International Journal of Innovative and Emerging Research in Engineering
Volume 2, Issue 2, 2015

Available online at www.ijiere.com

International Journal of Innovative and Emerging Research in Engineering

e-ISSN: 2394 - 3343 e-ISSN: 2394 - 5494

Efficient video streaming and data sharing using Cloud Computing

Suntej Singh[1], Aamod Pisat[1] and Gajanan Shewale[1], Prof. Sheetal Chaudhari[2]
[1]BE Student,Department of IT, Sardar Patel Institute of Technology, Mumbai, India
[2]Assistant Professor, Dept of IT, Sardar Patel Institute of Technology, Mumbai, India.

ABSTRACT:
This paper presents the video compression algorithm by exploiting the similarities and redundancies that exists in the video signal. Video delivery by video streaming attempts to solve the problems associated with file download. However, existing video streaming services face problems like loss of quality, buffering delay. In this paper, we have proposed efficient video streaming over internet by using video compression algorithm as well as tools such as Mencoder, FFmpeg. In the proposed scheme, video files are compressed using unquantized motion, but the decoder receives and uses only the most appropriate motion parameter quality layer.

Keywords: Video, Streaming, Sharing, Cloud, Player

I. INTRODUCTION

Internet has been evolving a new way of streaming multimedia application. Many of the web application that have been developed are heavyweight and not efficient for real time media application. Since, internet is highly heterogeneous environment in term of link capacity and many devices, video codecs need to generate bitstreams that are highly scalable in term of bandwidth[1].The performance of the proposed algorithm is evaluated by comparison with a non scalable codec and the penalty in compression efficiency that the scalability requirement imposes is analyzed.

Real-time transport of live video or stored video is the predominant part of real-time multimedia. In this paper, we are concerned with video streaming, which refers to real-time transmission of stored video. There are two modes for transmission of stored video over the Internet, namely the download mode and the streaming mode (i.e., video streaming). In the download mode, a user downloads the entire video file and then plays back the video file. However, full file transfer in the download mode usually suffers long transfer time. In contrast, in the streaming mode, the video content need not be downloaded in full, but is to be buffered which are being received and decoded.

Real-Time Streaming Protocol (RTSP)

The Real-Time Streaming Protocol (RTSP) establishes and controls either a single or several time-synchronized streams of continuous media such as audio and video. It does not typically deliver the continuous streams itself, although interleaving of the continuous media stream with the control stream is possible.[5]In other words, RTSP acts as a “network remote control” for multimedia servers.

There is no notion of an RTSP connection; instead, a server maintains a session labeled by an identifier. An RTSP session is in no way tied to a transport-level connection such as a TCP connection. During an RTSP session, an RTSP client may open and close many reliable transport connections to the server to issue RTSP requests. Alternatively, it may use a connectionless transport protocol such as UDP. The streams controlled by RTSP may use RTP , but the operation of RTSP does not depend on thetransport mechanism used to carry continuous media.[2]

II. Tools and Technology

A. FFmpeg

FFmpeg is the leading multimedia framework, which can be able to play streaming services on Web browser. Its help the application to encode, decode , stream and filter the media files accordingly. Our application uses ffmpeg plugin to support multimedia streaming services to webApp. It uses ffcodecs library which provides some generic global options, which can be set on all the encoders and decoders. In addition each codec may support so-called private options, which are specific for a given codec.
The list of supported options follow:

integer (encoding,audio,video)
Set bitrate in bits/s. Default value is 200K.

ab integer (encoding,audio)
Set audio bitrate (in bits/s). Default value is 128K.

bt integer (encoding,video)
Set video bitrate tolerance (in bits/s). In 1-pass mode, bitrate tolerance specifies how far ratecontrol is willing to deviate from the target average bitrate value. This is not related to min/max bitrate.

flags flags (decoding/encoding,audio,video,subtitles)
Set generic flags.
Possible values:
'mv4'
Use four motion vector by macroblock (mpeg4).

'qpel'
Use 1/4 pel motion compensation.

B. MEncoder
Mencoder features the same huge number of highly-configurable video and audio filters to transform the video and audio stream. Filters include cropping, scaling, vertical flipping, horizontal mirroring, expanding to create letterboxes, rotating, brightness/contrast, changing the aspect ratio, colorspaceconversion, hue/saturation, color-specific gamma correction, filters for reducing the visibility of compression artifacts caused by MPEG compression (deblocking, deringing), automatic brightness/contrast enhancement (autolevel), sharpness/blur, denoising filters, several ways of deinterlacing, and reversing telecine.

Frame rate conversions and slow-motion in MEncoder
Changing the frame rate is possible using the -ofps or -speed options and by using the framestep filter for skipping frames. Reducing the frame rate can be used to create fast-motion "speed" effects which are sometimes seen in films.
Doubling the frame rate of interlaced footage without duplicating or morphing frames is possible using the tfields filter to create two different frames from each of the two fields in one frame of interlaced video. This allows playback on progressive displays, while preserving the full resolution and framerate of interlaced video, unlike other deinterlacing methods. It also makes the video stream usable for frame rate conversion, and creating slow-motion scenes from streams taken at standard video/TV frame rates, e.g. using cheap consumer camcorders. If the filter gets incorrect information about the top/bottom field order, the resulting output will have juddering motion, because the two frames created would be displayed in the wrong order.

C. HTTP streaming to HTML5 video client
IP video camera RTSP H.264 stream is picked up by FFMPEG and remuxed into a mp4 container using the following FFMPEG settings in node, output to STDOUT. This is only run on the initial client connection, so that partial content requests don't try to spawn FFMPEG again. se the node http server to capture the STDOUT and stream that back to the client upon a client request. When the client first connects I spawn the above FFMPEG command line then pipe the STDOUT stream to the HTTP response.

```
        liveFFMPEG.stdout.pipe(resp);
```

Using the following HTTP header, httpCreateServerConnection(req,res) will have two parameters, passed one is HTTP request object and other one is HTTP response object.

```
        if(req.file!=undefined && req.file<bytes.size[999]) then
                var parts = range.replace(/bytes=/, "").split("-"); //Replace byte with nulll and splits the part.

//Intialise start and end bit
   var partialstart = parts[0];
   var partialend = parts[1];
}

var start = parseInt(partialstart, 10);
var end = partialend ? parseInt(partialend, 10) : total;   // fake a large file if no range request

var chunksize = (end-start)+1;
```

Streaming (any size) video files with fs.createReadStream
By utilizing fs.createReadStream(), the server can read the file in a stream rather than reading it all into memory at once. This sounds like the right way to do things, and the syntax is extremely simple:

```
            Server Snippet:
            var fs=require('fs')++
            movieStream = fs.createReadStream(pathToFile); // Read file stream from URL or from the specified path
//Open the stream connection
movieStream.on('open', function () {
   res.writeHead(206, {
      "Content-Range": "bytes " + start + "-" + end + "/" + total,
         "Accept-Ranges": "bytes",
         "Content-Length": chunksize,
         "Content-Type": "video/mp4"
   });
   // This just pipes the read stream to the response object (which goes
   //to the client)
   movieStream.pipe(res);
});
```

III. CONCLUSIONS

This application allows us to share videos and to stream videos online .It also takes into account that the security of the person sharing the video is maintained and that the cloud service provider is not able to intercept any of the information by encrypting the video.Still as security is the most important aspect nowadays some modifications might be needed to protect the system from new threats that may prop up in the future .In the future it might be possible to provide functionality of editing and clipping videos or creating videos in our application which would be something like an online movie maker. Also a video converter could be provided so that the video can be converted into a format that will play on your computer or on your tablet or even on your mobile phone.

ACKNOWLEDGMENT

We have immense pleasure in presenting the report for our paper entitled Efficient Video Streaming and Data Sharing using Cloud Computing. We would like to take this opportunity to express our gratitude to a number of people who have been sources of help and encouragement during the course of this project.

We are very grateful and indebted to our project mentor (Mrs.) Sheetal Chaudhari, for providing her enduring patience, guidance and invaluable suggestions. She was the one who never let our morale down and always supported us through our thick and thin. She is a constant source of inspiration for us and took utmost interest in our project.

We would also like to thank all the Staff Members of Sardar Patel Institute of Technology for their valuable cooperation and permitting us to work in the I.T. Lab.

REFERENCES

[1] Xiaofei Wang,Min Chen,Ted Taeyoung Kwon,Laurence.Yang, Victor C.M.Leung ,"AMES –cloud: A framework of adaptive mobile video streaming and efficient social video sharing in the clouds",‖ IEEE transaction on multimedia,Vol 15, no.4, June 13.
[2] Dapeng Wu (Carnegie Mellon University), Yiwei Thomas Houn (Fujitsu Laboratories) ,Wenwu Zhu (Microsoft Research China), Ya-Qin Zhang (Microsoft Research China), Jon M. Peha(Carnegie Mellon University, peha@andrew.cmu.edu), "Streaming Video over the Internet: Approaches and Directions"
[3] Andrew Secker and David Taubman, Member, IEEE , "Highly Scalable Video Compression With Scalable Motion Coding"
[4] N. Banerjee, W. Wu, S. Das, S. Dawkins, J. Pathak, ―"Mobility support in wireless Internet,‖ IEEE Wireless Commun", vol. 10, no. 5, pp. 54–61, Oct. 2003.