Pervasive and Mobile Computing 6 (2010) 607–622

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

ChunkStream: Interactive streaming of structured data

Justin Mazzola Paluska*, Hubert Pham, Steve Ward

MIT Computer Science and Artificial Intelligence Laboratory Cambridge, MA, USA

ARTICLE INFO

Article history:
Received 10 April 2010
Received in revised form 19 July 2010
Accepted 27 July 2010
Available online 6 August 2010

Keywords:
Chunks
Video editing
Cloud computing
Video streaming

ABSTRACT

We present ChunkStream, a system for efficient streaming and interactive editing of online video. Rather than using a specialized protocol and stream format, ChunkStream makes use of a generic mechanism employing *chunks*. Chunks are fixed-size arrays that contain a mixture of scalar data and references to other chunks. Chunks allow programmers to expose large, but fine-grained, data structures over the network.

ChunkStream represents video clips using simple data types like linked lists and search trees, allowing a client to retrieve and work with only the portions of the clips that it needs. ChunkStream supports resource-adaptive playback and "live" streaming of real-time video as well as fast, frame-accurate seeking; bandwidth-efficient high-speed playback; and compilation of editing decisions from a set of clips. Benchmarks indicate that ChunkStream uses less bandwidth than HTTP Live Streaming while providing better support for editing primitives.

1. Introduction

Users increasingly carry small, Internet-enabled computers with them at all times. Some of these small computers are highly capable smartphones like Apple's iPhone or Google's Android, while others are lightweight netbooks.

These computers are "small" in the sense that they have a small form factor as well as smaller than normal processing and storage abilities. Nonetheless, users expect the same functionality from them as their full-sized brethren. For example, netbook users run full-fledged operating systems and applications on their machines while smartphone users may use full-featured applications to edit photos and spreadsheets directly on their phone.

Such small machines may be computationally overwhelmed by "big" tasks like editing video or creating complex documents. Luckily, many of those big tasks are centered around highly structured data types, giving us an opportunity to present large data structures in smaller units that impoverished clients may more easily consume and manipulate. At the same time, a common network protocol for expressing structure may allow us to share work between small clients and cloud-based clusters, making use of always-on network connections to compensate for poor computational abilities.

In this paper, we explore one way of exposing and sharing structured data across the Internet, in the context of cloud-based video editing. We choose to explore video editing because video is highly structured – video files are organized into streams, which are further organized into groups of pictures composed of frames – yet there is no existing protocol for efficient editing of remote video. Additionally, many video editing operations (such as special effects generation) are computationally intensive, and as such may benefit from a system where clients can offload heavy operations to a cluster of servers. Finally, video is already an important data type for mobile computing because most small computers include special support for high-quality video decoding, and now commonly even video capture.

* Corresponding author.
E-mail address: jmp@mit.edu (J. Mazzola Paluska).

1574-1192/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2010.07.006

1.1. Pervasive video editing

Just as users capture, edit, mashup, and upload photos without touching a desktop computer, as video capabilities become more prevalent, we expect that users will want to edit and mashup videos directly from their mobile devices. However, currently, video editing is a single-device affair. Users transfer all of their clips to their "editing" computer, make all edits locally, and then render and upload their finished work to the web. In a modern pervasive computing environment, users should be able edit videos "in the cloud" with whatever device they may have at the time, much as they can stream videos from anywhere to anywhere at anytime.

Video editing is a much more interactive process than simple video streaming: whereas users of video streaming engage in few stream operations [1], the process of video editing requires extensive searching, seeking, and replaying. For example, a video editor making cutting decisions may quickly scan through most of a clip, but then stop and step through a particular portion of a clip frame-by-frame to find the best cut point. After selecting a set of cut points, the editor may immediately replay the new composite video to ensure that his chosen transitions make sense. In another work flow, an editor may "log" a clip by playing it back at a higher than normal speed and tagging specific parts of the video as useful or interesting. Each of these operations must be fast or the editor will become frustrated with the editing process.

1.2. Internet-enabled data structures

Existing streaming solutions generally assume that clients view a video stream at normal playback speeds from the beginning [2] and are ill-suited to the interactivity that video editing requires. A challenge to enabling pervasive video editing is allowing small clients to manipulate potentially enormous video clips.

Two general principles guide our approach. First, we expose the internal structures of uploaded video clips to the client so that each client can make decisions about what parts to access and modify. Second, we offer "smaller", less resource-intensive proxy streams that can be manipulated in concert with full-sized streams.

Most streaming systems already follow these principles using ad hoc protocols and specialized data formats. For example, Apple's HTTP Live Streaming [3] exposes videos to clients as a "playlist" of short ($\approx$10 s) video segments. Each segment may have individually addressable "variants" that allow clients to choose between an assortment of streams with different resource requirements.

While it is possible to design a specialized video editing protocol that adheres to these principles, we believe that the fine-grained interactivity required by video editing and the device-specific constraints imposed by each small client may be better served by a more generic framework that allows the client and server to decide, at run-time, how to export and access video data. To this end, rather than fixing the protocol operations and data formats, we explore an approach that uses generic protocols and data formats as a foundation for building video-specific network-accessible data structures.

1.3. Contributions

This paper presents ChunkStream, a system that allows frame-accurate video streaming and editing that is adaptive to varying bandwidth and device constraints. We propose the use of a generic primitive – individually addressable "chunks"– that can be composed into larger, but still fine-grained, data structures. Using chunks allows ChunkStream to reuse generic protocols and data structures to solve video-specific problems.

ChunkStream exposes video clips as a linked list of frame chunks. Each video clip may include multiple streams of semantically equivalent video at different fidelity levels, allowing resource-constrained clients to play and edit low-fidelity streams while more powerful cloud-based servers manipulate high-fidelity streams. To overcome network latencies inherent in cloud-based architectures, the linked list is embedded in a search tree that enables a client to quickly find individual frames for editing cut points without needing to download the entire video clip over the network.

Our goal in this paper is to explore how using generic structuring mechanisms may lead to streamable, flexible, and useful data structures. While we focus on video, we believe that using chunks to create structured data streams is generalizable to any structured data type.

2. Architecture

ChunkStream is built on top of a single data type—the chunk. Using chunks as a foundation, we construct composite data structures representing video streams and editing decisions.

2.1. Chunks

A chunk is a typed, ordered, fixed-size array of fixed-size slots. Each slot is typed and may be empty, contain scalar data, or hold a reference ("link") to another chunk. Every chunk is also annotated with a type. Chunks are stored on a central server and may be requested by clients over the network.

Chunk links are explicit and as such can be used to create large data structures out of networks of chunks. Each chunk is identified by an identifier determined by the creator of the chunk. The chunk identifier is guaranteed to be unique within