Asian Journal of Computer Science and Technology
ISSN 2249-0701 Vol. 2 No. 1, 2013, pp.28-31
© The Research Publication, www.trp.org.in

Fibonacci Based Chunks Distribution (FBCD) Scheme for Efficient P2P Interactive VOD Streaming

M. Arun[1] and P. Vaishnavi[2]
[1]*PG Scholar, Department of Computer Science and Engineering,*
[2]*Assistant Professor, Department of Computer Applications, Anna University Trichy, Tamil Nadu, India.*
E-mail : arunmadhavan.erode@gmail.com
(Received on 10 February 2013 and accepted on 18 April 2013)

Abstract - **One of the problems faced frequently by users in Video-on-Demand (VoD) streaming is long waiting time for voice and video files to load. This paper focuses on Fibonacci Based Chunks Distribution (FBCD) scheme to efficiently handle interactive VoD operations in peer-to-peer systems. In FBCD, videos are divided into number of chunks and stored at peers' local storage in a distributed manner. The chunk size is progressively increased to reduce delay and improve performance. The appropriate performance aspects are analyzed with respect to both user-responsiveness and provider system efficiency. In most existing methods, a new client must search for parent peers containing specific segments. However, FBCD uses the properties of middleware server to cache equivalent chunk in peers. A peer can discover and access a neighbor quickly through the middleware. So the client can pick any server without additional searches, since the middleware can monitor the distribution of server loads and client request. This eliminates direct communication between Client and server and therefore Server offline problems are also managed effectively. The proposed scheme achieves quick video display to the end user without any delay and requires fewer server resources for data storage.**

Keyword: **Fibonacci, Middleware, Chunks distribution, Video on demand**

I. Introduction

Normally a video-audio streaming activity transmits/receives audio, video and other types of media files over the Internet or intranet for efficient communication among users located in different parts of the world. Downloading of multimedia data usually takes place in the user computer through allocation of a small memory space called the 'buffer' which stores the downloaded data temporarily for playback. After a sizeable portion of the data is downloaded into the buffer, the video application starts playing the multimedia content and data in the buffer gets used up. Once the buffer becomes free, more streaming content is downloaded and this process repeats itself till all the content is downloaded and played. The critical requirement in this process is that the download speed should match or exceed the rate at which the video application uses data for playback. Otherwise the playback will not be smooth and will be interrupted frequently due to paucity of data in the buffer.

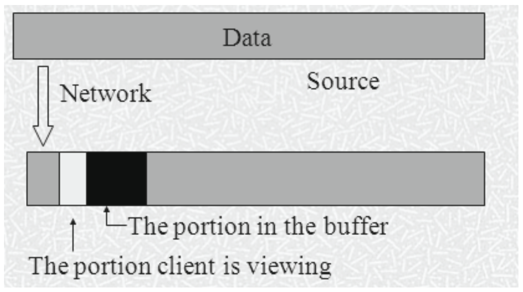

Fig. 1 Data Buffering model

In order to download and play a multimedia content from the Internet, we require a computer connected to the internet over a local area network (LAN) or a modem and a web browser with the proper player or plug-in installed. Plug-ins is useful utilities that can function by seamlessly integrating with the browser in playing multimedia file streams of a specific encoding format. A web server is the place where web pages or HTML files are stored. Generally, streaming and playback of video from the Internet is a function of network channel capacity, channel bandwidth and endpoint capabilities.

A.Network Channel Capacity

It is concerned with the quantity of multimedia data that can be transported from the server to the endpoint over a network or channel. Since there are a number of network types and different channel capabilities, the amount of data that can be transported can fall within a large range. For example a broadband network and radio channel combination is an instance of many such options available that determine the network channel capacity between a server and the endpoint.

B.Streaming Bandwidth and Storage

It may vary depending on a number of parameters, particularly the availability of bandwidth. Because of this reason there can be differences in quantity of data transported, even when the same streaming server streams the same content. Assuming a single file downloaded by a single user, we can arrive at the streaming media storage size as a function of streaming bandwidth and length of the media using the following formula:

Storage size (in megabytes) = length (in seconds) × bit rate (in bit/s) / (8 × 1024 × 1024

C. Endpoint Capabilities

It refers to other factors that have an impact on the downloading speed such as general hardware/software issues, website performance issues, Internet and firewall issues. Out-of-date software's, quality of plug-ins for playback of multimedia streams by web browser, graphics and sound card hardware capabilities, and decrease in internet speed are some of the factors that can affect endpoint capabilities.

In this paper, we focus on Fibonacci based chunk distributions (FBCD) scheme to efficiently handle interactive VoD operations in peer-to-peer systems. a) Videos are divided into no of chunks. b) Chunk data will be stored in server in a distributed manner. c) Chunk size progressively increased. d) The variant size of the chunks to reduce delay and improve performance. e) Performance and efficiency will be calculated between user and provider. f) Using middleware server to monitor the server load and balance.

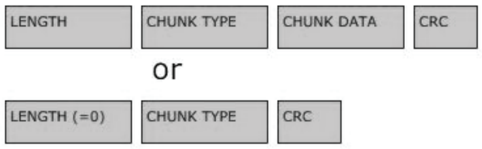

Fig.2 Chunk Format

II. Related Work

Generally, accessing video content randomly works well for on-demand multimedia content streaming. However, this process is rendered difficult and inefficient in cases where there is extensive interaction by users that are asynchronous and also in cases where peers are predominantly dynamic in nature. In this paper, we propose a network coding equivalent content distribution (NCECD) scheme that will effectively process on-demand streaming multimedia that is interactive in peer-to-peer systems. In this coding scheme multimedia content is divided into unique segments which are subdivided into blocks. The advantage of dividing the content into small blocks is that we are able to encode these uniquely identifiable blocks and store them in different peers resulting in a distributed local storage. This scheme facilitates the client in locating the whole video from among a sufficient number of parent peers; the need to hunt for new parents during random access by the client is eliminated. In traditional methods whenever a client hunts for video data on the web it has to look for parent peers storing particular segments. The characteristic feature of network coding scheme is to distribute the equivalent content across peers so that the client can choose any parent without looking for the one that has the correct segment. This way the scheme is able to reduce the time taken for content identification and startup/jump searching delays. In addition it requires fewer server resources.

In the illustration below, we present the analysis of system parameters to achieve reasonable block loss rates for the proposed scheme.

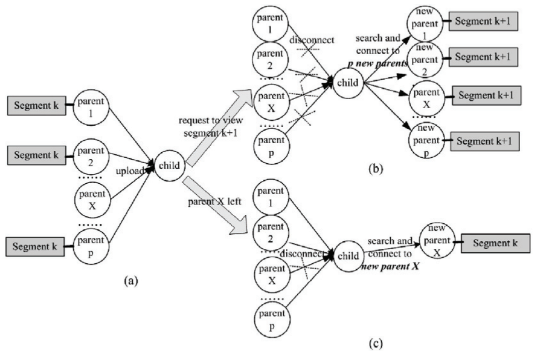

Fig. 3 Analysis of System parameters

Fig. c). the basic cache-and-relay approach with multiple parents. (a) The video content whether a whole segment or a block, is stored in each parent peer. (b) Child peers already connected to peers must first disconnect form the original p parent peers while wanting to view the next segment k þ 1 or say, switch over to another segment; subsequently it must locate p new parents peers for the service or (c) when a parent peer leaves, the child peer it needs to find a new parent peer also stores the same segment k.

III. Proposed System

In the proposed system, videos are divided into no of chunks and stored at peers' local storage in a distributed manner; the chunk size is progressively increased to reduce delay and improve performance. FBCD uses the properties of middleware to cache equivalent chunk in peers. A peer can discover and access a neighbor quickly through the middleware. Through the client one can pick any server without additional searches, because middleware will monitor all the distribution server loads and client request. This may not allow direct communication between client and server and the server offline problem is also effectively managed through this arrangement.

A. Fibonacci Method

In our development, Fibonacci concept will be used to reduce the buffering and video projection rate at the client end as well as to manage time. Once the amount of chunks is fixed, different performance levels can be detected corresponding to different splitting approaches, which refers to the judgments about the size of each chunk. Splitting approaches influence different aspects of the system performance. The Fibonacci sequence method is a set of numbers that starts with one or zero, followed by one, and proceeds based on the rule that each number (called a Fibonacci number) is equal to the sum of the preceding two numbers.

If the Fibonacci sequence is denoted F (n), where n is the first term in the sequence, the following equation obtains for n = 0, where the first two terms are defined as 0 and 1 by convention:

F (0) = 0, 1, 1, 2, 3, 5, 8, 13, 21, 34...

In some texts, it is customary to use n = 1. In that case the first two terms are defined as 1 and 1 by default, and therefore:

F (1) = 1, 1, 2, 3, 5, 8, 13, 21, 34...

Based on this concept, the video files will be divided from small to large size. So the data size increases progressively.

B. Middleware Technology

It is a computer / computer managed software that provides services between Client and Server. Middleware is not an operating system and also it's not a Database system. It makes easier for software developers to make communication between Client and Server. So it may focus on the specific purpose of their application like Client/Server Monitoring and status of that. Middleware is sometimes called plumbing because it connects two sides of an application and passes data between them.

This middleware concept will be used to make interaction between Client and Server without direct connection. All the communication will be done through the middleware server only. The middleware will monitor the Client and Server status, Load balance, Time schedule, Log maintenance, Server information's, request and response and related aspects. It supports to store chunks in all servers in a distributed manner.

C. Multimedia Upload

This Multimedia Upload activity is used to copy the multimedia contents on the server through the middleware First of all, the original media file is uploaded in the middleware server and then split (Splitting phase) into number chunks in order to allow the data dissemination. In essence, the amount of chunks and the accepted splitting methods affect the system performances in terms of delays during the streaming. This is then essential to sensibly carry out the splitting based on the presented Grid resources. Different chunk splitting of the same media file can help in answering different levels of Quality of Service requests. Once this stage is completed, each chunk is stored at the maximum possible quality.

Each chunk is sent to the Grid storage system and registered on the Replica Catalog. Scalability can be improved by creating more replicas of each chunk and by disseminating them in a geographically distributed manner.

D. Upload file collection

In this module middleware system maintains the video file list which is uploading the video files from the middleware system to back-up server system. When the client systems connect to the middleware to access the online videos the client should know the video files in server. The middleware system is responsible to show the video file list to the connected clients.

E. Multimedia Streaming

The Multimedia Streaming motion deals with end-user / Client requests. Users request to view Grid Video content with specified features in terms of resolution, size, audio reproduction, color depth, and media format supported using a GUI. The different chunks are recovered and tailored from the server through middleware according to the requested characteristics. Once the first part is tailored, the streaming phase allows users to watch the entire video streaming in a transparent way over a set of plain connections. So this module gathers the video files from different servers with in a network area and various video chunks are stored in a distributed manner in a number of servers. When end user/client selects a video from the list, videos will be displayed in the user window.

The user can select any video from the list to watch the video as per their wish. That particular request will be sent as a request to the middleware. The middleware tailors all the chunks from different servers as well as respond to the clients who demand the video. The media player is not aware of the entire Grid Video architecture. The application composes a playlist through the middleware that contains the reference to each chunk in the correct order.

F. Middleware Management

Multimedia streaming services involve managing some network related services like, video quality across the network. Throughput has to be managed through a careful control of network resources like bandwidth and through the adoption of client-side mechanisms (i.e., buffers) to overcome sudden service interruptions.

G. Critical attributes for Streaming

The critical attributes for any streaming are:

- Centralized server getting overloaded due to many request from the clients
- Clients directly connecting to server
- Loading every client at a time
- Buffering time when playing a video

H. Advantages of Fibonacci Streaming

- Designed with 3 tier architecture.
- Client and Server communication made through Middleware.
- Video files split into number of chucks using Fibonacci approach.
- Loading time and buffering time will be reduced
- Server offline problem also managed using middleware.
- Server Load balancing problems can be minimized
- Start, Stop, Jump sequence display.

IV. Conclusion and Future Works

In most existing methods, a new client must search for parent peers containing specific segments; however, FBCD uses the properties of middleware to cache equivalent chunk in peers, a peer can discover and access a neighbor quickly through the middleware. So a client can pick any server without additional searches as the middleware can monitor all the distribution server loads and client request. This may not allow direct communication between client and server and server offline problem is also managed efficiently. The proposed scheme achieves low startup and jump searching delays and requires fewer server resources.

References

[1] Yung-Cheng Kao, Chung-Nan Lee, Peng-Jung Wu, and Hui-Hsiang Kao - "A Network Coding Equivalent Content Distribution Scheme for Efficient Peer-to-Peer Interactive VoD Streaming", *International Journal of Computer Applications* Vol.23, No. 6, JUNE 2012.

[2] A.Shanmugam, L.M.Nithya, "A New Grid Architecture using JMF for Video-on-Demand Applications" *International Journal of Computer Applications* (0975 – 8887) Vol. 10, No.9, November 2010.

[3] Dario Bruneo, Giuseppe Iellamo, Giuseppe Minutoli, Antonio Puliafito, University of Messina, Messina – " A Practical Example of Nonscientific Application on the Grid", *International Journal of Computer Applications* Vol. 21, no. 5, May 2009.

[4] C. Xu, G.M. Muntean, E. Fallon, and A. Hanley, "A Balanced Tree - Based Strategy for Unstructured Media Distribution in P2P Networks," *Proc. IEEE Int'l Conf. Comm.* (ICC '08), pp. 1797-1801, May 2008.

[5] W.P.K. Yiu, X. Jin, and S.H.G. Chan, "VMesh: Distributed Segment Storage for Peer-to-Peer Interactive Video Streaming," *IEEE J. Selected Areas in Comm.,* Vol. 25, no. 9, pp. 1717-1731, Dec. 2007.

[6] H.V. Jagadish, B.C. Ooi, and Q.H. Vu, "BATON: A Balanced Tree Structure for Peer-to-Peer Networks," *Proc. Int'l Conf. Very Large Data Bases (VLDB '05),* pp. 661-672, Aug. 2005.

[7] C. Zheng, G. Shen, and S. Li, "Distributed Prefetching Scheme for Random Seek Support in Peer-to-Peer Streaming Applications," *Proc. ACM Workshop Advances in Peer-to-Peer Multimedia Streaming,* pp. 29-38, Nov. 2005.