to enable users at any time access data.

4. Overview Architecture of CDFS

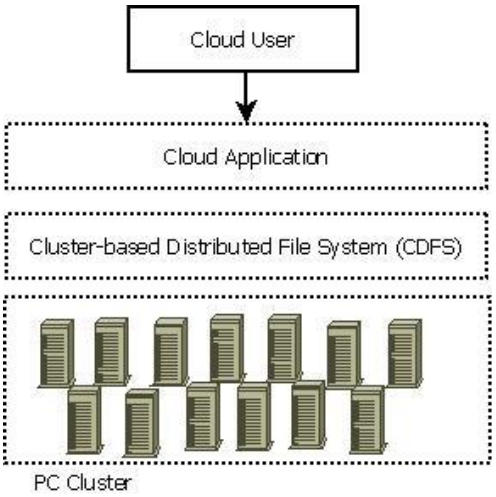

Figure 1. Overview framework architecture of CDFS

The overall framework architecture of the system in Figure 1, which consists of three layers. There are PC cluster layer, Cluster-based Distributed File System layer and data processing application layer. The PC cluster layer provides the hardware and storage devices for large scale data processing. The second layer is the main focus of proposed system, Cluster-based File System consisting of HDFS-based file system, Huffman

compression and MapReduce programming model. The application layer provides the services to users, where users can develop their own applications, such as user's data, files and so on.

4.1. System Deployed Architecture

In PC cluster, each individual machine of a cluster is referred to as a node. Our system is based on master-slave architecture and consists of one master node and many slave nodes in order to make them work as a single machine. Master node has two network cards: one is connected to the front-end infrastructure using giga switch and the other is connected to the slave nodes. Master node and slave nodes are shown in Figure 2. All of these nodes are installed Ubuntu Server 10.4 (64 bits) Linux distribution operating system which is an open source operating system built around the Linux kernel. This system is ongoing research which has been implementing Cloud-based File System (CDFS) in both single-node and multi-node cluster. Now we are testing this file system in multi-node cluster. The experiment contains several nodes among them. Some nodes are master nodes and others are slaves.

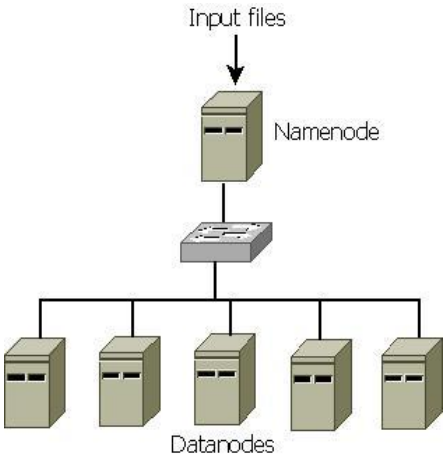

Figure 2. Physical configuration of CDFS

4.2. CDFS Architecture

CDFS (Cloud-based Distributed File System) is used to store data files in the collection of the computer nodes. In this system, it has name node and data nodes. Namenode runs on the server node that maintains the metadata of CDFS which contains the information of blocks, the current location of blocks, etc. Master node provides the monitoring of all slave nodes' states in system. Files are divided into multiple chunks and distributed across slave nodes. The slave nodes are responsible to store data. The large number of nodes is used in this system because the probability of some nodes can be failed.

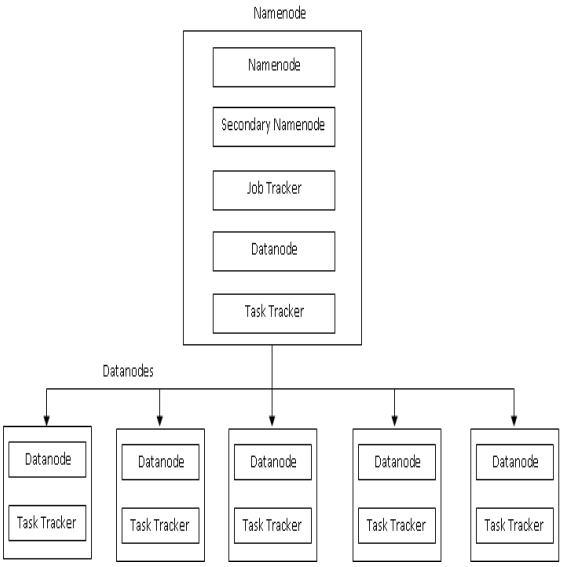

Figure 3. Detailed Architecture of CDFS

PC cluster has lower implementation cost and store large amount of data but major bottleneck is I/O operation for data processing. Due to I/O bottleneck problem, our design achieves a single I/O space for all of the blocks of data in the cluster. This allows any node to remotely access any I/O peripheral or disk devices without the knowledge of their physical location.

In this system, the master node or *Namenode* consists of a namenode, secondarynamenode, jobtracker, datanode and tasktracker. A slave or *compute node (Datanode)* consists of a datanode and tasktracker. Figure 3 shows the logical architecture design of CDFS.

4.2.1. Namenode and Secondary Namenode

Namenode works as the global controller and maintains the information of whole CDFS system. The namenode manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree. Secondary namenode does not act as a namenode. Its main role is to periodically merge the namespace image with the edit log to prevent the edit log from becoming too large. It keeps a copy of the merged name-space image, which can be used in the event of the namenode failing. However, the state of the secondary namenode lags that of the primary, so in the event of total failure of the primary data, loss is almost guaranteed.

4.2.2. Datanode

Data node stores the physical storage of the file. They store and retrieve blocks when they are told to (by clients or the namenode), and they report back to the namenode periodically with lists of blocks that they are storing.

4.2.3. Jobtracker and Tasktracker

The jobtracker coordinates all the jobs run on the system by scheduling tasks to run on tasktrackers. Tasktrackers run tasks and send progress reports to the jobtracker, which keeps a record of the overall progress of each job. If a task fails, the jobtracker can reschedule it on a different tasktracker.

4.3. MapReduce

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable manner.A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks,monitoring them and re-executes the failed tasks.

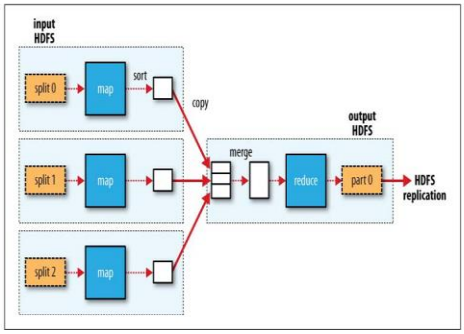

Figure 4. MapReduce data flow with a single reduce task

From a scheduling perspective, MapReduce reflects a framework for processing highly distributable problems across huge datasets by utilizing a large number of computers (nodes), collectively referred to as a cluster (homogeneous or heterogeneous HW).Computational processing can occur on data stored either in a file system (unstructured) or in a database (structured). The 2 steps are (see Figure 4):

1. The Map step: The master node receives the input, partitions the data into smaller sub-problems, and distributes the sub-parts to worker nodes. A worker node may partition the problem again, which may lead to a multi-level tree structure. The worker node processes the smaller problem, and passes the answer/data back to the master node.

2. The Reduce step: The master node then collects the answers/data from all the sub-problem providers and combines the data to form the output (the answer to the problem originally to be solved).

4.4. Compression

A commonly used method for data compression is Huffman coding. The method starts by building a list of all the alphabet symbols in descending order of their probabilities. It then constructs a tree, with a symbol at every leaf, from the bottom up. This is done in steps, where at each step the two symbols with smallest probabilities are selected, added to the top of the partial tree, deleted from the list, and replaced with an auxiliary symbol representing both of them. When the list is reduced to just one auxiliary symbol (representing the entire alphabet), the tree is complete. The tree is then traversed to determine the codes of the symbols.

4.5. Eucalyptus components on Private Cloud

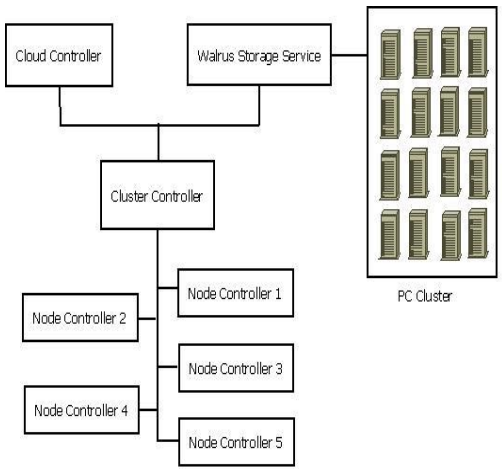

Figure 5. Eucalyptus components architecture

The main focus of the work is to build storage server using pc cluster for academic based private cloud. Cloud infrastructure is implemented by using Ubuntu Enterprise Cloud (UEC) architecture. The UEC is powered by Eucalyptus, an open source implementation for the emerging standard of the Amazon EC2 API. As shown above in Figure 5, Eucalyptus consists of four parts, namely, CLC (Cloud Controller), Walrus, CC (Cluster Controller) and NC (Node Controller).

Node Controller controls the execution, inspection, and terminating of VM instances on the host where it runs, and reports to the CC regularly with the information about virtual machine and host.

Cluster Controller gathers information about VM and schedules VM execution on specifically node controllers.

Storage Controller (Walrus) is a put/get storage service that implements Amazon's S3 interface, providing a mechanism for storing and accessing virtual machine images and user data.

Cloud Controller is the entry-point into the cloud for users and administrators. It's the key management module of the platform, responsible for the management of the entire IaaS.

5. Experiments on CDFS

Depending on the nature of the application there are various criteria to measure the performance of a compression algorithm. When measuring the performance the main concern would be the space efficiency. This system is tested for 30 files on text-based files. There are three types of file: text files (.txt), java code files (.java) and database script files (.sql). The following figure 6,7 and 8 show the comparison of file compressoin with different file sizes on HDFS and CDFS.

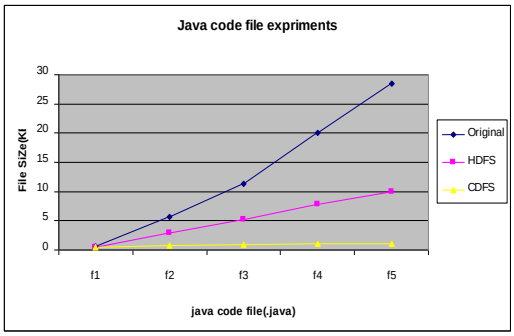

Figure 6. Experimental results for java code files(.java)

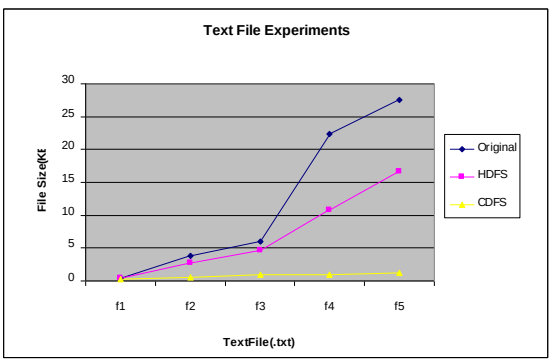

Figure 7. Experimental results for text files(.txt)

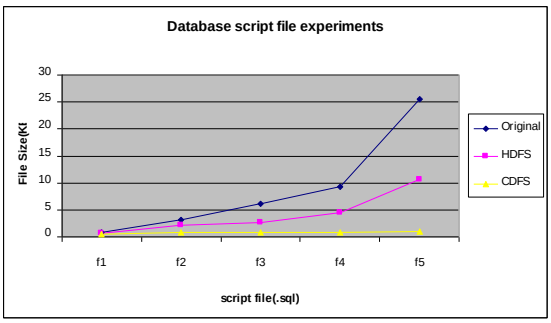

Figure 8. Experimental results for database script files (.sql)

In the above figures, x axis describes the number of files and y axis describes the file size (KB) of the corresponding files.

The following equation is the measurement used to evaluate the performance of the system. Saving percentage calculates the shrinkage of the source file as a percentage.

$$SP = \frac{S_{bc} - S_{ac}}{S_{bc}} \times 100$$

SP= Saving Percentage
S_{bc}*= *Size before compression
S_{ac}*= *Size after compression

Figure 9 shows the comparison of average saving percentage between HDFS and CDFS.

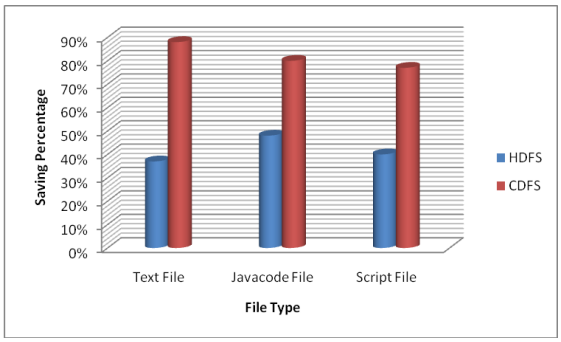

Figure 9. Average saving percentage comparison of HDFS and CDFS on different file types

5.1. Reliability of CDFS

This system configures a cluster that has five data nodes. For example, the replication factor of this cluster is 3. Furthermore, suppose that the design of the cluster is such that at least three nodes including name node are required to operate. This means that the nodes are reliability-wise in a k-out-of-n configuration, where k = 3 and n = 5. More specifically, they are in a 3-out-of-5 configuration.

The simplest case of nodes in a k-out-of-n configuration is when the nodes are independent and identical. In other words, all the nodes have the same failure distribution and whenever a failure

occurs, the remaining nodes are not affected. In this case, the reliability of the system with such a configuration can be evaluated using the binomial distribution:

$$R_s(k,n,R) = \sum_{r=k}^{n} \binom{n}{r} R^r (1-R)^{n-r}$$

Where:

n = the total number of units in parallel.

k = the minimum number of units required for system success.

R = the reliability of each unit.

According to the above equation, the reliability of the cluster can calculate with various values of k. The following table1 and Figure 10 show the reliability of the cluster with various values of k.

Table 1. Reliability for a k-out-of-5 system with different k values

k	Reliability
1	1
2	0.9984
3	0.9734
4	0.8352
5	0.4437

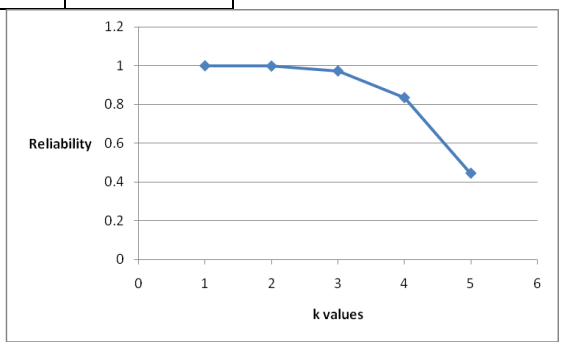

Figure 10. Reliability of a k-out-of-5 configuration with different k values.

6. Conclusion

This paper presents a development framework for cluster-based distributed file system (CDFS) which facilitates massive data processing on cluster storage of low cost computers. The main focus of CDFS is to reduce the storage space of the cloud server using Huffman Compression method and MapReduce programming model for text-based files. When the amount of data is growing in existing storage capacity, the system improves the storage utilization efficiently. According to evaluation results of the system, this system can reduce the total amount of storage space on the amount of existing physical storage capacity even when the number of files are increased. Moreover, this paper also evaluates the system reliability on a cluster that is deployed for cloud based data storage with different number of data nodes. It also provides the flexibility of large data storage on the private cloud system using low cost computers.

References

[1]Apache Hadoop Documentation http://hadoop. Apache . org/core

[2]F.Chang, J.Dean, S.Ghemawat, W.C.Hsieh, D. A.Wallach, M.Burrows, T.Chandra, A.Fikes, and R. E.Gruber. “Bigtable: a distributed storage system for structured data”. In OSDI ’06: Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, pages 15–15, 2006.

[3]G.DeCandia,D.Hastorun,M.Jampani,G.Kakulapati, A.Lakshma,A.Pilchin, S.Sivasubramanian, P. Vosshall, W.V..Dynamo, “Amazon's highly available key-value store”,Proceeding SOSP '07 Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles ACM, Volume 41 Issue 6, New York, NY, USA 2007 .

[4]G.Sanjay, G.Howard, and L.Shun-Tak. “The google file system”. In Proceedings of the 17th ACM Symposium on Operating Systems Principles, pages 29–43, 2003.

[5]J.Dean and S.Ghemawat. “Mapreduce: Simplified data processing on large clusters”. In OSDI ’04: Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation, pages 137–150,2004.

[6]Map/Reduce Tutorial (04/09/2009). Retrieved May 21, 2009 from http://hadoop.apache.org/core/docs/ current/ mapred tutorial.html

[7]Red Hat Global File System, White Paper, http://www.redhat.com/whitepapers/rha/gfs/GFS_INS0032US.pdf.

[8]S.Ghemawat, H.Gobioff, S.T.Leung, “The Google file system”, ACM SIGOPS Operating Systems Review, Volume 37 , Issue 5, pp. 29-43, December, 2003.