Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 12, December 2014, pg.549 – 554

RESEARCH ARTICLE

Distributed File Storage and Sharing using P2P Network in Cloud

Sapana V. Kapadnis
Computer Department, MET BKC Adgaon, Nashik, Savitribai Phule Pune University, Maharashtra India
sapanakapadnis@gmail.com

Abstract- Cloud Computing is the most important paradigm in IT industry in which most of the application follows Client/Server architecture. Client/Server architecture is costly in terms of server bandwidth and storage. Too many requests from the clients may lead to congestion, which rarely takes place in P2P network. Overload can lead to breaking-down of servers. In peer-to-peer, the total bandwidth of the network increases as the number of peers increase. There is need to provision file system on demand anytime & anywhere. File system which may contain any type of data i.e. video, images, virtual machine image file. These files are large in size as compare to other files. By using this concept we proposed file storage and sharing architecture to improve provisioning time, availability, scalability, speed while eliminating various bottlenecks such as Storage Size, Network Speed, Jitter, and count of Clients etc. Like socially connected people have high chance of seeing same video, in cloud also logically connected nodes would have high chance of using virtual machine files. A network coding based storage strategy is put forward, which can improve the utilization of each peer's limited storage space. Moreover, with the same storage space, it can increase the diversity of stored data, which can improve the efficiency of data sharing in cloud.

Keywords: Cloud computing, client/server architecture, virtual machine files

I. Introduction

Cloud computing is typically defined as a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications. Cloud computing focuses on maximizing the effectiveness of the shared resources. Cloud resources are usually not only shared by multiple users but are also dynamically reallocated per demand. This can work for allocating resources to users. Cloud computing is a general term for anything that involves delivering hosted services over the Internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). Cloud computing is probably the most cost efficient method to use, maintain and upgrade. Storing information in the cloud gives you almost unlimited storage capacity. Since all your data is stored in the cloud, backing it up and restoring the same is relatively much easier than storing the same on a physical device. In the cloud, software integration is usually something that occurs automatically.

Client/server architecture is network architecture with two components involved in it as client and server. Servers are responsible for managing server files, network traffic etc. Client relies on server for resources such as files, devices. Client is service requester. Server host does not have more resources than client which leads to centralized computing. Memory requirement, storage requirement and computing power of server scaled properly to serve workload of server. All files are stored at centralized server due to that file management becomes the task of server. Server can play differ role for different clients. When multiple client requests come at server it may get overloaded, forming traffic congestion. If centralized server fails files stored at centralized server may get deleted. While using client/server architecture it has to face multiple problems. So P2P system is used in most if the cloud application for storing and sharing files which are large in a size like video files, files with virtual machines.

Many problems of client/server architecture are overcome by using concept of distributed file storage and sharing using P2P network in cloud. Peer-to-peer networking is a distributed application architecture which partitions tasks or workloads between peers. Equal privileges are provided to Peers. They are said to form a peer-to-peer network of nodes. Peers are both suppliers and consumers of resources, in contrast to the traditional client server model in which the consumption and supply of resources is divided. Emerging collaborative P2P systems are going beyond the era of peers doing similar things while sharing resources, and are looking for diverse peers that can bring in unique resources and capabilities to a virtual community thereby empowering it to engage in greater tasks beyond those that can be accomplished by individual peers, yet that are beneficial to all the peers. In P2P networks, clients both provide and use resources. This means that unlike client-server systems, the content serving capacity of peer-to-peer networks can actually increase as more users begin to access the content. This property is one of the major advantages of using P2P networks because it makes the setup and running costs very small for the original content distributor. In this system chunk of file is shared among the users which can be videos or virtual machine files. A peer can access chunk or small prefix store at other peer. It reduces the workload of server. It is distributed protocol as to store and track data.

II. Literature Survey

Peer-to-peer file sharing technology has evolved through several design stages from the early networks like Napster, which popularized the technology, to the later models like the BitTorrent protocol[8].Several factors contributed to the widespread adoption and facilitation of peer-to-peer file sharing. These included increasing Internet bandwidth, the widespread digitization of physical media, and the increasing capabilities of residential personal computers. Users were able to transfer either one or more files from one computer to another across the Internet through various file transfer systems and other file-sharing networks.

A. Survey on P2P streaming System

In terms of scalability, cost and ease of deployment, the Peer-to-Peer (P2P)approach has emerged as a promising solution for video streaming applications. Its architecture enables end-hosts, called peers, to relay the video stream to each other. P2P systems are in fact networks of users who control peers. Thus, user behavior is crucial to the performance of these systems because it directly impacts the streaming flow. To understand user behavior, several measurement studies have been carried out over different video streaming systems. Each measurement analyzes a particular system focusing on specific metrics and presents insights. However, a single study based on a particular system and specific metrics is not sufficient to provide a complete model of user behavior considering all of its components and the impact of external factors on them. A comparison and a synthesis of these measurements are performed. First of all, review on video streaming architectures, followed by a survey on the user behavior measurements in these architectures [1].

B.Client Server Architecture

The client-server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Client Server Architecture is not time efficient and takes time to deliver files having large files.

C. GridCast

Video-on-Demand (VoD) is a compelling application, but costly. VoD is costly due to the load it places on video source servers. Many models are present which uses peer-to-peer (P2P) techniques to shift load from servers to peers. GridCast, is a real deployed P2P VoD system. GridCast has been live on CERNET since May of 2006. It provides seek, pause, and play operations, and employs peer sharing to improve system scalability. In peak months, GridCast has served videos to 23,000 unique users. GridCast with single video caching (SVC) can decrease load on

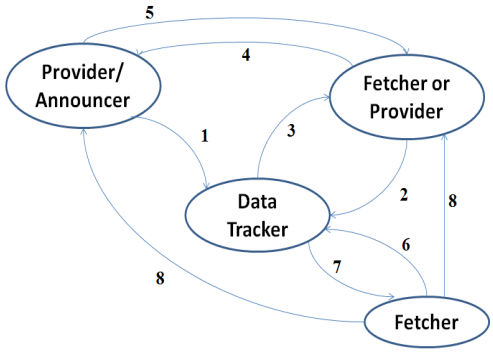
source servers by an average of 22 percent from client-server architecture[3]. The net effect on system resources and determine that peer upload is largely idle. This leads to changing the caching algorithm to cache multiple videos (MVC). MVC decreases source load by an average of 51 percent over the client-server. The improvement is greater as user load increases. This bodes well for peer-assistance at larger scales. A detailed analysis of MVC shows that departure misses become a major issue in a P2P VoD system with caching optimization. A framework for lazy replication is presented and evaluated in this article. In this framework, two predictors are plugged in to create the working replication algorithm.

III. Proposed System

In the proposed system the problem of delay in transmission and data distribution is addressed. We are using concept of video-on-demand it can be replaced by file-on-demand. In case of Infrastructure-as-a-service in cloud network, storage, virtual machines are provided to the user.

System Flow

1. Announcer who wants to distribute a file (which can be video file or file having virtual machine image) first creates a small announcement for that file. This announcement is done at data tracker. They make the file itself available through announcement and act as provider. Announcement contains metadata related to that particular file.
2. Fetcher who want particular file sends request to data tracker



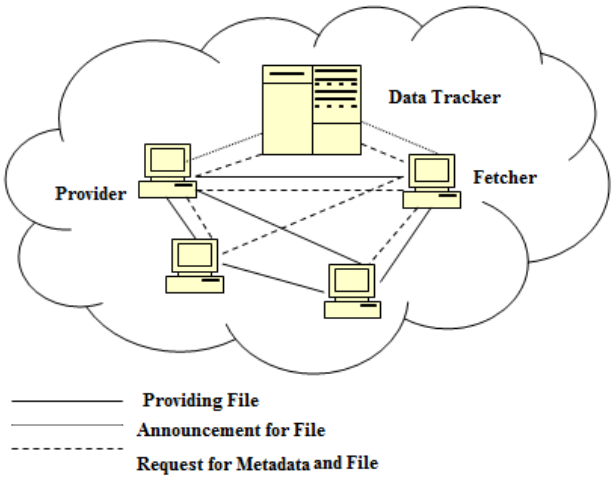
3. Data Tracker searches metadata for that particular file. If any provider related to that file founds data tracker sends metadata related to file and provider to fetcher. If metadata for particular file is not found file is not available with provider. Data Tracker notifies response to fetcher. After getting metadata related to file fetcher sends request to provider.
4. After arrival of request for file from provider the file being distributed is divided into segments called chunks. Each piece is protected by providing a checksum. This ensures that any modification of the piece can be reliably detected, and thus prevents both accidental and malicious modifications of any of the pieces received at other nodes. If any unwanted fetcher gets an authentic copy of file, it can it make changes to entire file it receives. To achieve integrity checksum is introduced.

5. Pieces are typically downloaded non-sequentially and are rearranged into the correct order by the fetcher, which monitors which pieces it needs, and which pieces it has and can upload to other peers. Pieces are of the same size throughout a single download (for example a 10 MB file may be transmitted as ten 1 MB pieces or as forty 256 KB pieces). Due to the nature of this approach, the download of any file can be halted at any time and be resumed at a later date, without the loss of previously downloaded information, which in turn makes system particularly useful in the transfer of larger files. This also enables the fetcher to seek out readily available pieces and download them immediately, rather than halting the download and waiting for the next piece in line, which typically reduces the overall time of the download. When a peer completely downloads a file, it becomes provider. This eventually shifted from peer to peer.

6. Now suppose new fetcher comes into picture same way it sends request for metadata to data tracker.
7. Data Tracker sends metadata for requested file to fetcher. Suppose that file is now available with two providers.
8. Fetcher will send request to both providers and as per the request half of the part fetcher gets from both providers.

IV. Modules

It is consists of 3 modules as provider, fetcher, data tracker. Provider provides data or which he wants to share with others. Provider announce file at data tracker. Data Tracker store metadata related to that file and whenever request comes for the file from fetcher, data tracker search into its metadata and provides information accordingly to fetcher. Fetcher directly contacts to respective provider.



Provider/Announcer: Announcer announces the file name which he wants to share with others. If any request comes to provider he will send corresponding file as per the request.

Data Tracker: At Data Tracker announcer announce the information about the file. Data Tracker maintain log about file and information related to provider, fetcher. Also keeps track related to each and every file.

Fetcher: Fetcher requests for metadata of required file which may contain filename, file type, information of provider, location of provider etc. If it gets metadata for file which belongs to particular provider fetcher directly connects to that provider and download the file once download is completed fetcher goes in provider's state.

V. Methodology

Th is the threshold value for percentage of file download. Flag is used for checking response from data tracker. Here fid is identification number of file, pid is identification number of provider and pname is name of provider.
Response saves the response for search of metadata.

Algorithm for Data Tracker: Data tracker is maintains metadata and changes role of fetcher. Th is the highest percentage of file download. Once announcement is received from announcer data tracker insert metadata for that announcer and file. At data tracker if any request comes from fetcher data tracker search into metadata and sets flag accordingly. If data is found related to file data tracker returns value as 1 and it send message as record not found. If fetcher reaches to Th value it notifies to data tracker and tracker will change its role from fetcher to provider. Data tracker updates it metadata accordingly.
Algorithm:
1. flag = 0
2. *Th* = Highest percentage of file download
3. If announce message receive
4. store announcer to metadata
5. If request comes from fetcher
6. flag = search(metadata)
7. If flag return value as 1
8. give(metadata)

9. else
10. record not found
11. If any notification from fetcher receives
12. Update(metadata)
13. Stop

Algorithm for Fetcher: Fetcher sends request to data tracker for file if positive response comes from data tracker that response contains provider's information else there is no information available for that file related provider at data tracker. For positive response fetcher sends request to provider. Provider gives file in chunk format and for each chunk integrity check is done at fetcher's side. At providers side for each chunk checksum is provided and fetcher uses that checksum for integrity check. If *Th* and T values are same means complete download of file is done and it notifies to data tracker.

Algorithm:

1. *Th* = Highest percentage of file download
MET's Institute of Engineering 12
A Scalable Server Architecture for Mobile Presence Services in Social Network
Applications
2. T = Percentage of file download
3. response = searchForMetadata(fname)
4. if positive response is received
5. sendRequestToProvider(*fid* , fname, *pid*)
6. T = getChunk ( )
7. Check integrity
8. if *Th* and T are equal
9. notifyToDataTracker(fetchId)
10. else
11. act as a fetcher
12. else
13. file does not exists
14. Stop

Algorithm for Provider: Sign is signature provided to that file. By using SHA algorithm hash is provided to file. Announcer announce file which he wants to distribute among nodes. After announcement announcer becomes provider. If request R arrives for file F at provider convert that file into equal size chunks and start distributing those chunks among the fetcher if all chunks are distributed and it reaches to end of file then chunk distribution is completed.

Algorithm:

1. R : request
2. F : file name
3. *fid* : file identification number
4. sign : signature of file
5. sign = SHA1(file)
6. reg= (Sign, reg)
7. If reg is true
8. if request R for file F arrives
9. start(provider)
10. chunkDistribution()
11. if end of file
12. sending completed
13. Stop

VI. Conclusion

This system stores files in a peer to peer network where each peer is contributing to serving others which leads to reduction in perfecting delay as well as eliminating bandwidth bottleneck. In this application files are shared among the nodes in a distributed way. A peer to peer approach is best suited for downloading or sharing large data files. It

is distributed protocol as to store and track data while continuously verifying the data integrity using check sums. It reduces workload of single node/peer also provides time efficient distribution and sharing of file having large size.

REFERENCES

[1] IEEE COMMUNICATIONS SURVEYS TUTORIALS on ”A Survey and Synthesis of User Behavior Measurements in P2P Streaming Systems” by Ihsan Ullah,Guillaume Doyen.
[2] Z. Li, H. Shen, H.Wang, G. Liu and J. Li,”SocialTube:P2P-assisted Video Sharing in Online Social Networks,” In Proc. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 25, NO. 9, SEPTEMBER 2014
[3] B. Cheng, L. Stein, H. Jin, X. Liao, and Z. Zhang, ”Gridcast: Improving Peer Sharing for P2P VoD,” ACM Trans. Multimedia Comput., Commun., Appl., vol. 4, no. 4, p. 26, Oct. 2008.
[4] Yaning Liu ,Gwendal Simon ,”Peer-to-Peer Time-shifted Streaming Systems”, Institut Telecom - Telecom Bretagne, France,November 6, 2009.
[5] Laizhong Cui and Nan Lu SocialStreaming: P2P-assisted Streaming in Social Networks In IEEE Trans. on Computers,2013.
[6] A.J. Ganesh, A-M.Kermarrec and L. Massoulie, Peer-to-Peer membership management for gossip-based protocols.In IEEE Trans. on Computers, vol.52, 2003.
[7] Jaakko Kangasharju, Jussi Kangasharju, ” An Optimal Basis For Efficient Peer-To- Peer Content Distribution Algorithms”.
[8] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, H.J. Sips,THE BITTORRENT P2P FILE-SHARING SYSTEM: MEASUREMENTS AND ANALYSIS,Department of Computer Science, Delft University of Technology, the Netherlands
[9] H. Wang, J. Liu, and K. Xu, On the Locality of Bittorrent-Based Video File Swarming, in Proc. P2P, 2009, p. 12. 37 A Scalable Server Architecture for Mobile Presence Services in Social Network Applications
[10] Peersim: A Peer-to-Peer Simulator. [Online]. Available: http:// peersim. sourceforge.net/.
[11] PlanetLab. [Online]. Available: http://www.planet-lab.org/.
[12] Facebook Users Average 7 hrs a Month in January as Digital Universe Expands. [Online]. Available: http://blog.nielsen.com/nielsenwire.

