A Cloud Computing Platform Based on P2P

Ke Xu [1], Meina Song [2], Xiaoqi Zhang [3], Junde Song[4]
School of Computer
Beijing University of Posts and Telecommunications
Beijing 100876, China
permit@263.net [1]; mnsong@bupt.edu.cn [2]; alphazxq@gmail.com [3]; jdsong@bupt.edu.cn[4]

Abstract

In recent years, the technology of cloud computing has been widely applied in e-business, e-education and etc.. Cloud computing platform is a set of Scalable large-scale data server clusters, it provide computing and storage services to customers. The cloud storage is a relatively basic and widely applied service which can provide users with stable, massive data storage space. Our research shows that the architecture of current Cloud Computing System is central structured one, all the data nodes must be indexed by a master server which may become bottle neck of the system. In this paper, we propose a new cloud storage architecture based on P2P and design a prototype system. The system based on the new architecture has better scalability and fault tolerance.

Keywords: cloud computing, P2P, storage

1. Introduction

A cloud computing platform dynamically provisions, configures, reconfigures, and provisions servers as needed. Servers in the cloud can be physical machines or virtual machines. Advanced clouds typically include other computing resources such as storage area networks (SANs), network equipment, firewall and other security devices. [1] This paper will focus on the storage service from cloud.

Some typical cloud systems, such as GFS of Google[2], Blue Cloud of IBM[1], Elastic Cloud of Amazon[3], have a similar architecture for storage. In the system architecture, there is a central entity to index or manage the distributed data storage entities. It is effective to simplify the design and maintenance of the system by a central managed architecture, but the central entity may become a bottleneck if the visiting to it is very frequent. Although systems in practice have used some technique as backup recovery to avoid the probably disaster from the central bottle neck, the flaw come from the architecture has not resolved essentially.

In this paper, we propose a cloud computing 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 has better scalability and fault tolerance.

Rest of the paper is organized as follows, in section 2, we will introduce some related work about cloud storage system and P2P storage system. In section 3 of this paper, we describe a typical scenario to explain the architecture of our proposed cloud computing storage environment. In section 4, there is an introduction on our prototype about the P2P cloud system. Section 5 is conclusion and proposal for future work.

This work is financially supported by the Program for New Century Excellent Talents in University No.NCET-08-0738 ; and the Innovation Technology Star Program of Beijing under Grant No.2007A045; and the National Key project of Scientific and Technical Supporting Programs of China under Grant No. 2008BAH24B04.

2. Related Works

In this section, we will introduce some related work about cloud system and P2P products for storage.

2.1 Google File System

The first to give prominence to the term cloud computing (and maybe to coin it) was Google's CEO Eric Schmidt, in late 2006[4]. Google Inc. has a proprietary cloud computing platform[5] which was first developed for the most important application of Google search service[6] and now has extended to other applications. Google cloud computing infrastructure has four systems which are independent of and closely linked to each other. They are Google File System for distributed file storage, MapReduce program model for parallel Google applications[7], Chubby for distributed lock mechanism[8] and BigTable for Google large-scale distributed database[9].

Figure 1. shows the architecture of Google file system. A GFS cluster consists of a single master and multiple chunk servers and is accessed by multiple clients. Chunk servers store chunks on local disks as Linux files and read or write chunk data specified by a chunk handle and byte range. The master maintains all file system metadata. This includes the namespace, access control information, the mapping from files to chunks, and the current locations of chunks. When a client wants to visit some data on a chunk server, it will first send a request to the Master, and the master then replies with the corresponding chunk handle and locations of the replicas. The client then sends a request to one of the replicas and fetch the data wanted. [2]

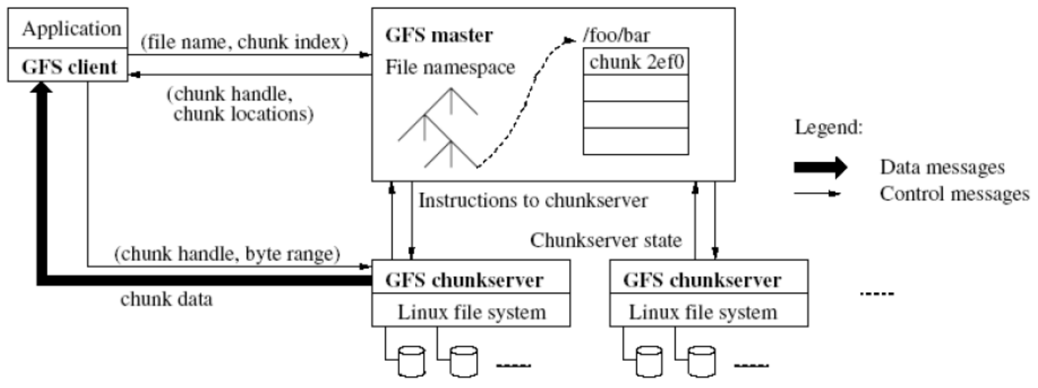


Figure 1 Architecture of Google File System[2]

The GFS above is actually a central indexed distributed storage system, GFS master work as an index server which can provide the global information about each chunk server for clients. The flaw of central index architecture is that the GFS master may become bottle neck of the system since all the request to the target data chunk must be originated from the index server which burdens the master.

2.2 P2P Storage System

The distributed P2P network indexed by DHT arithmetic can resolve the problems of bottle neck come from central index system. Since the management is distributed equality to every peers in the network, there is no bottle neck any more, but the new problem is how to keep the consistency of the replica when read/write. Some P2P systems for distributed storage have been developed now, such as Ivy[10], Eliot[11], Oasis[12], OM[13], Sigma[14], etc.. They keep the replica consistency in different way and index the data resource by DHT. In the following section, we will propose a cloud storage system based on

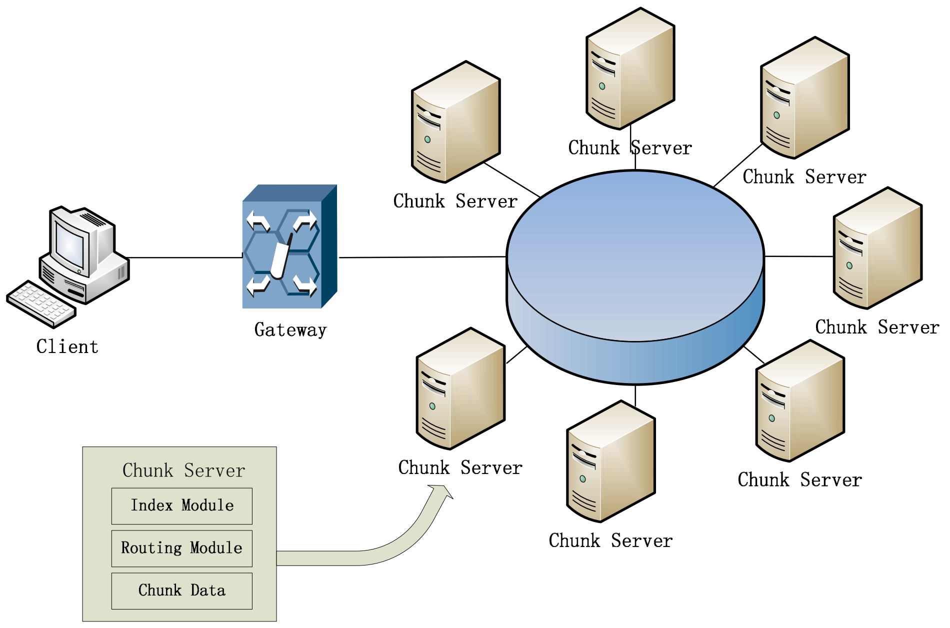
P2P which can keep the consistency with an innovative method.

3. Cloud Based on P2P

3.1 Architecture

We design a new system of P2P storage for cloud platform, which can take advantage of the P2P distribute architecture and do well in concurrent update.

Following figure 2 shows the architecture of the system:



Figrue 2 Cloud Storage Based on P2P

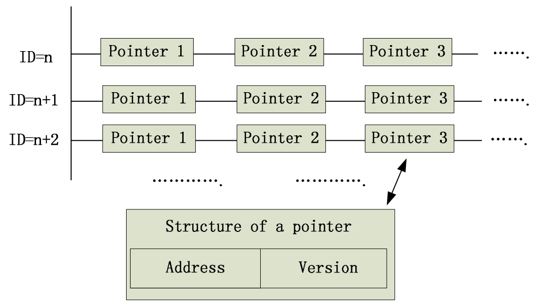
Roles involved in our architecture can be defined as follows and illustrated as below.

Client App: the client application which wants to get the data from the platform;

Gateway: the entity which can transfer the request or response between the Client App with the network and can lead the request to the nearest node in the network.

Chunk Server: the entity which is served as the data resource node and P2P node. Different with the function of pure data storage in GFS, the chunk server here has three function modules with separated interfaces. As shown in the figure above: Index Module, take charge of part of the global resource index which is assigned by DHT arithmetic such as Chord, Pastry and so on. Route Module, pass a lookup request by a next hop routing table which is also assigned by DHT. Data Module, provide the data resource stored in the local machine.

In the index module, as shown in the figure 3, there is a chain containing the data index information pointers to all of the data blocks with the same name ID will be linked in a sub chain. A pointer contains the address of a data block and the update version number of that block.



Figrue 3 Index Module Details

3.2 Typical Workflow

In this section, we will present the system working flow with a typical scenario.

At first, before a Client App can do its work, data blocks and the corresponding replica should be uploaded to the Chunk Servers. How to select the chunk servers for storage is the same with the traditional cloud computing platform. When a Client lookup data block, the work flow is different, as below:

1. Client App sends a request for a data block with logic identifier to Gateway;
2. Gateway analysis the request, parsing the identifier of the data block in the request, such as logic address, and change it to 128 bits logic ID by DHT algorithm which can be recognized by chunk server P2P network;
3. Gateway constructs a P2P search request data package including the logic ID, and sends the request to the chunk server P2P network;
4. The P2P search request package routed among the chunk servers following the P2P search protocol such as Chord[15], Can[16], Pastry[17], Tapestry[18] and so on. The chunk servers now act as a routing nodes of P2P and the routing interface will be taken used of;
5. The request reaches the server which contain the index information of the logic ID in searching;
6. The index includes all the pointers of the data replica with the same ID. The chunk server now acts as an index server and the index function interface will play its role. The chunk server will select a latest pointer by its version number, if there are more than one candidates, the server should select a nearest one by comparing the IP address of the client App and the data resource container, then return the best address to the client;
7. When the Client App gets the best address, it will then send its request to the address of the chunk server which contains the data block. Now the chunk server acts as a data provider as the traditional cloud storage platform does.

3.3 Replica Control

As we all know, when the cloud platform provides storage service to a user, a frequently met problem is write/read for mutual exclusion. In a central managed system as GFS, this can be resolved by lock mechanism, such as so called "lease and mutation order"[2]. But in distributed environment, it will be more complicated. In this section, we will discuss about the replica consistency control.

Here is an example for writing consistency in our P2P cloud storage system:

1. Client finds the chunk server node which contains the index information of the target data block. (We will call this chunk server index node later when it provides index function.)
2. Client tell the index node that it will do write operation.
3. The index node check the chain of replicas state to see whether there is another writing being processed. The state of the chain is lock or unlock.

If unlock, the index node will allow that write operation by returning the chunk server address of the latest version(if the candidate is multiple, the index node should select a nearest one to the client by comparing the IP addresses), and change the state to lock;

If lock, the index node will queue the requests until the state comeback unlock, the first write request in the queue then can be practiced.

4. Client gets the address of the newest version, connects that chunk server, write the update data to the block, then sends message to the index node to notify that the write operation has finished.
5. When the index node receives the finish message, the version number of the pointer to the just modified block will be increased. Then a procedure of consistency update to all the replicas will be start. In the procedure,

6. After a replica server finishes update, it will send an update response messages to the index node and the version in its pointer in the chain will be increased by the index node
7. When all the pointers in the chain have updated to the newest version, the state of the chain will be set unlock. If the period of lock state is overtime, the state of the chain will reset unlock by force. When the chain has set unlock, any delayed update response messages will be discarded and the version of the corresponding pointer in the chain will remain the old. This prevent the system from suspending if there is something wrong with some replica servers when update. Since the version of the data block on the delayed server is old, no client will visit that server for the old data later unless its update is confirmed.

From the example above, we can see that the consistency is controlled by the distributed index node. That is because all the data block with the same ID will be indexed at a same index node, the distributed separate chunk servers can be managed as a unit here. The replica update procedure is originated by the corresponding index node when any new write operation has been done. The procedure can also start when the index node finds version confliction in a same chain during its periodically check.

4. Prototype System

We developed a prototype system based on the proposed architecture of this paper. The design of the system is as follow: We deploy 100 PCs as data chunk servers with the operating system windows XP. After installed the chunk server side software, the PC nodes become Chunk servers of the cloud storage system. The Chunk servers in our prototype are organized by Chord arithmetic of P2P. Each node forwards query at least halfway along distance remaining to the target. With high probability, the number of nodes that must be contacted to find a successor in a N-node network is O(log N). Our new architecture provides a different mean to manage the chunks without any central master, the cost is more delay when lookup a node. The figure 4 shows the average lookup delay of our system.

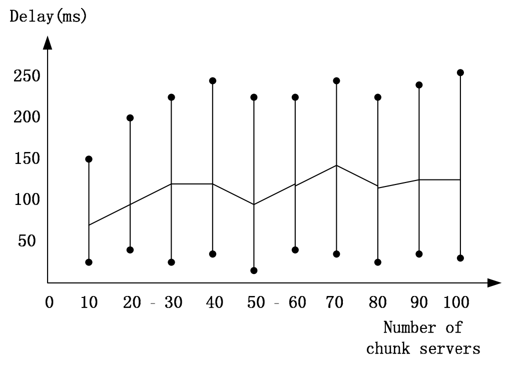


Figure 4 **Average Lookup Delay**

The deployed chunk servers is from 10 to 100, and each client issues 15 lookups for randomly chosen data blocks one-by-one. Suppose that the throughput of the central master cloud storage system is xMB/S, and each block size is yMB and the average lookup delay in our system is T second, so the new throughput of the P2P cloud system can be calculated as: $y/((y/x)+T)$. From Figure4 we can see that the median latency ranges about 100ms. The block size of our system is 64M. If the throughput of the central system is 50M/S, then the corresponding in P2P system is 46M/S

5. Conclusion and Future Work

In this paper, we propose a new architecture of cloud computing system based on P2P protocol, which resolve the problems of bottle neck come from central structure. In the future work, we will do some optimization about the throughput of the system by the technique such as pipelining read or write.

Reference

[1]Boss G, Malladi P, Quan D, Legregni L, Hall H. Cloud computing. IBM White Paper, 2007. http://download.boulder.ibm.com/ibmdl/pub/software/dw/wes/hipods/Cloud_computing_wp_final_8Oct.pdf

[2]Ghemawat S, Gobioff H, Leung ST. The Google file system. In: Proc. of the 19th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2003. 29−43.

[3]Amazon. Amazon elastic compute cloud (Amazon EC2). 2009. http://aws.amazon.com/ec2/

[4]Francesco Maria Aymerich, Gianni Fenu, Simone Surcis. An Approach to a Cloud Computing Network.

[5]Barroso LA, Dean J, Hölzle U. Web search for a planet: The Google cluster architecture. IEEE Micro, 2003,23(2):22−28.

[6]Brin S, Page L. The anatomy of a large-scale hypertextual Web search engine. Computer Networks, 1998,30(1-7):107−117.

[7]Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. In: Proc. of the 6th Symp. on Operating System Design and Implementation. Berkeley: USENIX Association, 2004. 137−150.

[8]Burrows M. The chubby lock service for loosely-coupled distributed systems. In: Proc. of the 7th USENIX Symp. on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2006. 335−350.

[9]Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE. Bigtable: A distributed storage system for structured data. In: Proc. of the 7th USENIX Symp. on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2006. 205−218.

[10]Ivy:A. Muthitacharoen, R. Morris, T. Gil, and B. Chen, "Ivy: A Read/write Peer-to-peer File System," in Proc. of the Symposium on Operating Systems Design and Implementation (OSDI), 2002.

[11]Eliot:C. Stein, M. Tucker, and M. Seltzer, "Building a Reliable Mutable File System on Peer-to-peer Storage," in Proc. of 21st IEEE Symposium on Reliable Distributed Systems (WRP2PDS), 2002.

[12]Oasis:M. Rodrig, and A. Lamarca, "Decentralized Weighted Voting for P2P Data Management," in Proc. of the 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access, pp. 85–92, 2003.

[13]OM:H. Yu. and A. Vahdat, "Consistent and Automatic Replica Regeneration," in Proc. of First Symposium on Networked Systems Design and Implementation (NSDI '04), 2004.

[14]Sigma:S. Lin, Q. Lian, M. Chen, and Z. Zhang, "A practical distributed mutual exclusion protocol in dynamic peer-to- peer systems," in Proc. of 3rd International Workshop on Peer-to-Peer Systems (IPTPS'04), 2004.

[15]I.Stoica, R.Morris, D.Karger, F.Kaashoek, and H.Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for Internet applications", IEEE/ACM Transaction on Networking, vol.11, no.1, Feb.2003, pp.17-32.

[16]Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp and Scott Shenker, "A Scalable Content-Addressable Network". In: P rocessing of ACMSIGCOMM , 2001, (8) : 1612172.

[17]Antony Rowstron and Peter Druschel, "Pastry: Scalable,decentralized object location and routing for large-scale peerto-peer systems", IFIP/ACM International Conference onDistributed Systems Platforms(Middleware 2001).

[18]Ben Y. Zhao, John Kubiatowicz, and Anthony D. Joseph,"Tapestry: An Infrastructure for Fault-tolerant Wide-areaLocation and Routing", Technical Report No. UCB/CSD -01-1141, University of California Berkeley.