Multimedia Streaming

Mike Zink
Technical Challenges

• Servers (and proxy caches)
  – storage
    • continuous media streams, e.g.:
      – 4000 movies * 90 minutes * 10 Mbps (DVD) = 27.0 TB
        15 Mbps = 40.5 TB
        36 Mbps (BluRay) = 97.2 TB
      – 2000 CDs * 74 minutes * 1.4 Mbps = 1.4 TB
Technical Challenges

- **Servers (and proxy caches)**
  - I/O
    - many concurrent clients
    - real-time retrieval
    - continuous playout
      - DVD (~4Mbps, max 10.08Mbps)
      - HDTV (~15Mbps, BlueRay ~36Mbps)
    - current examples of capabilities
      - disks:
        » mechanical: e.g., Seagate X15 - ~400 Mbps
        » SSD: e.g., MTRON Pro 7000 – ~1.2 Gbps
      - network: Gb Ethernet (1 and 10 Gbps)
      - bus(ses):
        » PCI 64-bit, 133Mhz (8 Gbps)
        » PCI-Express (2 Gbps each direction/lane, 32x = 64 Gbps)
  - computing in real-time
    - encryption
    - adaptation
    - transcoding
Outline

• Multimedia Servers
• Analysis of the YouTube streaming system
• Improving performance
  – Caching
  – Prefetching
  – Recommendation systems
Server Hierarchy

- Intermediate nodes or proxy servers may offload the main master server

- Popularity of data: not all are equally popular – most request directed to only a few

- Straight forward hierarchy:
  - popular data replicated and kept close to clients
  - locality vs. communication vs. node costs
General OS Structure and Retrieval Data Path

- Application
- File system
- Communication system

User space

Kernel space
Server Internals Challenges

- Data retrieval from disk and push to network for many users

- Important resources:
  - memory
  - busses
  - CPU
  - storage (disk) system
  - communication (NIC) system

- Much can be done to optimize resource utilization, e.g., scheduling, placement, caching/prefetching, admission control, merging concurrent users, ...
**Timeliness: Streaming**

- Start presenting data (e.g., video playout) at $t_1$

- **Consumed bytes (offset)**
  - variable rate
  - constant rate

- Must start retrieving data earlier
  - Data must arrive before consumption time
  - Data must be sent before arrival time
  - Data must be read from disk before sending time
Watch Global, Cache Local: YouTube Network Traffic at a Campus Network – Measurements and Implications
Overview

- Motivation
- Measurement
  - How YouTube Works
  - Monitoring YouTube Traffic
  - Measurement Results
- Distribution Infrastructures
  - Peer-to-Peer
  - Proxy Caching
- Conclusions & Future Work
Motivation

- YouTube is different from traditional VoD
- Access to YouTube from a campus network
- Influence on content distribution paradigms?
- Correlation between global and local popularity?

Methodology:
- Monitor YouTube traffic at campus gateway
- Obtain global popularity
- Video Clip traffic analysis
- Trace-driven simulation for various content distribution approaches
**How YouTube Works!**

1. **HTTP** Get MSG

2. **HTTP** Redirect MSG

3. **HTTP** Get MSG

4. Flash video stream

---

**YouTube Web server**

**CDN server located in YouTube or Limelight network**

**Monitor box**

**Client**

[Example of (1)]

Get /get_video?video_id=G_Y3y8escmA

HTTP/1.1

[Example of (2)]

HTTP/1.1 303 See other

Location: [http://sjc-v110.sjc.youtube.com/get_video?video_id=G_Y3y8escmA](http://sjc-v110.sjc.youtube.com/get_video?video_id=G_Y3y8escmA)
Monitoring YouTube Traffic

- Monitor web server access
  - Destination or source IP of YouTube web server pool
  - Analyze HTTP GET and HTTP 303 See Other messages

- Monitoring Video Stream
  - WWW access information to identify video stream
  - Construct flow to obtain:
    - Duration of streaming session
    - Average data rate
    - Amount of transferred payload data

<table>
<thead>
<tr>
<th>Trace</th>
<th>Date</th>
<th>Length (Hours)</th>
<th># of Unique Clients</th>
<th>Per Video Stats</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Total</td>
</tr>
<tr>
<td>1</td>
<td>05/08- 05/09</td>
<td>12</td>
<td>2127</td>
<td>12955</td>
</tr>
<tr>
<td>2</td>
<td>05/22-05/25</td>
<td>72</td>
<td>2480</td>
<td>23515</td>
</tr>
<tr>
<td>3</td>
<td>06/03-06/07</td>
<td>108</td>
<td>1547</td>
<td>17183</td>
</tr>
</tbody>
</table>
Measurement Results: Video Popularity

Request per video (Trace 1)

Request per video (Trace 2)

Request per video (Trace 3)
Measurement Results: Observations

- No strong correlation between local and global popularity observed: 0.04 (Trace1), 0.06 (Trace2), 0.06 (Trace3)
- Neither length of measurement nor # of clients observed seems to affect local popularity distribution
- Video clips of local interest have a high local popularity

http://www.youtube.com/watch?v=dp4MYii7MqA
Measurement Results: Requests per Client

Client in here means IP address (NAT, DHCP)

<table>
<thead>
<tr>
<th>Trace</th>
<th>Video clips with multiple requests from same client</th>
<th>Total number of requests</th>
<th>Max. number of requests per client</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2149</td>
<td>3100</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>3899</td>
<td>5869</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>3170</td>
<td>4893</td>
<td>47</td>
</tr>
</tbody>
</table>
Overview

- Motivation
- Measurement
  - How YouTube Works
  - Monitoring YouTube Traffic
  - Measurement Results
- Distribution Infrastructures
  - Peer-to-Peer
  - Proxy Caching
- Conclusions & Future Work
Distribution Infrastructures

- Trace-driven simulation based on traces 1, 2, and 3
- Create sequential list of requests
- Make use of results from stream flow analysis

<table>
<thead>
<tr>
<th>Trace</th>
<th>Duration (sec) (Length of viewing)</th>
<th>Packets</th>
<th>Payload Size (bytes)</th>
<th>Rate (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg</td>
<td>Max</td>
<td>Min</td>
<td>Avg</td>
</tr>
<tr>
<td>1</td>
<td>99.62</td>
<td>4421.00</td>
<td>0.04</td>
<td>5202</td>
</tr>
<tr>
<td>2</td>
<td>95.81</td>
<td>2359.83</td>
<td>0.53</td>
<td>4478</td>
</tr>
<tr>
<td>3</td>
<td>81.34</td>
<td>16956.28</td>
<td>0.04</td>
<td>4431</td>
</tr>
</tbody>
</table>
Simulation: Peer-to-Peer

- Peer availability based on flow trace file information
- Window-based availability approach
- Client availability influences hit rate
Simulation: Proxy Caching

- FIFO cache replacement
- Effective low cost solution since storage in the order of 100 GB is required
- Hit rates quite similar for all three traces compared to P2P results
Related Work

Parallel work to ours:

• Cha et al. (IMC 2007):
  • Only information from YouTube server is analyzed
  • No information about benefits of using caching in access networks

• Gill et al. (IMC 2007):
  • Similar motivation to ours
  • Only predefined set of content servers could be monitored
  • General trend between their and our results observable

No simulative study on different distribution architectures
Conclusions

• No strong correlation between local and global popularity observed
• Neither length of measurement nor # of clients observed seems to affect local popularity distribution
• Video clips of local interest have high local popularity
• Demonstrated implications of alternative distribution infrastructures
• Client-based caching, P2P-based distribution, and proxy caching can reduce network traffic and allow faster access
Watching User Generated Videos with Prefetching
**User Generated Videos**

- **Professional Produced Videos**
  - Netflix
  - Hulu

- **User Generated Videos**
  - YouTube, Youku, Tudou
  - Hundreds of millions of short video clips
  - Wide ranges of topics

- **Growing user generated videos**
  - Readily available device
  - Production cycle is short
Motivation

- User experience in watching videos is not satisfactory
  - Slow startup time
  - Many pauses during playback
Measuring User Experiences Watching YouTube

Video download traces from various environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Location</th>
<th>Network Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>University 1</td>
<td>Campus WLAN</td>
</tr>
<tr>
<td>E2</td>
<td>Company 1</td>
<td>DSL</td>
</tr>
<tr>
<td>E3</td>
<td>Home 1</td>
<td>DSL</td>
</tr>
<tr>
<td>E4</td>
<td>Apartment 1</td>
<td>Cable Internet</td>
</tr>
<tr>
<td>E5</td>
<td>Dormitory 1</td>
<td>Campus LAN</td>
</tr>
<tr>
<td>E6</td>
<td>Dormitory 2</td>
<td>Campus LAN</td>
</tr>
<tr>
<td>E7</td>
<td>Apartment 2</td>
<td>Cable Internet</td>
</tr>
<tr>
<td>E8</td>
<td>Town Library</td>
<td>Wireless Network</td>
</tr>
<tr>
<td>E9</td>
<td>Coffee shop</td>
<td>Wireless Network</td>
</tr>
<tr>
<td>E10</td>
<td>University 2</td>
<td>Campus WLAN</td>
</tr>
<tr>
<td>E11</td>
<td>Home 2</td>
<td>DSL</td>
</tr>
<tr>
<td>E12</td>
<td>Hotel</td>
<td>Wireless Network</td>
</tr>
</tbody>
</table>
Likelihood of Experiencing Pauses

- 10 out of 12 environments contain playbacks with pauses
- 41 out of 117 playbacks (35%) contain pauses

![Bar chart showing number of videos with disruptive and smooth playbacks across different environments (E1 to E12).]
Number of Pauses

- 31 out of 117 playouts (22.6%) contain more than 10 pauses
How to improve user experiences?
Video Prefetching Scheme

- Prefetching Agent (PA)
  - Select videos to be prefetched and retrieve their prefixes
  - Store prefixes of prefetched videos
  - At clients (PF-Client) or proxy (PF-Proxy)
- Predict videos that are most likely to be watched
  - PA determines videos to prefetch from incoming requests
How to select videos to prefetch?

- PA predicts a set of videos to be requested
- Two main sources of video requests
  - Search Result lists
  - Related Video lists
- Use top N videos from these lists
- Advantages
  - Simple
  - Require no additional data
  - Effectiveness?
Datasets for Evaluation

- Traces of data traffic between a campus network and YouTube servers

<table>
<thead>
<tr>
<th>Trace File</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>1 day</td>
<td>3 days</td>
<td>7 days</td>
</tr>
<tr>
<td>Start Date</td>
<td>20-Oct-09</td>
<td>8-Jan-10</td>
<td>28-Jan-10</td>
</tr>
<tr>
<td># Request</td>
<td>71,282</td>
<td>7,562</td>
<td>257,098</td>
</tr>
<tr>
<td># Unique Clients</td>
<td>7,914</td>
<td>607</td>
<td>10,511</td>
</tr>
<tr>
<td># Unique Videos</td>
<td>48,978</td>
<td>5,887</td>
<td>154,363</td>
</tr>
</tbody>
</table>

- Retrieve Search Result lists and Related video lists via YouTube data API
How Often Users Click on Related Videos and Search Results?

• Determine the referrers of each video request in the traces
  – From URL patterns, e.g., feature=related, feature=channel
  – From inference: look at a browse session to infer requests from Search Result list

• Related Video lists and Search Results lists are the most frequently used referrers
Evaluation Methodology

- Issue the requests based on real user request traces
- Keep track of the videos in PA's storage
- Evaluation metric
  - Hit ratio: How many requests we can serve from the PA's storage?

\[
\text{Hit ratio} = \frac{\text{Hit requests}}{\text{All requests}}
\]
Effectiveness of various scheme combinations

- Videos from a Related Video list of a user are watched by other users
- Best combination is using RV-N algorithm with PF-Proxy setting

![Graph showing hit ratio over N for different schemes](image)
Combining Caching with Prefetching

- Cache-and-Prefetch can reach up to 81% of hit ratio
- Improvement is smaller as N increases due to larger overlapping between prefetched videos and cached videos
Analyzing Hit Ratios

- Only half of the hit requests come from RV lists.
- Requests from SR lists is a large portion of the hit requests especially in PF-Proxy setting.
- Recommendation system is a good indicator of topic interest.
Analyzing the High Hit Ratios

- RV lists overlap with the video requests generated from other sources (esp. in PF-Proxy) up to 70%
• Measured in slots – a slot holds one prefix of a video
• One slot = 2.5 MB (for prefix size of 30% and average video size of 8.4 MB)
• Require only 5 TB to reach 81% of hit ratio (at N=25)
Impact of Storage space

- Hit ratio decreases with the storage space size
- Still can achieve hit ratio of around 60% with 125 GB (50k slots)
- Compared to caching, cache-and-prefetch always performs better
Do we need to prefetch the whole video?

- Prefetching the whole videos is not necessary.
- From analysis of video download traces, each location and each video requires different prefix size.
Feasibility – Traffic Overhead

- Suppose prefix size = 15%, N = 11 and caching whole videos

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Hit Ratio</th>
<th>Normalized load</th>
</tr>
</thead>
<tbody>
<tr>
<td>No scheme</td>
<td>0%</td>
<td>1.00</td>
</tr>
<tr>
<td>Cache-only</td>
<td>40%</td>
<td>0.60</td>
</tr>
<tr>
<td>Prefetch-only</td>
<td>66%</td>
<td>1.44</td>
</tr>
<tr>
<td>Cache-and-Prefetch</td>
<td>74%</td>
<td>1.02</td>
</tr>
</tbody>
</table>

- Caching helps reduce the traffic
- Pure prefetching yields higher hit ratio while increase traffic by 44%
- Combining the two results in highest hit ratio and only introduce 2% additional traffic
**Conclusion**

- Watching videos with prefix prefetching
  - Delay and Pauses are often
  - Prefix prefetching is feasible during browsing
  - Related videos are good interest predictors
  - Prefetching can reach hit ratio over 81% while caching can reach hit ratio of 40%
Cache-centric Video Recommendation: An Approach to Improve the Efficiency of YouTube Caches
Outline

• Motivation
• Approach
• Chain Analysis
• Cache Latency
• Related List Reordering
• Discussion
• Conclusion
**Motivation**

- YouTube is most popular user generated video service.

- Billions of videos with unequal popularity leads to long tail.

- Effective caching is difficult with such a long tail.

- Users usually select next video from related list.

- Caching and Prefetching of related list have shown to be effective.
Motivation (Contd.)
Approach

• Reordering of related list based on the content in cache.

• To verify the feasibility of reordering, we perform chain analysis.

• We also perform the RTT analysis to understand the origin of videos.
## Trace Details

<table>
<thead>
<tr>
<th>Trace File</th>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>3 Days</td>
<td>3 Days</td>
</tr>
<tr>
<td>Start Date</td>
<td>Feb 6(\text{th}) 2012</td>
<td>Jan 8(\text{th}) 2010</td>
</tr>
<tr>
<td>#Requests</td>
<td>105339</td>
<td>7562</td>
</tr>
<tr>
<td>#Related Videos</td>
<td>47986</td>
<td>2495</td>
</tr>
</tbody>
</table>

![CDF of the Ratio of Number of Videos in that Position](image-url)
Chain Analysis

• Loop Count – Video selection ending in loop.

• Chain Count – Video selection from related list until the last video selected by other means.
• Trace T1 – 84.76% chain count of 1 and 15.24% chain count of at least 2.

• Trace T2 – 48.2% chain count of 1 and 51.8% chain count of at least 2.

<table>
<thead>
<tr>
<th>Chain Count</th>
<th>Trace T1</th>
<th>Trace T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>1.195</td>
<td>2.304</td>
</tr>
<tr>
<td>Maximum</td>
<td>8</td>
<td>21</td>
</tr>
</tbody>
</table>
Loop Count

- Global analysis using PlanetLab.
- Loop length at fixed related video positions for 100 video requests.
Loop Count (Contd.)

- Loop length using random selections from the related list.
- Repeated 50 times for to obtain loop length.
Video Origin

- Requested 100 videos from Trace T1 and their related videos.
- Calculated RTT for the data session in the captured trace.

![Graphs showing CDF of video requests and cache levels](image-url)
Related List Reordering

From YouTube server:
- Videos already cached:
  - Video B
  - Video C
  - Video D

To client:
- Related video list from server:
  - Video A
  - Video B
  - Video C
  - Video D
  - Video E

Related video list to client:
- Video B
- Video C
- Video D
- Video A
- Video E

Position Centric selection
Content Centric selection
Reordering Approaches

• Content centric reordering
  – Related list selection based on content.
  – Position might change based on reordering.

• Position centric reordering
  – Related list selection based on position of original list.
  – Content might change based on reordering.
Reordering Results

<table>
<thead>
<tr>
<th>Trace</th>
<th>No Reordering</th>
<th>Content Centric</th>
<th>Position Centric</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>6.71%</td>
<td>6.71%</td>
<td>11.83%</td>
</tr>
<tr>
<td>T2</td>
<td>4.71%</td>
<td>4.71%</td>
<td>22.90%</td>
</tr>
</tbody>
</table>
Discussion

• Cost of Recommendation List Reordering.
  – Cost of cache depends on the cache structure and its size.
  – Using a plain hash table, worst case look up time will be $O(n)$.
  – Reordering comes with little extra cost but hit rate is more substantial.

• Reduction in Server Load.
  – Trace T1 cache hit rate increase from 6.71% to 11.83%, load reduction from 93.29% to 88.17%.
  – Trace T2 hit rate increase from 4.71% to 22.9%, load reduction of 18.19%.
Discussion (Contd..)

• Popularity based sorting of related list.
  - Reordering of related list is performed without taking into consideration of the popularity of videos in the cache.
  - Only significant differences in popularity would render the approach feasible.

• Adaptive video streaming.
  - Bandwidth adaptive video streaming contains different formats of same video.
  - Each format is a different file and caching them is not considered.
Conclusion

• We take advantage of user behavior of watching videos from related list.

• Our approach is to reorder the related list to move the content in the cache to top of the list.

• We present two approaches to reordering selection – Position centric and Content centric.

• Position centric selection leads to a high cache hit rate and reduction in server load due to reordering.