멀티미디어 데이터 Playback 최적화를 위한 Cross-Platform 어플리케이션

Mikhail Oparin*, 조영필*, 권용인*, 고광만** 백윤흥*
*서울대학교 전기컴퓨터공학부
**상지대학교 컴퓨터정보공학부
e-mail : micam1818@gmail.com

Cross-Platform Application for Multimedia Data Playback Optimization

Mikhail Oparin*, Yeongpil Cho*, Yongin Kwon*, Kwangman Ko** Yunheung Paek*
*Dept. of Electrical Engineering and Computer Science, Seoul National University
**Dept. of Computing Information & Engineering, Sangi University

Abstract

With the continuous growth of a number of high-quality multimedia services for handheld devices, the lack of power resources becomes an increasingly critical issue. One of the ways to overcome existing problem is to make multimedia data processing more efficient. In order to do that this paper introduces a video streaming application for Android platform which, while being used along with offloading technique, may provide an efficient progressive download service for user devices along with relief of media servers.

1. Introduction

The onrush of technology in recent years has led to growing popularity of handheld devices including smartphones and tablet PCs. To date people can use smartphones for a wide variety of purposes, starting from usual phone calls and exchanging messages to playing games and developing new software. Under the circumstances the usage of multimedia services, which gets more and more common among users, also increased drastically. According to the research made on power consumption in a smartphone, graphics indeed is one of major power consumers [1]. Therefore, the need for new more efficient and optimal designer solutions to process multimedia data becomes more intense.

There can be several approaches to the problem such as: improving video compression algorithms, optimization of video player work, additional hardware resources utilization, or use of remote computational resources for data processing. As an example for improving video compression, a new compression format known as the high-efficiency video coding (HEVC) standard can be considered. Its goal is a significant improvement of the compression efficiency compared to already existing H.264/AVC standards [2]. A new high-definition video player method based on GPU technology [3] serves as an example for video player elaboration. In that paper they proposed a new framework of hardware and software platforms utilizing a method which uses an advantage of VDPAU library and VAAPI to accelerate the decoding process. Additional hardware resource approaches are those, which use digital signal processors and FPGAs partitioning [4]. In this paper we will consider the last of all the above mentioned approaches to improve quality-energy ratio for mobile video viewing, using cloud computing. This method not only optimize the mobile device multimedia data processing, but also helps to unburden media servers by shifting part of their duties to more abundant cloud servers. In order to reach these goals we developed a video streaming application for Android 'x86' and ARM architectures, which is being installed in a smartphone and a remote high-performance computer allowing them to execute the heaviest parts of video viewing processes outside the mobile device.

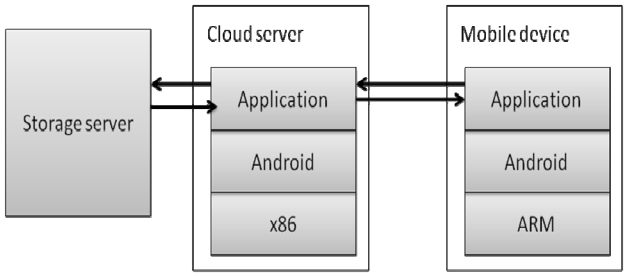

Figure 1 Structure of video offloading method

2. System framework

Framework of the method is shown in Figure 1. The entire workflow consists of the following stages: downloading source files from the remote storage server to the high-performance cloud server, adjusting the parameters of the video on the cloud server side and streaming of fully adjusted, perfectly matched video file for the end device. Applications,

running on the cloud server and mobile device are identical and implemented according to the 'Fast Dynamic Execution Offloading' technique [5]. That is, firstly initiated on the mobile device, the most heavy and power consuming parts of the execution process will be transferred to the cloud server. Thus, when we just start executing the application and we send a request for a video file from mobile device to the storage server, this part will be treated as heavy and in the end will be processed by the cloud server. Therefore, the whole video downloading and viewing process will look exactly as described below: the user starts downloading the video file from the handheld device whereupon the process migrates from mobile device to the high-performance cloud server; the cloud server sends the actual request to the remote storage server and downloads the video file; during the downloading process the cloud server starts editing the video file so that all the required parameters, such as quality, resolution and data bit rate fit well to the end device; when part of the video is processed it starts streaming it back to the user.

3. Video Streaming Application

We made two versions of the video streaming application, one of which is intended to run on the cloud server 'x86' architecture and the other on Android mobile device 'ARM' architecture. The data flow graph for it is represented in Figure 2. The application consists of four main logical blocks: downloading, processing, streaming and viewing of the video file.

At the initial stage we accessed the storage server and upload video data from there to the cloud server. In contrast to mobile devices, which generally use a 3G network to access the Internet, desktop PCs can be connected to broadband Internet service which makes it no problem at all to download multimedia files. Later the mobile device can get the file from the cloud server through a Wi-Fi connection which is also quite fast compared to 3G.

After we downloaded the video we start processing it. For that purpose we used FFmpeg multimedia framework which supports a huge variety of formats. Built together with x264 library for Android platform it allowed us to decode and later to encode the video file into H.264/MPEG-4 AVC format. At this stage we demultiplex the source multimedia file to get it out of the file container, following which we separated audio and video threads and transcoded video data into RAW format.

Next we optimized video parameters, customizing it for the end device screen resolution. By doing this we can reduce the size of the file and avoid excessive data transferring.

After optimization we start preparing our data for streaming. We encode the video stream with H.264 codec and multiplex it into 'mp.4' container format. We decided to stream multimedia data with 'mp4' format so that it can be played by the Android default media player with no additional software or transcoding required. The streaming part is implemented in accordance with the progressive download method, which is used in many worldwide, famous projects such as YouTube, ESPN or CNN. The main features of the method are that data is transferred by regular HTTP protocol and stored on the viewer's hard drive. To develop the HTTP server and client inside our application we used the Netty asynchronous event-driven network application framework. The streaming process is implemented so that every 2-3 seconds video will be sent to user straight away after the transcoding is finished. When the next piece of video arrives to the mobile device it gets joined to the previous ones. This allows the viewer to play the video smoothly once he clicks the play button.

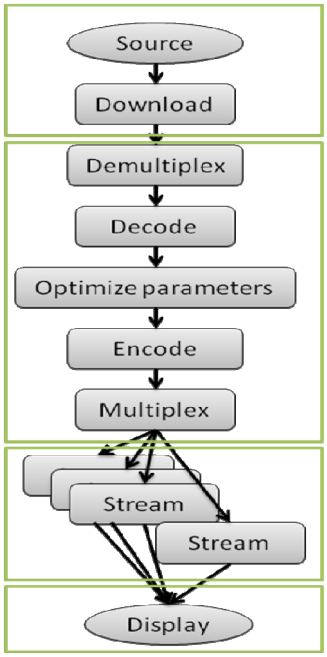

Figure 2 Data flow graph for Video Streaming

When the first part of the video is delivered to the mobile device, we can start playing it. For viewing video on Android devices we used a cross-platform multimedia library called Simple DirectMedia Layer (SDL). FFmpeg and SDL are libraries both written in C and they work just fine together.

4. Summary

In this paper we have considered various methods of optimizing multimedia data playback. Out of them we chose one which uses offloading technique as a main feature and after that we developed video streaming Android application to conduct the experiments on the optimization of multimedia data viewing processes. Also this model will be useful for further modification and elaboration of the offloading technique, where it can be used as a representative of the unique class of workbenches.

References

[1] Aaron Carroll, Gernot Heiser. "An Analysis of Power Consumption in a Smartphone". USENIX Conference on File and Storage Technologies, 2012

[2] Mahsa T.Pourazad, Colin Doutre, Maryam Azimi, Panos Nasiopoulos. "HEVC: The New Gold Standard for Video Compression". In IEEE Consumer Electronics Magazine, 2012

[3] Dong Min, Qiu Rongcai, Wei Ruiping, Bi Sheng, Cai

Wenyi, Xin Jiayi. “A New High-definition Video Player Method Based on GPU Technology”. In IEEE International Conference on Cyber Technology in Automation, 2012

[4] Cheng Peng, Thanh Tran. “HD Video Encoding with DSP and FPGA Partitioning”, Texas Instruments, 2007

[5] Seungjun Yang, Yongin Kwon, Yeongpil Cho, Jonghee Youn, Yunheung Paek. “Fast Dynamic Execution Offloading for Efficient Mobile Cloud Computing”, IEEE International Conference on Pervasive Computing and Communications, 2013

References

본 연구는 교육과학기술부/한국과학재단 우수연구센터 육성사업(No.2012-0000470), 2012 년도 정부(교육과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업(No. 0421-2012-0047), (재)스마트 IT 융합 시스템 연구단(글로벌프론티어사업) ((재)스마트 IT 융합 시스템 연구단-0543-20110012)) 및 IDEC 의 지원을 받아 수행된 것임.