How Do You "Tube"?

Vijay Kumar Adhikari, Sourabh Jain, Yingying Chen and Zhi-Li Zhang *
Department of Computer Science & Engineering, University of Minnesota
Minneapolis, MN
{viadhi, sourj, yingying, zhzhang}@cs.umn.edu

ABSTRACT

In this paper we "reverse-engineer" the YouTube video delivery cloud by building a distributed measurement infrastructure. Through extensive data collection and analysis, we deduce the key design features underlying the YouTube video delivery cloud. The design of the YouTube video delivery cloud consists of three major components: a "flat" *video id space*, multiple DNS namespaces reflecting a multi-layered *logical* organization of video servers, and a 3-tier physical cache hierarchy. By mapping the video id space to the logical servers via consistent hashing and cleverly leveraging DNS and HTTP re-direction mechanisms, such a design leads to a scalable, robust and flexible content distribution system.

Categories and Subject Descriptors

C.2.4 [**Distributed systems**]: Distributed applications

General Terms

Measurement, Performance

1. INTRODUCTION

Given the traffic volume, geographical span and scale of operations, the design of YouTube's delivery infrastructure is perhaps one of the most challenging engineering tasks. Little is known how Google leverages its resources to design and structure the YouTube video delivery cloud to meet the rapidly growing user demands. This paper attempts to "reverse-engineer" the YouTube video delivery cloud through large-scale active measurement, data collection and analysis. We are particularly interested in answering the following question: how does YouTube design and deploy a *scalable* and *distributed* delivery infrastructure to match the geographical span of its users and meet varying user demands?

Towards this goal, we have developed a novel distributed active measurement platform with more than 1000 vantage points spanning five continents. Our distributed measurement platform consists of two key components: i) PlanetLab nodes that are used to play YouTube videos and and to perform DNS resolutions and ii) open recursive DNS servers to provide additional vantages to perform DNS resolutions. Through data analysis and inference, and by conducting extensive "experiments" to test and understand the behavior of the YouTube video delivery cloud, we uncover and deduce the logical designs of the YouTube video id space, the DNS namespace structures and cache hierarchy, how they map to the physical infrastructure and locations, and what mechanisms they use to select a server for any given request.

Most existing studies of YouTube mainly focus on user behaviors or the system performance. For instance, the authors in [3] examined the YouTube video popularity distribution, popularity evolution, and its related user behaviors and key elements that shape the popularity distribution using data-driven analysis. The authors in [4] investigate the (top 100 most viewed) YouTube video file characteristics and usage patterns such as the number of users, requests, as seen from the perspective of an edge network. A more relevant to our work is the recent study carried in [2], where the authors utilize the Netflow traffic data *passively* collected at various locations within a tier-1 ISP to uncover the locations of YouTube data center locations, and infer the load-balancing strategy employed by YouTube at the time. As the data used in the study is from 2008, the results reflect the YouTube delivery infrastructure *pre Google re-structuring*. This work attempts to reverse engineer the current YouTube design.

2. MEASUREMENTS & DATASETS

We develop a distributed active measurement and data collection platform consisting of the 471 PlanetLab nodes and 843 open recursive DNS servers. We use PlanetLab nodes to run our distributed crawler to crawl YouTube video pages and collect $434K$ video ids. We then play all those videos on PlanetLab nodes using our video player emulator and collect video playback traces that include all the hostnames and IP addresses involved in the video delivery. We use the open recursive DNS servers as additional vantage points to resolve the hostnames that appear in the video playback trace. Additionally, we measure round-trip delay to all observed IP addresses from all PlanetLab nodes.

3. YOUTUBE SYSTEM DESIGN

Analysis of the sequence of hostnames and IP addresses in the playback traces reveals that the YouTube video delivery cloud consists of the following three components. Due to space limitations, we refer the readers to [1] for more detais.

Video Id Space. Each video is uniquely identified using a "flat" identifier of 11 literals long, where each literal can be [A-Z], [0-9], - or _, thus forming a space of total 64^{11} ids.

*This work is supported in part by the NSF grants CNS-0905037, CNS-1017647 and CNS-1017092 and the DTRA Grant HDTRA1-09-1-0050.

SIGMETRICS'11, June 7–11, 2011, San Jose, California, USA.
ACM 978-1-4503-0262-3/11/06.

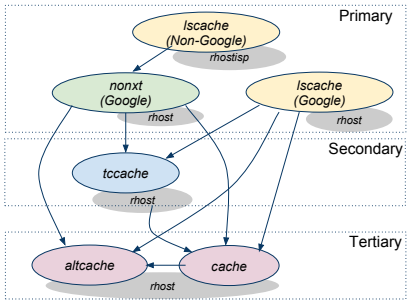

Figure 1: Namespace hierarchy and redirection order.

Table 1: Anycast (first 5) & unicast (last 2) namespaces.

namespace	format	hosts
lscache	v[1-24].lscache[1-8].c.youtube.com	192
nonxt	v[1-24].nonxt[1-8].c.youtube.com	192
tccache	tc.v[1-24].cache[1-8].c.youtube.com	192
cache	v[1-8].cache[1-8].c.youtube.com	64
altcache	alt1.v[1-24].cache[1-8].c.youtube.com	64
rhost	r[1-24].*cityid*.c.youtube.com	5,044
rhostisp	r[1-24].*isp-city*[1-3].c.youtube.com	402

Three-Tier Server Cache Hierarchy. Using the IP addresses seen in our datasets, we geo-map the "physical" video server cache locations, which are dispersed at five continents. In addition to cache locations inside Google, there are about a dozen physical caches hosted inside other ISPs such as Comcast and Bell-Canada. Based upon the roles of the servers we deduce that YouTube employs a 3-tier physical cache hierarchy with (at least) 38 *primary* cache locations, 8 *secondary* and 5 *tertiary* cache locations.

Multi-Layered Anycast DNS Namespaces. YouTube videos and (physical) cache hierarchy are tied together by a set of 5 (logical) *anycast* (can map to more than one IP address) namespaces as well as 2 *unicast* (maps to a unique IP address) namespaces as shown in Table 1.

4. MECHANISMS AND STRATEGIES

The layered organization of *logical* video servers enables YouTube to employ several mechanisms and strategies.

***Fixed* Mapping between Video Id Space and *Logical* Video Servers.** YouTube adopts a form of "consistent" hashing to map each video id uniquely to one of the hostname in each of the anycast namespaces. In other words, for *lscache* namespace, the video id space is uniformly divided into 192 sectors, and each *lscache* DNS name is responsible for a fixed sector. This *fixed* mapping between the *video id* space to the anycast namespaces makes it easier for individual YouTube front-end servers to generate – *independently and in a distributed fashion* – HTML pages with embedded URLs pointing to the relevant video users are interested in, regardless of where users are located or how logical servers are mapped to physical servers or cache locations. These fixed mappings make it easy for each (physical) video server to decide – given its logical name – what portion of videos it is responsible for serving.

Locality-Aware Video Cache Selection via DNS Resolution. YouTube employs *locality-aware* DNS resolution to serve user video requests regionally by mapping *lscache* hostnames to physical video servers (IP addresses) residing in cache locations reasonably close to users.

***Dynamic* HTTP Request Redirection.** The DNS resolution mechanism, while locality-aware, is generally agnostic of server load or caching status. When cache misses happen, depending on how busy a video server at the primary location, it may either directly fetch the missed video from another video server which has the video cached, or redirect the request to another video server at a secondary or tertiary location. Our analysis and experiments show that more than 18% times, a user video request is redirected from a primary video cache server selected via DNS *lscache* name resolution to another server.

YouTube employs a clever and complex mix of dynamic HTTP redirections and additional rounds of DNS resolution to perform finer-grained dynamic load-balancing and to handle cache misses. For instance, our investigation shows that YouTube utilizes the layered *anycast* namespaces to redirect video requests i) from one location to another location (especially from a non-Google primary cache location to a Google primary cache location via the use of *nonxt* namespace); and ii) from a Google cache location in one tier to another tier (the primary to secondary or tertiary, or the secondary to tertiary via the use of the *tccache* *cache* and *altcache* namespaces). There is a *strict ordering* as to how the *anycast* namespaces are used for redirection (see Fig. 1). At each step of the redirection process, the corresponding *anycast* hostname is resolved to an IP address via DNS. YouTube also utilizes the *unicast* namespaces to dynamically redirect a video request from one video server to a specific server usually (more than 90% of times) *within the same cache location*, and occasionally in a different location. The use of the layered *anycast* namespaces enables to enforce an strict ordering and control the redirection process.

On the other hand, each redirection (and DNS resolution) process incurs additional delay. Up to 9 redirections may happen, although they are rarely observed in the video playback traces we collected.

5. CONCLUSIONS

In this paper, we reverse-engineer the YouTube video delivery cloud by building a distributed active measurement platform. Through extensive data collection, measurement and analysis, we have uncovered and geo-located YouTube's 3-tier physical video server hierarchy, and deduced the key design features of the YouTube video delivery cloud.

6. REFERENCES

[1] How Do You "Tube"? Reverse Engineering the YouTube Video Delivery Cloud (Technical report). `http://www-users.cs.umn.edu/~viadhi/resources/youtube-tech-report.pdf`.

[2] V. K. Adhikari, S. Jain, and Z. Zhang. YouTube Traffic Dynamics and Its Interplay with a Tier-1 ISP: An ISP Perspective. In *IMC '10*. ACM, 2010.

[3] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *IMC '07*. ACM, 2007.

[4] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Youtube traffic characterization: a view from the edge. In *IMC '07*. ACM, 2007.