Video BenchLab Demo: An Open Platform for Video Realistic Streaming Benchmarking

Patrick Pegus II Emmanuel Cecchet Prashant Shenoy
University of Massachusetts Amherst
{ppegusii,cecchet,shenoy}@cs.umass.edu

ABSTRACT
In this demonstration, we present an open, flexible and realistic benchmarking platform named *Video BenchLab* to measure the performance of streaming media workloads. While Video BenchLab can be used with any existing media server, we provide a set of tools for researchers to experiment with their own platform and protocols. The components include a MediaDrop video server, a suite of tools to bulk insert videos and generate streaming media workloads, a dataset of freely available video and a client runtime to replay videos in the native video players of real Web browsers such as Firefox, Chrome and Internet Explorer. Various metrics are collected to capture the quality of video playback and identify issues that can happen during video replay. Finally, we provide a Dashboard to manage experiments, collect results and perform analytics to compare performance between experiments.
The demonstration showcases all the BenchLab video components including a MediaDrop server accessed by real web browsers running locally and in the cloud. We demo the whole experiment lifecycle from creation to deployment as well as result collection and analysis.

Categories and Subject Descriptors
D.2.5 [**Testing and Debugging**]: Testing tools, .D 2.8 [**Metrics**]: Performance measures.

General Terms
Measurement, Performance, Experimentation.

Keywords
Benchmarking, Video, Streaming, Web browsers.

1. INTRODUCTION
This demo presents some of the features described in the MMSys conference paper [5]. The BenchLab project seeks to offer an open, freely-available platform for realistic benchmarking of servers applications. While BenchLab was initially designed to support web-based applications and services (e.g., multi-tier web applications accessed from browser-based clients), *Video BenchLab* is an enhanced platform that provides similar functions for streaming media servers and protocols accessed via browser-based players. Towards this end, Video BenchLab uses real web browsers running real video players to request HTTP streaming content from the server. A key goal of Video BenchLab is to enable automation for running complex experiments where clients, servers or both may be distributed or running on remote machines such as cloud servers; an experiment comprises a set of clients that are controlled remotely by the platform and provided with a video request trace that they then inject on the specified server or set of servers. The content is streamed to HTML5 players supported by modern browsers and a range of statistics are gathered and uploaded to a central database. Real user behavior when watching video content such as pausing, skipping or clicking on related videos can be simulated as part of the trace replay. BenchLab provides the ability to inject requests onto real servers (e.g., YouTube) and also provides a synthetic server backend based on MediaDrop that emulates a YouTube-like system on small scale. Our goal is to enable a range of performance evaluation experiments related to server design, streaming protocol performance, network performance and client-side performance. Finally, Video BenchLab includes new tools that permit analysis of results across different runs of an experiment. We believe that the open nature of the BenchLab platform makes it an attractive choice for multimedia systems researchers for use in their own research and experimentation.

MMSys '15, Mar 18-20, 2015, Portland, OR, USA
ACM 978-1-4503-3351-1/15/03.
http://dx.doi.org/10.1145/2713168.2723146

2. VIDEO BENCHLAB
Video BenchLab provides multiple components and tools for researchers to experiment with media streaming environments: a streaming server virtual appliance (section 2.1), video data sets and workload generators to exercise these datasets (section 2.2), a client runtime to play videos in existing Web browsers using their native HTML5 players or the YouTube JavaScript API (section 2.3) and a Web Application hosting a database to define experiments, collect results and provide analytics (section 2.4).

2.1 MediaDrop Virtual Appliance
While Video BenchLab can be used with any existing video streaming service, there is no readily available virtual appliance of realistic video streaming servers that can be used by researchers to perform media streaming experiments. Recently, in-browser HTML 5 video playing has become standard since the switch of all major players to HTTP streaming [4]. To provide a relevant benchmark, we chose MediaDrop [3], a redistributable open source platform supporting HTML5 video streaming that also included social media features such as comments, view counts and likes. Like modern video streaming platforms, MediaDrop offers content management, statistics and popularity ranking and social media aspects such as comments, Twitter and Facebook integration.

We have built an Amazon EC2 virtual machine image with a complete installation of MediaDrop ready to use. We will perform a demonstration of this virtual appliance that we have made publicly available on EC2.

We also provide the Ganglia monitoring tools to report on cpu, memory, disk and network utilization if the cloud infrastructure does not already report these metrics. When network throttling is needed, we rely on the Linux traffic controller hierarchical token bucket (tc-htb) to limit the MediaDrop server bandwidth.

2.2 Video datasets, tools, and workloads

While Video BenchLab allows a researcher to upload any video dataset into the MediaDrop appliance, for ease of use, we provide an initial video dataset for research use. All videos in our dataset were obtained from a set of videos on Vimeo[6] that are distributed under a Creative Commons Attribution and thus can be used and modified by researchers without licensing restriction. Table 1 summarizes our video dataset, which contains videos on topics such as news, general entertainment and sports.

Table 1. Video dataset specifications

Category	# of videos	Duration	Video size
Small	20 (10SD-10HD)	0-5 min	2.9-68MB
Medium	8 (4SD-4HD)	5-10 min	20MB-73MB
Large	5 (3SD-2HD)	10min – 1H+	18MB-203MB

Importing videos through the MediaDrop Web interface is a multistep process that involves a user uploading a video and an administrator approving the video for it to be available for others to stream. This process is too cumbersome for bulk video inserts or large dataset creations. We provide tools to perform bulk inserts of videos directly in the MediaDrop database to make them readily available with their thumbnails.

Typical BenchLab workload traces include a unique request id, a timestamp indicating when the request must be played, a client id identifying the browser, the URL to visit as well as optional interactions with Web elements in the visited page. The trace format for video workloads extends the original format by adding video manipulation commands. The supported commands are:

- `play`: play the video from the current position
- `pause`: pause the video at the current position
- `change_quality(`*`quality`*`)`: changes the quality of the video (player specific, e.g. in MediaDrop implemented as pause/load new media file/seek to previous current position/resume, in YouTube implemented as a call to setPlaybackQuality). *`quality`* is one of the pre-defined values `lowest`, `highest`, `lower` or `higher`, which changes to the lowest, highest, next lower available and next higher available quality, respectively.
- `skip_ads`: skip the in-stream advertisement (player specific, currently only available for YouTube).
- `quit`: end the playing right now and proceed to the next entry in the trace
- `seek(`*`video_position`*`)`: seek the video to the given position in seconds since the beginning of the video. The position is defined by a numerical value or a formula using the video `length` and `current_position` provided by the BCR. Example:
  - Seek to 1 minute into the video: `seek(60)`
  - Seek to the middle of a video: `seek(length/2)`
  - Jump 30 seconds: `seek(current + 30)`
- `wait_for(`*`timeout,[video_position]`*`)`: wait for the timeout to expire or the current video position to reach the given value, whichever comes first. The `wait_for` command should always be followed by another command. The timeout can either be a relative value in seconds or an absolute timestamp. The *`video_position`* parameter is defined in a similar way as the seek command:
  - Wait for the video to reach 1 minute playback within 2 minutes: `wait_for(120, 60)`
  - Wait for the video to reach the middle of the playback with a maximum 20% delay: `wait_for((length/2)*1.2, length/2)`
  - Wait until the given timestamp before executing the next command: `wait_for('2015-01-16 12:30:10.5')`
  - Wait until the video ends: `wait_for(0,length)`

The default script that just plays a video until its end is as follows:|

```
VideoBenchLab={player_type:html5|youtube,
  commands:{play, wait_for(0,length), quit}}
```

Multiple browsers can be synchronized by using the timestamp field in the trace. Recall that the trace format is: `<request id>,<client id>,<timestamp>,<URL>,<parameter>`

The following trace starts 3 clients at a 10 seconds interval, each client going first to the YouTube homepage and 5 seconds later playing a video (URLs are abbreviated):

```
1, 1, 2015-01-16 00:00:00.0, http://youtu.be,null
2, 1, 2015-01-16 00:00:05.0, http://youtu.be/…,VideoBenchLab={…}
3, 2, 2015-01-16 00:00:10.0, http://youtu.be,null
4, 2, 2015-01-16 00:00:15.0, http://youtu.be/…,VideoBenchLab={…}
5, 3, 2015-01-16 00:00:20.0, http://youtu.be,null
6, 3, 2015-01-16 00:00:25.0, http://youtu.be/…,VideoBenchLab={…}
```

2.3 Video BenchLab Client Runtime

A central contribution of Video BenchLab is the ability to replay traces through real Web browsers. The Video BenchLab Client Runtime (BCR) extends the open source Selenium [4] framework with functionalities to download a video trace, replay it via a real web Browser by issuing HTTP requests for video, record streaming performance statistics for each page and upload the results at the end of the replay. Unlike traditional load injectors that work at the network level, replaying through a Web browser accurately performs all the complex interactions between the browser and the server. When playing videos, the unmodified native player or plugin extensions of the browser are used giving insights on real world performance on any platform or software combination.

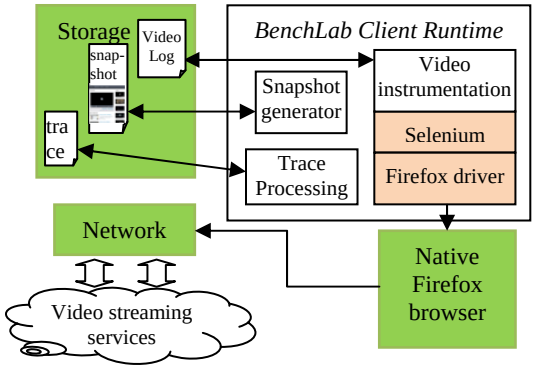

Figure 1.Video BenchLab Client Runtime (BCR) architecture

Figure 1 gives an overview of the architecture of the BCR with a Firefox browser as an example. We also support Chrome and Internet Explorer browsers. In order to benchmark HTML5 video, the BCR executes JavaScript on the video web page to collect performance metrics. The script instructs the browser to record video metadata, including MIME type, resolution and duration, current time, video position, buffer start and end positions, and, if supported by the player, when the video has stalled. This data can be used to extract information about potential lags or skips during the video replay. As each player might store these values differently, the data collection code is specific to each video player. We have implemented data collectors for the native standard HTML5 player found in every browser and the YouTube player.

2.4 BenchLab Dashboard to orchestrate experiments

The BenchLab Dashboard is used to setup and automatically run performance experiments as well as gathering experimental results. It provides a Web interface (standard Java WebApp) to interact with experimenters that want to create experiments using an arbitrary number of browsers and servers. The Dashboard gives an overview of the browsers currently connected, the experiments (created, running or completed) and the Web traces that are available for replay.

Video workload trace files are uploaded by the experimenter through a Web form and stored in the Dashboard database. The BenchLab Dashboard does not deploy, configure or monitor any server-side software. There are a number of deployment frameworks (Gush, WADF, JEE, .Net deployment service, etc) and monitoring tools (Ganglia and fenxi are popular choices) that users can choose from depending on their preferences.

Anyone can deploy a BenchLab Dashboard and therefore build his or her own experiment repository. An experiment defines what trace should be played and how. The user defines how many browsers should replay the sessions with eventual constraints (specific platform, version, location…). The experiment can start as soon as enough clients have registered to participate in the experiment. The Dashboard does not deploy client web browsers, rather it waits for the browsers to connect and its scheduler assigns them to experiments.

3. Demo

For this demo, we are deploying a BenchLab Dashboard and a BCR on the local laptop. We also deploy another BCR in Amazon EC2 as well as our MediaDrop server as shown on Figure 2. The trace file used for the demo is shown in Figure 3.

The first browser plays a video from a MediaDrop player running on EC2, waits for 5 seconds, then seeks to the middle of the video and waits for the video to finish with a maximum delay of 20%. The second browser plays a YouTube video that has in-stream ads. After 10 seconds the ads are discarded and the video is played for 5 seconds before being paused. The playing resumes for 5 seconds before switching the quality to a higher quality and waits for another 30 seconds before terminating.

Figure 2 gives an overview of the BenchLab components and how they interact to run an experiment. The Video BenchLab Client Runtime (BCR) starts and controls the native Web browser on the client machine. On startup, a BCR connects the browser to a BenchLab Dashboard (step 1 in Figure 2). When the browser connects to the Dashboard, it provides details about the exact browser version and platform runtime it currently executes on as well as its IP address and location if available.

A new experiment is created using the Web interface of the Dashboard that automatically redirects the BCR to a download page where it gets the trace for the session it needs to play. The BCR stores the trace on the local storage and makes the Web browser regularly poll the Dashboard to get the experiment start time. There is no communication or clock synchronization between BCRs, they just get a start time as a countdown in seconds from the Dashboard that informs them 'experiment starts in x seconds' through a Web form. The status of browsers is recorded by the Dashboard and stored in a database.

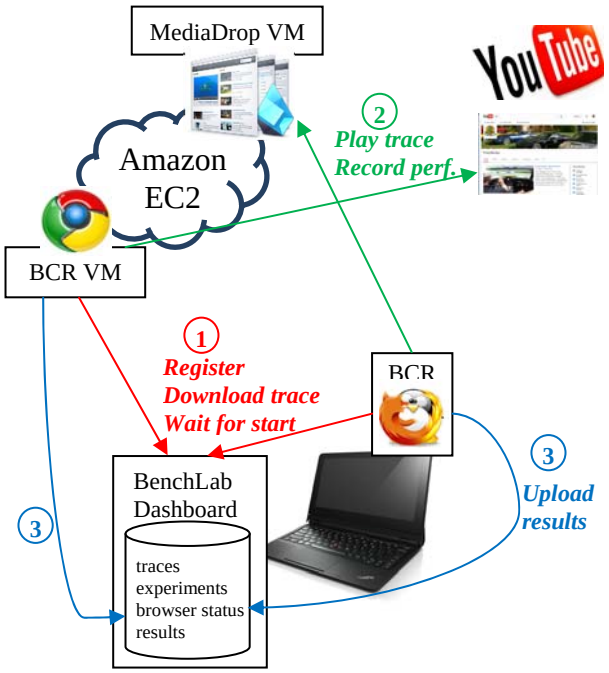

Figure 2.Video BenchLab experiment flow overview

```
1, 1, 2015-01-16 00:00:00.0,
   http://ec2.amazon.com/mediadrop/…,
   VideoBenchLab = {player_type: html5,
       commands: [
           play,
           wait_for(5),
           seek(length/2),
           wait_for(1.2*length/2,length),
           quit ]}
2, 2, 2015-01-16 00:00:00.0,
   http://youtube.com/…,
   VideoBenchLab = {player_type: youtube,
       commands: [
           play,
           wait_for(10),
           skip_ads,
           wait_for(5),
           pause,
           wait_for(5),
           play,
           wait_for(5),
           change_quality(higher or hd720),
           wait_for(30),
           quit ]}
```

Figure 3. Trace used for the experiment playing a video from MediaDrop and another from YouTube

When the experiment start time has been reached, the BCRs play their trace through their Web browser monitoring each interaction (step 2 in Figure 2). For each URL visited, BCRs record Web and video performance metrics. The results are uploaded to the Dashboard at the end of the experiment (step 3 in Figure 2). The Dashboard provides a number of ways of visualizing data and comparing results between experiments. The entire database with experiment configuration, traces and results can also be easily exported to be shared with other researchers.

Figure 4 shows a smooth HTML5 video streaming from our MediaDrop server running on EC2. The jump at the 5 second mark corresponds to the command to seek to the middle of the video. There is no buffer underrun and the video plays smoothly all along.

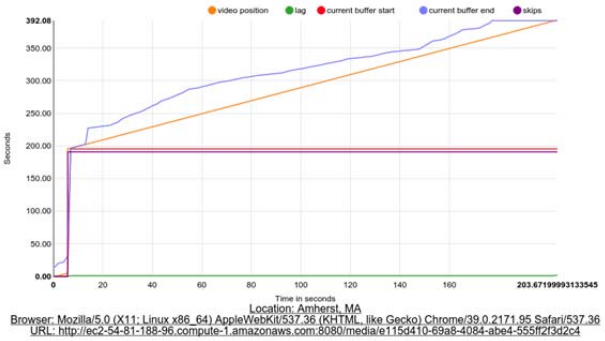

Figure 4. Analysis of an HTML5 video streaming from a MediaDrop server running on EC2

Figure 5 shows an example of the video analysis graphs computed by the BenchLab Dashboard. Besides the video buffer size and video current position over time, the amount of user perceived lag and the seconds of video skipped during replay are also shown to diagnose issues.

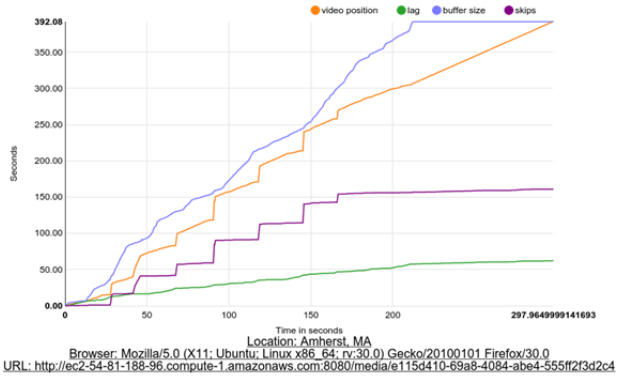

Figure 5. Sample of a MediaDrop video playing analysis using the BenchLab Dashboard

The Dashboard also allows the experimenter to compare different experiments and browsers to easily analyze experimental results at a large scale. We are currently working on expanding the set of analysis tools to automate result processing.

4. Implementation and Platform Availability

Video BenchLab is an open platform that is designed for benchmarking streaming media workloads. Hence, all components of the Video BenchLab software suite are freely available under an open source license and downloadable from SourceForge at http://sf.net/projects/benchlab. Additional details of our platform are available from the BenchLab project page at http://lass.cs.umass.edu/projects/benchlab. Our software suite comprises several components summarized in Table 2.

Table 2. Summary of tools in the Video BenchLab suite

Tool	Description	Platforms
MediaDrop	Video streaming server	Linux
mediaload	Bulk video import for MediaDrop	Linux/python
metadump	Video metadata export	Linux/python
loadgen	Video workload generator	Linux/python
BCR	Video BenchLab Client Runtime	Firefox+Chrome on Linux/Windows, IE on Windows
HTML5 collector	BCR data collector for HTML5 video players	All browsers
YouTube collector	BCR data collector for YouTube video player	All browsers
Dashboard	Experiment management, result db and analytics	Java Web container (e.g. Tomcat)

These components are available individually and as a packaged platform to permit flexibility in how they are used for experimentation. We have also published on SourceForge our experimental Dashboard database with all the results contained in our MMSys 2015 paper [5]. To help with reproducibility of experimental results, anyone can import this database in their own Dashboard and rerun the experiments. We have also provided Amazon EC2 images (AMIs) for MediaDrop, BCR and Dashboard so that anyone can reproduce the experiments conducted in this paper.

Acknowledgements: This research and the development of BenchLab and Video BenchLab is supported in part by a National Science Foundation grant 1339839.

5. REFERENCES

[1] ACM Multimedia Systems conference Dataset archive, http://traces.cs.umass.edu/index.php/Mmsys/Mmsys

[2] Philip Bräunlich and Gerrit van Aaken – *HTML5 Video Player Comparison* – http://praenanz.de, last update 2014-07-09.

[3] M. Larson, M. Soleymani, M. Eskevich, P. Serdyukov, R. Ordelman, and G. Jones. "The community and the crowd: Developing large-scale data collections for multimedia benchmarking." IEEE *Multimedia*, (2012).

[4] Li, Mingzhe, Mark Claypool, Robert Kinicki, and James Nichols. "Characteristics of streaming media stored on the Web." *ACM Transactions on Internet Technology (TOIT)* 5, no. 4: 601-626, 2005.

[5] P. Pegus, E. Cecchet, and P. Shenoy, "Video BenchLab: An Open Platform for Realistic Benchmarking of Streaming Media Workloads", in Proc. ACM Multimedia Systems Conference (MMSys), Portland, OR, 2015.

[6] Slingerland, Nathan T., and Alan Jay Smith. "Design and characterization of the Berkeley multimedia workload." *Multimedia Systems* 8, no. 4: 315-327, 2002.