Available online at www.sciencedirect.com

ScienceDirect

Computer Networks 51 (2007) 336–356

Computer Networks

www.elsevier.com/locate/comnet

Modeling and generating realistic streaming media server workloads

Wenting Tang [a,1], Yun Fu [b,2], Ludmila Cherkasova [a,*], Amin Vahdat [b]

[a] *Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94303, United States*
[b] *Department of Computer Science and Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093, United States*

Received 28 May 2004; received in revised form 27 February 2006; accepted 2 May 2006
Available online 14 June 2006

Responsible Editor: U. Krieger

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 propose a framework for modeling *long-term* behavior of network services by capturing the process of file introduction, non-stationary popularity of media accesses, file duration, encoding bit rate, and session duration. (ii) We propose a variety of practical models based on the study of the two workloads. (iii) We develop an open-source synthetic streaming service workload generator to demonstrate the capability of our framework to capture the models.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Streaming media server workload; Synthetic workload generator; Media access patterns; Temporal and static properties; Non-stationary popularity; Zipf–Mandelbrot law; File life span; Modeling

[*] Corresponding author. Tel.: +1 650 857 3753; fax: +1 650 857 7029.
E-mail addresses: wenting.tang@hp.com, wtang@arcsight.com (W. Tang), fu@cs.ucsd.edu, yfu@yahoo-inc.com (Y. Fu), lucy.cherkasova@hp.com (L. Cherkasova), vahdat@cs.ucsd.edu (A. Vahdat).
[1] This work was originated and largely completed while W. Tang worked at HPLabs. Currently, W. Tang is with Arcsight Inc. 5 Result Way, Cupertino, CA, United States.
[2] This work was mostly done while Y. Fu worked at HPLabs during his summer internship. Currently, Y. Fu is with Yahoo! Inc. 2821 Mission College Blvd., Santa Clara, CA 95054, United States.

1389-1286/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2006.05.003

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 [10] 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. Consider the example of a news service when an important event takes place; load on CNN reportedly doubled every seven minutes shortly after 9 AM on September 11, 2001 [7].

Thus, we are pursuing a vision where large-scale hosting infrastructures simultaneously provide "resource-on-demand" capabilities to competing Internet services [19,8]. 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. For instance, surplus resources resulting from "troughs" in accesses to one service may be reallocated to satisfy the requirements of a second service experiencing a peak. Further, Service Level Agreements (SLAs) may specify that, under resource constraints, one service should preferentially receive resources over other services.

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 [13,18]. 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. For MediSyn, we develop a number of novel models to capture a broad range of characteristics 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 simultaneous competing services (consider a CDN simultaneously hosting hundreds of individual services).
- It was observed [2,12] 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 showed that a special version of Zipf–Mandelbrot law can be used to capture the popularity distribution in such workloads. The traditional Zipf-like distribution is a special case of the Zipf–Mandelbrot 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 presents the workload generation process adopted by MediSyn.

Section 3 outlines the real-world workloads used tin our study and 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 and their generation in MediSyn

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 mathematical 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 how these properties are composed together during workload generation process in MediSyn.

We partition media workload properties in **two groups**: *static* and *temporal* properties.

- **Static properties** provide the characteristics of the underlying media fileset, reflect the aggregate, quantitative properties of client accesses (independent of the access time), and present the properties of individual file accesses. Static properties include:
  - *file duration* that represents the advertised duration of the file (in seconds),
  - *file encoding bit rate* that reflects the rate (in bits/s) used for file encoding and that defines bandwidth requirements for the file transfer,
  - *file access popularity* that defines the number of accesses to a file within a certain period of time,
  - *file access prefix* that represents the elapsed time of the requested media file when the play ended (a play is ended prematurely when the client hits the stop button).
- **Temporal properties** reflect the dynamics and evolution of accesses to media content over time, and determine the ordering and the timing of session arrivals. The temporal properties of media workloads include:
  - *new file introduction process* that reflects at what rate the new content is introduced at the media site (and hence, when it appears in media workload),
  - *file life span* that defines the file popularity changes over a daily time scale within a certain period of time,
  - *diurnal access pattern* that specify how the number of accesses to a site varies during a given period of time, e.g., a day.

MediSyn's goal is to generate a synthetic trace representing a sequence of file accesses to media service. This process consists of generating values/distributions for all the properties introduced above for each media file.

Once all the file's properties are generated, MediSyn generates a sequence of accesses to each file accordingly to the assigned popularity distributions and file temporal properties. At the end, all the media sessions for all the files are combined and sorted according to a global time and merged together to generate the synthetic trace.

3. 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.

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 mathematical models introduced to reflect these properties. The streaming media server logs represent two different media services: *HP Cor-*

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

	HPC	HPL
Log duration	29 months	21 months
Number of files	2999	412
Number of sessions	666,074	14,489

porate Media Solutions Server (HPC) and *HPLabs Media Server (HPL)*. We define a *session* as a client access to a particular file. Table 1 briefly summarizes the workloads.

In Table 1, the HPC media server shows more activities than the HPLabs server. While the HPLabs server serves a small number of research-oriented communities within HP, the HPC workload represents a reasonably busy media server with 300–800 client sessions everyday with peak rate at 12,000 sessions per day. Given this difference, it becomes even more interesting whether we can design common models that are capable of generating these diverse enterprise media workloads.

In the paper, we chosed to use a visualization help for presentating our models and results. Most of the figures are used to visually characterize the nature of studied workloads and help in understanding the main workload properties and their specifics. We used MatLab for fitting our data with distribution candidates. In particular, we used χ^2 (chi-squared) and λ^2 for goodness-of-fit measurement based on [22,21].

3.1. Static properties

3.1.1. Duration

Prior studies [12,3] 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 min, 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, the same 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 [15], the same distribution fails to capture the file duration distribution of the HPL trace.

As shown in Figs. 1(a) and 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 a particular type 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 Figs. 1(a) and 2(a). Here, each group is modeled by a normal distribution with the mean (μ) of each distribution 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.

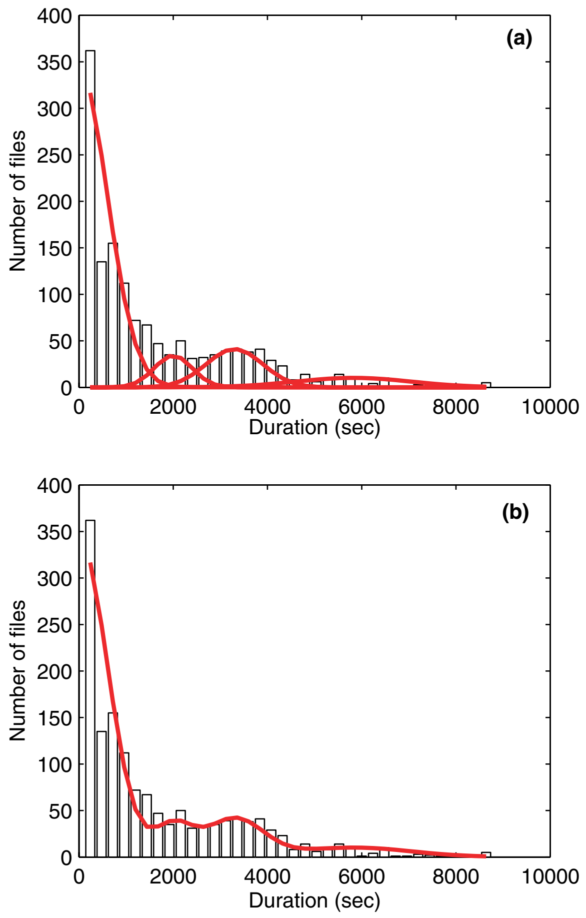


Fig. 1. PDF of the HPC duration distribution: (a) four normal distributions to capture the four peaks and (b) the aggregate distribution of the four normal distributions.

Note that we do not use segmented probability density functions (PDFs) 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

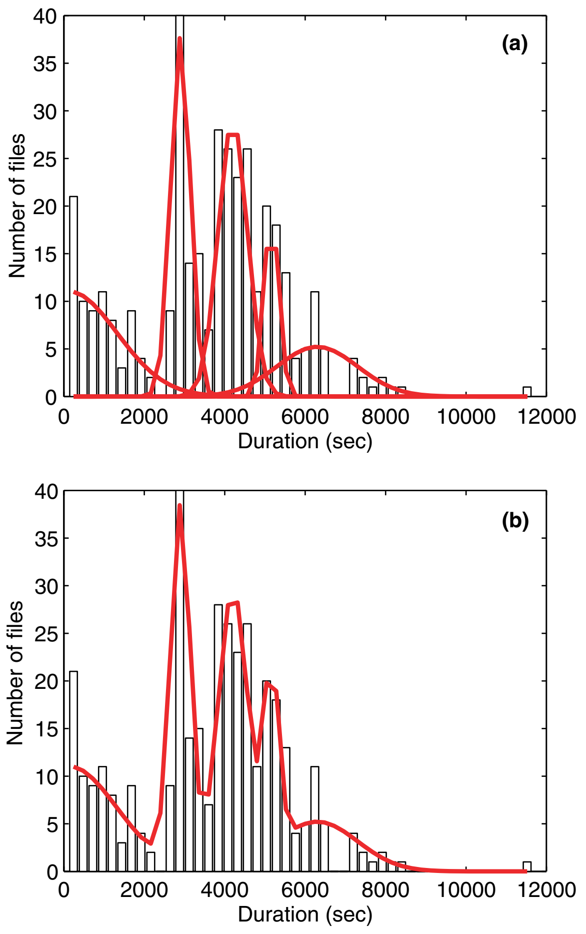


Fig. 2. PDF of the HPL duration distribution: (a) five normal distributions to capture the five peaks and (b) the aggregate distribution of the five normal distributions.

Table 2
Parameters of the normal distributions for the HPC trace

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 HPL 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%

the first group in Table 2 is 0, only half of the normal distribution is used. Tables 2 and 3 present the

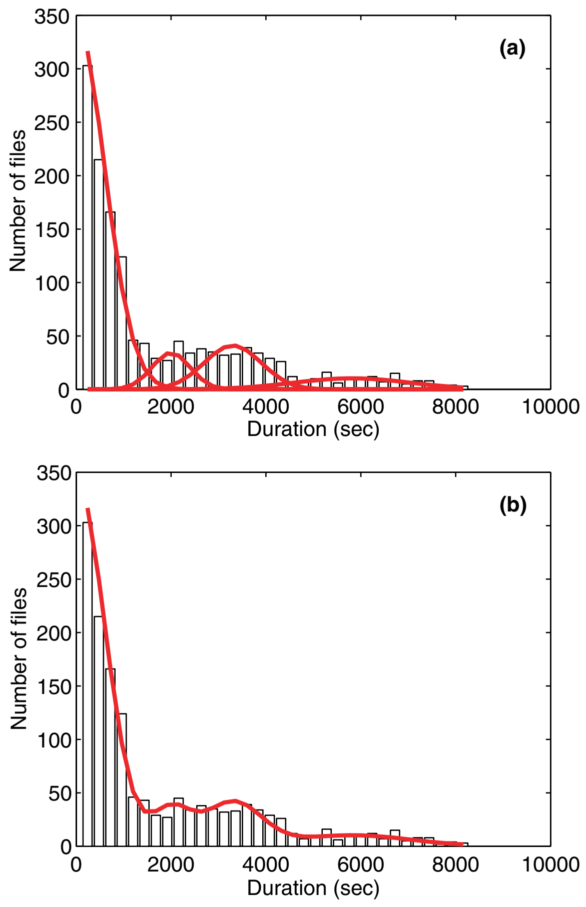


Fig. 3. PDF of the MediSyn duration distribution: (a) Four normal distributions to capture the four peaks and (b) the aggregate distribution of the four normal distributions.

mean (μ), the standard deviation (σ) and the ratio of each normal distribution for the HPC and HPL trace 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 [15] to generate the duration sequence according to the parameterized normal distribution. Fig. 3(a) and (b) shows the durations generated by MediSyn to simulate the HPC workload based on the parameters presented in Tables 2 and 3.

We use λ^2 discrepancy measure introduced by Vern Paxson [26] to compare the duration set generated by MediSyn with the original data set. The λ^2 value is 0.1140.

3.1.2. 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 fileset.

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 fileset can be specified. Based on the discrete distribution provided by users, MediSyn generates a sequence of bit rates for the fileset 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).

3.1.3. Popularity

Earlier studies [13,18] 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 ith 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 line. A larger α implies more sessions are concentrated on the most popular files. Some synthetic workload generators [6,17] also adopt a Zipf-like distribution in generating file popularity.

However, several studies [2,12,3,5,9] analyzing the properties of workloads collected over significant periods of time 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.

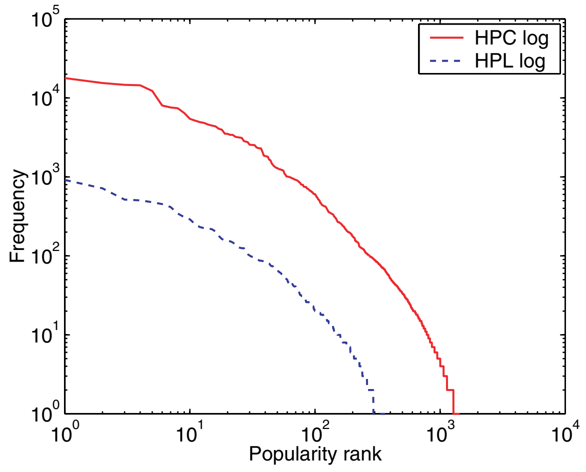


Fig. 4. The original popularity distributions of the HPC trace and the HPL trace on a log–log scale.

For reference, Fig. 4 shows the file popularity distributions of the HPC and the HPL traces over the entire trace periods on a log–log scale. They are more like circular curves similar to those distributions noticed in previous web studies [5,9] and media workload studies [2,12].

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. [9] 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.

- **Zipf–Mandelbrot law**
  Zipf–Mandelbrot law [25] is a discrete probability distribution, which is a generalized Zipf distribution. The law can be described as
  $$f(x) = \frac{C}{(x+k)^{\alpha}}, \tag{1}$$
  where x is the file popularity rank, k is a constant, C is a normalization constant, α is the same as the parameter of Zipf distribution. $C = \sum_{i=1}^{N} 1/(i+k)^{\alpha}$. We observe that circular curves of file popularity can be captured by Zipf–Mandelbrot law.
- ***k*-transformation**
  To facilitate users to select popularity distributions in our workload generator, we provide a simple transformation (k-transformation) that can assist the parameter fitting of Zipf–Mandelbrot law and intuitively illustrate the meaning of the parameters. The k-transformation is defined as follows: given x as a file rank, y as the corresponding access frequency for the file, the following k-transformation can transform x and y to a Zipf-like distribution between x_k and y_k with the same α,
  $$x_k = \frac{x + k_x - 1}{k_x} \tag{2}$$
  $$y_k = \frac{y + k_y - 1}{k_y} \tag{3}$$
  where k_x and k_y are scale parameters. Since $y_k = C_k/x_k^{\alpha}$ (C_k is the normalization constant),

$$(y + k_y - 1)/k_y = \frac{C_k}{((x + k_x - 1)/k_x)^{\alpha}}, \tag{4}$$

$$y = \frac{C_k k_x^{\alpha} k_y}{(x + k_x - 1)^{\alpha}} + 1 - k_y. \tag{5}$$

Eq. (5) is a centered Zipf–Mandelbrot distribution. The parameters C and α of the Zipf–Mandelbrot distribution on x and y can be described as $C = C_k {k_x}^{\alpha} k_y$, $k = k_x - 1$. We also introduce a constant $a = 1 - k_y$.

Fig. 5 shows the relationship between x_k and y_k of the HPC and HPL traces on a log–log scale respectively. We observe that they are perfectly straight lines. The α value of the Zipf k-transformation is derived through linear regression [14]. Table 4 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.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 ith rank and reflect now the corresponding file group rank. 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 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 *general* Pareto distribution [1].

An 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.

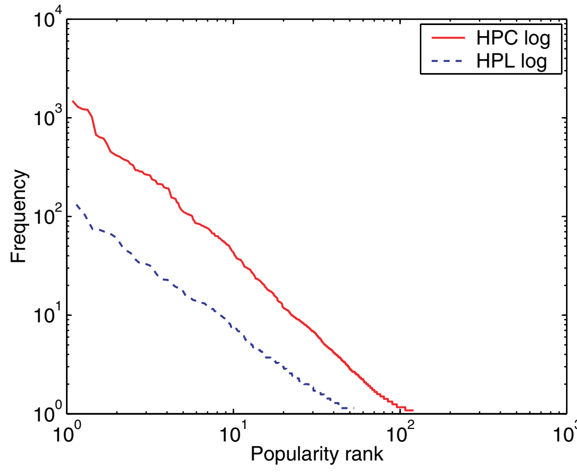


Fig. 5. The popularity distributions after Zipf k-transformation on a log–log scale.

Table 4
Parameters of Zipf k-transformation of the HPC and the HPL traces

Trace	α	R^2	k_x	k_y	Maximum frequency	Number of files
HPC	1.561	0.995	12	12	17831	1434
HPL	1.23	0.995	7	7	961	364

For popularity generation, instead of specifying the total number of requests, we choose to follow traditional load generator style to ask users to specify the maximum access frequence of the most popular file and then generate other file's popularity based on Zipf–Mandelbrot law. To generate a sequence of frequencies, users of MediSyn only need to 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 xth file ($x \in [1,N]$) using the following formula:

$$\left(\frac{M_k}{\left(\frac{x+k-1}{k}\right)^{\alpha}} - 1\right)k + 1, \tag{6}$$

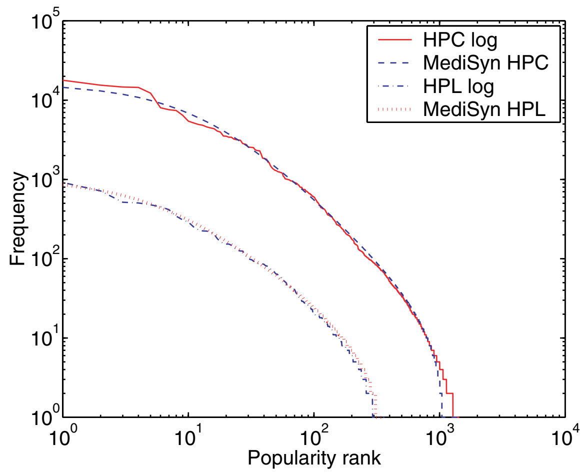


Fig. 6. Comparison between the popularity distribution generated by MediSyn and the original traces on a log–log scale.

Table 5
Correlation coefficient between file popularity and file duration

Workload	HPC frequency	HPL frequency	HPC rank	HPL rank
Correlation coefficient	−0.03	0.05	−0.20	−0.002

where $M_k = \frac{M-1}{k} + 1$. Fig. 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 5 shows these results. We use both the file frequency and the file rank as the popularity metric to compute the correlation coefficient.

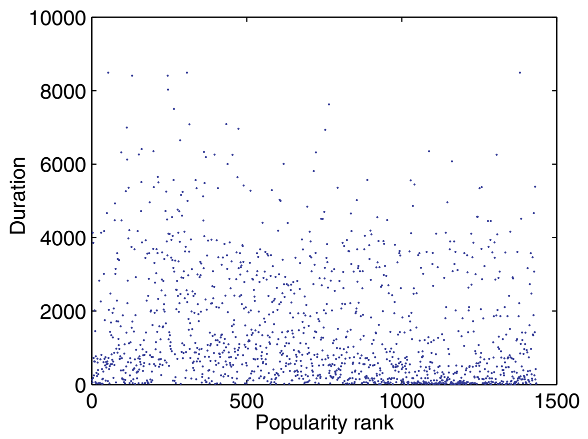


Fig. 7. Popularity rank vs. duration for the HPC trace.

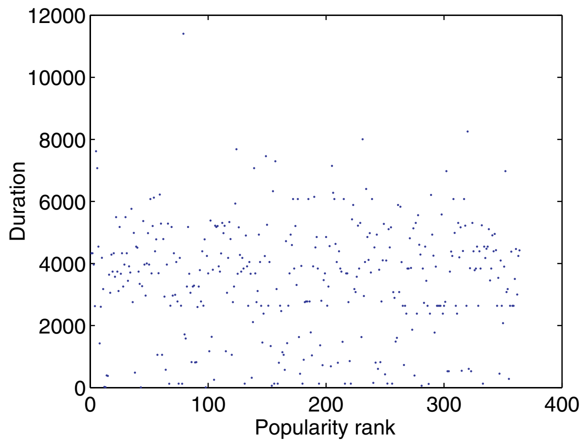


Fig. 8. Popularity rank vs. duration for the HPL trace.

Figs. 7 and 8 show the relationship between popularity and file duration for the HPC and HPL traces.

Overall, we observe no strong correlation between file popularity and file duration. So file duration and file popularity are randomly matched in MediSyn.

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.

3.1.4. *Prefix*

One major characteristics of streaming workloads is that a significant amount of clients do not finish playing an entire media file [12,3]. Typically, this reflects the browsing nature of client accesses, client time constraints, or QoS-related issues. Most incomplete sessions (i.e. terminated by clients before the video is finished entirely) 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 in the HPC (HPL) trace last less than 2 min. This high percentage of incomplete accesses as well as a high number of sessions accessing only the initial part of the file create a very special resource usage model, which is widely considered for streaming media cache design [24].

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*. Figs. 9 and 10 show the histogram for the prefixes of two typical example files in the HPC trace. The “spikes” in the

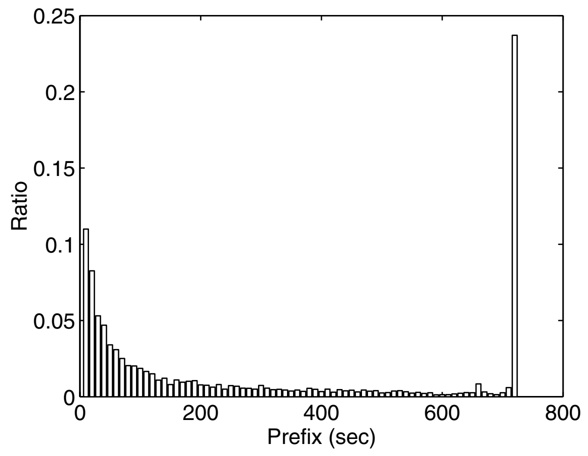


Fig. 9. A typical access prefix distribution of short duration media file.

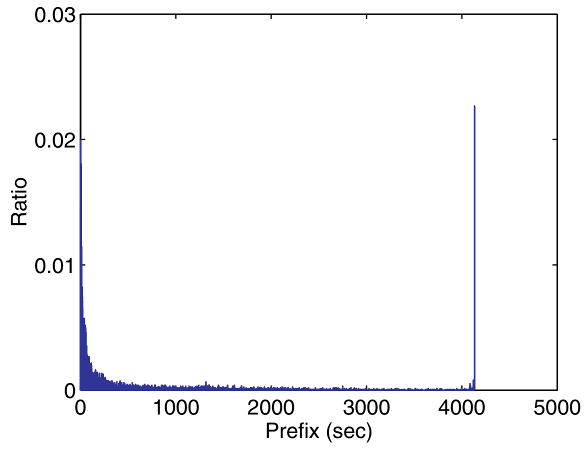


Fig. 10. A typical access prefix distribution of long duration media file.

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 Figs. 9 and 10 are 723 s and 4133 s respectively. The file in Fig. 9 has more complete sessions than that in Fig. 10. We use r_c to denote the ratio of complete sessions for a file compared with the total number of sessions for the file. Fig. 11 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. Fig. 9 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 Fig. 10, the prefixes of incomplete sessions 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 Fig. 11. We observe that the contour of the dotted area in Fig. 11 follows a Zipf-like distribution. To obtain this curve, we segment the durations of all files into 1-min bins. The maximum r_c value of each bin (denoted as r_c^{max}) constitutes the contour of the dotted area in Fig. 11. Fig. 12 shows the relationship between r_c^{max} of each bin and the duration (in minutes) for the corresponding bin on a log–log scale. Because the maximum value of r_c^{max} is 0.74, the curve is flat in the beginning. The other points can be fitted with a straight line. Thus, we can use a Zipf-like distribution to capture the

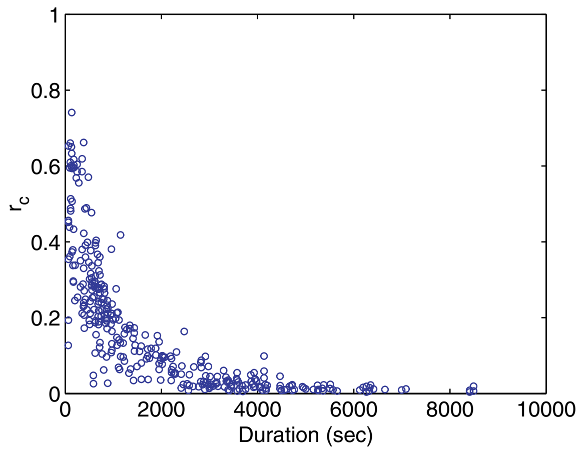


Fig. 11. Fraction of complete sessions (r_c's) versus the corresponding media file durations.

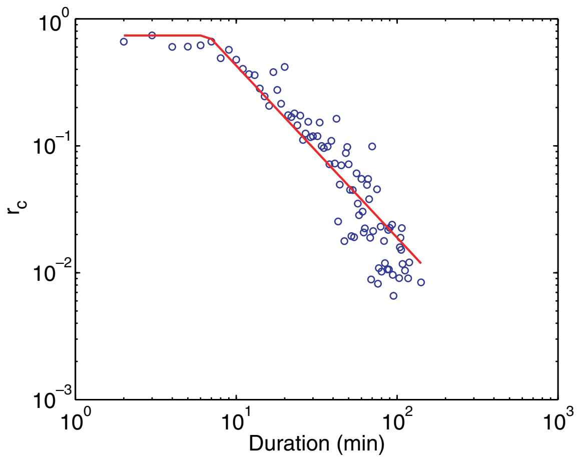


Fig. 12. r_c^{max} of all bins versus the corresponding bin duration on a log–log scale.

distribution of r_c^{max} for all bins. We also observe that the r_c values for other files within a bin, are uniformly distributed between 0 and the r_c^{max} value of the bin. So, to generate r_c for each file, MediSyn first classifies files into 1-min bins based on their durations. Then, MediSyn generates r_c^{max} for each bin. For files in each bin, their r_c values are chosen according to a uniform distribution between 0 and r_c^{max} of the bin. Through this process, the r_c of each file can be determined.

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 a mix of an exponential and a uniform distribution. Additionally, both Figs. 9 and 10 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 min 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 min) 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 min. Users of MediSyn can specify their own cut-off point. We use the following denotations:

- r_e defines the ratio of incomplete sessions whose prefixes are within the cut-off point compared with the total number of sessions of the file,
- r_u defines the ratio of incomplete sessions whose prefixes are longer than the cut-off point compared with the total number of sessions of the file. If a file duration is less than the cut-off point, r_u is 0.

Given r_c for a file, MediSyn needs to generate the values of r_e and r_u for the file. The strategy is to generate r_e for the file and to set $r_u = 1 - r_c - r_e$. Fig. 13 shows the relationship between r_c and r_e for all files in the HPC trace. We can see that for a given r_c, the value of r_e is bounded on both the lower and upper sides. If we denote the maximum of all r_c values as R_c^{max} (0.74 for the HPC trace), the upper bound r_e^{upper} and the lower bound r_e^{lower} of a given r_c can be computed by Eqs. (7) and (8) respectively.

$$r_e^{upper} = 1 - r_c \tag{7}$$

$$r_e^{lower} = r_e^{upper} * (0.6 + 0.4 * r_c/R_c^{max}) \tag{8}$$

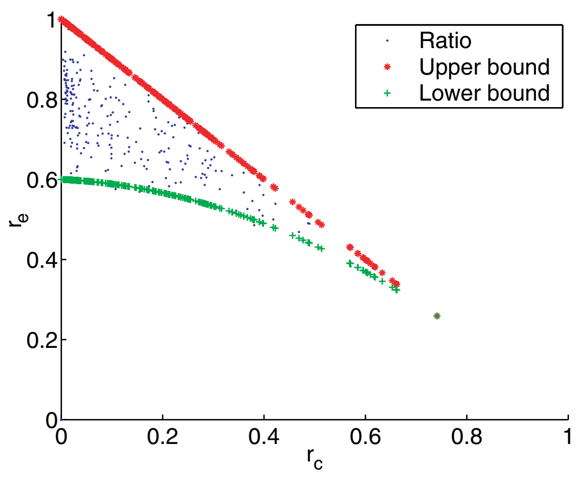


Fig. 13. r_e versus r_c.

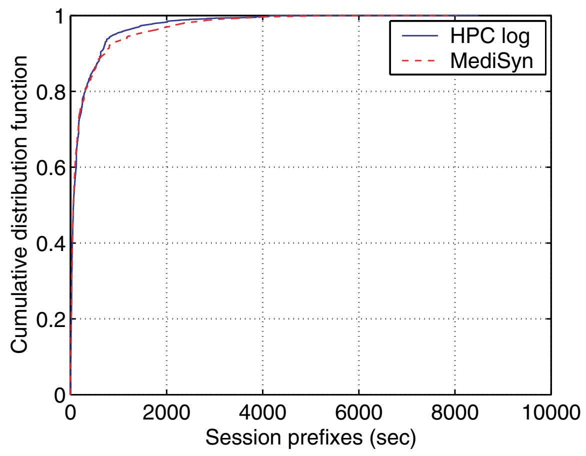


Fig. 14. CDFs of the prefixes generated by MediSyn and the HPC trace.

Fig. 13 plots these calculated upper and lower bounds. The upper bound is caused by the limitation that the sum of r_c, r_e, and r_u is 1. The lower bound changes from 60% of the upper bound to the same as the upper bound while r_c increases from 0 to R_c^{max}. So during workload generation, for a given r_c, MediSyn generates the value of r_e according to a uniform distribution between the corresponding upper bound r_c^{upper} and the lower bound r_c^{lower}.

After generating r_c, r_e and r_u for each media file, MediSyn still needs to generate the mean (μ) of the exponential distribution for the incomplete prefixes. Since each file has its own exponential distribution for prefixes, we have to derive the distribution for the μ's of all media files. Analysis of session prefixes in the HPC trace shows that μ's of all media files follow a *normal* distribution.

MediSyn generates a sequence of prefixes according to the generated ratios of r_c, r_u, r_e and μ for each file. Then, these prefixes are randomly matched with all the sessions of the file. In this way, MediSyn can generate all session prefixes for every file. Fig. 14 shows the cumulative distribution function (CDF) of the prefixes in the HPC trace compared with the CDF of the prefixes generated by MediSyn.

3.2. Temporal properties

3.2.1. Causes of temporal locality in media workloads collected over long period of time

Temporal reference locality, which is universally observed in web and media workloads [13,12,4], 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* [11,16]. Since popular files have a higher probability to be accessed within the access stream, a file's popularity contributes to its temporal locality. If we randomly permute the access stream, the temporal locality caused by skewed popularity is still preserved under reordering. However, temporal locality caused by temporal correlation cannot be preserved under random permutation. In MediSyn, to generate a stream of session arrivals exhibiting proper temporal locality, we need to clearly understand and distinguish the causes of temporal locality.

To check the existence of possible temporal correlation among sessions for the same files in our traces, we compare reference distances [4] between the original HPC trace and a randomly permuted HPC trace. Let $s_1, s_2, \ldots, s_n$ be the stream of accesses representing the media sessions of the entire HPC trace. Let $s_{i_1}, s_{i_2}, \ldots, s_{i_m}$ be the sequence of sessions to the same file f_i. The *reference distances* of s_{i_2}, $s_{i_3}, \ldots, s_{i_m}$ are defined as $i_2 - i_1$, $i_3 - i_2, \ldots$, $i_m - i_{m-1}$. We calculate the reference distances for all the files and their sessions over the entire HPC trace. Then we apply similar procedure to the randomly permuted HPC trace.

Fig. 15(a) shows the histogram of the reference distances in the original HPC trace on a log–log scale. The *X*-axis shows the reference distances, and the *Y*-axis shows the number of sessions in the original HPC trace for the corresponding reference distance. Fig. 15(b) shows the histogram of the reference distances in the randomly permuted HPC trace on a log–log scale. It can be observed that two curves are not the same. In Fig. 15(b), there are less references with small distances. For example, there are 122,765 references with distance 1 in Fig. 15(a). But there are only 5468 references with distance 1 in Fig. 15(b). Since in Fig. 15(b), reference distances are only determined by the file popularity, we assume the reason that Fig. 15(a) and (b) are not the same is that there is temporal correlation among sessions for the same files over the entire trace.

To verify whether our media traces exhibit short-term temporal correlation, we calculate reference distances within every day and sum the number of references with the same distance over all days for the HPC trace. Then, we permute accesses within

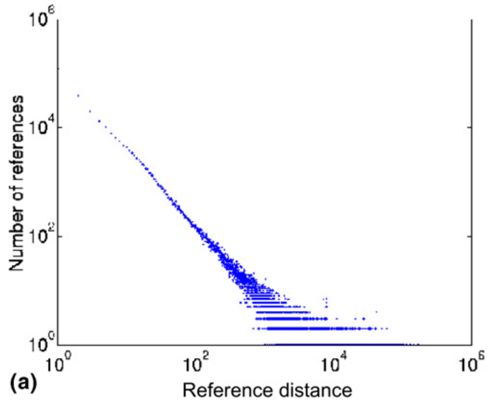


Fig. 15. (a) Reference distances calculated over the entire trace period in the original HPC trace and (b) reference distances calculated over the entire trace period in the permuted HPC trace.

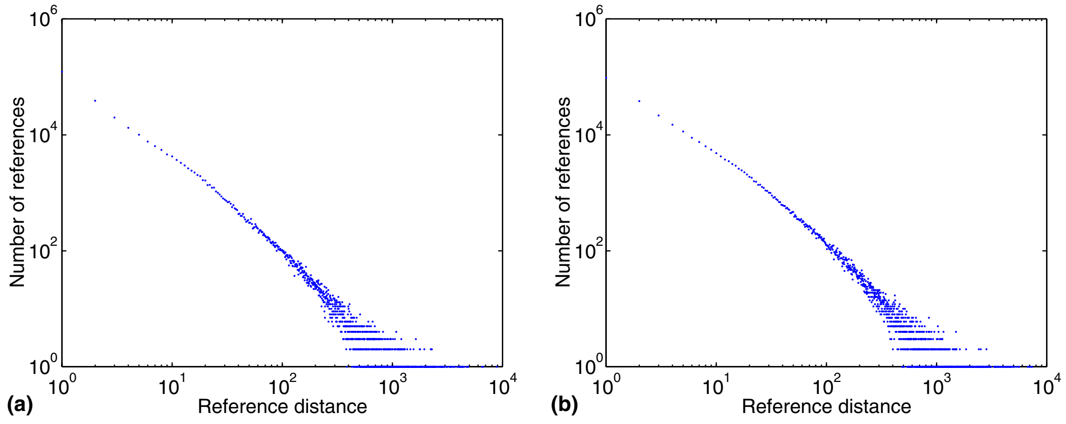


Fig. 16. (a) Reference distances calculated within each day in the original HPC trace and (b) reference distances calculated within each day in the permuted HPC trace.

every day and calculate reference distances again for the permuted trace. The results are shown in Fig. 16(a) and (b). We observe that there is almost no difference between the original trace and the permuted trace on reference distances. This implies that there is no temporal correlation for sessions within a single day and that temporal locality of sessions within a single day is purely determined by the file popularity distribution within that day.

The analysis above implies that temporal correlation in media workloads collected over long period of time is due to long-term temporal correlation exhibited on a daily time scale, and that there is no temporal correlation for sessions within a single day. These observations motivate our choice of temporal properties in MediSyn. The temporal properties described below intend to reflect the dynamics and evolution of accesses to media content over time and to define the proper temporal locality and long-term temporal correlation found in media workloads.

3.2.2. New file introduction process

One recent study [12] observes that accesses to new files constitute most of the accesses in any given month for enterprise media servers. We envision that this access pattern is even more pronounced for media news and sports sites. While for educational media sites the rate of new file introduction and accesses to them might be different, we aim to design a generic parameterized model capable of capturing the specifics of new content introduction for different media workloads. Among the design goals of our synthetic generator is the ability to reflect the evolution of media content provided by different media sites over a long period of time (months). Since we design MediSyn to support detailed resource allocation studies, we must account for the dynamic introduction of new content and its relative popularity.

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. Fig. 17 shows the distribution of new file introduction gaps measured in days for the HPC trace. The distribution depicted in Fig. 17 can be captured by a *Pareto* distribution with $\alpha = 2.0164$. Fig. 20 shows the histogram of the original trace and the PDF of the fitted Pareto distribution. Fig. 18 shows the introduction gap distribution for the HPL trace, which can be captured by an *exponential* distribution with $\mu = 4.2705$. Fig. 21 shows the histogram of the original trace and the PDF of the fitted exponential distribution. We center the exponential distribution at $x = 1$.

MediSyn can generate new file introduction time gaps according to one of three possible distribu-

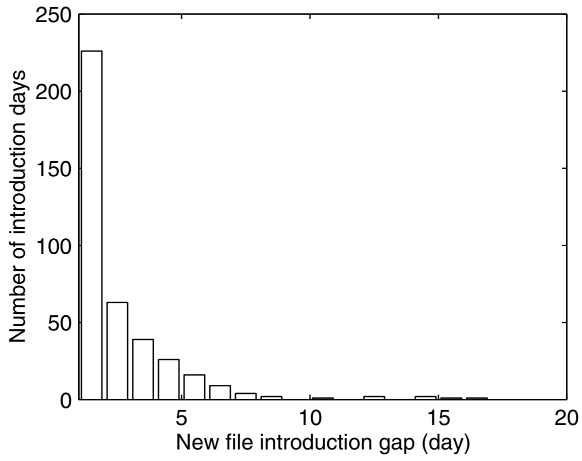


Fig. 17. New file introduction gaps measured in days for the HPC trace.

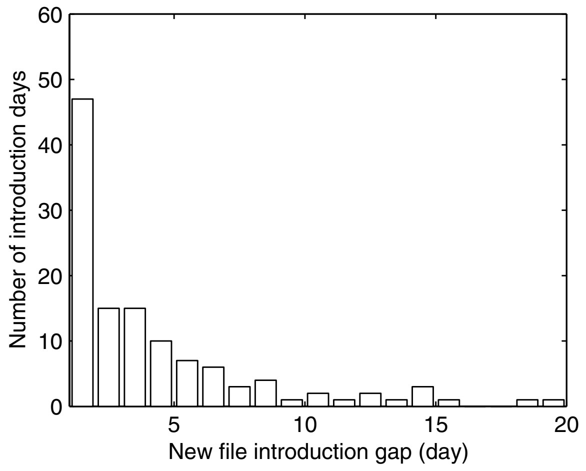


Fig. 18. New file introduction gaps measured in days for the HPL trace.

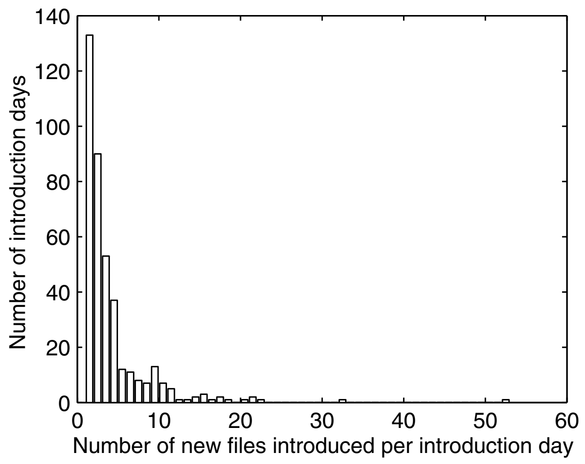


Fig. 19. The number of new files introduced per introduction day.

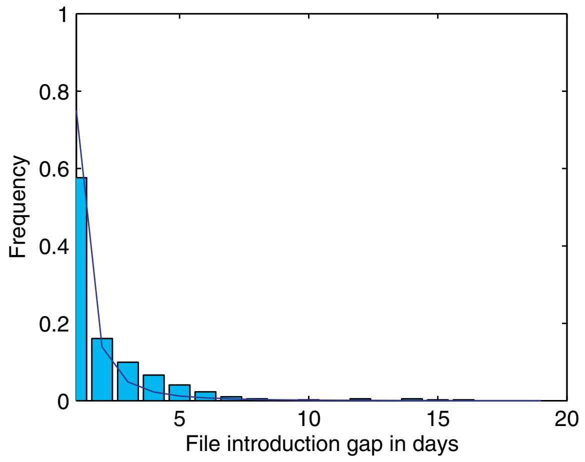


Fig. 20. The histogram and the PDF of the new file introduction gap for the HPC trace.

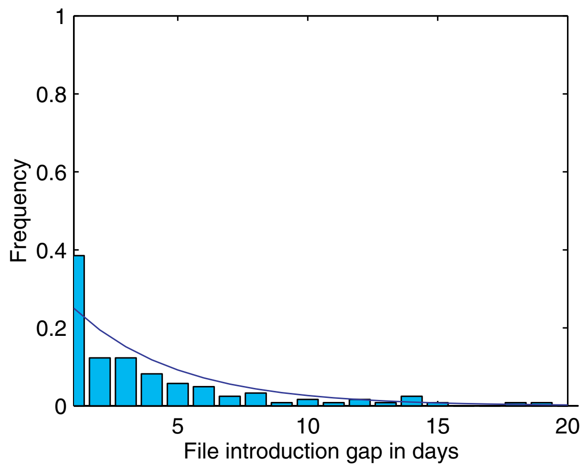


Fig. 21. The histogram and the PDF of the new file introduction gap for the HPL trace.

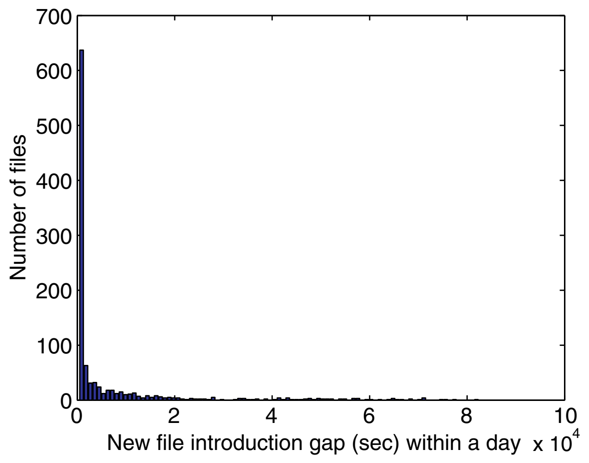


Fig. 23. New file introduction time gaps within an introduction day.

tions: (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. Fig. 19 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 $\alpha = 1.1323$. Fig. 22 shows the histogram and the PDF of the fitted Pareto distribution.

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. Fig. 23 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 s (15 min). 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

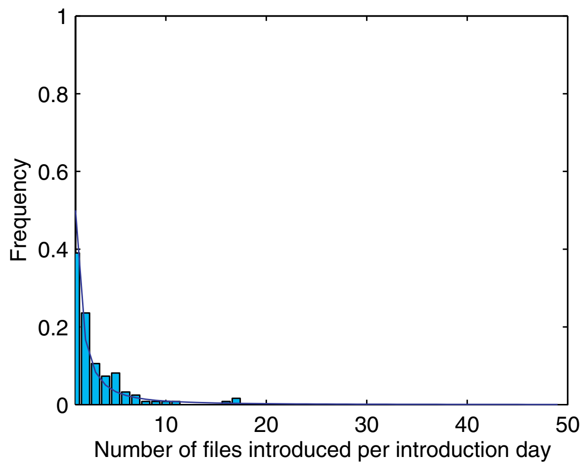


Fig. 22. The histogram and the PDF for the number of new files introduced per day.

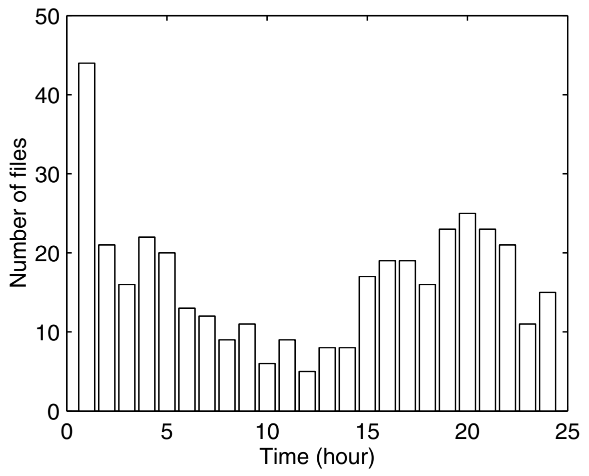


Fig. 24. The start times of new file introduction within introduction days.

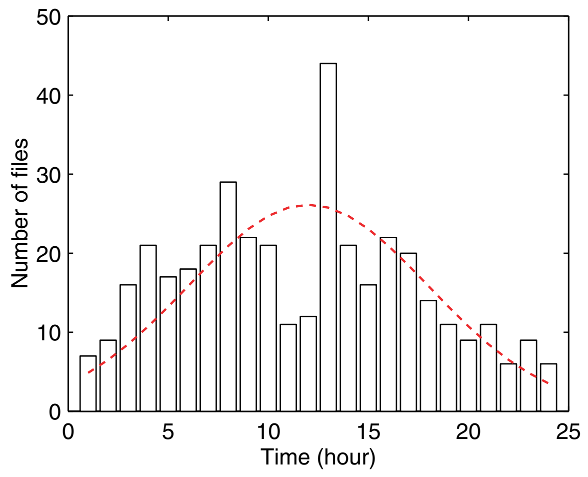


Fig. 25. The rotated start times of new file introduction within introduction days.

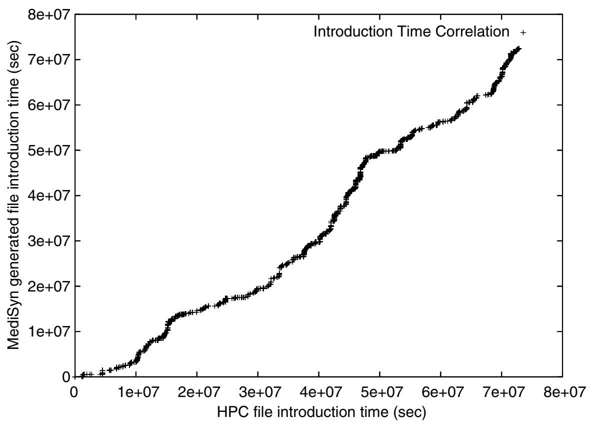


Fig. 26. The correlation of file introduction time between the HPC workload and the workload generated by MediSyn.

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. Fig. 24 shows this distribution. Since it looks like a rotated *normal* distribution with the peak at 0, we rotate the distribution by 12 h as shown in Fig. 25. This is a *normal* distribution with mean 43,200 s and standard deviation 21,600 s.

Fig. 26 shows the correlation of file introduction time between the HPC workload and a workload generated by MediSyn to simulate the HPC workload.

The correlation coefficient is 0.9889. So the generated file introduction time sequence matches the original trace quite well.

3.2.3. Life span

Since temporal correlation is observed in media workloads, an independent reference model combined with a global popularity distribution [9] 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 [9] 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.

A new property called *life span* has recently been proposed in [12] to measure the change in access rate of newly introduced files. Life spans reflect the timeliness of accesses to the introduced files. We observe that accesses to a media file are not uniformly distributed over the entire trace period. Instead, most accesses for a file occur shortly after the file is introduced, with access frequency gradually decreasing over time. For example, 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. This phenomenon implies that session arrivals might be very bursty when new files are introduced at a site.

To accurately model the non-stationarity of file popularity, we use the new file introduction process to mimic how media files are introduced at the 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 global 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 Figs. 27 and 28 respectively.

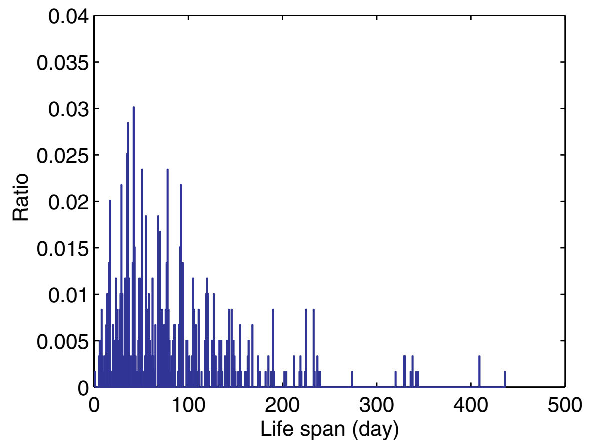


Fig. 27. A *regular* lifespan.

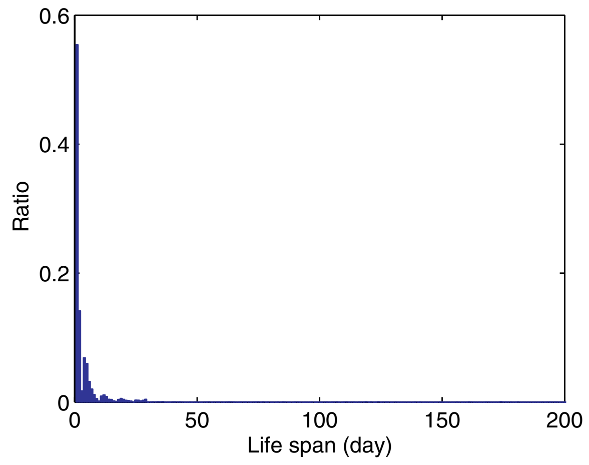


Fig. 28. A *news-like* lifespan.

Since most files in our traces have life spans similar to Fig. 27, we call this type of life span a *regular life span*.

News-like streaming contents typically have life span distributions similar to Fig. 28, 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 *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 6 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).

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 for having a *news-like life span*. Fig. 29 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.

Table 6
The parameters for the distributions (normal distributions) of the parameters in lognormal and pareto life span distributions

Normal distribution paramaters	Lognormal μ	Lognormal σ	Pareto α
μ	3.0935	1.1417	0.7023
σ	0.9612	0.3067	0.2092

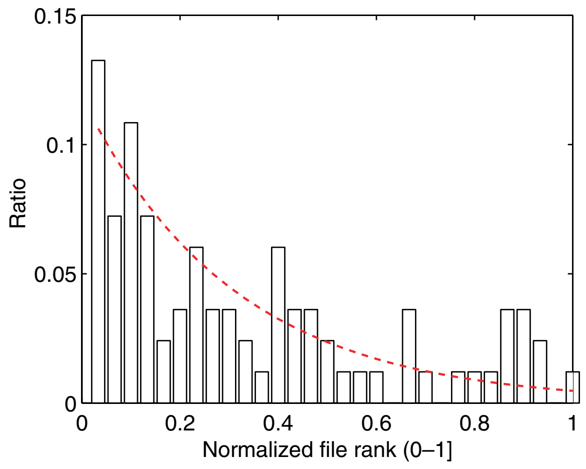


Fig. 29. PDF that a file at a certain rank has a Pareto life span distribution.

3.2.4. Diurnal access pattern

Earlier studies observed the diurnal access patterns for streaming media workloads [13,18,2,3,17]. The diurnal access pattern defines how the number of accesses to a site varies during a given period of time, e.g., a day. Diurnal access patterns might significantly vary for different media sites. For mixed media workloads utilizing a shared infrastructure, the diurnal access patterns have to be taken into account when designing the optimal support for efficient resource allocation. Additionally, 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.

After determining the life span and the global 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 wish to model both session interarrival time and diurnal access patterns.

Fig. 30 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 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. Fig. 31 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.

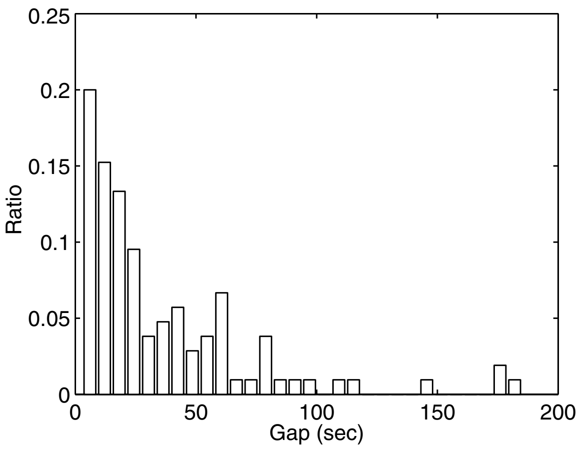


Fig. 31. The PDF of session access interarrival time gaps for a file measured in an hour.

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. Fig. 32 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 Fig. 32, which contains a set of bins.

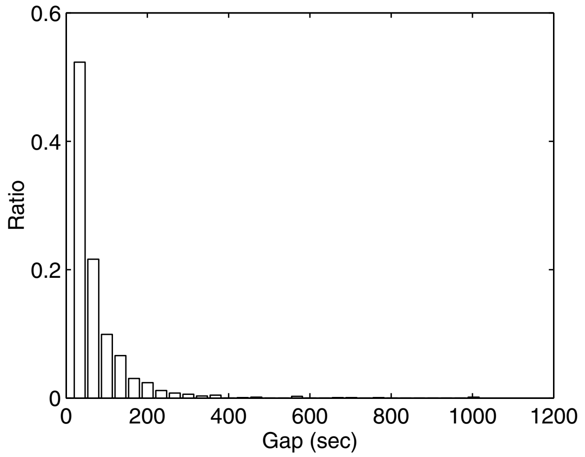


Fig. 30. The PDF of session access interarrival time gaps for a file measured in a day.

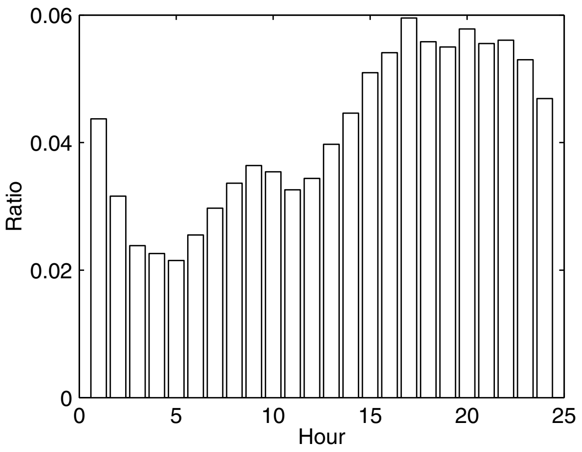


Fig. 32. The session access diurnal pattern for the HPC trace. Each bin is an hour.

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 *non-homogeneous Poisson* process [23], 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.

4. Workload generation process in MediSyn

MediSyn's goal is to generate a synthetic trace representing a sequence of file accesses to media service. This process consists of **two steps**: *file property generation* and *file access generation.*

- **File property generation**
  The first step is to generate values for all properties introduced above for each media file. Static properties define file set parameters (duration and encoding bit rate) and the principal access patterns (popularity and prefix). Temporal properties define the ordering and timing of media sessions. MediSyn defines these properties using a parameterized set of distributions in the input configuration file. If a property can be described by a value such as global file popularity, duration, encoding bit rate, MediSyn first generates a sequence of values according to the given distribution, and then selects a value for each file. If a property is modeled as a distribution, the choice of the distribution and parameter(s) of the distribution are generated for each file.Thus, 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 first step, the set of corresponding static and temporal properties shown in Table 7 is generated for each file. Section 3 describes each property generation and correlations among the properties in detail.
- **File access generation**
  Taking the assigned file popularities as the basis, MediSyn independently generates the arrival of media sessions to each file using: (i) the initial file introduction time, (ii) the life span of the file, and (iii) the diurnal access pattern of the file. Each file access 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 together to generate the synthetic trace.

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 [2,3,13,12,18,20].

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

Table 7
Properties generated for each file

File id	Duration	Bit rate (Kbps)	Popularity	File introduction time (s)	Life span	Life span parameters	⋯
1	3600	112	20,000	100	Pareto	1.0	⋯
2	200	350	14,300	50	Lognormal	2.0,10.0	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	600	28.8	1	10,000	Lognormal	1.0, 1.0	⋯

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. [13] 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 min), 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 [12]. 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.

Since HTTP requests and streaming media sessions are very different, streaming media workloads exhibit many new properties relative to traditional web workloads. Thus existing synthetic web workload generators [6] are not suitable for generating streaming media workloads.

The only synthetic workload generator for streaming media reported in literature is GISMO [17]. 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 Internet hosting centers and CDNs, serving streaming media content, requires performing the experiments with realistic streaming media workloads which need to be scaled, parametrized, and mixed in a controllable and desirable way.

In this work, we present a synthetic streaming media workload generator, MediSyn, which is specially designed to accomplish 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, non-stationary file popularity, new file introduction process, and diurnal access pattern. Among the primary 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 the sites over time. Our evaluation, based on two long-term traces of streaming media services, demonstrates that MediSyn accurately captures the essential properties of media workloads, which are chosen to represent the unique (while generic) properties of streaming media workloads and their dynamics over time.

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

As part of MediSyn, we plan to release a workload analysis tool reflecting the property profiles generated by MediSyn. These profiles can be conveniently used for tuning the workload generator parameters to specify the desired properties.

Acknowledgements

Our sincere thanks to anonymous referees for useful remarks and numerous suggestions during several thorough revisions of the paper. Their insightful questions and constructive criticism helped us to significantly improve the content and presentation of the paper.

References

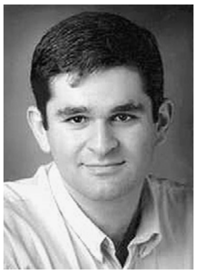
[1] General Pareto Distribution. Available from: <http://www.math.uah.edu/stat/special/special12.html>.

[2] Soam Acharya, Brian Smith, Peter Parnes, Characterizing user access to videos on the world wide Web, in: Proceedings of ACM/SPIE Multimedia Computing and Networking, January 2000.
[3] Jussara Almeida, Jeffrey Krueger, Derek Eager, Mary Vernon, Analysis of educational media server workloads, in: Proceedings of NOSSDAV, June 2001.
[4] Virgílio Almeida, Azer Bestavros, Mark Crovella, Adriana de Oliveira, Characterizing reference locality in the WWW, in: Proceedings of PDIS, December 1996.
[5] Virgilio Augusto Almeida, Marcio Cesirio, Rodrigo Fonseca, Wagner Meira Jr., Cristina Murta, Analyzing the behavior of a proxy server in the light of regional and cultural issues, in: Proceedings of WCW, June 1998.
[6] Paul Barford, Mark Crovella, Generating representative Web workloads for network and server performance evaluation, in: Proceedings of SIGMETRICS, June 1998.
[7] Dave Bianchi, CNN.com: Facing A World Crisis. Available from: <http://www.tcsa.org/lisa2001/cnn.txt>.
[8] Rebecca Braynard, Dejan Kostić, Adolfo Rodriguez, Jeffrey Chase, Amin Vahdat, Opus: an overlay peer utility service, in: Proceedings of the 5th International Conference on Open Architectures and Network Programming (OPENARCH), June 2002.
[9] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, Scott Shenker, Web caching and Zipf-like distributions: evidence, and implications, in: Proceedings of INFOCOM, March 1999.
[10] Jeffrey Chase, Darrell Anderson, Prachi Thakar, Amin Vahdat, Ronald Doyle, Managing Energy and Server Resources in Hosting Centers, in: Proceedings of the 18th ACM SOSP, October 2001.
[11] Ludmila Cherkasova, Gianfranco Ciardo, Characterizing temporal locality and its impact on Web server performance, in: Proceedings of ICCCN, October 2000.
[12] Ludmila Cherkasova, Minaxi Gupta, Characterizing locality, evolution, and life span of accesses in enterprise media server workloads, in: Proceedings of NOSSDAV, May 2002.
[13] Maureen Chesire, Alec Wolman, Geoffrey Voelker, Henry Levy, Measurement and analysis of a streaming-media workload, in: Proceedings of USITS, March 2001.
[14] Morris DeGroot, Mark Schervish, Probability and Statistics, third ed., Addison-Wesley, 2002.
[15] Raj Jain, The art of computer systems performance analysis: technique for experimental design, measurement, simulation and modeling, John Wiley & Sons, 1992.
[16] Shudong Jin, Azer 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.
[17] Shudong Jin, Azer 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.
[18] Dario Luperello, Sarit Mukherjee, Sanjoy Paul, Streaming media traffic: an empirical study, in: Proceedings of Web Caching Workshop, June 2002.
[19] Hewlett Packard, Utility Data Center. Available from: <http://www.hp.com/go/hpudc>.
[20] Jitendra Padhye, Jim Kurose, An empirical study of client interactions with a continuous-media courseware server, in: Proceedings of NOSSDAV, June 1998.
[21] V. Paxson, Empirically-derived analytic models of wide-area TCP connections, IEEE/ACM Transactions on Networking 2 (4) (1994).
[22] S. Pederson, M. Johnson, Estimating model discrepancy, Technometrics 32 (3) (1990) 305–314.
[23] Sheldon Ross, Introduction to Probability Models, Academic Press, 1997.
[24] Subhabrata Sen, Jennifer Rexford, Don Towsley, Proxy prefix caching for multimedia streams, in: Proceedings of INFOCOM, March 1999.
[25] Zipf–Mandelbrot law. Available from: <http://en.wikipedia.org/wiki/Zipf-Mandelbrot_law>.
[26] Vern Paxson, Empirically-derived analytic models of wide-area TCP connections, IEEE/ACM Transactions on Networking 2 (4) (1994).

Wenting Tang currently works at Arcsight Inc. to develop next-generation Security Event and Information Management (SEIM) system. Before that, he worked at VMware to develop virtual infrastructure management software. He conducted research in areas of utility computing, content delivery networks, streaming media and scalable web server systems in HPLabs before he joined VMware. He received his Ph.D. from Michigan State University.

Yun Fu received his B.S. and M.Eng. in Computer Science from Nankai University, China, in 1995 and 1998 respectively and his M.S. and Ph.D. in Computer Science from Duke University, USA, in 2001 and 2004 respectively. He is currently a senior engineer at Yahoo! Inc. His research interests include distributed systems, operating systems, computer networks, databases, and data mining.

Ludmila Cherkasova is a senior scientist in the Enterprise Software and Systems Laboratory at HPLabs, Palo Alto. She joined Hewlett-Packard Laboratories in 1991. Before joining HPLabs, she was a senior researcher at Institute of Computing Systems, Russia, and adjunct associate professor at Novosibirsk State University. Her current research interests are in distributed systems, internet technologies and networking, performance measurement and monitoring, characterization of next-generation system workloads and emerging applications in the large-scale enterprise data centers.



Amin Vahdat is an Associate Professor in the Computer Science and Engineering Department at UC San Diego and the Director of the Center for Networked Systems. He received his Ph.D. from the University of California, Berkeley and then was on the faculty at Duke University from 1998 to 2003 before joining UC San Diego in 2004. Vahdat received the NSF CAREER award in 2000, the Alfred P. Sloan Fellowship in 2003 and the Duke University David and Janet Vaughn Distinguished Teaching Award in 2003. He co-founded the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI) in 2003. His research focuses broadly on computer systems and networks, with recent focuses on availability, resource allocation, programming models and languages for distributed systems, and scalable network emulation environments.