Cloud Based Video-on-Demand Service Model Ensuring Quality of Service and Scalability

Carlos Barba-Jimenez, Raul Ramirez-Velarde, Andrei Tchernykh, Ramón Rodríguez-Dagnino, Juan Nolazco-Flores, Raul Perez-Cazares

PII: S1084-8045(16)30086-8
DOI: http://dx.doi.org/10.1016/j.jnca.2016.05.007
Reference: YJNCA1649

To appear in: Journal of Network and Computer Applications

Received date: 7 September 2015
Revised date: 29 February 2016
Accepted date: 10 May 2016

Cite this article as: Carlos Barba-Jimenez, Raul Ramirez-Velarde, Andrei Tchernykh, Ramón Rodríguez-Dagnino, Juan Nolazco-Flores and Raul Perez-Cazares, Cloud Based Video-on-Demand Service Model Ensuring Quality of Service and Scalability, Journal of Network and Computer Applications http://dx.doi.org/10.1016/j.jnca.2016.05.007

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting galley proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.
Cloud Based Video-on-Demand Service Model Ensuring Quality of Service and Scalability

Carlos Barba-Jimenez\textsuperscript{a}, Raul Ramirez-Velarde\textsuperscript{a,}\textsuperscript{*}, Andrei Tchernykh\textsuperscript{b}, Ramón Rodríguez-Dagnino\textsuperscript{a}, Juan Nolazo-Flores\textsuperscript{a}, Raul Perez-Cazares\textsuperscript{a}

\textsuperscript{a}Tecnologico de Monterrey, Campus Monterrey, Ave. Eugenio Garza Sada 2501, Monterrey, N.L., Mexico, 64849
\textsuperscript{b}CICESE Research Center, Carretera Ensenada-Tijuana 3918, Ensenada, B.C., Mexico, 22860

Abstract

Increasing availability and popularity of cloud Storage as a Service (STaaS) offers alternatives to traditional on-line video entertainment models, which rely on expensive Content Delivery Networks (CDNs). In this paper, we present an elastic analytic solution model to ensure Quality of Service (QoS) when providing Video-on-Demand (VoD) using several third party elastic cloud storage services. First, we individually gather cloud storage start-up delays, and characterize them to show that they are heavy-tailed. Then, we perform a meta-characterization of these delays using Principal Component Analysis (PCA) to create a characteristic cloud delay trace. By using different estimation techniques of the Hurst Parameter, we demonstrate that this new trace (also heavy-tailed) exhibits self-similarity, a property not sufficiently studied in cloud storage environments. Finally, we pursue stochastic modeling using different heavy-tailed probability distributions to derive prediction models and elasticity parameters from the cloud VoD system. We obtain a stochastic self-similar model and compare it with trace based simulation results by testing different heavy-tailed probability distributions, meta-cloud elasticity values and Hurst parameters. Since our approach optimizes QoS, we guarantee a specific video start-up delay for a number of arriving clients. This is a strong commitment for a VoD service, because traditional cloud approaches often focus on a best-effort paradigm optimizing performance, cost, and bandwidth, among other parameters.

Keywords: Video-on-Demand, Cloud computing, Elasticity, Heavy-tails, PCA, Self-similarity.

1. Introduction

Nowadays, the cloud paradigm has become increasingly popular, offering new services and products every year. The concept has reached areas beyond traditional IT environments, research, software development, and even entire business models. Providers such as Amazon offer elastic computing and cloud Storage as a Service (STaaS) commercially. These services have increased interoperability, usability and reduced cost of application hosting, content storage and delivery (Buyya et al., 2010). These conditions open the door for creating new services that can satisfy existing and future user demands. This is especially important if one considers the trends of IP traffic reported in Cisco (2013), where it is shown that Video-on-Demand (VoD) traffic will be tripled in 2017 with rising mobile content consumption (Passarella, 2012).

VoD and online video content services are generally supported by a centralized delivery architecture based on private or rented servers with fixed costs and little flexibility (Buyya et al., 2009). Such a model poses a challenge, since predicting the user demand erroneously could cause performance bottlenecks with an under-estimation, and high costs with an over-estimation. The model has to be adapted to include a Content Delivery Network (CDN), usually operated by a third party in multiple external sites (Buyya et al., 2009). CDNs have a central entity that can enforce Quality of Service (QoS) levels, but this comes at a non-negligible financial cost (Passarella, 2012).

There are Peer-to-Peer (P2P) alternatives, but rarely provide guaranteed services (Passarella, 2012). They are used primarily in live streaming, where they help with flash crowds that need to access certain content at the same time (Mansy and Ammar, 2011). This is not always the case in a VoD service. Now, the CDN model is still the most prevalent, with services like Akamai. The recent P2P propositions in Thomas et al. (2015) have included a hybrid cloud component to facilitate video streaming by taking advantage of characteristics from both technologies, but only focus on a best-effort approach to QoS.

There are third party services (e.g. YouTube, Vimeo, etc.) for delivering video content. However, even with large infrastructures behind, they only offer best-effort QoS. It makes them not suitable for all businesses and entertainment conditions.

STaaS can be used as an alternative, with some commercially available services that use characteristics and ad-
vantages of the cloud, in a similar manner as proposed under the MetaCDN project (Broberg et al., 2009). The authors describe characteristics of the model, but left out details of the algorithms. The described statistical analysis is also limited for accurate predictive models. Our work has a similar motivation.

We extend the preliminary work presented in Barba-Jiménez et al. (2014) by widening the scope of our study and the problem definition. We include new parameters and characteristics to the proposed end-to-end VoD service. Furthermore, in this paper, we consider the derivation of the different stochastic models for predicting user capacity under certain QoS level and start-up delay taking into account the self-similarity property of the Characteristic Cloud Delay Trace (CCDT), and elasticity characteristics of the cloud. Additionally, we compare and evaluate the model and uncertainties of its parameters against simulation results. These aspects were not fully elucidated neither in Broberg et al. (2009) nor in Barba-Jiménez et al. (2014).

In this paper, we use third party cloud services to create a meta-VoD service (similar to MetaCDN) using a gateway as suggested in Islam and Grégoire (2012). This gateway creates a CDN like functionality sitting in a layer above the clouds. It takes into account some of the main challenges of VoD content delivery, namely, response time, and start-up delay.

As a metric for the meta-VoD service quality, we use start-up delay. We take into account the abandonment rate described in Krishnan and Sitaraman (2012), where the authors found that VoD clients start leaving the service after a start-up delay of 2000ms (milliseconds), losing 5.8% of users for each additional second. This delay time $T_D$ includes all network times, server times, and additional overheads.

We propose a methodology to model and analyze storage and elasticity in VoD content delivery, considering their statistical characteristics. Then we develop a meta-VoD elastic model that estimates the number of users that can be served for a given QoS and start-up delay time. This model also uses self-similarity and heavy-tail properties, following Ramirez-Velarde et al. (2013) and Ramirez-Velarde and Rodríguez-Dagnino (2010).

We provide background of our work and introduce an elasticity concept in Section 2. We discuss related work in Section 3. Then, in Section 4, we present the basic solution model. Section 5 includes a detailed statistical analysis regarding real cloud data delay traces. It includes the determination of heavy-tails. We characterize real cloud data using statistical analysis, determine the heaviness of each individual cloud delay time tails and then reduce the dimensionality of these data sets using Principal Component Analysis (PCA). We provide a meta-characterization of the individual clouds with the CCDT. The objective is to simplify the model for the delay time $T_D$, which enables us to make predictions of the probability of successful service under a certain threshold of abandonment rate. Section 6 introduces the concept of self-similarity and different estimations using the CCDT. Additionally, the self-similarity and elasticity are included to derive delay stochastic models using sub-exponential probability distributions. Section 7 validates the models by experimentation. Sections 8 and 9 describe the results and conclusions.

2. Background

2.1. Cloud Computing

The cloud is defined as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort and service provider interaction. Cloud computing has three service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) (Mell and Grance, 2009).

Cloud providers offer service level agreements (SLAs), which guarantee a level of QoS. Resource usage is monitored, controlled, and reported, providing transparency for both the provider and consumer (Espadas et al., 2013). This makes the idea of using the existing third-party cloud services for content distribution very attractive. Paying only for storage and computing, instead of having a potentially expensive contract with one CDN or an under-provisioned private server.

2.2. Cloud Elasticity

One of the main characteristics of cloud computing is the pay-per-use model. In order to provide metered services and resources under SLAs, the cloud providers must be able to match the resource demand with the resource offer as close as possible. Elasticity can be defined as the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible (Herbst et al., 2013).

In real scenarios, clouds are not perfectly elastic (Brebner, 2012). The infrastructure cannot respond instantly to sudden, significant increases in demand. There is a delay between the time when resources are requested, and when the application starts running.

Elasticity has been studied in several works, including Almeida et al. (2013), Costa et al. (2013), Herbst et al. (2013) and Kaur and Chana (2014), for frameworks considering costs, QoS, under/over provisioning and task execution. Ideally, there would be an external way of measuring or polling a measure of elasticity at any given time. However, obtaining these values from outside the cloud black-box is not easy.

We denote this elasticity metric as $\xi$, where $0 < \xi \leq 1$. In this definition, 1 describes a 100% elastic cloud system, where the resources always match the demand in every
3. Related Works

We address the video CDN based on the cloud. In Broberg et al. (2009), the authors present the price comparison of delivering content through a CDN versus different cloud providers. The traditional CDN model is the most expensive in TB data/month, while cloud and cloud CDN options come as the cheaper alternatives. The authors presented interesting points related to QoS provisions. However their proposal is vague for a proper mathematical setting.

In a recent work, the use of cloud CDN-like functionality has also been reported (Guan and Choi, 2014). However, it is aimed at minimizing bandwidth and cost in the content placement problem (from a provider point of view). In contrast, we consider the latency and QoS as user centric criteria.

Additionally, on the topic of CDN modeling, in Pathan and Buyya (2009), the authors explore resource discovery and request redirection in a multi provider content delivery network environment. They show that CDNs evolve as a solution for Internet service degradations and bottlenecks due to large user demands to certain web services. They address some of the internal problems that CDN providers face, like the break down of system locations, increase of utilization rates, over-provisioning and external resource harnessing to satisfy a certain SLA. The proposal is to interconnect a constellation of CDNs that collaborate for short or long periods of time to handle the different workload situations. This constellation uses a load distribution scheme with a request redirection and a mediator or coordinator agent that redirects load according to the workload of the different CDNs.

In Qi et al. (2012), the authors describe a QoS aware composition method supporting cross-platform service invocation in cloud environments, which explores a web service composition attaining a QoS optimal solution.

In Calheiros et al. (2012), a coordinator for scaling elastic applications across multiple clouds is used. The main focus is in cloud elastic work applications, where the application characteristics and usage in the cloud are better known (although provisioning is still a challenge). The authors do not address content delivery, like a VoD service, since the application characteristics of a web application and content storage in the cloud have different behavior. However, it does give some ideas to take into consideration for our work. We also take into consideration previous work related to VoD specific topics like the specification of an integrated QoS model for VoD applications (Sujatha et al., 2007).

4. Cloud Based VoD Service Solution Model and Problem Definition

In this paper, we address a cloud based CDN-like VoD service. Our solution model uses the concept of peering CDNs, where several content delivery networks cooperate to fulfill various requests. In our case, the cooperating CDNs are the different clouds without the concepts of authoritative rights. Our model works with STaaS and uses IaaS for required computing. The solution model consists of the following main components:

1. Asynchronous Resource Discovery: Since each cloud provider operates individually, this module keeps track of available resources.
2. Asynchronous Resource Monitoring: This module takes care of probing the different providers.
3. QoS Aware Resource Redirector: This module is in charge of analyzing the information available from the clouds and redirecting the load to the best re-source location and provider.

Figure 1 shows the basic components of the solution model, with details of the gateway and data connections flows. User requests go through the gateway, which has the redirection and resource monitoring logic. The gateway gives back an address of redirection for the content to one of the cloud providers. Here, the various cloud storage providers are working in a similar manner as the Content Distribution Inter networking model, in which the gateway considers the clouds and their networks as black-boxes (Pathan and Buyya, 2009).

It differs from the main model because cloud providers usually do not have mechanisms that will allow peering, and there is no central entity or supervisor with access to intra cloud information. One of the main challenges is to work without knowing the request and response loads of each inter cloud networks in real time, which is essential for the peering CDN scheme and algorithms (Pathan and Buyya, 2009). In order to overcome these limitations, we make conclusions based on the response or start-up delay. We use the total delay times, since they can be monitored and discovered from outside of the black-boxes. Additionally, our assumptions are based on the published requests per second statistics from some cloud providers like Amazon (Barr, 2012).

To determine how many users the service can handle under a maximum delay time, we have to take into consideration the abandonment rate times (Krishnan and Sitaraman, 2012). Once we make predictions about users and system scalability, the problem of the redirector is to minimize the delay time that the end user has when connecting to the final cloud provider, taking into consideration the redirection time caused by the decision making time in the gateway. In addition, the downtime status and QoS levels of each provider are aggregated to the model parameters.

The delay time $T_{D_{ij}}$ of the user $i$ to access the content of a certain provider $j$ is determined by:

$$T_{D_{ij}} = T_{r_{ij}} + T_{n_{ij}} + T_{O_{g}}$$ (1)

where $T_{r_{ij}}$ is the response time from the cloud storage, $T_{O_{g}}$ is the delay in the gateway caused by the redirection overhead, and $T_{n_{ij}}$ is the network time the request takes to travel from client $i$ to cloud $j$. We follow a simple scheme to minimize the redirection time and cost by using the following objective function (similar to Pathan and Buyya (2009)):

$$\min \sum_{i=1}^{n} \sum_{j=0}^{m} R_{c}(i,j)S_{ij}C_{j}Q_{j}$$ (2)

They are located. $C_{j}$ is the completion rate of the provider $j$ and can take a value in $[0,1]$, and finally $Q_{j}$ is the quality of service that provider $j$ has. It can take a value in $[0,1]$ as well.

The solution model from Eq. (2) does not consider the stochastic nature of the response and network times or its probability distribution. Considering the probabilistic nature of $T_{D_{ij}}$, taking into account viewer behavior (Krishnan and Sitaraman, 2012), and the 2000ms abandonment start time $\theta$ as a measure of a successfully redirected request, we create a new model for a client $i$ connected to cloud $j$ using the following premise:

$$P(T_{D_{ij}} > \theta) < \varphi$$ (3)

where $\varphi$ is the probability that the delay time is over the $\theta$ limit (2000ms in our case). Using Eq. (3) as a base, and assuming that $T_{D_{ij}}$ is heavy-tailed, we can model it using Pareto, Lognormal or Weibull distributions. However, we first provide a statistical analysis of different real life cloud delay times to use the best fitted probability distribution.

5. Cloud Data Analysis

We use real cloud delay times to have a good estimation of the extreme values and distribution. These delay times are obtained using the PRTG Network Monitor from Paessler (Paessler, 2014). The measurements are taken from an Amazon EC2 instance (small) located in N. Virginia, acting as a client, which is connected to different cloud providers to gather the total delay time (consisting of disk access times, memory access times, and network times) required to retrieve a 65 Kilobyte file (representing a worst case video frame size). These measurements have been taken every 60s over a period of 40 days for 6 cloud storage providers. In Table 1, we see their descriptive statistics.

<table>
<thead>
<tr>
<th>Provider</th>
<th>Mean Delay (ms)</th>
<th>Max Delay (ms)</th>
<th>Std Deviation (ms)</th>
<th>Kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>65.74</td>
<td>3425</td>
<td>97.8</td>
<td>170</td>
</tr>
<tr>
<td>GoGrid</td>
<td>22.88</td>
<td>11768</td>
<td>175.1</td>
<td>959.6</td>
</tr>
<tr>
<td>Rackspace</td>
<td>118.17</td>
<td>8790</td>
<td>101.2</td>
<td>1168</td>
</tr>
<tr>
<td>CloudFront</td>
<td>37.26</td>
<td>3001</td>
<td>86.6</td>
<td>152.4</td>
</tr>
<tr>
<td>S3 USA</td>
<td>188.24</td>
<td>27059</td>
<td>284.6</td>
<td>2725.9</td>
</tr>
<tr>
<td>S3 EU</td>
<td>652.48</td>
<td>22653</td>
<td>411.4</td>
<td>386.66</td>
</tr>
</tbody>
</table>

In Figure 2, we see an example of a time series (which we call trace) for the Cloudfront delay times.

The bursty behavior of the delay times is clearly observed. However, to make a conclusion, we have to look at the distribution of the values. Figures 3, 4 and 5 show the histograms for each of the clouds. They reveal delay times that are several orders of magnitude above the mean. In addition, the high kurtosis values presented in
Table 1 could indicate heavy-tailed distributed data (DeCarlo, 1997). To confirm it, more tests are conducted.

Taking a look at the definition of heavy-tails (Cooke and Nieboer, 2011; Crovella et al., 1998; Willinger et al., 1998), let X be a random variable with cumulative distribution function (CDF) \( F(x) = P[X \leq x] \) and its complement \( F(x) = 1 - F(x) = P[X > x] \). \( F(x) \) is heavy-tailed if \( F(x) \sim cx^{-\alpha} \), where \( c \) is a positive constant and \( \alpha \) is generally 0 < \( \alpha \) < 2. By extending the range to 0 < \( \alpha \) < \( \infty \), we obtain the sub-exponential distributions class, which is also considered as having heavy-tail distributions (Crovella, 2001).

An estimation of the tail index is usually provided by parametric methods, where a cut off point \( x_0 \) value is determined. The tail index is then calculated using Log-Log Complementary Functions (Cirillo, 2013; Crovella, 2001), or Hills Estimators (Adler et al., 1998; Borak et al., 2005; Resnick and Rootzen, 2000; Resnick, 1998, 1997; Willinger et al., 1998). However, Crovella and Taqqu (1999) propose a method based on scaling estimation, which is non-parametric. It produces a single \( \alpha \) estimate and has been proved to be effective even if the empirical data does not exactly follow a Pareto or power law in all curve sections of its distribution. We present the results of the tail index \( \alpha \) estimations for the cloud data sets in Table 2.

### Table 2: Cloud Delay Times Distribution Tail Index Estimations

<table>
<thead>
<tr>
<th>Cloud</th>
<th>( \alpha ) Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>CloudFront</td>
<td>1.445</td>
</tr>
<tr>
<td>Rackspace</td>
<td>1.472</td>
</tr>
<tr>
<td>GoGrid</td>
<td>0.697</td>
</tr>
<tr>
<td>Google</td>
<td>1.265</td>
</tr>
<tr>
<td>S3usa</td>
<td>1.138</td>
</tr>
<tr>
<td>S3eur</td>
<td>1.758</td>
</tr>
</tbody>
</table>

All cloud traces exhibit heavy-tailed behavior according to their tail index calculations. Additionally, if we take into consideration that their histograms are not symmetrical, and the high variability, we conclude that the data cannot be modeled by a symmetric probability distribution (such as a Gaussian) or a short-term memory process like a Markov one (Ramírez-Velarde and Rodríguez-Dagnino, 2010).

Now, in order to generate a single model of the VoD system, we have to select a cloud with the most representative behavior. The various clouds have similar statistics and histograms, but have different kurtosis, means, etc. We deal with the existence of several cloud data sets as a problem of multidimensionality. We consider each cloud as a dimension for the total delay time. Therefore, we use Principal Component Analysis (PCA) to determine a Characteristic Cloud Delay Trace (CCDT), as in Ramírez-Velarde et al. (2013), to capture as much of the variability of all clouds as possible. The goal is to reduce dimensionality to at most 2 components, from our available 6, to
generate a single trace that can be analyzed and used for modeling.

To find the Principal Components (PCs), we build a correlation matrix using the delay traces for each cloud. Traces are used as columns to indicate the dimensions with \( \sim 50k \) different observations. We subtract the overall mean \( \mu \) from each element. We use the mean of all values, not on a per cloud basis (180.83 ms). Then we use Pearson’s correlation coefficient, since all observations are in the same ms units.

Figure 6 shows a 3D representation of the scores against observations and components from the PCA procedure. The resultant PCs, eigenvalues and variability are shown in Table 3.

We select \( m \) components that will keep most of the variability present in \( p \) variables, with \( m \ll p \) for a dimensionality reduction. From Table 3, we see that selecting 1 PC will give us \( \sim 50\% \) of the original data variability, but if we select 2 PCs we have \( \sim 81\% \). After that, the percentage of added variability of each extra PC is \(< 10\% \). Under these conditions, choosing 2 PCs give us significantly more information than the other 4, so the CCDT is generated by using PC1 and PC2.

Using these PCs, we reconstruct a single trace that has the same variance that the whole set of cloud delay time traces \( \sigma_{\text{original}}^2 = 98894.582 \). The reconstruction also has to take into account the subtracted overall mean \( \mu \) while doing the PCA procedure. To reconstruct it, we start with the following:

\[
E[PC1] = 0 \quad E[PC2] = 0 \quad (4)
\]

\[
\text{Var}[PC1] = \lambda_1 \quad \text{Var}[PC2] = \lambda_2, \quad (5)
\]

where \( E \) is the expected value, \( \text{Var} \) is the variance, and \( \lambda_i \) is the eigenvalue for the corresponding PCi. We propose to create a new variable \( C \):

\[
C = PC1 + PC2. \quad (6)
\]

The variable \( C \) has the following expected value, variance, and standard deviation:

\[
E[C] = E[PC1] + E[PC2] = 0 \quad (7)
\]

\[
\text{Var}[C] = \text{Var}[PC1] + \text{Var}[PC2] = \lambda_1 + \lambda_2 \quad (8)
\]

\[
\sigma_c = \sqrt{\lambda_1 + \lambda_2}. \quad (9)
\]

Then, to create the final reconstructed trace \( X \), which must have a \( \sigma_X^2 = \sigma_{\text{original}}^2 = 98894.582 \), we use the following transformation (taking into account \( \mu \)):

\[
X = \mu + \sigma_{\text{original}} \cdot C \sqrt{\lambda_1 + \lambda_2}. \quad (10)
\]

Obtaining the expected value and variance for \( X \), we get:

\[
E[X] = E[\mu] + \frac{\sigma_{\text{original}}}{\sqrt{\lambda_1 + \lambda_2}} E[C] = \mu \quad (11)
\]

\[
\text{Var}[X] = \text{Var}[\mu] + \frac{\sigma_{\text{original}}^2}{\lambda_1 + \lambda_2} \text{Var}[C] = \sigma_{\text{original}}^2. \quad (12)
\]

The variance is the same in our new reconstructed trace \( X \) as in the \( \sigma_{\text{original}}^2 \), evidenced by Eq. (12) and numerically tested. This transformation and properties hold for more than 2 PCs, they have to be added to Eq. (6) and their respective eigenvalues to Eq. (10). We apply the transformation to each row of the \( PC1 + PC2 \) to obtain the Characteristic Cloud Delay Trace (CCDT). We see the result in Figure 7. The CCDTs behavior is similar to the
6. Self-Similarity

To characterize the cloud delay times, we have to take into account its probability distributions and statistical properties. From Park and Willinger (2000) and Leland et al. (1994), we know that to analyze network performance and video traffic, we should consider the self-similarity of the data. Self-similarity in this context is a property that denotes that the data is bursty over a range of time scales. It contradicts the notion made in most traffic and service models where exponential models are used, and burst modeling only holds for very limited range of time scales (Willinger et al., 1998).

There are several, not equivalent, definitions for self-similarity. The most common one for data traffic flows in networks establishes that a continuous-time stochastic process \( Y(t), t \geq 0 \) is self-similar, with a self-similarity or Hurst parameter \( H \), if it satisfies the following conditions (Willinger and Paxson, 1998).

\[
Y(t) \overset{d}{=} Y(at), \forall a > 0, \quad 0 < H < 1.
\]  

The equality means that the expressions have an equivalent probability distribution. A process satisfying Eq. (13) can never be stationary, but it is assumed to have stationary increments. The Hurst parameter is further studied by Lobo et al. (2013), Kirichenko et al. (2011), Clegg (2006) and Abry and Veitch (1998). They present similar conclusions about values that indicate self-similarity \((0.5 < H < 1)\).

6.1. Self-Similar nature of the CCDT

The Hurst parameter \( H \) estimation is defined mathematically, however, it is not easy to measure it using real data traces. Several Hurst parameter estimators, rather than just one should be used to avoid false conclusions (Clegg, 2006). It must be said that all estimators are vulnerable when the data has a trend or a periodic component (Clegg, 2006; Kirichenko et al., 2011).

In this paper, we use the data filtering techniques described in Clegg (2006) (Linear de-trend and Poly de-trend) and nine methods to estimate the Hurst parameter: the R/S (Rescaled/Range), aggregate variance, periodogram, boxed periodogram, Abry-Veitch method (Abry and Veitch, 1998), Whittle’s estimator, absolute moments, Higuchi’s Method, and Peng’s variance of residuals. More information on each individual estimator can be found in Clegg (2004). Table 4 summarizes the results.
We can conclude that (for the CCDT) $0.67 < H < 0.82$. This is a definite indication of self-similarity.

### 6.2. Self-Similar Client Delay Model

Now, when the self-similar nature of the data has been established, we present a model to determine the number of clients $n$ for a given maximum delay time $\theta$. These clients have to be handled concurrently by a VoD service residing on a cloud represented by the CCDT. Since we are simplifying the multivariate nature of multiple clients connecting to multiple clouds to one single cloud, we define the $T_{Di}$ delay time that the user $i$ needs to access the content in a provider $j$ as $T_{Dij}$.

We characterize $T_{Di}$ in the following manner. Let $Y_i$ ($i = 1, \ldots, n$) be a discrete-time random process representing the total cumulative delay for client $i$ connecting to a video in a cloud. Let $X_i = Y_i - Y_{i-1}$ be the strictly positive increment process. We find the client load $n$ such that:

$$P[T_{Di} > \theta] = P[X_1 + X_2 + X_3 + \cdots + X_n > \theta] = \varphi. \quad (14)$$

However, this model does not take into account the elastic nature of clouds (see Section 2.2), when they dynamically scale capacity and available resources according to the demand from $n$ clients. This paper introduces the elasticity $\xi$ into the model under the assumption that the cloud is not perfectly elastic, this means a non-linear relationship between cloud resources and $n$ clients or $\xi < 1$. Taking this into account for the right side of the inequality, we propose:

$$P[X_1 + X_2 + X_3 + \cdots + X_n > \theta n^\xi] = \varphi. \quad (15)$$

Now, let $Y_j = Z_t$ be the delay time process for client $t$. Let $Z_t$ be a self-similar process with $E[Z_0] = 0$ and $E[Z_t^2] = \sigma^2 t^2 H$. The accumulation process is then:

$$\sum_{i=1}^{n} X_i = Y_n - Y_{n-1} + \cdots + Y_1 - Y_0. \quad (16)$$

Following Park and Willinger (2000), we know that this process can be expressed as:

$$\sum_{i=1}^{n} X_i \overset{d}{=} n^H Z_1. \quad (17)$$

Substituting the accumulation process from Eq. (17) into the model from Eq. (15), we obtain:

$$P[X_1 + \cdots + X_n > \theta n^\xi] = P[n^H Z_1 > \theta n^\xi] = \varphi. \quad (18)$$

If we define $a = \frac{\theta}{n^{\alpha-\xi}}$, we get

$$P[X_1 + \cdots + X_n > \theta n^\xi] = P[Z_1 > a] = \varphi. \quad (19)$$

$Z_t$ can have different probability density functions (PDFs), such as Pareto (Type I), Lognormal or Weibull. Next subsections, deal with each one of them.

### 6.3. Pareto Distribution

Let us assume that $Z_t$ has a Pareto (Type I) PDF denoted by:

$$f(a) = \frac{\alpha A^\alpha}{a^{\alpha+1}}, \quad (20)$$

where $A$ is the scale parameter (the lower bound at which the distribution starts), and $\alpha$ is the shape parameter, where $0 < A < a$ and $\alpha > 0$. The probability that $Z_t > a$, also called the survival function or tail function, can be expressed in terms of its complementary CDF:

$$\bar{F} = 1 - F = P[Z_t > a] = \left(\frac{A}{a}\right)^\alpha = \varphi. \quad (21)$$

Substituting Eq. (19) and $a$ into Eq. (21)

$$P[Z_1 > a] = \left(\frac{A n^H - \xi}{\theta}\right)^\alpha = \varphi. \quad (22)$$

Solving it for $n$, we get the number of clients using the Pareto I distribution.

$$n = \left(\frac{\theta \varphi + A}{A}\right)^\frac{1}{\alpha - \xi}. \quad (23)$$

### 6.4. Lognormal Distribution

Now assume that $Z_t$ has a Lognormal PDF denoted by

$$f(a) = \frac{1}{a\sigma\sqrt{2\pi}}e^{-\frac{(\ln a - \mu)^2}{2\sigma^2}}, \quad (24)$$

where the two parameters $\mu$ and $\sigma$ are the mean and standard deviation, respectively. The survival function for a lognormal random variable is

$$P[Z_t > a] = 1 - \frac{1}{2} \text{erfc}\left(\frac{-\ln a - \mu}{\sigma\sqrt{2}}\right) = \varphi. \quad (25)$$
The term \( \text{erfc} \) refers to the complementary error function. Working with that survival function is difficult, especially if we want to analytically solve the inequality for a particular term. To help with that, we use the asymptotic upper bound presented in Liu et al. (1999)

\[
\lim_{a \to \infty} \sup \frac{1}{\ln a} \ln P[Z_t > a] \leq -\frac{1}{2\sigma^2}.
\]  

(26)

Solving for \( P[Z_t > a] \) and substituting Eq. (19) into Eq. (26), we obtain

\[
\ln P[Z_t > a] = \ln\varphi \leq \frac{\ln a^2}{2\sigma^2}.
\]  

(27)

Now, substitute \( a \) into Eq. (27)

\[
\ln \frac{\theta}{n^{H-\xi}} \leq (-2\sigma^2 \ln \varphi)^{\frac{1}{2}}.
\]  

(28)

Solving it for \( n \), we get the number of clients using the Lognormal distribution.

\[
n \leq \left( e^{-(-2\sigma^2 \ln \varphi)\frac{1}{2} + \ln \theta} \right)^{\frac{1}{2\sigma^2}}.
\]  

(29)

Other option to manage the \( \text{erfc} \) in Eq. (25) is to use the tight and simple approximations for the \( \text{erfc} \) and inverse \( \text{erfc} \) found in Chiani et al. (2003), and denoted by:

\[
\text{erfc}(x) \leq \frac{1}{6} e^{-x^2} + \frac{1}{2} e^{-\frac{4}{3}x^2}
\]  

(30)

\[
\text{inverfc}(y) \leq \sqrt{-\ln(y)}.
\]  

(31)

A final option is to apply a numerical approximation of the \( \text{erfc} \) for a specific \( \varphi \) and then solving for \( n \).

6.5. Weibull Distribution

Finally, assume that \( Z_t \) has a Weibull PDF with a survival function denoted by

\[
P[Z_t > a] = e^{(-\frac{a}{\lambda})^k} = \varphi,
\]  

(32)

where \( k > 0 \) is the shape parameter, and \( \lambda > 0 \) is the scale parameter of the distribution. Substituting \( a \) and Eq. (19) into Eq. (32)

\[
\frac{\theta}{n^{H-\xi}} = \lambda(-\ln \varphi)^{\frac{1}{k}}.
\]  

(33)

Solving for \( n \), we can get the number of clients using the Weibull distribution.

\[
n = \left( \frac{\theta}{\lambda(-\ln \varphi)^{\frac{1}{k}}} \right)^{\frac{1}{H-\xi}}.
\]  

(34)

6.6. Heavy-Tail Fits

After defining the PDFs used for modeling, we need to get the distribution parameters by fitting the CCDT data. Figure 9 shows obtained CDFs versus empirical CDF from the CCDT. The fit for all three probability distributions is obtained using the maximum likelihood estimation (MLE) method. Table 5 shows the obtained fit parameter values for each PDF using the MLE estimates.

<table>
<thead>
<tr>
<th>Pareto</th>
<th>Lognormal</th>
<th>Weibull</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A = 459 \text{ ms} )</td>
<td>( \mu = 5.51 )</td>
<td>( \lambda = 313.71 )</td>
</tr>
<tr>
<td>( \alpha = 3.388 )</td>
<td>( \sigma = .81963 )</td>
<td>( k = .99 )</td>
</tr>
</tbody>
</table>

7. Description of Experiments

The main objective of the experiments is to validate the stochastic self-similar models defined in the previous Section. It requires a simulation of the arrival of \( n \) client video requests to a cloud represented by the CCDT. The simulation determines the probability \( \varphi \) of exceeding the tolerance time \( \theta \) for a number of clients \( n \). These probabilities are then used to evaluate the proposed self-similar models.

Another objective is to empirically determine the elasticity parameter \( \xi \) described in Section 2.2 and the best fit for the models.
The arrival of clients with heavy-tailed delays has to be Poisson, so the aggregation process can be self-similar (Park and Willinger, 2000). Therefore, the simulation uses a Poisson process for the clients arrival over a 1s total period, and delay times taken directly from the CCDT instead of being simulated by a Markov process as in a classic $M/M/1$ system. The $\lambda$ is changed as the amount of $n$ clients increases for each simulation.

The simulation is fed with the client Poisson arrivals and delay times. Output is the probability of exceeding the maximum delay after sampling 500 random 1s periods from the total simulation. The resulting $\phi$'s and corresponding values for $n$ from the total simulation. The resulting $\phi$ the maximum delay after sampling 500 random 1s periods and delay times. Output is the probability of exceeding $\phi$'s and corresponding values for $n$ from the total simulation. The resulting $\phi$ the maximum delay after sampling 500 random 1s periods and delay times. Output is the probability of exceeding $\phi$'s and corresponding values for $n$ from the total simulation.

The simulation is fed with the client Poisson arrivals and delay times. Output is the probability of exceeding the maximum delay after sampling 500 random 1s periods from the total simulation. The resulting $\phi$'s and corresponding values for $n$ are then fed into the analytical models to make several evaluations. Additionally, 95% confidence intervals (CI) are calculated for the $\phi$'s obtained from the different $n$ clients simulations.

For the Lognormal model, the erf$c$ approximations described in Eq. (26), Eq. (30) and Eq. (31) are not as tight as expected and give preliminary results at least one order of magnitude larger than with simulation and other analytical models. So, we use a numerical approximation for the erf$c$.

The obtained $\phi$ values used for evaluating the model are in the common range for SLAs, as seen in Google (2015).

8. Results

As previously discussed, an elasticity metric ($\xi$) is not publicly available or easily measured for the commercial third party clouds used to create the CCDT. This prompted us to use several scenarios to mitigate the uncertainty in our analytic model. Another parameter with uncertainty is the Hurst parameter that was set in a range $0.67 < H < 0.82$. Evaluations must be performed in order to determine the best approximation for each one.

8.1. The Elasticity Uncertainty

First, we have to analyze the impact of the uncertainty on the elasticity $\xi$. Figure 10 shows results of simulation and modeling with four values of $\xi$ ($0.5, 0.6, 0.7, 0.8$) and $H = 0.82$. It is clear that assuming a $\xi = 0.5$ flattens the modeling results, not approaching the simulation results. While $\xi = 0.8$ makes the models to over-estimate the number of clients. The elasticity values $\xi \sim 0.7$ perform better. The models obtain results closer to the simulation line.

Figure 11 presents more detailed analysis. It shows that elasticity values in the range of 0.65-0.68 approximate results without significant under-estimating or over-estimating $n$.

8.2. The Hurst Parameter Uncertainty

To understand the uncertainty of the Hurst Parameter, let us consider two scenarios with $\xi = 0.68$. We set $H = 0.71$ and $H = 0.74$ (low and mid range values for $H$) to estimate $n$. Figure 12, in Log scale for better visualization, shows results of simulation and modeling. These values do not approximate simulation results with good accuracy. In our models, we set $H = 0.82$ (at the top range of possible $H$ values).

8.3. Evaluation and Discussion

From the results obtained in all scenarios, we can see that the Pareto model gives a very pessimistic prediction with a clear lower bound. It gets far from the simulation results with a $\phi > 0.01$. This can be a result of the Pareto fit not being adequate for the whole distribution of values. It can also be a problem with the number of samples being $< 10^6$, which could limit extreme values empirically gathered and described by this tail distribution.

The Lognormal model prediction is the second most pessimistic. However, it does follow the simulation results, and can over-estimate if the model assumes a very elastic cloud. As seen in Figure 11, it establishes a lower bound prediction to the simulation throughout the different excess probabilities ($\phi$'s). This is a desired outcome since it allows to generate conservative modeling predictions, where no under-provisioning occurs in the video service with promises of serving more clients than it can.

A similar assessment can be obtained from the Weibull model results presented in Figure 11. The prediction comes very close to the simulation results, with tighter results in comparison to the Lognormal model. However, it can give over-estimation for different values of $\xi$, for $\phi < 0.007$ and $\phi > 0.017$. This turns the Lognormal model with ($H = 0.82$ and $\xi = 0.68$) to be the closest one, it can be used as a lower bound with respect to the simulation results.

However, the model does not follow the simulation trends or even be within the CI. To obtain a better approximation, we take into account three principal components (PCs) for the CCDT instead of two.

The analysis and methodology are valid with more PCs. The Hurst parameter is still $0.67 < H < 0.82$, and the elasticity value is $0.6 < \xi < 0.7$. Figure 13 shows the main results. The Lognormal model with $\mu = 5.51$ and $\sigma = .81963$ (similar to the parameters with 2 PCs) follows the simulation curve more closely, falling within the 95% CI.

With this fully parametrized model, we make predictions with less restrictive values of $\phi$, and process more clients per second for the VoD cloud service. Some of these predictions can be seen in Table 6. To get those predictions in context of real cloud data and give them some validity, one can look at the public data from cloud storage services like Amazon S3 as an example (Barr, 2013, 2012). It indicates that, in peak conditions, they were serving 650,000 requests per second in 2012 and 1.1 million requests per second in 2013. So, the prediction model, with a little relaxation in the probability of not exceeding a delay threshold is close to the real data of a single massive cloud storage provider. However, their quality assurances
are in uptime, while all other services are provided in a best-effort manner. Our cloud based VoD service model can give QoS assurances for start-up delay time.

9. Conclusions and Future Work

Using cloud computing storage services for efficient implementation of VoD and online video content demand is a new approach. The main idea is to use the CDN paradigm and adapt it to the challenges and advantages of cloud computing and storage.

We present an analytical model and describe the elements necessary to create a CDN-like service in a logical layer above different third party public storage clouds. We conclude that it can be based on a gateway redirector that accepts requests from incoming clients and provides the optimal cloud redirection based on a set of available metrics.

We propose to use video start-up delay to optimize the VoD cloud service and reduce the loss of customers. This allows us to treat each cloud as a black-box, with very little visibility. In order to make a proper redirection scheme, we propose a performance and scalability model based on real life cloud start-up delay data.

Facing with overwhelming multidimensional information, we perform a PCA of the multiple cloud data, and generate a Characteristic Cloud Delay Time Trace. This trace acts as a proxy. We show that it retains the same variance as the original cloud traces, and contains up to 80% of the variability from all the data with 2 PCs. Together with simplification of the statistical analysis, it allows to model clouds as a single one maintaining important statistical characteristics. Under this PCA conditions, the Lognormal assumption provides a lower bound, but not tight to the simulation trends. We found that with 3 PCs,
Figure 11: Detailed Simulation vs Models Scenarios with H = 0.82
the Lognormal model matches simulation results within the 95% CI.

We found that the cloud delay times can be modeled using heavy-tailed probability distributions, and they exhibit self-similarity. We reached this last conclusion by taking advantage of different Hurst parameter estimators and data filtering techniques.

Self-similarity is a well established property of traditional network communications, web traffic and video. But the role of self-similarity in cloud computing and modeling VoD cloud services has not yet been adequately addressed in the scientific literature. We propose a methodology that allows us to obtain consistent values for the degree of self-similarity in clouds for video start-up delays. We expect that it could be applied in other areas. However, further study is needed to analyze more cloud metrics that help us to better understand QoS.

We propose and evaluate a novel cloud scalability model (using three different heavy-tailed distributions) that takes the self-similar and elastic nature of the cloud into account to determine the amount of clients that can be served, maintaining a maximum delay time. The main result is the prediction model for \( n \) clients using the Lognormal self-similar model that closely matches simulation results. Another contribution is an empirical measurement of the elasticity \( \xi \) of the cloud using different uncertainty scenarios.

We validate the assumptions and derivations through an evaluation of the best found model predictions against the known data of clients per second of real storage cloud providers. These results help us to understand its elastic property for consumer entertainment applications. This could improve under/over estimation of the number of clients and serve as a tool for providing SLA restrictions other than uptime or best-effort.
However, further study is required for better understanding the role of PCA and variability for the improvement of the CCĐT and the model.

Another direction for future work is to apply these concepts and stochastic self-similar models to improve the efficiency of the user demand allocation in the VoD cloud broker (gateway) taking into account the elasticity of the clouds, self-similarity and the tail data characteristics.

References


Barr J. Amazon web services blog: Amazon s3 - 905 billion objects and 650,000 requests /second 2012. URL: http://aws.typepad.com/aws/2012/04/amazon-s3-905-billion-objects-and-650000-requests-second.html.


Google . Google cloud storage, google prediction API, and google BigQuery SLA. 2015. URL: https://cloud.google.com/storage/sla.


