Model of Video on Demand Service Provisioning on Multiple Third Party Cloud Storage Services

Carlos Barba Jimenez 1, Raúl V. Ramírez V. 1, and Andrei Tchernykh 2
1 Monterrey Institute of Technology and Higher Education, ITESM, Monterrey, México
{a00613412, rramirez}@itesm.mx
2 CICESE Research Center, Ensenada, Mexico, chernykh@cicese.mx

Abstract. In this paper, we present a solution model to tackle the problem of providing Video-on-Demand (VoD) using cloud computing storage service composition. We present related works, the problem motivation and some preliminary results. As part of the problem, we study the performance and scalability model for this VoD cloud service by performing a statistical analysis and Principal Component Analysis (PCA) on real cloud data. In order to simplify the stochastic modeling, we created a Characteristic Cloud Delay Time trace (using PCA), and determined the self-similarity nature of the data itself to pursue modeling using heavy-tailed probability distributions.

Keywords. Video on Demand, Cloud Computing, Service Composition, heavy-tails, PCA.

1 Introduction

The cloud is a term that has become increasingly popular in IT and consumer trends, generating buzz. Each year we see new products coming out that take advantage of it, are based on it, or even named after them. The scope of the concept has reached enterprises, journalism, research, software development, and business models; as well as the outlook for technology products. Commercially we have seen the rise of new IT services described as the cloud or cloud computing, with companies like Amazon offering it as elastic computing and storage. These services have increased interoperability, usability and reduced the cost of computation, application hosting, and content storage and delivery by several orders of magnitude [3]. These conditions open the door to creating new services that satisfy existing and future user’s demands.

The trends on IP traffic are changing constantly, posing new challenges to deliver, as the amount of newer mobile internet enabled devices increases and content consumption rises [2]. Cisco [1] reports that:

- Global IP traffic has increased eightfold over the past 5 years, and will increase fourfold over the next 5 years;
- In 2015, the gigabyte equivalent of all movies ever made will cross global IP networks every 5 minutes;
- Internet video is now 40 percent of consumer Internet traffic, and will reach 62 percent by the end of 2015, not including the amount of video exchanged through P2P file sharing
- Video-on-demand traffic will triple by 2015

Content services, like Video-on-Demand (VoD), are usually supported by a centralized delivery architecture based on private or rented servers with fixed costs and little flexibility [9]. However, facing the problem of performance bottlenecks caused by an unpredictable usage of the service, they had to adopt a Content Distribution Network (CDN) model, usually operated by another company [9]. CDNs have a central entity that can enforce Quality of Service (QoS) levels, but this comes at a non-negligible financial cost [2].

There are also Peer-to-Peer (P2P) alternatives, some even free, but they can rarely provide guaranteed services [2]. There have been complimentary or hybrid solutions, like [4] or [5]. They are used in live streaming, where they help with flash crowds, in events, where certain content needs to be accessed at the same time by several users [5], which is no always the case in a VoD service. As of now, the usual CDN
centralized model is still the most prevalent, with services like Akamai [6].

CDN providers are mostly priced out of reach for small to medium enterprises (SMEs), smaller government agencies, universities and charities [6]. A very small scale business could in theory use a private server or contract space in a hosting service. However, it is difficult to size them properly initially. They are either expensive, because of an overprovision, or they are not robust enough and will have problems under heavy demand. They could be under the Slashdot effect or some other popularity phenomenon.

There are other options like third party services (e.g. Youtube, Vimeo, etc.), but content rights management and QoS are not their priorities, even if they have big infrastructures behind them. This situation could not be suitable for all businesses.

To help us to cope with some of these factors, we go back to the concept of the cloud and different available services. There are new solutions commercially available that take the characteristics and advantages of the cloud, like the one proposed in [6] with the MetaCDN project. The authors published some of the characteristics of this model, but left out the proprietary code and algorithms that make it work. Their statistical analysis is also limited, if we are to generate a predictive model.

In this paper, we use different cloud services available in composition to create an Edge Cloud Service (similar to MetaCDN [6]), but in a fashion as [8] describes it: a gateway. This gateway will create a CDN like functionality sitting in a layer atop the cloud, and will be taking into account some of the main challenges of VoD delivery like response time.

The idea is to tackle a similar problem motivation and approach as established in [6]. However, the objective of our work is not to solve the programmatic problem, but rather tackle the mathematical performance predictive model for the final service. This model has to determine maximum number of clients that can be served under certain cloud conditions and a fixed maximum delay time. In addition, it has to handle request redirections to different cloud providers, under different conditions keeping a given QoS. These aspects were not fully explained and explored in [6].

As a metric for the meta VoD service, we take mainly into account the times for the abandonment rate described in [20], which found that VoD viewers start leaving after a startup delay of 2000ms (milliseconds) losing 5.8% of users for each additional second of wait. This delay time $T_D$ is total, it includes all network times, server times, and additional overhead.

This paper proposes a scalability model and analysis using self-similarity and heavy tails similar to ones used in [15][19]. We start it by characterizing real cloud data using statistical analysis, and dimensionality reduction using Principal Component Analysis. The objective is to get a model for the delay time $T_D$, which will enable us to make predictions for the probability of successful service under a certain threshold of abandonment rate.

## 2 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 [7].

Cloud computing offers three service models according to [7]: SaaS, PaaS, IaaS.

**Cloud Software as a Service (SaaS):** The capability provided to the consumer is to use the providers applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email). The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

**Cloud Platform as a Service (PaaS):** The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming...
languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.

Cloud Infrastructure as a Service (IaaS): The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

We focus on the storage part of Cloud Infrastructure as a Service (IaaS), where VoD is potentially a very heavy user. Cloud storage providers offer service level agreements (SLAs), which guarantee a level of Quality of Service (QoS) and operate on a utility computing model. This model is defined in [7] as the metering capability for a pay-per-use-basis. Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service, as explored in [14]. This situation makes the idea of using the existing cloud services for content distribution very attractive, as we will only pay for the amount of storage and computing we use, instead of a potentially expensive contract with a CDN or an under provisioned private server.

3 Related Works

The basic idea we want to address is the video CDN based on the cloud as explained in [6]. One of the most interesting figures it presents is the price comparison of delivering content through a CDN versus different cloud providers (Figure 1). It also has interesting points related to QoS provisions, but the model in [6] is still a little broad for a proper mathematical definition.

On the topic of CDN and modelling, the authors in [10] explore resource discovery and request redirection in a multi provider content delivery network environment.

They explain that CDNs evolved as a solution for internet service degradations and bottlenecks due to large user demands to certain web services. In addition, they address some of the internal problems that CDN providers face, like the required break down of system locations, the need to increase utilization rates and the over provisioning and external resource harnessing need that can happen to satisfy a certain SLA. The proposition is a constellation of CDNs that collaborate for short or long term periods to handle the different workload situations. This constellation uses a load distribution scheme with a request redirection approach where there is a mediator or coordinator agent that redirects load according to the state of the different CDNs [10].

Continuing on the topic of cloud computing service aggregations, the authors in [11] describe a QoS aware composition method supporting cross-platform service invocation in cloud environments, which explores a web service composition and how to attain a QoS optimal solution.

There have also been more recent developments where a coordinator for scaling elastic applications across multiple clouds is used, as proposed in [12]. The main focus is in elastic web applications where the application characteristics and usage in the cloud providers are better known (although provisioning is still a challenge). The authors in [12] don't address
much on content delivery, like in a VoD service, and the application characteristics of a web application and content storage in the cloud are not the same. However, it does give ideas to take into consideration for the work of this research. This paper also takes into consideration previous work related to VoD specific topics like the specification of an integrated QoS model for VoD applications found in [13].

4 Solution Model and Preliminary Results

Given that we are developing a CDN-like service, it is fitting to use similar techniques to the ones that this type of systems use and apply them to Cloud Environments. The model stems from the basis of peering CDNs where several CDN networks cooperate to fulfill requests. In this case the CDNs will be the different clouds available, but without the concepts of authoritative rights.

Our model works by sitting on a layer atop the different Cloud Storage providers (SaaS) and using IaaS for some of the calculations and computing required. The solution model to the problem consists of the following main components:

- Asynchronous Resource Discovery: Since each Cloud provider operates individually this module keeps track of different points of presence and resources available.
- Asynchronous Resource Monitoring: This module takes care of probing the different providers to keep historic information about different performance parameters (these are explained with more detail later).
- QoS Aware Resource Redirector: This agent is the one in charge of weighting in the information available from the clouds and redirecting the load to the optimal resource location and provider.

Figure 2 shows the basics of the solution model, with all the principal elements of the gateway and how most of the data and connections flow. User requests go through the gateway, which has the redirection and resource monitoring logic inside, then that gateway gives back an address redirection for the final content in one of the several cloud providers. In this model the several cloud providers and their infrastructure are working in a similar manner as the Content Distribution Inter networking model, in which the gateway will consider the clouds and its networks as black boxes [10].

![Figure 2. Cloud VoD System](image)

However, it differs from the main model in that there really aren’t mechanisms implemented from the cloud providers that will allow peering, and there is no central entity or supervisor with access to intra cloud information.

This is one of the main challenges, working without knowing the request and response loads that are present in each of the inter cloud networks in real time; essential to work within the peering CDN scheme and algorithms [10]. In order to overcome these limitations, we draw conclusions from the information we can get.

We use the total delay times, since they can be monitored and discovered from outside the black boxes. Additionally, we make assumption based on the published requests/second statistics from some cloud providers [27]. The performance parameters that the redirector has access for the different cloud providers available are:

- Delay Time: measured in ms as captured by probing done by the monitor service.
- Request Origin/Destination similarity: Established in a [0,1] scale where 1 matches the origin of the request to an available destination cloud available and progressively gets reduced as the distance difference between both is greater.
- Rejection rate: number of dropped requests due to service unavailability or timeouts as captured by the monitor service.
- Completion rate: number of completed services that each of the providers have on record.
Provider QoS: the QoS as stated by the service level agreement that each provider gives.

Mean Peak requests/second and the percentage that VoD type requests represent. We need to be able determine how many users the service can handle under a maximum delay time, taking into consideration the abandonment rate times and data from [20]. This will be the scalability model.

Once we can make predictions about users and system scalability, the problem that we must solve inside the redirector logic 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. We must also add the wildcard variables of the downtime status and QoS levels each provider has, as a continually changing modifier. Finally we evaluate them all under the values and statement of abandonment rate and time described by [20] and different cloud load scenarios using the requests/second information.

To begin, we define a basic model where the delay time \( T_{Di} \) that the user \( i \) needs to access the content in a certain provider \( j \) is determined by:

\[
T_{Di} = T_{rij} + T_{nij} + T_{Og}
\]  

(1)

\( T_{rij} \) is the response time from the cloud storage (disk access, memory and computations needed), \( T_{nij} \) is the delay time in the gateway caused by the overhead of redirection, and \( T_{Og} \) is the network time the request takes to travel from client \( i \) to cloud \( j \). From here, we follow a simple scheme similar to [10], minimizing the redirection time and cost, using the following:

\[
\min \sum_{i=1}^{n} \sum_{j=0}^{m} R_c(i,j) S_{ij} C_j Q_j
\]  

(2)

In this case \( R_c = T_{Di} \) if the cloud provider is not rejecting requests at that moment, else it is \( \infty \). \( S_{ij} \) can take a value between [0,1] according to how close the locations from user and cloud provider are together geographically, \( C_j \) is the completion rate of the provider \( j \) and can take a value between [0,1], and finally \( Q_j \) is the quality of service that provider \( j \) has and can take a value between [0,1].

However, this solution model (2) would be naive, since it would in theory base its results in the most recent data from the different clouds available. It does not consider the stochastic nature of the response and network times; its probability distribution and chaos measurements.

Thus, considering the probabilistic nature of \( T_{Di} \), and taking into account [20] and the 2s abandonment start time \( \theta \) as a measure of a successfully redirected petition, we could create a new model for a client \( i \) connected to \( m \) different clouds using the following premise:

\[
P(T_D < \theta) = \phi
\]  

(3)

\( \phi \) is the probability that the delay time is under \( \theta \) or 2s (in our case). Using (3) as a base and assuming \( T_p \) is heavy-tailed we could model it, using one of the Pareto, Lognormal or Weibull distributions. However, we first must do a statistical analysis of different real life cloud delay times, to properly characterize them.

4.1 Cloud Data Analysis

In order to have a good estimation and idea of the extreme values and distribution that the clouds delays can exhibit, we used real cloud delay times. From now on, when we talk about Delay Time we only refer to \( T_{nij} + T_{rij} \), since only the network times and cloud response times interest us to model from the cloud data.

Real cloud delay times were obtained using the PRTG Network Monitor from Paessler [26]. The measurements are taken from an Amazon EC2 instance (small) located in N. Virginia, as an analog for a client, which then connects to different cloud providers and gathers the delay time (comprising, disk access times, memory access times, and network times) to get a 65kb file (representing a worst case video frame size). The measurements were taken every 60s over a period of 40 days leaving us with over 52,000 measurements for 6 cloud storage providers. In Table 1, we see different clouds and some descriptive statistics:
### Table 1. Cloud Providers Statistics

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

In Figure 3, we see an example time series, which we call trace, for the Google cloud.

![Time Series for Google Cloud delay times in ms](image)

**Figure 3. Time Series for Google Delay Times**

We clearly see the chaotic behavior of the delay times; however we still need to look at the distribution of the values. Figures 4 and 5 have the histograms for each of the clouds.

Figures 4 and 5 show that the delay times for the different cloud provider samples exhibit a heavy-tailed distribution which is corroborated in combination with the high kurtosis values presented in Table 1.

Moreover, if we take into consideration that the histograms presented are not symmetrical and the high variability in the data, (with max delay values several times the mean value), we conclude in concordance with [15] that the data cannot be modeled by a symmetric probability such as a Gaussian or a short-term memory process like a Markov one. This would be a good sign if a scalability analysis similar to [19][15], shows that the data are self-similar.

The problem is now converted to finding a cloud to model the delay. Clouds have similar statistics and histograms, but some have different kurtosis and different means, etc.

In this case, we treat the existence of several different clouds as a problem of multidimensionality [19]. 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 [19][16] to capture as much of the variability of all clouds as possible. The goals is to reduce dimensionality to at most 2 components, from our available 6, and then regenerate a single trace that can be analyzed further and used for modeling.
To find the Principal Components we followed a procedure similar to [16][19], and built a correlation matrix using the delay traces for each cloud. Traces are used as columns to indicate the dimensions with ~50k different observations. The procedure then subtracts the overall mean \( \mu \) from each element. We used the mean from all values, not in a per cloud basis (180.83 ms), and then used Pearson’s correlation coefficient, since all the observations are in the same ms units [19].

The results for the PCA (for the first 3 components) are visualized in Figure 6, where we show the orthonormal principal component coefficients for each variable and the principal component scores for each observation in a single plot.

Figure 6. Component Scores

The resultant principal components (PCs) and their respective effect are shown in Table 2:

<table>
<thead>
<tr>
<th>Variables</th>
<th>PC1</th>
<th>PC2</th>
<th>PC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenvalue</td>
<td>169,231.8</td>
<td>80,962.5</td>
<td>30,646.5</td>
</tr>
<tr>
<td>Variability(%)</td>
<td>54.920</td>
<td>26.274</td>
<td>9.947</td>
</tr>
<tr>
<td>Cumulative (%)</td>
<td>54.920</td>
<td>81.200</td>
<td>91.140</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variables</th>
<th>PC4</th>
<th>PC5</th>
<th>PC6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenvalue</td>
<td>10,249.8</td>
<td>9,555.8</td>
<td>7,490.3</td>
</tr>
<tr>
<td>Variability(%)</td>
<td>3.326</td>
<td>3.102</td>
<td>2.431</td>
</tr>
<tr>
<td>Cumulative (%)</td>
<td>94.470</td>
<td>97.570</td>
<td>100.000</td>
</tr>
</tbody>
</table>

Following [19], we have to select \( m \) number of components that will keep most of the variability present in \( p \) variables, where we want \( m \ll p \), effectively doing a dimensionality reduction. In the case of our data, we see that selecting 1 PC will give us ~50% of the original data variability, but if we go to 2 PCs we have ~81% of it. After that, the percentage of added variability of each extra PC is < than 10%.

Considering these conditions, choosing 2 PCs will give us significantly more information than the other 4, so the Characteristic Cloud Delay Trace will be generated using PC1 and PC2.

Using these PCs we reconstruct a single trace that will have the same variance that the whole set of cloud delay time traces: \( \sigma_{\text{original}}^2 = 98,894.582 \) and \( \sigma_{\text{original}}^2 = 314.475 \). The reconstruction also has to take into account that we subtracted the overall mean \( \mu \) while doing the PCA. To reconstruct it, we start from the following:

\[
E[PC1] = 0 \quad E[PC2] = 0
\]

\[
VAR[PC1] = \lambda_1 \quad VAR[PC2] = \lambda_2
\]

Where \( \lambda_i \) is the eigenvalue for the corresponding PCI. We propose creating a new variable \( C \) :

\[
C = PC1 + PC2
\]
This variable \( C \) has the following expected value, variance, and standard deviation:

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

\[
VAR[C] = VAR[PC1] + VAR[PC2] = \lambda_1 + \lambda_2 \quad (8)
\]

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

Then we create the final reconstructed trace \( X' \), which must have a \( \sigma^2_{x'} = \sigma^2_{original} = 98894.582 \) using the following transformation, and taking into account \( \mu \):

\[
X' = \mu + \frac{\sigma_{original} C}{\sqrt{\lambda_1 + \lambda_2}} \quad (10)
\]

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

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

\[
VAR[X'] = VAR[\mu] + \left( \frac{\sigma_{original}^2}{\lambda_1 + \lambda_2} \right) VAR[C] = \sigma_{original}^2 \quad (12)
\]

Since the standard deviation is the same in our new reconstructed trace \( X' \) as in the \( \sigma_{original}^2 \), evidenced by (12) and numerically tested, we apply the transformation to each row of the PC1+PC2 and get our Characteristic Cloud Delay Trace. We see the result in Figure 8.

Figure 8 shows that the CCDT’s behavior is similar to the Google delay times in Figure 3. Figure 9 shows the histogram for the new reconstructed trace:

![Reconstructed Characteristic Cloud Delay Times Histogram](image)

The data has a heavy-tailed distribution, which is not symmetric. The kurtosis parameter is 472.451, so in accordance to [15], we assume that the data cannot be modeled using traditional symmetric probability distributions and Markovian processes. Therefore, we use the Pareto, Lognormal and Weibull probability distributions and fit them to characterize the reconstructed data and obtain the cdf’s that we see in Figure 10.

![Comparison of Lognormal, Pareto, and Weibull cdf against data trace](image)

Going back to the scree plot in Figure 7, even if limited to only 6 components, it shows that...
there’s one mayor PC and a second one that has still some considerable percentage of variation. After them, the third to sixth component exhibit a tail of slowly decaying eigenvalues with what [19] calls a rule similar to $k^{-d}$, power law rule. This is an indication of self-similarity.

According to [15], there are several, not equivalent, definitions for self-similarity. The most commonly accepted for flows related to networks is found in [25], it states that a continuous-time process $Y = \{Y(t), t \geq 0\}$ is self-similar, with a self-similarity or Hurst parameter $H$, if it satisfies the following conditions.

$$Y(t) \sim a^{-H}Y(t) \ \forall t \geq 0 \ \forall a > 0, \ \ 0.5 < H < 1 \ \ (13)$$

This self-similarity measurement is further studied by [21][22][23][24], having similar conclusions about the values of the Hurst parameter that indicate self-similarity (0.5 < $H$ < 1). However, from [21], we also know that even if the Hurst parameter is perfectly defined mathematically, measurements and estimations using real traces is not easy. This calls for the use of several Hurst parameters estimators rather than just one, if we are to avoid false conclusions [21]. It must be said, that all estimators are vulnerable when the data has a trend and periodic component [21][24].

In this paper, we use the following methods to estimate the Hurst parameter: the R/S (Rescaled/Range), the aggregate variance, the boxed periodogram, the Abry Veitch wavelet method [23] and Peng’s variance of residuals. Table 3 summarizes results.

<table>
<thead>
<tr>
<th>Method</th>
<th>Hurst Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/S</td>
<td>0.777</td>
</tr>
<tr>
<td>Aggvar</td>
<td>0.789</td>
</tr>
<tr>
<td>Box Per</td>
<td>0.679</td>
</tr>
<tr>
<td>Peng’s</td>
<td>0.786</td>
</tr>
<tr>
<td>Abry-Veitch</td>
<td>0.680</td>
</tr>
<tr>
<td>Mean</td>
<td>0.742</td>
</tr>
</tbody>
</table>

We concluded that $0.670 < H < 0.789$, which is a definite indication of self-similarity.

5 Conclusions and Future Work

The use of cloud computing services is a fundamentally new approach to solve the increased VoD and online video content demand. The main idea is to use some of the knowledge that the CDN paradigm has to offer and adapt it to the challenges and advantages of cloud computing.

We presented a solution model and its elements to create a CDN like service that exists in a layer atop of the Clouds, using storage service composition from different providers. We concluded that it can be based on a gateway redirector that accepts requests from incoming clients and gives out the optimal cloud redirection based on a set of available metrics.

We also show that this system can treat each cloud as a black box where we have very little visibility, and that in order to make a proper redirection scheme, we would need to make a proper performance and scalability model of real life cloud data.

We demonstrate that PCA is a powerful tool. It allows making a dimensionality reduction of the multiple cloud data, and generating a Characteristic Cloud Delay Time Trace. This trace can act as a proxy, as we showed that it retains the same variance as the original cloud traces, and that it contains up to 80% of the variability from all the data. This not only simplifies statistical analysis, but enables future work in the devising of the scalability and performance model.

Finally, we found that cloud delay times present a heavy-tailed probability distribution, and that they exhibit self-similarity (taking advantage of different Hurst parameter estimators to achieve the measurement). This are important results, since we can use probability distributions like the Weibull, Lognormal and Pareto to model performance and establish the request redirector gateway logic for the proposed solution model.

As a future work we want to evaluate the different probability distributions goodness of fit to either select one or two of them to generate the final scalability model. This model will then be
validated through simulation, using the real cloud delay time traces and several load assumptions. Finally, further work is needed in the model to account for the stochastic nature of the delay times and to integrate the abandonment rate and times into it, as another measure of success.

Acknowledgements

The authors would like to acknowledge the support provided by Tecnologico de Monterrey through the Google Research Chair with grant number 0020CAT027.

References


Carlos Barba Jimenez obtained his Bachelor’s Degree in Electronic Systems Engineering from Tecnologico de Monterrey, Campus Monterrey in 2009. Currently he is a PhD student at the same institution. His interests include cloud computing, performance modeling, and microprocessor manufacturing, developing, and testing.

Raúl V. Ramírez Velarde is a researcher in the Computer Science Department, Tecnológico de Monterrey, Monterrey, Mexico. He is part of the Consejo Nacional de Acreditación Informática y Computación (CONAIC) and Asociación Nacional de Instituciones de Educación Tecnología de Información (ANIEI) in Mexico. He received his PhD in Distributed Multimedia Systems in 2004. His research interests include analysis, modeling and design of multimedia systems.

Andrei Tchernykh is a researcher in the Computer Science Department, CICESE Research Center, Ensenada, Baja California, Mexico. From 1975 to 1990 he was with the Institute of Precise Mechanics and Computer Technology of the Russian Academy of Sciences (Moscow, Russia). He received his Ph.D. in Computer Science in 1986. In CICESE, he is a coordinator of the Parallel Computing Laboratory. He is a current member of the National System of Researchers of Mexico (SNI), Level II. He leads a number of national and international research projects. He is active in grid-cloud research with a focus on resource and energy optimization. He served as a program committee member of several professional conferences and a general co-chair for International Conferences on Parallel Computing Systems. His main interests include scheduling, load balancing, adaptive resource allocation, scalable energy-aware algorithms, green grid and cloud computing, eco-friendly P2P scheduling, multi-objective optimization, scheduling in real time systems, computational intelligence, heuristics and meta-heuristic, and incomplete information processing.