MediSyn: A Synthetic Streaming Media Service Workload Generator *

Wenting Tang
Hewlett Packard Labs
1501 Page Mill Rd
Palo Alto, CA 94303
wenting.tang@hp.com

Yun Fu
Dept of Computer Science
Duke University
Durham, NC 27708
fu@cs.duke.edu

Ludmila Cherkasova
Hewlett Packard Labs
1501 Page Mill Rd
Palo Alto, CA 94303
lucy.cherkasova@hp.com

Amin Vahdat
Dept of Computer Science
Duke University
Durham, NC 27708
vahdat@cs.duke.edu

ABSTRACT

Currently, Internet hosting centers and content distribution networks leverage statistical multiplexing to meet the performance requirements of a number of competing hosted network services. Developing efficient resource allocation mechanisms for such services requires an understanding of both the short-term and long-term behavior of client access patterns to these competing services. At the same time, streaming media services are becoming increasingly popular, presenting new challenges for designers of shared hosting services. These new challenges result from fundamentally new characteristics of streaming media relative to traditional web objects, principally different client access patterns and significantly larger computational and bandwidth overhead associated with a streaming request. To understand the characteristics of these new workloads we use two long-term traces of streaming media services to develop MediSyn, a publicly available streaming media workload generator. In summary, this paper makes the following contributions: i) we model the *long-term* behavior of network services capturing the process of file introduction and changing file popularity, ii) we present a novel *generalized* Zipf-like distribution that captures recently-observed popularity of both web objects and streaming media not captured by existing Zipf-like distributions, and iii) we capture a number of characteristics unique to streaming media services, including file duration, encoding bit rate, session duration and non-stationary popularity of media accesses.

Categories and Subject Descriptors

H.1.0 [**Models and Principles**]: General

General Terms

Measurement,Performance,Design,Algorithms

Keywords

streaming media, workload analysis, workload generator

*This work was partially completed while Y. Fu was a summer intern at HP Labs during 2002. A. Vahdat and Y. Fu are supported in part by the National Science Foundation (EIA-9972879). A. Vahdat is also supported by an NSF CAREER award (CCR-9984328).

NOSSDAV'03, June 1–3, 2003, Monterey, California, USA.

1. INTRODUCTION

Two recent trends in network services motivate this work, a move toward shared service hosting centers and the growing popularity of streaming media. Traditionally, service providers over-provision their sites to address highly bursty client access patterns. These access patterns can vary by an order of magnitude on an average day [9] and by three orders of magnitude in the case of flash crowds. In fact, services are often most valuable exactly when the unexpected takes place. Thus, we are pursuing a vision where large-scale hosting infrastructures simultaneously provide "resource-on-demand" capabilities to competing Internet services [17, 7]. The idea is that the system can use statistical multiplexing and efficient resource allocation to dynamically satisfy the requirements of services subject to highly bursty access patterns.

A second emerging trend is the growing popularity of streaming media services. Streaming media takes the form of video and audio clips from news, sports, entertainment, and educational sites. Streaming media is also gaining momentum in enterprise intranets for training purposes and company broadcasts. These workloads differ from traditional web workloads in many respects, presenting a number of challenges to system designers and media service providers [12, 16, 11]. For instance, transmitting media files requires more computing power, bandwidth and storage and is more sensitive to network jitter than web objects. Further, media access lasts for a much longer period of time and allows for user interaction (pause, fast forward, rewind, etc.).

The long-term goal of our work is to study resource provisioning and resource allocation at the confluence of the above two trends: network service hosting infrastructures for next-generation streaming workloads. A key obstacle to carrying out such a study is the lack of understanding of changing client access patterns over a long period of time. For both hosting centers and content distribution networks (CDNs), we require such an understanding to determine, for example, how to place objects at individual sites (potentially spread across the network) and how to allocate resources to individual streams and to individual clients.

Thus, we use long-term traces from two streaming media services to construct an open-source media workload generator called MediSyn[1]. For MediSyn, we develop a number of novel models to capture a broad range of characteristics

[1]Currently MediSyn is implemented on UNIX platforms. The software can be downloaded from `http://www.hpl.hp.com/research/iii/projects/medisyn.html`.

for network services. We also demonstrate how these models generalize to capture the characteristics of traditional web services. Overall, this paper makes the following contributions:

- A primary contribution of our work is its focus on the *long-term* behavior of network services. Among the features of our synthetic generator is the ability to reflect the dynamics and evolution of content at media sites and the change of access rate to this content over time. Existing workload generators assume that there is a set of active objects fixed at the beginning of the "trace". Similarly, existing techniques assume that object popularity remains the same over the entire duration of the experiment. While these are reasonable assumptions for experiments designed to last for minutes, we are interested in long-term provisioning and resource allocation, as well as the resource allocation for simultaneously competing services (consider a CDN simultaneously hosting hundreds of individual services).
- It was observed [2, 11] that the popularity distribution in media workloads collected over significant period of time (more than 6 months) does not follow a Zipf-like distribution. We present a novel *generalized* Zipf-like distribution to capture the popularity distribution in such workloads. The traditional Zipf-like distribution is a special case of the proposed *generalized* Zipf-like distribution.
- We designed a set of new models to capture a number of characteristics critical to streaming media services, including file duration, file access prefix duration, non-stationary file popularity, new file introduction process and diurnal access patterns.

The rest of this paper is organized as follows. Section 2 outlines the workload properties that MediSyn attempts to capture and the workloads used to develop the models for MediSyn. We present the workload generation process adopted by MediSyn in Section 3. Section 4 introduces the models used in MediSyn and discusses their specifics. We review previous related work in Section 5. Finally, we conclude with a summary and future work in Section 6.

2. MEDIA WORKLOAD PROPERTIES

Accurate workload characterization is critical for successful generation of realistic workloads. A synthetic media workload generator can produce traces with targeted, controllable parameters and desired distributions for performance experiments studying effective streaming media delivery architectures and strategies. For such experiments, the generated workload must not only mimic the highly dynamic resource-utilization patterns found on today's media systems but also provide flexible means to generate more intensive, bursty and diverse workloads for future media systems. Challenges to designing a useful analytical workload generator include:

- identifying essential properties of workloads targeted by synthetic workload generators, and those that most affect the behavior of hosting centers;
- designing appropriate statistical models that closely reproduce the identified workload properties from real traces.

In this section, we highlight the main properties of streaming media workloads modeled in MediSyn, and explain why we believe these properties are important. Throughout this paper, we use two representative streaming media server logs, collected over a period of years, to demonstrate the chosen properties and to validate our statistical models introduced to reflect these properties. The streaming media server logs represent two different media services: *HP Corporate Media Solutions Server (HPC)* and *HPLabs Media Server (HPL)*. The HPC site hosts diverse video coverage of major event, keynote speeches, hardware and software releases of HP, etc. The HPL site provides video coverage about HP Labs, such as coffee talks, presentations, seminars, etc. Table 1 briefly summarizes the workloads.

	HPC	HPL
Log duration	29 months	21 months
Number of files	2,999	412
Number of sessions	666,074	14,489

Table 1: Summary for two media logs used to develop property models in MediSyn.

We partition media workload properties in two groups: *static* and *temporal* properties. **Static properties** provide the characteristics of the underlying media file set, reflect the aggregate, quantitative properties of client accesses (independent of the access time), and present the properties of individual file accesses. **Temporal properties** reflect the dynamics and evolution of accesses to media content over time, and determine the ordering and the timing of session arrivals. Sections 2.1 and 2.2 briefly describe these properties.

2.1 Static Properties

- **File Duration**. Different from web, media files have two dimensions: *duration* of the file measured in *time* and the corresponding media file *size* measured in *bytes*, a direct product of the file duration and the corresponding encoding bit rate. Earlier media workload studies report various distributions characterizing file durations. However, the duration distribution is largely determined by the nature of the media content stored at a site, which could be a short (a few minutes) media clip or a long (one to two hours) movie. Thus, any single distribution, e.g, the heavy-tail distribution used to capture web object size, may fail to characterize media file duration.
- **File Encoding Bit Rate**. The encoding bit rate of a media file determines the bandwidth and system resources required to deliver the file. Typically, there is a set of popular encoding bit rates offered by commercial encoding software and guided by the bandwidth from different groups of Internet audience.
- **File Popularity**. File popularity is defined as the number of accesses to a file over the trace period. Recent studies observe highly uneven file popularity both in web and streaming media workloads [12, 11, 4], implying that most accesses to a site are concentrated on a relatively small set of files. 14%(30%) of the files accessed on the server account for 90% of the media sessions for the HPC(HPL) trace. Traditionally, a Zipf-like distribution is used to capture the file popularity for

web workloads. However, several media workload studies [2, 11] report that the popularity distribution for media workloads over significant periods of time (more than 6 months) does not follow a Zipf-like distribution. Instead, the distribution curve (on a log-log scale) exhibits a "circular"-shape. Accurately reflecting these skewed file popularity distributions is very important.

- **File Access Prefix**. Prior studies [11, 3] show that many clients do not finish the playback of a full media clip. Typically, this reflects the browsing nature of client accesses, client time constraints, or QoS-related issues. Most incomplete sessions access only the initial segments of media files. In the HPC(HPL) trace, only 29%(12.6%) of the accesses finish the playback of the files. 50%(60%) of the accesses last less than 2 minutes [11]. This high percentage of incomplete accesses and the existence of a large number of sessions accessing only the initial part of the file are widely exploited in streaming media cache design [20].

2.2 Temporal Properties

Temporal reference locality, which is universally observed in web and media workloads [12, 11, 4], is the primary factor that affects session arrival ordering. Temporal locality states that recently accessed objects are likely to be accessed in the near future in the access stream. Two factors can cause the temporal locality in the access stream: *skewed popularity distribution* and *temporal correlation* [10, 14].

We observe that our traces only exhibit long-term temporal correlation on daily time scale, and that there is not obvious temporal correlation within a single day [21]. We introduce temporal properties, *new file introduction process*, *life span*, and *diurnal access pattern*, to model temporal locality. The temporal locality generated by MediSyn is a combined *effect* of the skewed popularity distribution, new file introduction process and life span distributions of media files. The concrete session arrival times are determined by *diurnal access pattern*.

- **New File Introduction**. Among the design goals of our synthetic generator is the ability to reflect the evolution of media content provided by various media sites over a long period of time (months). Thus we must enable MediSyn to model the dynamic introduction of new content and its relative popularity over the entire trace period.
- **File Life Span**. A new property *life span* has recently been proposed [11] to measure the change of access rate for media files. We observe that accesses to a media file are not uniformly distributed over the entire trace period. Most accesses occur shortly after the file is introduced, with access frequency gradually decreasing over time. For the HPC(HPL) log, 52%(51%) of the accesses occur during the first week after file introduction, while only 16%(10%) of the accesses occur during the second week, etc. Hence, the file access frequency (file popularity) changes over time. In other words, file popularity is *non-stationary* over the trace period.
- **Diurnal Access Patterns**. The diurnal access pattern defines how the number of accesses to a site varies during a given period of time, e.g., a day. Earlier studies observed the diurnal access patterns for streaming media workloads [12, 16, 2, 3, 15]. Diurnal access patterns might significantly vary for different media sites. The diurnal access pattern is important for capturing the burstiness of resource consumption within a given time period. The diurnal access patterns are defined using the second time scale in our synthetic workload generator, e.g. within a day.

3. WORKLOAD GENERATION PROCESS IN MEDISYN

MediSyn's goal is to generate a synthetic trace representing a sequence of file accesses (sessions) to a set of media files. During the generation process, we incorporate all the properties we introduced in Section 2. A workload generated by MediSyn is comprised of a file set and sessions. Each file has its own popularity, duration, and encoding bit rate. In our session model, a session may be terminated without finishing the playback of a media file. Media session interactivity is not modeled. So each session specifies an arrival time, the accessed media file, and a prefix duration.

Overall, the generation process consists of two steps: *file property generation* and *file access generation*.

- **File Property Generation**

  The first step is to generate values for file-based properties introduced in Section 2 for each file. MediSyn generates these properties based on a parameterized set of distributions specified in a configuration file. If a property can be described by a value such as file popularity, duration, or encoding bit rate, MediSyn first generates a sequence of values according to the specified distribution, and then selects a value for each file. There are well-defined methods to generate a series of values following a particular distribution [13]. For example, we use the rejection method [13] to generate a series of values following *normal* or *lognormal* distributions.

  If a property of a file is modeled as a distribution, the choice of the distribution and the parameter(s) of the distribution are generated for the file. For example, during the generation of life span property, for each file, we first generate the choice of the distribution (either *Pareto* or *lognormal*). If a file's life span follows a *Pareto* distribution, the α parameter is generated. If it follows a *lognormal* distribution, both μ and σ for this *lognormal* distribution are generated.

  Overall, a file is the basic unit to which the property values are propagated at the first step of workload generation. At the end of the this step, a set of static and temporal properties shown in Table 2 is generated for each file. Section 4 describes each property generation and correlations among the properties in detail.

- **File Access Generation**

  In MediSyn, the session generation process first generates sessions for each individual file and then simply merges them according to session arrival time. To generate sessions for a particular file, MediSyn takes the assigned file popularities as the basis and generates the arrivals of media sessions to the file using: i) the initial file introduction time, ii) the life span of the file, and iii) the diurnal access pattern of the file. We also generate the prefix duration for each session. Details of generation are given in Section 4.

File id	Duration	Bit rate (Kbps)	Popularity	File Introduction Time (sec)	Life span	Life span parameters	···
1	3600	112	20000	100	*Pareto*	1.0	···
2	200	350	14300	50	*Lognormal*	2.0,10.0	···
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	600	28.8	1	10000	*Lognormal*	1.0, 1.0	···

Table 2: Properties generated for each file.

Each session includes the following three fields:

- *timestamp* indicating the session arrival time,
- *file id* specifying the target file accessed during the media session,
- *file access prefix* describing the duration of the media session.

Once a sequence of media sessions is generated for each file, all the media sessions are sorted according to a global time and merged to generate a synthetic trace.

4. MAIN MODELS OF WORKLOAD GENERATION IN MEDISYN

This section describes the models used in MediSyn to capture static and temporal properties of streaming media workloads.

4.1 Duration

Prior studies [11, 3] observed that media files might be classified into a set of groups according to their durations. Different workloads can be grouped based on the content of media files hosted by a streaming service. For example, music sites may have file durations from 3 to 5 minutes, while movie sites may have file durations from one and half to two hours. While a particular workload might be captured by a certain statistical distribution, this distribution may fail to capture another workload. In our case, although the file duration distribution of the HPC trace can be modeled by a heavy-tail distribution such as a *Weibull* distribution [13], the HPL trace cannot be captured by it at all.

As shown in Figure 1 (a) and Figure 2 (a), the file durations in our traces are concentrated around a set of hot points. These hot points are usually some common durations, semantically meaningful to some particular types of media content. Based on this observation, we classify these hot points into a set of groups and use a set of *normal* distributions to model the grouped file duration distribution as shown in Figure 1 (a) and Figure 2 (a). Here, each group is modeled by a *normal* distribution with the mean (μ) defined by the hot point of that group. The standard deviation (σ) of each *normal* distribution determines the concentration of the durations within that group.

Note that we do not use segmented PDFs (Probability Density Functions) to model the duration distribution. We assume a hot point can affect the entire duration scope rather than just a segment. Thus, we use an aggregated distribution, whose PDF sums the PDFs of all *normal* distributions proportionally. To proportionally sum all duration groups, we associate each group with a ratio determined by the number of files in the group compared with the total number of files in the trace. So the *normal* distribution PDF of each group is normalized against the ratio of that group. If only a fraction of a *normal* distribution for a group is used, normalization is performed on the adopted fraction of the distribution. For example, since the mean of the first group in Table 3 is 0, only half of the *normal* distribution is used. Tables 3 and 4 present the mean (μ), the standard deviation (σ) and the ratio of each *normal* distribution for the HPC and HPL traces respectively. They show that the HPC and HPL traces have different hot points.

In MediSyn, users can specify a set of duration groups with different μ, σ and ratios based on the nature of the media workload they want to generate. For each duration group, MediSyn generates a sequence of durations according to the ratio and the *normal* distribution of the group. We use the rejection method [13] to generate the duration sequence according to the parameterized *normal* distribution. Figure 3 (a) and Figure 3 (b) show the durations generated by MediSyn to simulate the HPC trace based on the parameters in Table 3.

Group	1	2	3	4
μ	0	2000	3300	5821
σ	600	400	600	1223
ratio	63%	10%	18%	9%

Table 3: Parameters of the *normal* distributions for the HPC trace.

Group	1	2	3	4	5
μ	117	2900	4200	5160	6300
σ	1200	240	360	180	1000
ratio	19%	26%	30%	10%	15%

Table 4: Parameters of the *normal* distributions for the HPL trace.

4.2 Encoding Bit Rate

Since most of commercial media servers are designed to stream media files encoded at some constant bit rates, the current version of MediSyn is designed to only generate a set of constant bit rates for the underlying file set. MediSyn models encoding bit rates by a discrete distribution, where the value of each bit rate and the ratio of the bit rate occupied in the file set can be specified. Based on the discrete distribution provided by users, MediSyn generates a sequence of bit rates for the file set and matches the bit rate with the file duration randomly, since we observe that there is no correlation between them in our traces (the correlation coefficient is 0.0144).

4.3 Popularity

Earlier studies [12, 16] found that media file popularity can often be captured by a *Zipf-like* distribution. A Zipf-like distribution states that the access frequency of the i-th most popular file is proportional to $1/i^{\alpha}$. If the frequencies of files and the corresponding popularity ranks are plotted on a log-log scale, a Zipf-like distribution can be fitted by a straight

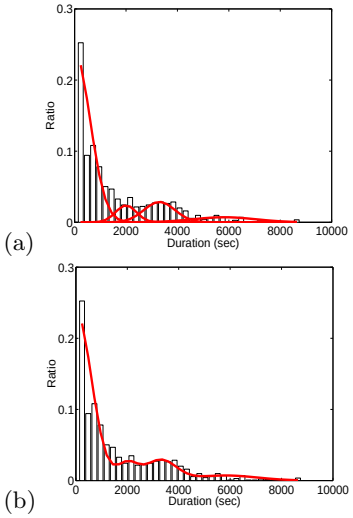

Figure 1: PDF of the HPC duration distribution. (a) 4 *normal* distributions to capture the 4 peaks. (b) The aggregate distribution of the 4 *normal* distributions.

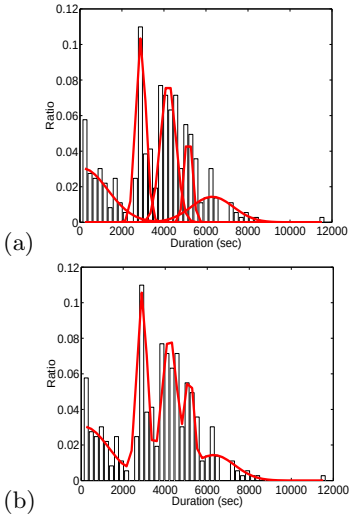

Figure 2: PDF of the HPL duration distribution. (a) 5 *normal* distributions to capture the 5 peaks. (b) The aggregate distribution of the 5 *normal* distributions.

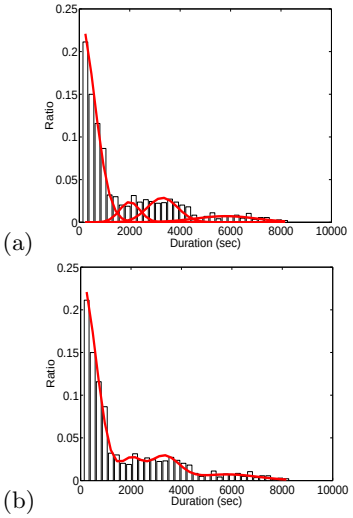

Figure 3: PDF of the MediSyn duration distribution. (a) 4 *normal* distributions to capture the 4 peaks. (b) The aggregate distribution of the 4 *normal* distributions.

line. A larger α implies more sessions are concentrated on the most popular files. Some synthetic workload generators [6, 15] also adopt a Zipf-like distribution in generating file popularity.

However, recent studies [2, 11, 3, 5, 8] observed that for some web and streaming media workloads, a Zipf-like distribution does not accurately capture the file popularity distribution. The popularity distribution of these workloads shows a circular curve on a log-log scale. For instance, Figure 4 shows the file popularity distributions of the HPC and the HPL traces over the entire trace periods on a log-log scale. They exhibit circular curves similar to those studies.

If we use a straight line (a Zipf-like distribution) to fit the circular curve and generate session frequencies based on the value of α obtained by curve fitting, the generated frequencies must be skewed from the original session frequencies. Breslau et al [8] calculated α by excluding the top 100 files. For our traces, not only the beginning but also the end of the curves cannot be fitted by straight lines. Moreover, since the most popular files are especially important for synthetic streaming media workloads, we cannot ignore the first 100 files.

Thus, an important contribution of this work is that a *generalized Zipf-like* distribution is proposed as a unified method to capture file popularity distributions of both Zipf-like and circular-curve shapes. A generalized Zipf-like distribution can be fitted by a straight line on a log-log scale after a Zipf k-transformation. The Zipf k-transformation for file popularity is defined as follows: assume x is a file rank, y is the corresponding access frequency for the file, k_x and k_y are scale parameters, the k-transformation transforms the original x and y as follows:

$$x_k = \frac{x + k_x - 1}{k_x}, \quad y_k = \frac{y + k_y - 1}{k_y} \tag{1}$$

After the k-transformation, x_k and y_k denote the transformed file rank and frequency respectively. Figure 5 shows the relationship between x_k and y_k of the HPC and HPL traces on a log-log scale respectively. It shows perfectly straight lines. The α value of the Zipf k-transformation is derived through linear regression [13]. Table 5 shows some critical parameters related to the curve fitting of the two workloads. As shown in the table, R^2 is 0.995 for both the HPC and HPL traces, indicating that straight lines fit both the distributions very well[2].

Trace	α	R^2	k_x	k_y	Max freq	Files
HPC	1.561	0.995	12	12	17831	1434
HPL	1.23	0.995	7	7	961	364

Table 5: Parameters of Zipf k-transformation of the HPC and the HPL traces.

The reason that the original traces do not show perfectly straight lines at the heads of the curves is that there is little differentiation in the frequencies of the most popular files (with smaller x). It can be attributed to the fact that a long-term trace can collect enough files with similar popularities over time, and thus these files can be considered as a group (equivalence class), where a group rank will be a better reflection of the file popularities. Intuitively, the effect of the k-transformation is that the popularity follows a Zipf-like distribution if we check every group of k_x files. We divide x by k_x to scale the file ranks so that the $((i-1)k_x + 1)$-th rank becomes the i-th rank and reflects the corresponding file group rank. Other file ranks are transformed to float numbers evenly distributed among integral ranks. So we actually move all points on the log-log scale along the x-axis to the left and squeeze the points to a more straight line. Similarly, the reason that the traces do not show perfectly straight lines at the tails of the curves is that there is not enough differentiation in the number of files with the lowest frequencies. So we divide y by k_y to squeeze those points along the y axis

[2]The coefficient of determination R^2 measures the goodness of a regression. If R^2 =1, the model is perfect. A larger R^2 implies a more accurate model [13].

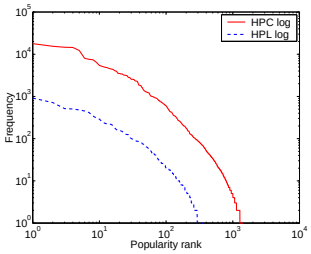

Figure 4: The original popularity distributions of the HPC trace and the HPL trace on a log-log scale.

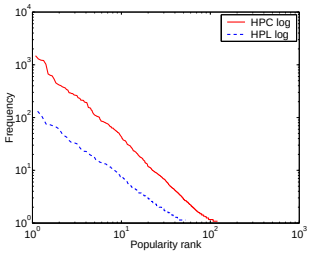

Figure 5: The popularity distributions after Zipf k-transformation on a log-log scale.

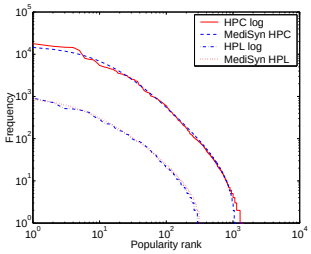

Figure 6: Comparison between the popularity distribution generated by MediSyn and the original traces on a log-log scale.

on the log-log scale. The value of k_y is not necessarily the same as k_x. However, MediSyn uses the same value for k_x and k_y based on our observations for both the HPC and HPL traces, which we simply refer to as k. The scale parameter k of our k-transformation is similar to the scale parameter k of a *generalized Pareto* distribution [1].

Another explanation for the k-transformation is that the original frequency sequence cannot be fitted by a Zipf-like distribution starting from rank 1, but it can be fitted into part of a Zipf-like distribution starting from rank k_x. To describe this file rank starting from k_x by a Zipf-like distribution, we have to divide its original rank by k_x. Similar explanation can be applied for k_y. Clearly, Zipf-like distributions are a special case of generalized Zipf distributions when $k = 1$.

To generate a sequence of frequencies following a generalized Zipf-like distribution, users of MediSyn specify the maximum frequency M for the most popular file, the number of files N, the scale parameter k and the Zipf-like distribution parameter α. MediSyn computes the frequency of the most popular x-th file ($x \in [1, N]$) using the following formula

$$\left(\frac{M_k}{(\frac{x+k-1}{k})^{\alpha}} - 1 \right) k + 1, \tag{2}$$

where $M_k = \frac{M-1}{k} + 1$. Figure 6 compares the frequencies generated by MediSyn with the original frequencies in our traces.

To determine whether there is a correlation between file duration and file popularity, we compute the correlation coefficient between file popularity and file duration for both of our workloads. Table 6 shows these results. We use both the file frequency and the file rank as the popularity metric to compute the correlation coefficient. Overall, we observe no strong correlation between file popularity and file duration. So file duration and file popularity are randomly matched in MediSyn.

Workload	HPC Freq	HPL Freq	HPC Rank	HPL Rank
Correlation Coefficient	-0.03	0.05	-0.20	-0.002

Table 6: Correlation coefficient between file popularity and file duration.

We also check for possible correlation between popularity and encoding bit rate. Once again, there is no correlation between them, so MediSyn matches popularity and encoding bit rate randomly.

4.4 Prefix

One major characteristics of streaming workloads is that a significant amount of clients do not finish playing an entire media file. We refer to the duration between the start of a media session and the time when the session is terminated by the client as the *prefix duration* of the session, or simply the *prefix*. Figure 7 and Figure 8 show the PDF for the prefixes of two typical example files in the HPC trace. The "spikes" in the figures correspond to successfully completed media sessions for the files, while the other prefixes in the figures are incomplete sessions. We observe that there is a strong correlation between the file duration and the prefix distribution:

- *Complete sessions.* The fraction of complete sessions of a file highly depends on the file duration. A short file tends to have more complete sessions. For example, the file durations in Figure 7 and Figure 8 are 723 seconds and 4133 seconds respectively. The file in Figure 7 has more complete sessions than that in Figure 8. We use r_c to denote the ratio of complete sessions for a file compared with the total number of sessions for the file. Figure 9 shows the relationship between file duration and r_c. We can observe that the r_c of each file highly depends on the file duration.

- *Incomplete sessions.* The prefix distribution of incomplete sessions of a file depends on the file duration. Figure 7 reflects that the prefixes of incomplete sessions for a short-duration media file can be captured by an *exponential* distribution. While for a long-duration file as shown in Figure 8, the prefixes of incomplete sessions follows a heavy-tail distribution, which cannot be captured by an *exponential* distribution.

Thus, the overall prefix distribution of a media workload highly depends on each file's prefix distribution, which in turn depends on the duration of the file. There is not a straightforward solution to directly capture the overall prefix distribution for the entire workload. To generate each file's prefix distribution, we first generate r_c for the file, then model the distribution of incomplete sessions for the file based on the assigned r_c.

To generate r_c for a file, we need to determine the relationship between r_c and the file duration as shown in Figure 9. We observe that the contour of the dotted area in Figure 9 follows a Zipf-like distribution. We use a bin-based process to generate r_c values following this contour. The detail can be found in [21].

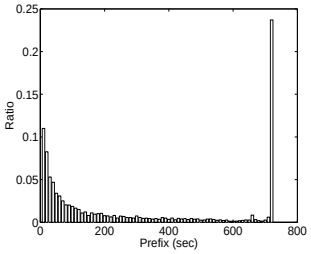

Figure 7: A typical access prefix distribution of short duration media file.

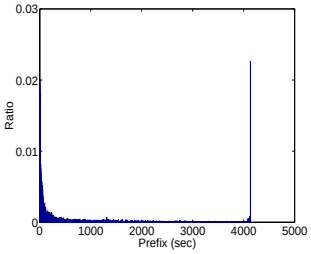

Figure 8: A typical access prefix distribution of long duration media file.

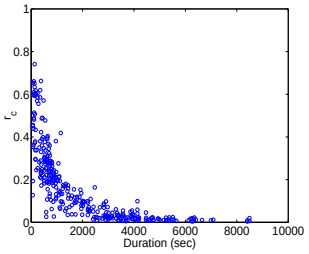

Figure 9: Fraction of complete sessions (r_c's) versus the corresponding media file durations.

After the r_c value of each file is determined, the distribution of incomplete sessions needs to be determined. As mentioned above, depending on the file duration, it could be captured by an *exponential* distribution or some heavy-tail distribution. Additionally, both Figure 7 and Figure 8 show a similar shape in the beginning of the distributions within a certain range of duration. We observe that for more than 90% of the media files in the HPC trace, the distributions of prefixes within the first 5 minutes can be fitted by *exponential* distributions. These results confirm similar findings for an educational workload studied by Almeida et al [3]. Given the fact that prefixes within a certain duration range (e.g., the first 5 minutes) occupy a high percentage of total incomplete sessions, we introduce a cut-off point and use the following method to model the prefix distribution of a given media file:

- If a media file duration *is less than the cut-off point*, its incomplete prefixes are modeled by an *exponential* distribution.
- If a media file duration is *longer than the cut-off point*, the distribution of incomplete prefixes is modeled by the *concatenation of two distributions*. The distribution of incomplete prefixes less than the cut-off point is modeled by an *exponential* distribution. The distribution of the remaining incomplete prefixes longer than the cut-off point is approximated by a *uniform* distribution.

In the HPC trace, the cut-off point is 5 minutes. Users of MediSyn can specify their own cut-off point. The detail of prefix distribution generation can be found in [21]. After generating a sequence of prefixes for each file, MediSyn randomly matches the prefixes with all the sessions of the file.

4.5 New File Introduction Process

The process of new file introduction mimics how files are introduced at a media site and attempts to answer the following questions:

- What is the new file arrival process on a daily time scale?
- What is the new file arrival process within an introduction day?

To model new file arrival on a daily level, we capture the time gap measured in days between two introduction days and the number of new files introduced in each introduction day. Figure 10 shows the distribution of new file introduction gaps measured in days for the HPC trace. The distribution depicted in Figure 10 can be captured by a *Pareto* distribution with $\alpha = 2.0164$. Figure 11 shows the introduction gap distribution for the HPL trace, which can be captured by an *exponential* distribution with $\mu = 4.2705$.

MediSyn can generate new file introduction time gaps according to one of three possible distributions: 1) a *Pareto* distribution, 2) an *exponential* distribution, 3) a *fixed interval*. If users specify a *Pareto* distribution for the new file introduction process, files tend to be introduced into the system clustered over time. If the introduction process is specified by an *exponential* distribution, the new file arrival process is a *Poisson* arrival process, which means the interarrival times are independent. The fixed interval is used to model some artificial introduction process with regular patterns.

Since there may be multiple new files introduced in a given day, we must also model the number of files introduced per introduction day. Figure 12 shows the distribution for the number of files introduced in a given day for the HPC trace. The distribution can be fitted by a *Pareto* distribution with $\alpha = 1.1323$.

After determining the number of files introduced in a given day, MediSyn needs to model the new file arrival process within that day. We model this process by capturing the gap between two file arrivals. Figure 13 shows the time gaps for new files introduced within a day. Since the distribution is too sparse on time scale of seconds, we measure the time gaps at multiples of 900 seconds (15 minutes). The distribution can be captured by a *Pareto* distribution with $\alpha = 1.0073$.

Due to the properties of *Pareto* distribution, if we only model the time gap between two file arrivals and start to introduce new files from the beginning of a day, then most of the files will be introduced in the beginning of every day. So we also capture the start times of new file introduction process within every introduction day. Figure 14 shows this distribution. Since it looks like a rotated *normal* distribution with the peak at 0, we capture this by a *normal* distribution as shown in Figure 15 after rotating the original distribution by 12 hours. The mean of the rotated *normal* distribution is 43200 seconds and the standard deviation is 21600 seconds.

4.6 Life Span

Since temporal correlation is observed in media workloads, an independent reference model combined with a popularity distribution [8] is insufficient for a synthetic workload generator to generate a file access stream. SURGE [6] uses a stack distance model to generate web reference streams with reference locality. Both the independent reference model [8] and the stack distance model [6, 4] assume that each file's popularity is stationary over the entire trace period and that each file is introduced at the start of the trace. Since we observe non-stationary popularity in streaming media workloads, such models are unsuitable for generating session arrivals in streaming media workloads.

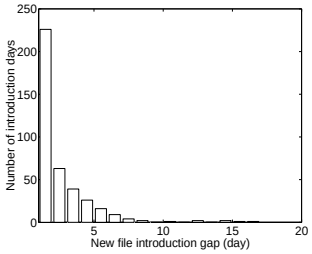

Figure 10: New file introduction gaps measured in days for the HPC trace.

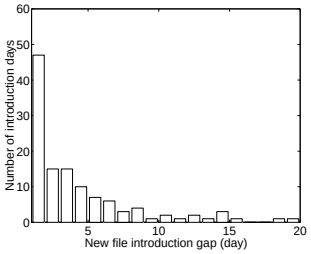

Figure 11: New file introduction gaps measured in days for the HPL trace.

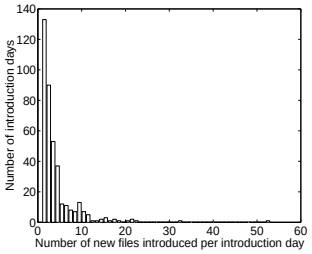

Figure 12: The number of new files introduced per introduction day.

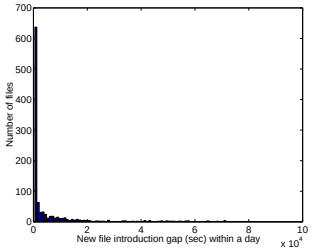

Figure 13: New file introduction time gaps within an introduction day.

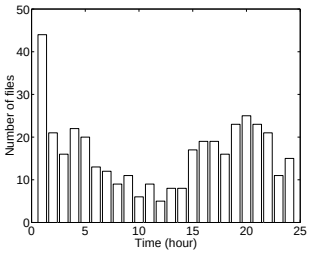

Figure 14: The start times of new file introduction within introduction days.

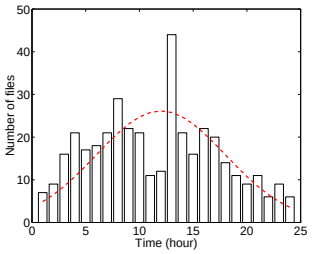

Figure 15: The rotated start times of new file introduction within introduction days.

To accurately model the non-stationarity of file popularity, we use the new file introduction process to mimic how media files are introduced at media sites as we described above. In addition, each file has its own life span, which characterizes its changing popularity after the file's introduction. Thus, the file popularity distribution, the file introduction process and life spans of individual files, all together capture the popularity change of media files over the entire trace.

We define the *relative access time* of a file as a random variable whose value is the time measured in days when the file is accessed by a client after the file is introduced. The distribution of a file's *relative access times* describes the temporal correlation of all accesses to the file. We also call this distribution the life span distribution of the file. In our traces, we observe two types of life span distributions as illustrated in Figure 16 and Figure 17 respectively. Since most files in our traces have life spans similar to Figure 16, we call this type of life span a *regular life span*.

News-like streaming contents typically have life span distributions similar to Figure 17, where most accesses occur shortly after the file introduction and the access frequency diminishes relatively quickly. So we refer to this kind of life span as a *news-like life span*.

We experimented with *gamma*, *Pareto*, *exponential* and *lognormal* distributions to fit the *relative access times* of our traces. Although *gamma* distributions can somehow capture both news-like and regular life spans, the combination of *Pareto* and *lognormal* distributions can better fit them. Thus, news-like life spans follow *Pareto* distributions, and regular life spans follow *lognormal* distributions.

To generate a sequence of regular life spans and news-like life spans, we need to model the distributions of the mean (μ) and the standard deviation (σ) for regular life spans, and the distributions of α for news-like life spans. Our analysis of the HPC and HPL traces shows that these parameters follow *normal* distributions. Table 7 shows the parameters for these *normal* distributions derived from the HPC log to capture the parameters of regular life spans (μ and σ) and news-like life spans (α).

Normal dist. paramaters	**lognormal** μ	**lognormal** σ	**Pareto** α
μ	3.0935	1.1417	0.7023
σ	0.9612	0.3067	0.2092

Table 7: The parameters for the distributions (*normal* distributions) of the parameters in *lognormal* and *Pareto* life span distributions.

There is a strong correlation between file popularity and life span shape. A file with a higher popularity rank tends to have a higher probability to have a *news-like life span*. Figure 18 shows the PDF for this probability. The distribution can be captured by an *exponential* distribution. File ranks have been transformed between 0 and 1 so that μ for the *exponential* distribution is independent of the number of media files generated. In the HPC trace, we observed 82 *news-like life spans* out of the 400 most popular files. Users of MediSyn can specify their own ratio according to the workload they want to generate. A workload including more files with *news-like life spans* has a more bursty access pattern.

4.7 Diurnal Access Pattern

After determining the life span and the popularity of every file, MediSyn can generate the number of accesses for every day of a file's life span. Distributing these accesses over a day is challenging because we want to model both session interarrival time and diurnal access patterns.

Figure 19 shows a typical session interarrival time distribution for a file measured in a day. It is a heavy-tail distribution and can be fitted by a *Pareto* distribution better than

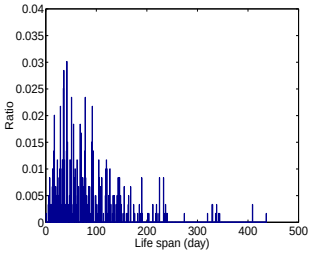

Figure 16: A *regular* lifespan.

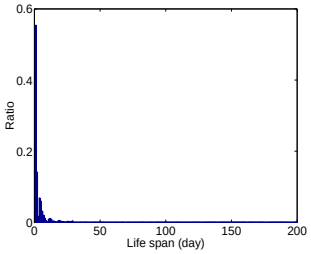

Figure 17: A *news-like* lifespan.

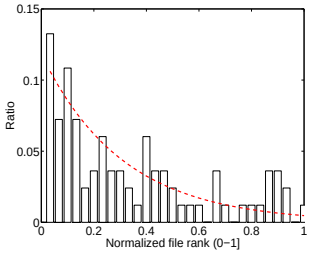

Figure 18: PDF that a file at a certain rank has a *Pareto* life span distribution.

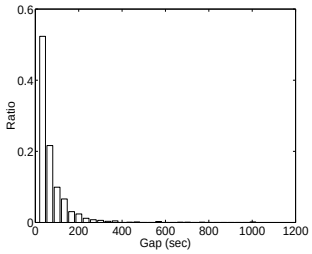

Figure 19: The PDF of session access interarrival time gaps for a file measured in a day.

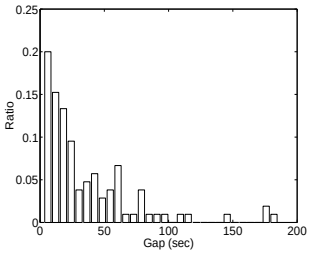

Figure 20: The PDF of session access interarrival time gaps for a file measured in an hour.

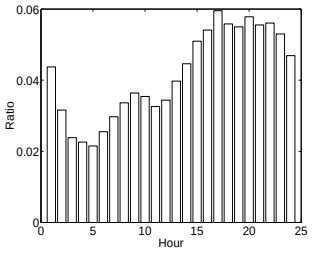

Figure 21: The session access diurnal pattern for the HPC trace. Each bin is an hour.

an *exponential* distribution. However, if we generate all interarrival times within a day based on this *Pareto* distribution, it is difficult to simultaneously ensure diurnal pattern. Figure 20 shows the interarrival time distribution for the same file within one hour of the same day. This distribution is not a heavy-tail distribution and can be captured by an *exponential* distribution. Thus, if we can determine the number of accesses in each hour of a day according to a certain diurnal pattern, we can use an *exponential* distribution to generate the interarrival times of the accesses in this hour. Thus, we can both generate the diurnal pattern and satisfy the observed *exponential* distribution for interarrival times.

Diurnal access patterns are universally observed by other streaming workload analyses. But we do not explicitly find diurnal patterns for single files. We only observe an aggregate diurnal access pattern for all file accesses. Figure 21 shows the average ratios of accesses in each hour for all files in the HPC trace.

In MediSyn, a user can specify a global diurnal pattern like Figure 21, which contains a set of bins. Each bin specifies a time period and the ratio of accesses in this bin. Since we believe there is no temporal correlation among file accesses within a day (i.e., the temporal locality within a day is entirely determined by file popularities), we can make every file follow the diurnal pattern. Essentially, each file's session arrival process within a given day is modeled as a *nonhomogeneous Poisson* process [19], where only the session arrivals within each bin can be modeled by a *Poisson* process. The session arrival rate of the file for a given bin is computed based on the diurnal pattern specified by the user and the number of accesses within a day determined by the file life span. MediSyn generates the interarrival time gaps within each bin and constructs a sequence of sessions for the file on the scale of seconds.

5. RELATED WORK

Accurate workload characterization lays down a foundation for a successful synthesis of realistic workloads. A number of studies on multimedia workload analysis have been reported in literature [2, 3, 12, 11, 16, 18].

Acharya et al [2], presented the analysis of the six-month trace data from mMOD system (the multicast Media on Demand) which had a mix of educational and entertainment videos. They observed high temporal locality of accesses, the special client browsing pattern showing clients preference to preview the initial portion of the videos, and that rankings of video titles by popularity do not fit a Zipfian distribution.

Almeida et al [3] performed an analysis of two educational media server workloads. The authors provide a detailed study of client session arrival process: the client session arrivals in one workload can be characterized as a *Poisson* process, and the interarrival times in the second workload follow a heavy-tail *Pareto* distribution. They also observed that media delivered per session depends on the media file length.

The study by Chesire et al [12] analyzed the media proxy workload at a large university. The authors presented a detailed characterization of session duration (most of the media streams are less than 10 minutes), object popularity (78% of objects are accessed only once), sharing patterns of streaming media among the clients, and that popularity distribution follows a Zipf-like distribution (trace duration covers one week).

Two enterprise media server workloads have been extensively studied in [11]. The data was collected over significant period of time. Thus authors concentrated on the analysis of media server access trends, access locality, dynamics and evolution of the media workload over time. They reported non-Zipfian and non-stationary popularity of files observed in their data.

In our work, we attempt to summarize findings from the earlier work, and build a general, unified model for workload

characteristics capturing unique properties of streaming media workloads as well as the dynamics in media workloads observed over long period of time.

The only synthetic workload generator for streaming media reported in literature is GISMO [15]. MediSyn adopts similar approach chosen in GISMO to organize the synthetic trace generation in two steps: *i)* defining the individual session characteristics, and *ii)* determining the media session arrival process. GISMO operates over a "fixed" set of media files already "introduced" at a media site, with the assumption that object popularity follows a Zipf-like distribution and remains the same over the entire duration of the experiment. Since we pursue the goal of developing a synthetic workload generator which reflects the dynamics and evolution of media workloads over time, we propose a set of new models to reflect these new temporal properties of streaming media workloads in MediSyn.

6. CONCLUSION AND FUTURE WORK

Development of efficient resource allocation mechanisms for streaming media hosting centers and CDNs requires scalable, configurable, and realistic streaming media workloads. In this work, we present a synthetic streaming media workload generator, MediSyn, which is specially designed for accomplishing this goal. In MediSyn, we develop a number of novel models to capture a set of characteristics critical to streaming media services, including file duration, file access prefix, file popularity, new file introduction process, and diurnal access pattern. Among the primary features of MediSyn is the ability to reflect the dynamics and evolution of content at media sites and the change of access rate to the content over time. We introduce a *generalized* Zipf-like distribution to capture the recently-observed skewed popularity of both web objects and streaming media that cannot be captured by existing Zipf-like distributions. Based on two long-term traces of streaming media services, our evaluation demonstrates that MediSyn can accurately capture the essential properties of the media workloads.

MediSyn implementation is based on a modular design allowing particular system properties to be customized, enhanced or extended to reflect the requirements of various scenarios. In the future, we plan to enable MediSyn to support client interactivities within media sessions.

7. REFERENCES

[1] General Pareto Distribution. `http://www.math.uah.edu/stat/special/special12.html`.

[2] S. Acharya, B. Smith, and P. Parnes. Characterizing User Access to Videos on the World Wide Web. In *Proceedings of ACM/SPIE Multimedia Computing and Networking*, January 2000.

[3] J. Almeida, J. Krueger, D. Eager, and M. Vernon. Analysis of Educational Media Server Workloads. In *Proceedings of NOSSDAV*, June 2001.

[4] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira. Characterizing Reference Locality in the WWW. In *Proceedings of PDIS*, December 1996.

[5] V. Almeida, M. Cesirio, R. Fonseca, W. Meira Jr., and C. Murta. Analyzing the behavior of a proxy server in the light of regional and cultural issues. In *Proceedings of WCW*, June 1998.

[6] P. Barford and M. Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Proceedings of SIGMETRICS*, June 1998.

[7] R. Braynard, D. Kostić, A. Rodriguez, J. Chase, and A. Vahdat. Opus: an Overlay Peer Utility Service. In *Proceedings of OPENARCH*, June 2002.

[8] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence, and Implications. In *Proceedings of INFOCOM*, March 1999.

[9] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle. Managing Energy and Server Resources in Hosting Centers. In *Proceedings of the 18th ACM SOSP*, October 2001.

[10] L. Cherkasova and G. Ciardo. Characterizing Temporal Locality and its Impact on Web Server Performance. In *Proceedings of ICCCN*, October 2000.

[11] L. Cherkasova and M. Gupta. Characterizing Locality, Evolution, and Life Span of Accesses in Enterprise Media Server Workloads. In *Proceedings of NOSSDAV*, May 2002.

[12] M. Chesire, A. Wolman, G. Voelker, and H. Levy. Measurement and Analysis of a Streaming-Media Workload. In *Proceedings of USITS*, March 2001.

[13] R. Jain. *The art of computer systems performance analysis: technique for experimental design,measurement,simulation and modeling*. John Wiley & Sons, 1992.

[14] S. Jin and A. Bestavros. Temporal Locality in Web Requests Streams: Sources, Characteristics, and Caching Implications. Technical Report BUCS-TR-1999-009, Department of Computer Science, Boston University, August 1999.

[15] S. Jin and A. Bestavros. GISMO: A Generator of Internet Streaming Media Objects and Workloads. Technical Report BUCS-TR-2001-020, Department of Computer Science, Boston University, October 2001.

[16] D. Luperello, S. Mukherjee, and S. Paul. Streaming Media Traffic: an Empirical Study. In *Proceedings of WCW*, June 2002.

[17] Hewlett Packard. Utility Data Center. `http://www.hp.com/go/hpudc`.

[18] J. Padhye and J. Kurose. An Empirical Study of Client Interactions with a Continuous-Media Courseware Server. In *Proceedings of NOSSDAV*, June 1998.

[19] S. Ross. *Introduction to probability models*. Academic Press, 1997.

[20] S. Sen, J. Rexford, and D. Towsley. Proxy Prefix Caching for Multimedia Streams. In *Proceedings of INFOCOM*, March 1999.

[21] W. Tang, Y. Fu, L. Cherkasova , and A. Vahdat. Long-term Streaming Media Server Workload Analysis and Modeling. HP Laboratories, Technical Report HPL-2003-23, February 2003.