

Chapter 1

GENERATING INTERNET STREAMING MEDIA OBJECTS AND WORKLOADS

Shudong Jin

*Computer Science Department
Case Western Reserve University
jins@cwru.edu*

Azer Bestavros

*Computer Science Department
Boston University
best@cs.bu.edu*

Abstract Characterization and synthetic generation of streaming access workloads are fundamentally important to the evaluation of Internet streaming delivery systems. GISMO is a toolkit for the generation of synthetic streaming media objects and workloads that capture a number of characteristics empirically verified by recent measurement studies. These characteristics include object popularity, temporal correlation of request, seasonal access patterns, user session durations, user interactivity times, and variable bit-rate (VBR) self-similarity and marginal distributions. The embodiment of these characteristics in GISMO enables the generation of realistic and scalable request streams for use in benchmarking and comparative evaluation of Internet streaming media delivery techniques. To demonstrate the usefulness of GISMO, we present a case study that shows the importance of various workload characteristics in evaluating the bandwidth requirements of proxy caching and server patching techniques.

Keywords: Streaming media delivery; performance evaluation; Internet characterization.

1. Introduction

The use of the Internet as a channel for the delivery of streaming (audio/video) media is paramount. This makes the characterization and synthetic generation of streaming access workloads of fundamental importance in the evaluation of

Internet streaming delivery systems. While many studies have considered the characterization and synthesis of HTTP workloads [Almeida et al., 1996, Arlitt and Williamson, 1996, Barford et al., 1999, Breslau et al., 1999, Crovella and Bestavros, 1996, Cunha et al., 1995, Gribble and Brewer, 1997, Jin and Bestavros, 2000, Padmanabhan and Qiu, 2000, Barford and Crovella, 1998, The Standard Performance Evaluation Corporation,], just to name a few, very few studies focused on characterizing streaming media workloads [Acharya et al., 2000, Almeida et al., 2001, Chesire et al., 2001, Padhye and Kurose, 1998, Cherkasova and Gupta, 2002, Veloso et al., 2002, van der Merwe et al., 2002, Costa et al., 2004, Li et al., pear], and just two [Jin and Bestavros, 2001, Tang et al., 2003] have tried to generate representative streaming media workloads. Because HTTP requests and streaming accesses are different, HTTP request generators are not suitable for generating streaming access workloads. These differences include the duration of the access, the size of the objects, the timeliness requirements, *etc.*

In the absence of synthetic workload generators, and in order to evaluate the performance of streaming access techniques, one has to seek alternatives, such as using real traces, or using analysis/simulation under simplifying and often incorrect assumptions (*e.g.*, using independent reference model, sequential access, *etc.*). Indeed, these alternatives have been used in prior work on caching and on patching [Hua et al., 1998, Gao and Towsley, 1999], for example. While the use of such alternatives allows analysis and performance evaluation, the resulting conclusions may not be accurate enough, and certainly could not be reliable enough to assess performance when conditions under which the traces were collected (or modeling assumptions made to simplify analysis) are violated. For example, when a limited trace is used in a trace-driven simulation, it may not be possible to generalize the conclusions of such a simulation when the system is subjected to scaled-up demand, or when the distribution of some elements of the trace (*e.g.*, size and popularity distributions of objects) are changed. Synthetic workload generators have the advantage of being able to produce traces with controllable parameters and distributions. The challenge is in ensuring that such synthetic workload generators reflect (in a parameterizable fashion) known characteristics of streaming media and their access patterns.

This chapter overviews GISMO—a tool for synthesizing streaming access workloads that exhibit various properties observed in access logs and in real traces. One of the salient features of our work is the independent modeling of both session arrival processes and individual session characteristics. For session arrival processes, we use a Zipf-like distribution [Breslau et al., 1999, Zipf, 1929] to model reference correlation due to streaming object popularity. For individual sessions, we use a model that exhibits rich properties, including session durations, user interactivity times, VBR self-similarity and heavy-tailed marginal distributions. These properties have been observed and validated by

studies on streaming access workload characterization. Using GISMO, we are able to generate synthetic workloads with parameterizable characteristics. To demonstrate the usefulness of GISMO, we present results from a case study comparing the effectiveness of recently proposed proxy caching and server patching techniques. We show how workload characteristics affect the performance of these techniques.

2. Related Work

HTTP Workload Characterization. Workload characterization is fundamental to the synthesis of realistic workloads. Many studies [Almeida et al., 1996, Arlitt and Williamson, 1996, Barford et al., 1999, Breslau et al., 1999, Cunha et al., 1995, Gribble and Brewer, 1997, Jin and Bestavros, 2000] focused on the characterization of HTTP requests. Main findings include the characterization of Zipf-like document popularity distribution [Barford et al., 1999, Breslau et al., 1999, Cunha et al., 1995], the characterization of object and request size distributions [Barford et al., 1999, Cunha et al., 1995], and the characterization of reference locality properties [Almeida et al., 1996, Arlitt and Williamson, 1996, Jin and Bestavros, 2000].

Web traffic is self similar, exhibiting burstiness at different time scales [Crovella and Bestavros, 1996, Leland et al., 1994]. A representative self-similar Web traffic generator, SURGE [Barford and Crovella, 1998] models the overall request streams as the aggregation of many individual user request streams, which have heavy-tailed inter-arrival time distribution, and/or heavy-tailed request size distribution. Request streams generated in such a way have significantly different characteristics than the ones from the workloads generated by HTTP benchmark tools such as SpecWeb96 [The Standard Performance Evaluation Corporation,].

Streaming Media Workload Characterization. As we mentioned at the outset, there have been few studies that considered the characteristics of streamed media on the Web [Acharya and Smith, 1998] and the characteristics of access patterns for streamed media [Acharya et al., 2000]. These studies revealed several findings that are also known for non-streamed Web media, including: high variability in object sizes, skewed object popularity, and temporal locality of reference. In addition, these studies highlighted the preponderance of partial accesses to streamed media—namely, a large percentage of responses to user requests are stopped before the streamed object is fetched in its entirety. [Chesire et al., 2001] analyzed a client-based steaming-media workload. They found that most streaming objects are small, and that a small percentage of requests are responsible for almost half of the total transfers. They also found that the popularity of objects follows a Zipf-like distribution and that requests during periods of peak loads exhibit a high degree of temporal local-

ity. [Almeida et al., 2001] analyzed workloads from two media servers for educational purposes. They studied the request arrival patterns, skewed object popularity, and user inter-activity times. Examples of characterization efforts targeted at non-web environments include the work of Padhye and Kurose [Padhye and Kurose, 1998], which studied the patterns of user interactions within a media server, and the work of [Harel et al., 1999], which characterized a workload of media-enhanced classrooms, and observed user inter-activity such as “jumping behavior”. More recently, Cherkasova and Gupta [Cherkasova and Gupta, 2002] analyzed enterprise media server workloads, in particular locality of access and evolution of access patterns. They pointed out a number of differences from traditional Web server workloads. Based on this, they have also designed the MediSyn [Tang et al., 2003] workload generator. [Velooso et al., 2002] presented the first study on live streaming access workload characterization at different granularity level, namely users, sessions, and individual transfers. In another work, [van der Merwe et al., 2002] analyzed streaming access logs from commercial service. They assessed the potential benefit of using distribution infrastructures to replicate streaming media objects and to improve bandwidth efficiency. A recent study [Costa et al., 2004] provides a more thorough analysis of pre-stored streaming media workloads, focusing on the typical characteristics of client interactive behavior.

In Section 3, we incorporate many of these characteristics in the models we use for workload generation in GISMO. To the best of our knowledge, GISMO is the first streaming workload generator developed and made available to public. Another more recent effort, the MediSyn project by [Tang et al., 2003] adopted similar methodology and models.

Evaluation Methodologies. In the absence of a unified model for workload characteristics, various proposals for streaming media protocols and architectures have used a variety of assumptions and models. We discuss these below, focusing only on caching and patching [Carter and Long, 1997, Hua et al., 1998, Gao and Towsley, 1999] protocols—protocols we will be contrasting in a case study using GISMO in Section 5.

In particular, the patching technique [Carter and Long, 1997, Hua et al., 1998, Gao and Towsley, 1999, Eager et al., 2001] leverages large client buffer to enable a client to join an ongoing multicast for prefetching purposes, while using unicast communication to fetch the missed prefix. A few patching protocol studies have considered the effect of Zipf-like popularity distributions on performance [Gao and Towsley, 1999, Hua et al., 1998]. In these studies, the arrival processes for requests were assumed to follow a Poisson distribution [Gao and Towsley, 1999, Hua et al., 1998]. None of the studies we are aware of considered other workload characteristics, such as stream length or user inter-activity.

3. Workload Characteristics in GISMO

Accurate workload characterization is essential to the robust evaluation of streaming access protocols. In fact, several studies on streaming access workload characterization [Acharya et al., 2000, Almeida et al., 2001, Chesire et al., 2001, Padhye and Kurose, 1998] considered the implications of observed characteristics on the performance of various protocols, including caching, prefetching, and stream merging techniques.

In order to generate realistic synthetic streaming access workloads, we need to adopt an access model. We define a *session* as the service initiated by a user's request for a transfer and terminated by a user's abortion of an on-going transfer (or the end of the transfer). The workload presented to a server is thus the product of the *session arrivals* and the *properties of individual sessions*. The first three distributions in Table 1.1 specify the characteristics of session arrivals, whereas the remaining distributions characterize properties of individual sessions.

Session arrivals could be described through the use of appropriate models for: (1) object popularity, (2) reference locality, and (3) seasonal access characteristics. In GISMO, and given the preponderance of findings concerning the first two of these models, we use a Zipf-like distribution to model object popularity, implying a tendency for requests to be concentrated on a few "popular" objects, and we use a heavy-tailed Pareto distribution to model reference locality (*i.e.*, temporal proximity of requests to the same objects). Given the application-specific nature of seasonal access characteristics, we allow the overall request arrival rate to vary with time according to an arbitrary user-supplied function.

An individual session could be described through the use of appropriate models for: (1) object size, (2) user inter-activity, and (3) object encoding characteristics. In GISMO, we model object size (which determines the total playout time of the streamed object) using a Lognormal distribution. We model user inter-activity times which reflect user interruptions (*e.g.*, VCR stop/fast-forward/rewind functionalities) using a Pareto distribution. Finally, we model object encoding characteristics by specifying the auto-correlation of the variable bit rate needed to transfer that object in real-time. Multimedia objects are known to possess self-similar characteristics. Thus, in GISMO, we model the VBR auto-correlation of a streaming object using a self-similar process. Also, we use a heavy-tailed marginal distribution to specify the level of burstiness of the bit-rate.

Modeling Session Arrivals

The first aspect of a workload characterization concerns the model used for session arrivals. We define the *session inter-arrival time* to be the time between two session arrivals. We consider both the inter-arrival time of *consecutive sessions* (*i.e.*, general inter-arrival time), and the inter-arrival time of *sessions*

Component	Model	PDF $f(x)$	Params
Popularity	Zipf-like	$\frac{1}{x^\alpha}, x = 1, 2, \dots, N$	α, N
Temporal Correlation	Pareto	$\frac{\alpha k^\alpha}{1-k^\alpha} x^{-\alpha-1}, k < x < 1$	α, k
Seasonal Access Frequency	User-specified		
Object Size	Lognormal	$\frac{e^{-(\ln x - \mu)^2 / 2\sigma^2}}{x\sigma\sqrt{2\pi}}, x > 0$	μ, σ
User Inter-activities	Pareto	$\frac{\alpha k^\alpha}{1-k^\alpha} x^{-\alpha-1}, k < x < 1,$	α, k
VBR Auto-correlation	Self-similarity		H
VBR Marginal Dist.(body)	Lognormal	$\frac{e^{-(\ln x - \mu)^2 / 2\sigma^2}}{x\sigma\sqrt{2\pi}}, 0 < x < C$	μ, σ
VBR Marginal Dist.(tail)	Pareto	$\alpha k^\alpha x^{-\alpha-1}, x \geq C$	α, k

Table 1.1. Distributions used in the workload generator

requesting the same objects, which is a measure of temporal locality of reference [Almeida et al., 1996, Arlitt and Williamson, 1996, Breslau et al., 1999].

General inter-arrival times can be generated by distributing the requests over the spanning time of the synthetic workload. If the requests are distributed uniformly, then general inter-arrival times roughly follows the exponential distribution. However, several studies have shown that streaming accesses exhibit diurnal patterns [Acharya et al., 2000, Almeida et al., 2001, Chesire et al., 2001, Harel et al., 1999, Luperello et al., 2001]. We call such phenomena *seasonal patterns*, *i.e.*, there are, hourly, daily, and weekly patterns. Users are more likely to request streaming objects during particular periods, making a uniform distribution of requests over the spanning time of the synthetic workload unrealistic.

For HTTP requests, the distribution of inter-arrival time of requests to the same object was found to be the result of two phenomena: the popularity distribution of objects and the temporal correlation of requests [Jin and Bestavros, 2000]. The skew in Web object popularity was found to be directly related to the skew in the inter-arrival time distribution [Breslau et al., 1999, Jin and Bestavros, 2000]. This skew was further increased by temporal correlations of requests. For streaming media accesses, we need to model both of these phenomena.

Popularity Distribution. The skewed popularity of streaming media objects was documented in [Acharya et al., 2000, Aggarwal et al., 1996, Almeida et al., 2001, Chesire et al., 2001, Luperello et al., 2001]. In particular, several studies observed a Zipf-like distribution of streaming object popularity [Almeida et al., 2001, Chesire et al., 2001, Luperello et al., 2001]. Zipf-like distributions imply that the access frequency of an object is inversely proportional to its popularity (rank), *i.e.*, $P(r) \sim r^{-\alpha}$, $1 < r \leq N$, where N is the number of objects, r is the rank, and P is the access frequency of the r -ranked object. A discrete form

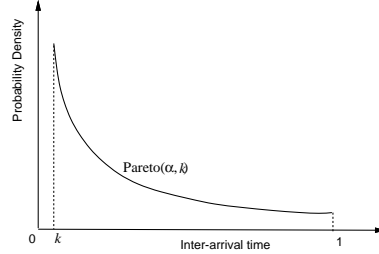


Figure 1.1. Truncated Pareto PDF for interarrival time of correlated requests. The cutoff point is unity, the maximum possible inter-arrival time.

of the probability density function is $f(x) = \frac{1}{\Omega x^\alpha}$, $x = 1, 2, \dots, N$, where $\Omega = \sum_{i=1}^N i^{-\alpha}$. The parameter α is called the *shape parameter* since it determines the level of skewness in the popularity profile. The parameter N is called the *scale parameter*.

Temporal Correlation. If requests to the same object are independent, then they are distributed randomly. This was shown not to be accurate enough for HTTP requests [Jin and Bestavros, 2000]. Similarly, in a number of recent studies, streaming media accesses were shown to exhibit temporal correlations [Acharya et al., 2000, Almeida et al., 2001, Chesire et al., 2001]. For example, it was observed that streaming accesses have much higher overlap during peak loads. To reflect this, we assume that a portion of all request arrivals are correlated, while the remaining request arrivals are independent.

To model correlated inter-arrival times, we use a Pareto distribution. The Pareto distribution has a density function $f(x) = \alpha k^\alpha x^{-\alpha-1}$, where $\alpha, k > 0$ and $x > k$. In [Jin and Bestavros, 2000], it was observed that temporal correlations were stronger when request inter-arrival times were shorter. The Pareto distribution models such a condition well. The Pareto distribution used to characterize temporal correlations has two parameters. The *shape parameter* (α) indicates the skewness of inter-arrival time distribution. The *scale parameter* (k) indicates the time scale of observations. Since we are only interested in a finite period but the random variable with a Pareto distribution can have arbitrarily large values, we need to cut off the Pareto distribution at unity (corresponding to the maximum possible inter-arrival time, or the spanning time of synthetic request stream). Introducing a cutoff for the Pareto distribution necessitates that we normalize it. We do so by defining a *truncated Pareto distribution* with a PDF $f(x) = \alpha \frac{k^\alpha}{1-k^\alpha} x^{-\alpha-1}$, where $\alpha, k > 0$ and $k < x < 1$. In implementation, we use inverse method to generate Pareto-distributed random values.¹ Figure 1.1 illustrates such a truncated PDF.

Seasonal Access Frequency. In GISMO, we do not make any assumptions related to the seasonal patterns of the overall access frequency. Such patterns are application-specific, and depend on various aspects of location and time. For example, several studies [Acharya et al., 2000, Almeida et al., 2001, Chesire et al., 2001, Harel et al., 1999, Luperello et al., 2001] observed such patterns over significantly different time scales (from hours to months). Hence, we assume that a histogram of access frequency (request arrival rate) at different times is provided by users of GISMO. Namely, given the histogram of the overall request arrival rate at different times, we can approximate the CDF $F(t)$, $t \in (0, 1)$. For each request generated in the last step, assume $t \in (0, 1)$ is the request time, then we transform t to another request time $F^{-1}(t)$.

Modeling Individual Sessions

The second aspect of a workload characterization concerns the model used for determining the specifics of each user session.

First, the distribution of object sizes is a main determinant of session duration—the larger the object, the longer the session. HTTP requests are usually shorter, while streaming accesses have much longer durations (typically a few KB for Web objects but up to hundreds of MB for streaming objects). Several studies have found that the session length has heavier tails than an exponential distribution [Almeida et al., 2001, Luperello et al., 2001, Padhye and Kurose, 1998, Chesire et al., 2001].

Second, user activities (including VCR-like stop/fast-forward/rewind/pause functionalities) affect session duration. User interventions are not unique to streaming access and were documented for HTTP requests (*e.g.*, “interrupted” transfers). Such effects are much more common for streaming accesses. For example, it has been observed that nearly a half of all video requests are not completed [Acharya et al., 2000]. In addition, jumps become popular in streaming media access workloads [Almeida et al., 2001, Harel et al., 1999, Padhye and Kurose, 1998].

Third, the bit-rate of streaming objects exhibits important properties which may have implications on transfer time. Specifically, streaming media bit rates exhibit long-range dependence. With long-range dependence, the autocorrelation function decays slowly, meaning that burstiness persists at large time scales. In addition, high variability of frame sizes (a property of the encoding scheme used) can be modeled using a heavy-tailed distribution. Both long range dependence and high variability of VBR have been characterized in [Garrett and Willinger, 1994].

Object Size Distribution. In GISMO, we use the Lognormal distribution to model streaming object sizes. Several studies on workload characterization [Almeida et al., 2001, Luperello et al., 2001, Padhye and Kurose, 1998] found

that the Lognormal distribution fits the distribution of object sizes well. The Lognormal distribution has two parameters, μ , the mean of $\ln(x)$, and σ , the standard deviation of $\ln(x)$. To generate a random variable that follows the Lognormal distribution, we first generate x from an approximation of the standard Normal distribution, and then return $e^{\mu+\sigma x}$ as the value of the Lognormally-distributed random variable representing the object size.

Notice that GISMO allows our choice of the Lognormal distribution to be changed. Specifically, several other distributions (*e.g.*, Pareto and Gamma) were found to provide a good fit for streaming object sizes measured empirically [Almeida et al., 2001, Padhye and Kurose, 1998]. This is one way in which GISMO is extensible: Users of GISMO can easily replace the module for generating object sizes for the synthetic workload with their own module.

User Inter-activity Times. In GISMO, two forms of user interventions (or activities) are modeled—namely, partial accesses due to “stop” activity and jumps due to “fast forward and rewind” activities.

For partial accesses (resulting from a “stop” activity), we need to model the duration of an aborted session. Unfortunately, there are very few empirical studies characterizing partial accesses. The work presented in [Acharya et al., 2000] implies that the stopping time (time until a session is stopped) is not uniformly or exponentially distributed. Instead, stopping is more likely to occur in the beginning of a stream playout. We model such a behavior with a Pareto distribution. We make this choice since stopping probability decreases as the session grows longer (indicating interest in the streamed content, and hence a lower probability of stoppage). A Pareto distribution models this behaviors very well.

For intra-session jumps (resulting from a “fast forward” or “rewind” activity), we need to model the distribution of *jump distances*. In previous work [Padhye and Kurose, 1998], it was found that jump distances tend to be small but that large jumps are not uncommon. In our current implementation of GISMO, we model jump distances using Pareto distributions. In addition to jump distances, we also need to model the duration of continuous play (*i.e.*, intra-jump times). In our current implementation of GISMO, we assume that the duration of continuous play follows an exponential distribution $e^{-\lambda t}$, where λ is the *frequency* of jumps.

Two previous studies [Almeida et al., 2001, Padhye and Kurose, 1998] have used active period (ON period) and silent period (OFF period) in modeling user interactivities. The duration of continuous play (ON period) tends to be heavier-tailed, but for small objects exponential distribution is the most observed [Almeida et al., 2001]. The duration of the silent period is best fit by a Pareto distribution.

VBR Self-Similarity. We model the sequence of frame sizes for a streaming object as a self-similar process [Garrett and Willinger, 1994]. A time series X is said to be *exactly second-order self-similar* if the corresponding “aggregated” process $X^{(m)}$ has the same correlation function as X , for all $m \geq 1$, where the process $X^{(m)}$ is obtained by averaging the original X over successive non-overlapping blocks of size m . The variance of the aggregated process behaves for large m like $Var(X^{(m)}) \approx m^{-\beta}(\sigma^2)_X$, resulting in a single Hurst parameter $H = 1 - \beta/2$. A property of self-similar processes is that the auto-correlation function decays much slower when $H > 0.5$. This means that burstiness persists at large time scale, and implies the ineffectiveness of buffering to smooth out burstiness.

In GISMO, we generate fractional Gaussian noise by generating a fractional Brownian motion (FBM), and then obtaining FGN from the increments of FBM. We implemented a simple and fast approximation of FBM called “Random Midpoint Displacement” (RMD). The RMD method was proposed in [Lau et al., 1995]. RMD works in a top-down fashion. It progressively subdivides an interval over which to generate the sample path. At each division, a Gaussian displacement, with appropriate scaling (d^H , where d is the length of the interval and H is the target Hurst parameter), is used to determine the value of the midpoint. This recursive procedure stops when it gets the FBM process of the required length. The time complexity for RMD is only $O(n)$, where n is the length of the FBM process. Note that RMD generates a somewhat inaccurate self-similar process and that the resulting Hurst parameter may be slightly smaller than the target value. Other methods such as the fast Fourier Transform [Paxson, 1997] can be implemented.

VBR Marginal Distribution. To model the high variability of streaming media bit rates, we use a heavy-tailed marginal distribution to characterize the bit rate. A heavy-tailed distribution is one whose upper tail declines like a power law, *i.e.*, $P[X > x] \sim x^{-\alpha}$, where $0 < \alpha < 2$. In [Garrett and Willinger, 1994], it was found that the tail of the VBR marginal distribution can be modeled using a Pareto distribution. The CDF of Pareto distribution is $F(x) = P[X \leq x] = 1 - (k/x)^\alpha$, where $k, \alpha > 0$ and $x \geq k$. Pareto distributions yield random variables with high variability. If $1 < \alpha < 2$, the random variable with Pareto distribution has finite mean and infinite variance; if $\alpha \leq 1$, it has infinite mean and variance.

In addition to modeling the “tail” of the distribution, we also need to model the “body” of the distribution. Garrett and Willinger [Garrett and Willinger, 1994] found that the Gamma distribution is a good fit for the body, so they used a hybrid Gamma/Pareto for the marginal distribution. We use a Lognormal distribution for the body along with a Pareto tail.

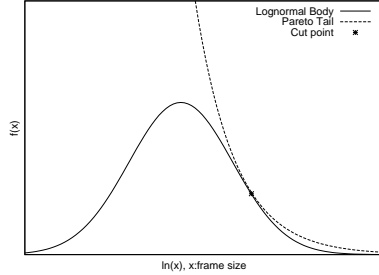


Figure 1.2. A hybrid distribution with Lognormal body/Pareto tail.

Finally, to complete our model of VBR marginal distribution, we use the following approach to “connect” the body to the tail. Given the Lognormal distribution for the body with parameters u and σ , and the cut point between the body and the tail, we can derive the scale and shape parameter of the Pareto tail by equalizing both the value and the slope of the two distributions at the cut point. The resulting hybrid distribution needs to be normalized. One can get different tail distributions by moving the cut point. Figure 1.2 illustrates the fit of a Lognormal distribution and a Pareto distribution.

We use a transformation to generate the required marginal distribution from the FGN Gaussian marginal distribution (CDF $G_{\mu,\sigma}$). The parameters μ and σ can be computed from FGN samples. Then we transform it to a hybrid Lognormal/Pareto distribution with CDF F_{hybrid} . To do this, for each sample value x in the FGN process, the new value is computed as $F_{hybrid}^{-1}(G_{\mu,\sigma}(x))$.

We test the Hurst parameter of the resulting VBR frame size series using *variance-time plot*. A variance-time plot should show that if the sample is aggregated by a factor of m , then the variance decreases by a factor of $m^{-\beta}$, where $\beta = 2 - 2H$. Since the RMD algorithm is an approximation, and the transformation of marginal distribution may not preserve the Hurst parameter very well, we repeat the last two steps if the resulting H value is not close enough to the target value.

For example, we generate a VBR series for 100000 frames with target Hurst parameter 0.8. The given parameters are $\mu = 6$, $\sigma = 0.4$, and cut point 560. We derive other parameters $\alpha = 2.05$ and $k = 335$ for the Pareto tail. The hybrid distribution needs to be normalized by a factor 0.876. Figure 1.3(a) shows the marginal distribution of the synthetic trace. It fits the target distribution well. We test the Hurst parameter using larger number of samples. Figure 1.3(b) shows the variance-time plot from a sequence of one million frame sizes. It shows that the resulting H value is smaller than the target value when the aggregation level is low. At intermediate and high aggregation level, the difference between the target Hurst value and the resulting is less than 0.01.

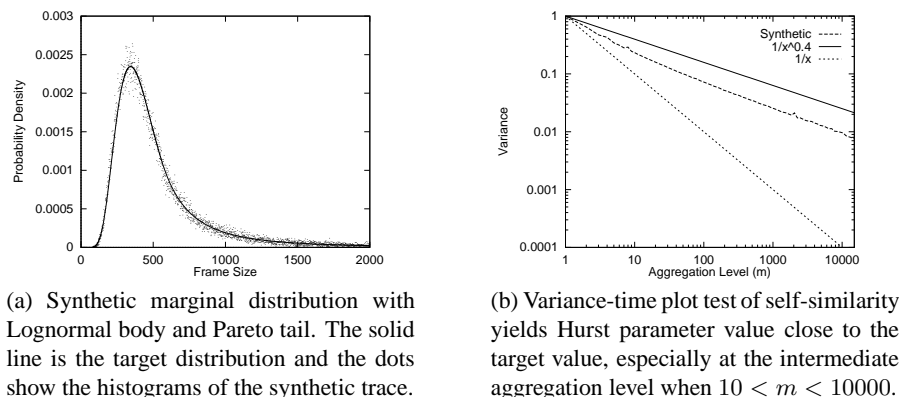


Figure 1.3. Comparisons of synthetic VBR sequence with target parameters.

4. Adapting GISMO to Various Architectures

GISMO was designed as a “toolbox” that allows the evaluation of various streaming delivery architectures. A typical architecture would involve a set of *users* accessing a set of streaming *objects* stored on a set of streaming *servers* via a *network*. Figure 1.4 illustrates such an architecture. The media players are usually the *Plug-ins* of the Web browsers (we show them coupled). When a user is browsing an HTTP page with links to streaming objects, a media player is launched. The media player may be using different protocols to stream the data from the streaming server, *e.g.*, UDP, TCP, and RTSP. In addition to the entities shown in Figure 1.4, there could be other components that may play a role in the delivery of streaming media (*e.g.*, caching proxies inside the network or replicated servers for parallel downloads).

The workload generated by GISMO for the performance evaluation of a given architecture consists of two parts: (a) the set of phantom streaming objects² available at the server(s) for retrievals, and (b) a schedule of the request streams generated by various clients.³ To use such a workload, the set of streaming objects are installed on the servers and schedules specifying client accesses are installed on the clients. Once installed, a GISMO workload can be played out simply by having clients sending requests to the server(s) according to the schedule of accesses at such a client.

By virtue of its design, GISMO allows the evaluation of any “entity” in the system (lying between request generating clients and content providing servers). To do so requires that such entities be “coded” as part of an end-to-end architecture to be evaluated. While a user of GISMO is expected to develop one or more modules for the entity to be evaluated (*e.g.*, a caching or patching algorithm), he/she is not expected to provide the many other entities necessary to complete the end-to-end architecture. To that end, and in addition to the

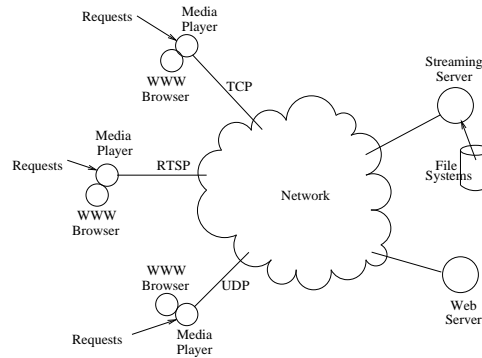


Figure 1.4. Synthetic workload components: (1) Access schedules generated by client requests, and (2) Streaming objects stored on servers.

above two main components of a workload (the objects on the servers and the schedules at the clients), GISMO provides support for various other ingredients of a streaming media delivery system. Examples of these include modules to implement simple transport protocols (*e.g.*, UDP, TCP, RTP) and modules to interface clients and server to an emulated network (*e.g.*, NistNet).

The following are examples of “use cases” for GISMO: (1) To evaluate the capacity of a streaming server, a number of clients are used to generate requests to the server under test. This can be done on a LAN and clients do not have to be real media players. The interesting aspects of the server performance (that a GISMO user may want to evaluate using simulations) may include its scheduling, its memory and CPU behavior, and caching, etc. (2) Evaluating network protocols for streaming data transmission. For this purpose, the data is streamed using the protocol under investigation, but one may use simple implementation of media players and streaming servers. An example of using GISMO in such a study is our work on stream merging and periodic broadcasting protocols [Jin and Bestavros, 2002]. (3) Evaluating streaming data replication techniques. For this purpose, one can study how streaming objects are replicated via the Internet to provide better services to the users. The replication techniques include proxy caching, prefetching, work-ahead smoothing, and multicasting etc. An example of using GISMO in such a study is our work on partial caching [Jin et al., 2002] as well as the case study we present in the next section.

5. GISMO in Action: Evaluating Caching versus Patching

To demonstrate its usefulness, we describe how GISMO was used to generate realistic workloads, which were used to compare the effectiveness of proxy caching and server patching techniques in reducing bandwidth requirements.

We conducted a base experiment to measure the server bandwidth requirements for a system using neither caching nor patching. GISMO was used to

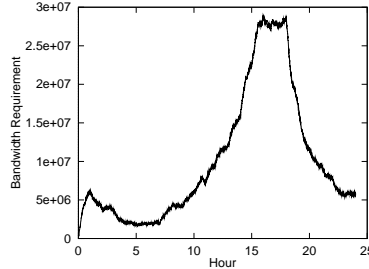


Figure 1.5. Base server bandwidth requirements.

generate a total of 50000 requests to 500 streaming objects stored on the server. Requests were over a one-day period, with three hours of peak activities. We used $\alpha = 0.7$ to describe the popularity skew. Requests were not temporally correlated and the streams were played out without interruptions. We used a Lognormal distribution with $\mu = 10.5$ and $\sigma = 0.63$ to model the streaming object sizes (in number of frames), resulting in a mean object size of approximately 43K frames. To model the VBR frame sizes, we used Lognormal with $\mu = 5.8$, $\sigma = 0.4$ to model the body of the distribution and a Pareto with $\alpha = 1.82$ and $k = 248$ bytes to model its tail, with the cut point between the body and the tail set to 400 bytes. Under this model, the mean bit-rate was close to $100Kbps$ with 24 frames per second. The sequences of frame sizes were generated with a target Hurst parameter $H = 0.8$. Figure 1.5 shows the base bandwidth (bytes per second) needed by the server.

Next, we conducted a number of experiments to study the effectiveness of proxy caching and server patching techniques. To that end, we considered *bandwidth reduction ratio* as the metric of interest. This metric is computed by normalizing the mean bandwidth requirement for a system using caching or patching with respect to the base bandwidth requirement (similar to that shown in Figure 1.5). In our experiments, we varied various parameters of the workload and report the bandwidth reduction ratio (as a function of such parameters), focusing only on the 3-hour period of peak load.

To study the effectiveness of caching, we considered a system with 100 proxies, each with infinite cache size. A proxy can satisfy a request if it has a previously-fetched copy of the streaming object in its cache. To study the effectiveness of patching, we considered a system in which the server patches its response to requests it receives (to the same object) within a short period of time. This was done using the optimal threshold-based patching schemes proposed in [Gao and Towsley, 1999] (assuming that clients had enough buffer space).

Figure 1.6 shows the performance of proxy caching and server patching when the total number of requests and the skewness parameter α change. We

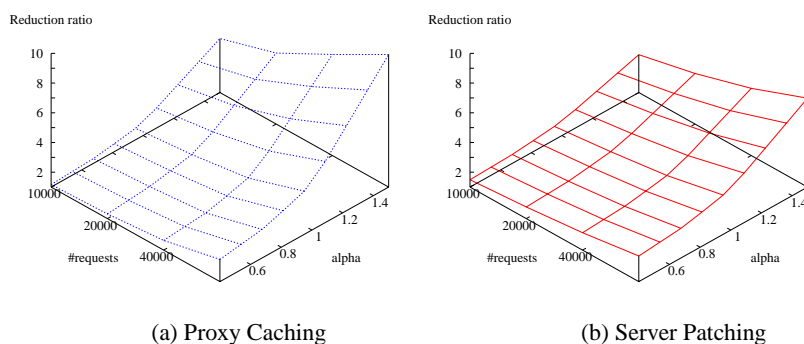


Figure 1.6. Server bandwidth reduction ratios of proxy caching and server patching schemes when popularity parameters change. Larger α is more important for caching.

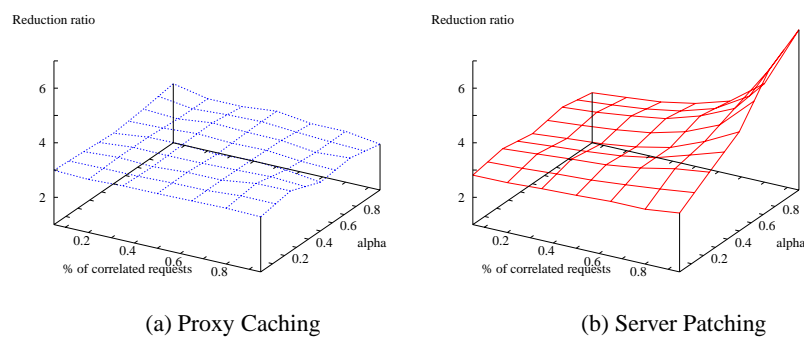


Figure 1.7. Server bandwidth reduction ratios of proxy caching and server patching schemes when correlation parameters change. Strong temporal correlation favors server patching.

observe that for proxy caching, a larger α results in higher bandwidth reduction ratio. This means that for proxy caching, the concentration of requests on a smaller number of “popular” objects is much more important than it is for server patching techniques. Recent studies [Acharya et al., 2000, Chesire et al., 2001, Almeida et al., 2001] of streaming access logs suggest that such popularity skew for streaming media access is limited, *i.e.*, α is likely to have small values. This suggests that it is difficult to achieve high bandwidth reduction ratios using proxy caches. From Figure 1.6, we also observe that increasing the number of requests in the workload increases the efficiency of both techniques. Since we assume a fixed number (100) of proxies, increasing the number of requests in effect increases sharing among users.

Figure 1.7 shows the performance of proxy caching and server patching when the percentage of temporally correlated requests and the correlation skewness α

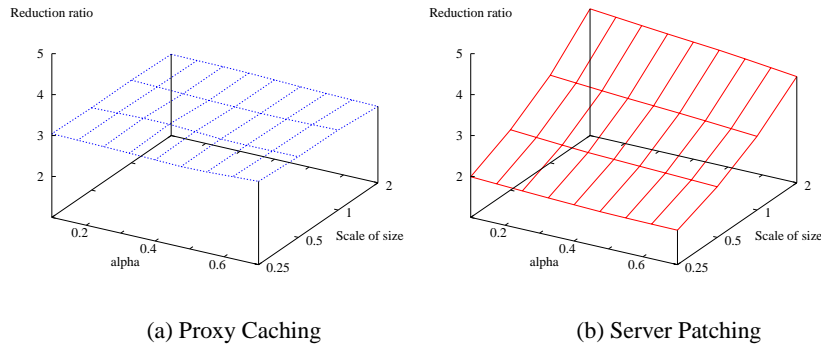


Figure 1.8. Server bandwidth reduction ratios of proxy caching and server patching schemes when size distribution parameters change. Larger sizes favors server patching.

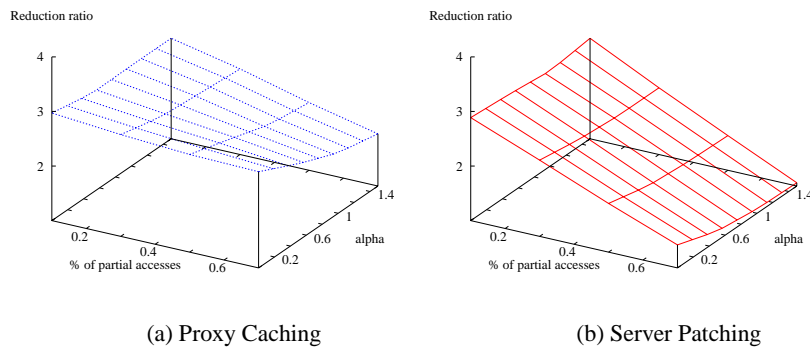


Figure 1.9. Server bandwidth reduction ratios of proxy caching and server patching schemes when partial access parameters change. Early stops degrade server patching performance significantly, but only affect caching moderately.

are changed. For proxy caching, the correlation of requests is almost irrelevant.⁴ For server patching, increasing the percentage of correlated requests or increasing the skewness of correlated inter-arrival times results in higher reduction ratios. Nevertheless, when correlation is not strong, the reduction ratio is only slightly higher than when no correlation exists. Thus, for evaluating server patching techniques, Poisson arrivals are adequate in workloads with weak correlations.

Figure 1.8 shows the performance of proxy caching and server patching when object sizes are scaled and the size skewness parameter α changes. Again, the effectiveness of proxy caching is not affected by size distribution. For server patching, the larger the objects, the higher the reduction ratio. This is expected since long streams offer more opportunities for patching. However,

the skewness parameter has less of an effect, suggesting that it is adequate to use a mean-size streaming object to study the effectiveness of server patching. One implication from this experiment is that a good hybrid strategy would involve using caches for smaller objects and patching for longer streams.

Figure 1.9 shows the performance of proxy caching and server patching, when the probability of partial accesses and the partial access skewness parameter α are varied. Increasing the fraction of partially-accessed objects (*i.e.*, probability of early stops) hurts the performance of both proxy caching and server patching. While the impact on proxy caching performance is marginal, the impact on server patching is disastrous. This suggests that for streaming access allowing a high degree of user inter-activity, server patching is not a promising technique at all.

To summarize, our case study demonstrates the importance of a realistic and scalable streaming access workload generator by showing that the characteristics of a workload may have great impacts on the effectiveness of a streaming content delivery solution. Changing the workload characteristics does indeed change the relative performance of various techniques.

6. Summary

GISMO generates streaming access workloads, which are parameterized so as to match properties of real workloads, including object popularity, temporal correlation of requests, seasonal access patterns, user session durations, user inter-activity, and VBR long-range dependence and marginal distribution. We demonstrated the value of GISMO by showing that the relative performance of proxy caching and server patching techniques is inherently dependent on properties of the workload used to evaluate them.

Notes

1. To generate a random variate following Pareto distribution $f(x)$, we first compute the inverse CDF $F^{-1}(x)$. A random variable $r \in (0, 1)$, *i.e.*, uniformly-distributed r is generated, and the inter-arrival time is $F^{-1}(r)$.
2. While the contents of “phantom” objects generated by GISMO are not comprehensible (not real audio or video), their characteristics conform to the specific parameters of desired distributions (*e.g.*, VBR auto-correlation, VBR marginal distributions, sizes, *etc.*)
3. A GISMO client is a software entity that mimics a configurable set of *real users*, each generating requests conforming to the various distributions of popularity, inter-activities, *etc.*
4. Request correlation (a.k.a. locality of reference) would be relevant for finite-size proxy caches because it impacts the effectiveness of cache replacement algorithms.

References

- [Acharya and Smith, 1998] Acharya, Soam and Smith, Brian (1998). An experiment to characterize videos stored on the Web. In *Proceedings of MMCN*.

- [Acharya et al., 2000] Acharya, Soam, Smith, Brian, and Parns, Peter (2000). Characterizing user access to videos on the World Wide Web. In *Proceedings of MMCN*.
- [Aggarwal et al., 1996] Aggarwal, Charu C., Wolf, Joel L., and Yu, Philip S. (1996). On optimal batching policies for video-on-demand storage servers. In *Proceedings of ICMCS*.
- [Almeida et al., 2001] Almeida, Jussara, Krueger, Jeffrey, Eager, Derek, and Vernon, Mary (2001). Analysis of educational media server workloads. In *Proceedings of NOSSDAV*.
- [Almeida et al., 1996] Almeida, Virgilio, Bestavros, Azer, Crovella, Mark, and de Oliveira, Adriana (1996). Characterizing reference locality in the WWW. In *Proceedings of PDIS*.
- [Arlitt and Williamson, 1996] Arlitt, Martin and Williamson, Carey (1996). Web server workload characteristics: The search for invariants. In *Proceedings of SIGMETRICS*.
- [Barford et al., 1999] Barford, Paul, Bestavros, Azer, Bradley, Adam, and Crovella, Mark (1999). Changes in Web client access patterns: Characteristics and caching implications. *World Wide Web*, 2(1):15–28.
- [Barford and Crovella, 1998] Barford, Paul and Crovella, Mark (1998). Generating representative Web workloads for network and server performance evaluation. In *Proceedings of SIGMETRICS*.
- [Breslau et al., 1999] Breslau, Lee, Cao, Pei, Fan, Li, Phillips, Graham, and Shenker, Scott (1999). Web caching and Zipf-like distributions: Evidence and implications. In *Proceedings of INFOCOM*.
- [Carter and Long, 1997] Carter, S. W. and Long, D. D. E. (1997). Improving video-on-demand server efficiency through stream tapping. In *Proceedings of ICCCN*.
- [Cherkasova and Gupta, 2002] Cherkasova, Ludmila and Gupta, Minaxi (2002). Characterizing locality, evolution, and life span of accesses in enterprise media server workloads. In *Proceedings of NOSSDAV*.
- [Chesire et al., 2001] Chesire, Maureen, Wolman, Alec, Voelker, Geoff, and Levy, Henry (2001). Measurement and analysis of a streaming workload. In *Proceedings of USITS*.
- [Costa et al., 2004] Costa, Cristiano P., Cunha, Italo S., Borges, Alex, Ramos, Claudiney Vander, Rocha, Marcus M., Almeida, Jussara M., and Ribeiro-Neto, Berthier A. (2004). Analyzing client interactivity in streaming media. In *Proceedings of WWW Conference*.
- [Crovella and Bestavros, 1996] Crovella, Mark and Bestavros, Azer (1996). Self-similarity in World Wide Web traffic: Evidence and possible causes. In *Proceedings of SIGMETRICS*.

- [Cunha et al., 1995] Cunha, Carlos, Bestavros, Azer, and Crovella, Mark (1995). Characteristics of WWW client-based traces. Technical Report BU-CS-95-010, Computer Science Department, Boston University.
- [Eager et al., 2001] Eager, Derek, Vernon, Mary, and Zahorjan, John (2001). Minimizing bandwidth requirements for on-demand data delivery. *IEEE Transactions on Data and Knowledge Engineering*, 13.
- [Gao and Towsley, 1999] Gao, Lixin and Towsley, Don (1999). Supplying instantaneous video-on-demand services using controlled multicast. In *Proceedings of ICMCS*.
- [Garrett and Willinger, 1994] Garrett, Mark W. and Willinger, Walter (1994). Analysis, modeling and generation of self-similar VBR video traffic. In *Proceedings of SIGCOMM*.
- [Gribble and Brewer, 1997] Gribble, Steven D. and Brewer, Eric A. (1997). System design issues for Internet middleware services: Deductions from a large client trace. In *Proceedings of USITS*.
- [Harel et al., 1999] Harel, Nissim, Vellanki, Vivekanand, Chervenak, Ann, Abowd, Gregory, and Ramachandran, Umakishore (1999). Workload of a media-enhanced classroom server. In *Proceedings of Workshop on Workload Characterization*.
- [Hua et al., 1998] Hua, Kien A., Cai, Ying, and Sheu, Simon (1998). Patching: A multicast technique for true video-on-demand services. In *Proceedings of ACM MULTIMEDIA*.
- [Jin and Bestavros, 2000] Jin, Shudong and Bestavros, Azer (2000). Temporal locality in Web request streams: Sources, characteristics, and caching implication (poster). In *Proceedings of SIGMETRICS*.
- [Jin and Bestavros, 2001] Jin, Shudong and Bestavros, Azer (2001). GISMO: Generator of Streaming Media Objects and Workloads. *Performance Evaluation Review*, 29(3).
- [Jin and Bestavros, 2002] Jin, Shudong and Bestavros, Azer (2002). Scalability of multicast delivery for non-sequential streaming access. In *Proceedings of ACM SIGMETRICS*.
- [Jin et al., 2002] Jin, Shudong, Bestavros, Azer, and Iyengar, Arun (2002). Accelerating Internet streaming media delivery using network-aware partial caching. In *Proceedings of IEEE ICDCS*.
- [Lau et al., 1995] Lau, W.-C., Erramilli, A., Wang, J. L., and Willinger, W. (1995). Self-similar traffic generation: The random midpoint displacement algorithm and its properties. In *Proceedings of ICC*.
- [Leland et al., 1994] Leland, Will E., Taqqu, Murad S., Willinger, Walter, and Wilson, Daniel V. (1994). On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Trans. on Networking*, 2(1).

- [Li et al., 2005] Li, Mingzhe, Claypool, Mark, Kinicki, Robert, and Nichols, James (To appear). Characteristics of streaming media stored on the web. *ACM Transactions on Internet Technology (TOIT)*.
- [Luperello et al., 2001] Luperello, Dario, Mukherjee, Sarit, and Paul, Sanjoy (2001). Streaming media traffic: An empirical study. In *Proceedings of Web Caching Workshop*.
- [Padhye and Kurose, 1998] Padhye, J. and Kurose, J. (1998). An empirical study of client interactions with a continuous-media courseware server. In *Proceedings of NOSSDAV*.
- [Padmanabhan and Qiu, 2000] Padmanabhan, Venkata N. and Qiu, Lili (2000). The content and access dynamics of a busy Web site: Findings and implications. In *Proceedings of SIGCOMM*.
- [Paxson, 1997] Paxson, V. (1997). Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic. *Computer Communication Review*, 27:5–18.
- [Tang et al., 2003] Tang, Wenting, Fu, Yun, Cherkasova, Ludmila, and Vahdat, Amin (2003). Medisyn: A synthetic streaming media service workload generator. In *Proceedings of NOSSDAV*.
- [The Standard Performance Evaluation Corporation,] The Standard Performance Evaluation Corporation. Specweb96. available from <http://www.specbench.org/org/web96>.
- [van der Merwe et al., 2002] van der Merwe, Jacobus, Sen, Subhabrata, and Kalmanek, Charles (2002). Streaming video traffic: Characterization and network impact. In *Proceedings of Web Caching Workshop*.
- [Velooso et al., 2002] Velooso, Eveline, Almeida, Virgilio, Meira, Wagner, Bestavros, Azer, and Jin, Shudong (2002). A hierarchical characterization of a live streaming media workload. In *Proceedings of SIGCOMM Internet Measurement Workshop*.
- [Zipf, 1929] Zipf, George Kingsley (1929). *Relative Frequency as a Determinant of Phonetic Change*. Reprinted from Harvard Studies in Classical Philology, XL.