SearchGen: a Synthetic Workload Generator for Scientific Literature Digital Libraries and Search Engines

Huajing Li[1] Wang-Chien Lee[1] Anand Sivasubramaniam[1] C. Lee Giles[1,2]
[1]Department of Computer Science and Engineering
[2]College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802, USA
{huali, wlee, anand}@cse.psu.edu, giles@ist.psu.edu

ABSTRACT

Due to the popularity of web applications and their heavy usage, it is important to obtain a good understanding of their workloads in order to improve performance of search services. Existing works have typically focused on generic web workloads without putting emphasis on specific domains. In this paper, we analyze the usage logs of CiteSeer, a scientific literature digital library and search engine, to characterize workloads for both robots and users. Essential ingredients that contribute to workloads are proposed. Among them we find the access intervals show high variance, and thus cannot be predicted well with time-series models. On the other hand, client visiting path and semantics can be well captured with probabilistic models and Zipf-law. Based on the findings, we propose SearchGen, a synthetic workload generator to output traces for scientific literature digital libraries and search engines. A comparison between synthetic workloads and actual logged traces suggests that the synthetic workload fits well.

Categories and Subject Descriptors

H.3.6 [**INFORMATION SYSTEMS**]: Library Automation—*Large text archives*; H.3.4 [**INFORMATION SYSTEMS**]: Systems and Software—*Distributed systems*; C.4 [**COMPUTER SYSTEM ORGANIZATION**]: PERFORMANCE OF SYSTEMS—*Modeling techniques*

General Terms

Measurement, Verification, Performance

Keywords

Synthetic Workload, Modeling, Benchmark

1. INTRODUCTION

Search engines have become an integral part of daily life for nearly all Internet users[1]. Many different types of search engines, being general purpose or specialized, commercial or open source, autonomous or manually administrated, have appeared on the Web. Moreover, many web applications, such as digital libraries and e-commerce websites, also provide a search functionality as part of their services. Thus, there is a need to better understand search engine usage to improve performance.

The operating nature and functionalities of search engines make them different from other websites which are designed to host web contents specific to the site. Typically, search engines have autonomous data producers (e.g., crawlers, metadata harvesters, etc.) that fetch data from Internet continuously. The amount of stored data (including indices, cached data, extracted metadata, etc.) behind a search service is much larger than common websites and typically has a very large user base, which makes performance a significant concern for such systems.

Obviously, the performance of search engines is highly dependent on the workload. An in-depth understanding of workloads for search engines can facilitate system tuning, enhance system performance and reduce maintenance cost. Although some studies on web workloads exist in the literature, they do not focus on search engines and thereafter cannot provide adequate insights. To our knowledge there is no extensive workload analysis for search engines. The goal of this research work is to characterize search engine workloads to achieve better system design and operations.

We focus on domain search engines for scientific uses, which have an archive of scholarly publications in electronic form and serve the scientific community as digital libraries with their dedicated document query and retrieval utilities. Automatic citation indexing (ACI) [11], together with the citation query utility, is an important feature of such systems. As a well-known example, CiteSeer[2] and its successor, CiteSeer$^{\mathcal{X}}$ [15], are web-based scientific literature search engines as well as digital libraries which focus on computer and information science. Other examples include the Google Scholar[3] and the REXA Project[4]. They are different from general purpose search engines like Google in that they are dedicated in a limited scope of online resources and serve a specific community, with value-added services.

Due to its heavy use and popularity (over 500,000 hits a day), CiteSeer has been considered as a representative in the space of scientific literature search engines. Thus, we choose

[1]Pew Internet Study: http://www.pewinternet.org/

[2]http://citeseer.ist.psu.edu
[3]http://scholar.google.com
[4]http://rexa.info/

JCDL'07, June 18–23, 2007, Vancouver, British Columbia, Canada.

to use its logs in this study. By characterizing typical workloads of CiteSeer, we aim at developing models to capture the behaviors of users to such systems. Our synthetic workload generator, SearchGen, has the capability to generate workloads not only desirable for testing and performance tuning of a search engine system, but essential for advancing research in related areas. The primary contributions of this work are:

- We define the basic characteristics of the workload for scientific literature digital libraries and search engines, each of which has corresponding impact on system performances.
- The characteristics of a synthetic workload are used to decompose the entire stream of events into a group of single fine-grained streams. The model for each stream is proposed and is used in predicting a portion of the synthetic workload.
- We develop SearchGen, which merges all individual streams corresponding to the proposed characteristics into an integral trace. Our generated workload is compared with actual traces from CiteSeer. The synthetic workload, with characteristics observable in real logs, can be used for system testing, tuning and evaluation.
- Although we focus on scientific literature search engines, the analysis procedure described in this paper is appropriate for other application domains.

The rest of the paper is organized as follows. Related research is reviewed in Section 2, while additional background is provided in Section 3. The overall modeling strategy is presented in Section 4. The preliminary data preparation to the system logs is described in Section 5. Modeling methodology and results for each single workload characteristic are presented in Section 6, followed by a discussion in Section 7 of how we merge and validate synthetic traces. Concluding remarks and future plans are in Section 8.

2. RELATED WORK

Workload analysis is used in many applications for system performance tuning and benchmarking. World Wide Web traffic has been analyzed and modeled by previous works [1], in which it is suggested that web access can be modeled by heavy-tailed distributions [9]. The study also suggests that web traffic shows a self-similarity characteristic. With the property of self-similarity in web traces, the correlational structure in a time series of events remains unchanged regardless measure scales. This feature can be explained by facts such as self-similar property of underlying file system statistics and user behavior. Self-similarity has been found in I/O workloads [12] as well.

The synthetic workload generator, SURGE [3], presents representative web workloads for network and server performance evaluation. *User Equivalence* (UE) is generated in SURGE to represent a population of a known number of users. In each UE, six web characteristics are statistically modeled, which include file sizes, request sizes, popularity, embedded references, temporal locality, and OFF times. The generated trace is compared with other synthetic workloads, showing that SURGE well models actual web traffic. More importantly, a SURGE-generated trace is found to have the self-similarity feature of actual workload data. However, SURGE cannot be applied to generate user traffic for search engines because of the unique running nature of search engines versus generic websites. Query processing inside a distributed search engine architecture is studied in [2], which focuses on issues such as performance bottleneck and index server load balancing. However, there still lacks a complete model for the application genre. Other network workload generators, such as [19], emulate protocol-based traffic to benchmark the performance of web servers. From an ISP's perspective, [4] analyzes characteristics of a server that hosts many websites.

A recent workload study [24] emphasizes generating accurate I/O requests for TPC-H queries to evaluate the performance of disk subsystems with 22 queries of the TPC-H workload. The analysis is performed at the disk block-level. The request arrival-pattern (inter-arrival times) and access-pattern (request location and requested data size) are synthesized. In database fields, [8] addresses the problem of identifying primitives which best summarize SQL workloads sent to databases. The interest in workload analysis continues to grow as more applications benefit from synthetic workload generation, such as adaptive system tuning [16, 20], testing [23] and diagnosis [25]. [17] shows the extracted user workload models can be applied in improving search engine services.

There also exist many web-oriented benchmark projects where typical web application scenarios are identified and the corresponding web workload traffic is synthetically generated. Well-known examples include TPC-App[5] (an application server and web service benchmark), TPC-H[6] (a benchmark for decision-support applications), TPC-W[7] (a transactional web benchmark), and SPECweb2005[8] (the next-generation SPEC benchmark for evaluating the performance of web servers). However, none of them aims at providing benchmark suites for search engines.

3. BACKGROUND

Two key concepts are introduced in this section for their relevance in summarizing web workloads.

3.1 Self-Similarity

Self-similarity [9] is common in web traffic, which can be explained by assuming the existence of a heavy-tail distribution in workload characteristics, including file sizes, transfer times and user think times. A random variable X follows a heavy-tailed distribution if

$$P[X > x] \sim x^{-\alpha}$$

as $x \to \infty, 0 < \alpha < 2$.

Namely, regardless of the behavior of the distribution for small values of the random variable, it is heavy-tailed if the asymptotic shape of the distribution is hyperbolic.

Intuitively, self-similarity means that the features are independent of scale. For time-series, a process is called self-similar when the statistics are independent of the time scale. It can be expected that averaging over equal periods of time does not influence the statistical characteristics of the process. Formally, a time-series X_t $(t = 1, 2, ...)$ is said to

[5]http://www.tpc.org/tpc_app/default.asp
[6]http://www.tpc.org/tpch/default.asp
[7]http://www.tpc.org/tpcw/default.asp
[8]http://www.spec.org/web2005/

be *exactly second-order self-similar* if

$$X_t \stackrel{d}{=} m^{-H} \sum_{i=m(t-1)+1}^{mt} X_i$$

for $1/2 < H < 1$ and all $m > 0$, where $\stackrel{d}{=}$ means an equivalent distribution. This suggests a methodology for testing self-similarity in a time-series. The series is separated into non-overlapping blocks with the same size m. All observations in each block are aggregated, by which a new m-aggregated time-series is constructed. If the new series is statistically identical with the original series, scaled by a factor of m^{-H}, the original time-series is *self-similar*. This method is adopted by self-similarity tests in Section 6.

3.2 Short-Range and Long-Range Dependence

In a self-similar time-series where observation bursts can be observed in a wide range of time-scales, the distribution exhibits *long-range dependence* [9]. Long-range dependence (LRD) exists when the current observation is highly correlated with observations far away in time. On the other hand, *short-range dependence* (SRD) [9] shows that the current observation is only correlated with recent observations. If we analyze the correlation functions of an SRD time-series, the correlation value decreases dramatically to a very small value, while the LRD series retains considerably significant correlation values for distant observations.

4. WORKLOAD CHARACTERIZATION

In order to construct a synthetic workload for scientific literature digital libraries and search engines, we need to determine what characteristics are essential to the performance evaluation.

According to [3], a typical web workload consists of six characteristics: (1) File Size, (2) Request Size, (3) Popularity, (4) Embedded Reference, (5) Temporal Locality, (6) OFF Time. This model can accurately represent the workload on a web server which serves incoming HTTP requests. However, the focus of our study is not on a web server, but a search engine. Due to different operating nature of search engines, the above cannot be directly used. Typically, there are not many static web pages stored in a search engine and the contents are often dynamically provided by indices and data providers in the system. The minimal unit of a synthetic workload for a search engine is not a file with a numbers of file accesses and file size. On the other hand, a search engine workload provides requests, according to which the corresponding web objects are constructed on-the-fly and returned to users.

To characterize CiteSeer workload, it is imperative to look into the usage logs. A typical logging entry in CiteSeer is listed below:

> 1114070813.127 event context 0 1782747 0 446930
> ip: 128.255.54.*[9] agent: Generic

This entry can be divided into five parts:

- **Time Stamp**: records the arrival time of the request.
- **Request Type**: decides which service is invoked to answer the request.
- **Request Parameters**: the actual meaning of parameters is dependent on request types. For example, they provide query terms for a document query and identifiers for document retrieval.
- **IP Address**: provides address information of clients.
- **Agent Type**: suggests the types of clients. We can infer from the agent type whether the request is from a user or a robot.

[9]We suppressed the last section of the IP address.

Not all the above information will be generated in our synthetic workload. Our goal is to develop a workload generator which can be used to simulate user access patterns and test the performance of search engines. Aspects with little influence on the system's performance such as browser type are not considered in our study. From a server's perspective, what really matters is: the time behavior and the semantic behavior of requests. The time behaviors decide how many simultaneous requests are accepted by the application at any time. The semantics of requests determine what services of the search engine are invoked to fulfill the requests and what data are returned. From the previous list, the time stamp can be used to model the time behavior of the workload. The request types and request parameters are components of semantic behaviors. Other parts of a log entry are removed from the synthetic workload because they are not relevant to performance. However, we will make use of the agent type information in our analysis to differentiate client types, as shown in Section 5.

It is extremely difficult to model the synthetic workload as a whole. The time and semantic behaviors need to be modeled separately as they are independent from each other. Furthermore, the time behavior of requests can be further decomposed. In system logs, all visits are collected and merged into a continuous record, which is a stream of the requests imposed on the server. However, keep in mind that this stream is comprised of many individual sessions, which provides a method to divide the trace into smaller units.

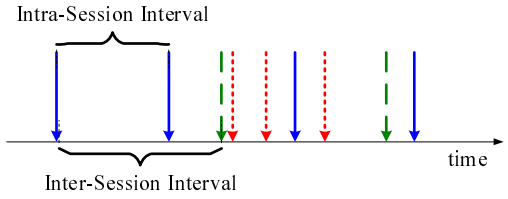


Figure 1: Request stream received by the server.

Figure 1 illustrates the composition of the request stream, in which requests from different session are represented by different line patterns. Following this direction, two sets of time intervals are identified, as marked in the figure.

- **Intra-Session Interval**: the time interval between two subsequent requests in one session.
- **Inter-Session Interval**: the time interval between subsequent sessions.

The two streams are studied independently because the intra-session interval is a client side behavior, decided solely by an individual client, whereas the inter-session interval is a server side aggregate. We do not model the session length in

the time behavior analysis. This problem will be addressed in Section 6 by using Markov models. When modeling time intervals, the logged request entries are segregated and organized into appropriate sessions. We first study each time stream (intra-session interval and inter-session interval) to test assumptions such as the existence of *IID* (Independent Identically Distributed) and self-similarity. These assumptions decide what modeling strategy is appropriate. Based on the testing results, we propose models that best fit in the Kolmogorov-Smirnov (K-S) [14] sense.

Semantic behavior of a client can also be put into two categories: request type and request parameter. Second-order Markov model is used to predict the "next-step" of requests. Two types of parameters, query terms and requested document identifiers, are studied respectively. Once the request type is decided, SearchGen acquires a parameter value from the corresponding parameter generator, which is built upon distributional study for request parameter frequencies of each representative request type. After combining with the time intervals, all workload streams are merged to output the synthetic trace.

5. DATA PREPARATION

To model workload characteristics, we prepare a one-month CiteSeer log. Three preliminary data preparation tasks are performed before analysis.

5.1 Application Model Simplification

As a complex web application, CiteSeer provides many services from the web interface, some of which are either irrelevant with search engine functionalities (e.g., user feedback) or seldom used (e.g., online paper submission). These requests are eliminated from the scope of the study. On the other hand, we do not want to construct a synthetic workload that is specifically tailored for CiteSeer. To achieve this, some CiteSeer-specific requests should not be considered.

Originally, CiteSeer has more than 40 request types recorded in its logs. After comparing available services provided both by CiteSeer and Google Scholar, some unique services of CiteSeer are excluded. Also, some similar service types are grouped into a more generic category. For instance, in CiteSeer, after finding a document, a user can ask for documents from the same source, co-citation documents and text-based similar documents. In the simplified application model, these requests are all mapped into *related* request type, suggesting this service returns users with related documents with any metric.

The final simplified application model consists of six unique request types, which are (1)Query Document, (2)Query Citation, (3)Document Retrieval, (4)Related, (5)External URLs, and (6)Document Specification. They are also unique states in the Markov model proposed in Section 6.

5.2 Sessionization

As stated in Section 4, to study the time behavior, the entire trace is decomposed into a series of sessions to analyze the models of intra-session interval and inter-session interval. This procedure can be divided into two tasks: correct mapping of activities to different users and correct separation of activities belonging to different visits of the same user. However, the original CiteSeer system does not record user identities as well as session information in its logs. Without the necessary information, heuristics can assist to construct sessions. [6] suggests that page stay interval performs well for short but temporally dense sessions, which are common for search engines. So the following heuristics are used:

- The IP address information is used as the identity of users. We regard every IP address as a single user, for which requests from a same IP address are taken as from the same user. Although it is not definitely accurate, this is the best available approach.
- The log entries that are belonged to an IP address are grouped and recorded sequentially by their time stamps. If we find the successive visit interval by a user exceeds a time threshold, the latter request is taken as the start of a new session. In our experiment, we set the time threshold to be $1,800$ seconds (30 minutes).

5.3 Robot Detection

It has been observed that the requests initialized by robots (crawlers, web spiders, etc.) contribute a considerable portion of the network traffic. Unlike users, robots start their crawling from some given seed URLs, and continue to probe the Internet using web links. From the nature of their behaviors it can be inferred that robot access patterns will be dramatically different from users. As far as we know previous web workload studies do not single out robots from the traffic and model their behavior separately. Considering the significant web traffic generated by robots, we study the behaviors of robots and users respectively, and propose models for both.

Web robot session studies [21] indicate that by collecting web access attributes such as agent type, access average time, and total request pages, most robot sessions can be accurately labeled. Based on what are recorded in the logs, some representative heuristics are adopted. First, we infer the type of a client from the agent type section in a log entry. Basically, we used a robot agent string list[10] to identify obvious robots. However, some robots do not declare themselves when they access CiteSeer services, disguising as real users. To locate such unfriendly robots, we gather statistical information from sessions. If we find that either the average session length is extremely long (more than 100) or a portion (10%) of intra-session intervals are particularly small (less than 0.5 second), the IP address is labeled with "robots" because they show extremely odd behaviors other than normal users.

6. CHARACTERISTIC MODELING

6.1 Time Behavior

We begin by examining the characteristics related to time behavior of the workload (i.e., request intervals). In the following discussion, we first perform statistical tests on the logged streams to decide the appropriate modeling strategy for the time series. Based on the test results, we give models to the time characteristics.

6.1.1 Studying Correlations

Before choosing a modeling strategy, it is necessary to understand the underlying nature of a given time-series because different stream models (LRD, SRD or IID) have their appropriate modeling techniques. To reveal the inherent nature, correlation studies are widely used. If we find the correlation in a time-series is not significant, we can assume

[10]http://www.pgts.com.au/pgtsj/pgtsj0208d.html

safely that the time-series is IID, for which we can use *statistical models* (such as normal distribution, pareto distribution, etc.) to fit the stream. On the other hand, if the correlation result suggests that an LRD or SRD exists, a simple statistical modeling will not be sufficient to capture the dependency between observations. In such a situation, we will opt to use time-series models to generate the output stream.

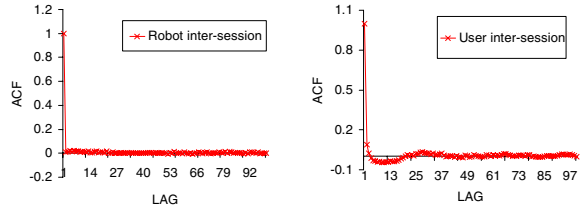
Generally, given a time-series as the input, the *autocorrelation function* (ACF) is used to study the correlation within the stream. ACF measures the similarity between the series X_t and its shifted version X_{t+k}, where k is the lag. The formal definition of a sample autocorrelation function is given as:

$$\rho(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\sigma^2}$$

where μ, σ are the sample mean and standard deviation respectively. ACF values can be plotted with different lags. If the ACF decays hyperbolically to zero, then the process shows LRD, where distant observations have significant influences. On the contrary, if the ACF is large for small lags and decreases dramatically when the lag increases, SRD exists. Intuitively, if none of the ACF values for shifted series is significant, we can assume the time-series to be IID.

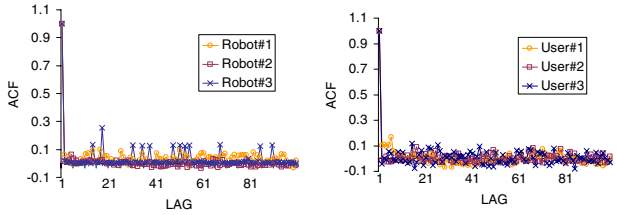
In our analysis, we treat robots and users separately. Figure 2 gives the ACF plots for inter-session streams. It is evident that the ACF values drop dramatically as the lag increases and remain close to 0 for large lags, which shows that the session intervals are independent.

To study the nature of intra-session streams, we randomly select representative sessions from robot and user categories. Here we deliberately select those sessions that do not show abnormal behaviors. The corresponding ACFs are plotted in Figure 3.



(a) Robot. (b) User.

Figure 2: ACF for inter-session intervals.

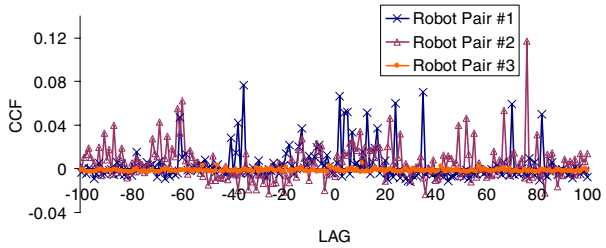


(a) Robot. (b) User.

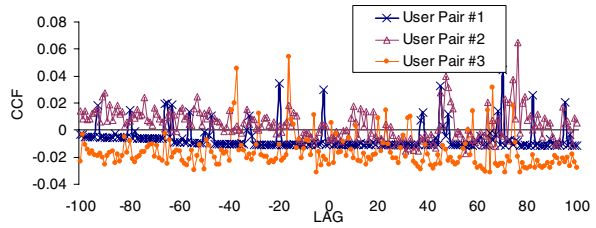
Figure 3: ACF for intra-session intervals.

It can be seen from the ACF plots that most of the absolute correlation values are extremely low. Figure 3(a) shows periodical spikes, which can reach a maximum value of 0.2, implying that some periodical pattern may exist for this robot. However, the spikes never reach a value of 0.25, indicating the correlation is still weak. Hence, all the results give us strong indications that no significant LRD or SRD exists and IID seems a better choice.

It is useful as well to study the correlation among multiple streams. Here we need to know if the correlation exists for the intra-session intervals of different sessions. The *cross-correlation function* (CCF) computes the similarity between two streams given a lag k. Our analysis results are shown in Figure 4, in which multiple pairs of intra-session intervals are compared.



(a) Robot.



(b) User.

Figure 4: CCF for intra-session interval pairs.

Again we study the behavior of robots and users independently. In each category, we randomly select three pairs of intra-session intervals as input streams. The results in Figure 4 indicate no strong correlation between different sessions exists. Most of the CCF values are smaller than 0.1, with only a few spikes reaching 0.12. Although only a small portion of sessions are compared in this experiment, considering the diverse use nature of clients and the fairness of our session selection policy, we can safely conclude that for most cases the sessions are not heavily correlated. This finding suggests that when we model the workload, we do not need to take the correlation between sessions into consideration.

Although the ACF results strongly suggest the existence of IID, we can use self-similarity tests to reinforce our beliefs in the conclusion that IID exists in the analyzed streams. Four methods [9] are recommended to test self-similarity: *variance-time plot*, *R/S plot*, *periodogram method* and *Whittle estimator* [5]. In all of the above tests, the degree of self-similarity is expressed with a single parameter H, the *Hurst parameter*. For self-similar series with LRD, $1/2 < H < 1$. As the H value goes up, the degree of both self-similarity and LRD increases.

We apply all the four tests on the streams which are extracted and studied in the previous ACF tests, with H values listed in Table 1.

In Table 1, the first column marks the streams under study, in which character "R" represents robot streams and "H" stands for user streams. Inter-session intervals are ab-

Stream	Varience -Time	R/S	Periodogram	Whittle
Inter(R)	0.689	0.636	0.547	0.521
Inter(H)	0.721	0.559	0.257	0.539
Intra(R1)	0.553	0.507	0.482	0.500
Intra(R2)	0.768	0.520	0.499	0.553
Intra(R3)	0.706	0.591	0.555	0.532
Intra(R4)	0.403	0.571	0.575	0.522
Intra(R5)	0.410	0.534	0.812	0.599
Intra(H1)	0.499	0.653	0.736	0.592
Intra(H2)	0.499	0.556	0.563	0.500
Intra(H3)	0.520	0.585	0.706	0.500
Intra(H4)	0.633	0.507	0.431	0.504
Intra(H5)	0.455	0.554	0.443	0.500

Table 1: Self-similarity test results.

breviated as "Inter" and intra-session intervals are written as "Intra". This naming schema is followed by all the following tables.

In addition to the H values listed in Table 1, the variance-time plot, R/S plot and periodogram method all return a correlation coefficient to estimate the reliability of the test. The Whittle estimator can produce confidence intervals, but its values are not shown here due to space limit. As we look to the values in Table 1, we can find that most are around 0.5, suggesting that the degree of self-similarity is not significant. Although some tests return a higher H value, its corresponding correlation coefficient indicates that the measure is not very reliable. For example, the value periodogram method only has a 39.05% correlation coefficient for intra-session intervals of robot5.

In summary, the results of ACF and self-similarity tests reveal that both LRD (self-similarity) and SRD do not exist in our target time streams. Thus, we assume them to be IID and use statistical models.

6.1.2 Prediction Using Time-Series Models

To validate our assumptions made in Section 6.1.1, we use RPS toolkit [10] to fit various time-series models to each time interval stream in our study. Nine models are implemented in the RPS prediction library, which includes three categories. MEAN (long-range mean), LAST (last-value), and $BM(p)$ (mean over "best" window) are widely applied simple models. $AR(p)$ (auto regressive), $MA(p)$ (moving average), $ARMA(p,q)$, and $ARIMA(p,d,q)$ are examples of the Box-Jenkins linear time-series models. The final model, $ARFIMA(p,d,q)$ is a good choice to capture LRD feature and thus can be used to predict self-similarity streams. Detailed introduction about these models can be found in [7, 22].

In this experiment, our time-series intervals are segmented into identical-length blocks, which are used as the training sets for RPS toolkit. Based on the training sets, parameters of each model can be determined. The produced models are used to predict a number of future observations. The predicted values are afterward compared with real observations recorded in the stream to find the absolute errors. We take different sections from a stream and perform each test for multiple times to reduce bias in the results. For inter-session intervals, each time a size of $2,000$ continuous observations are selected to predict the next 100 values. Because the size of intra-session interval stream is generally small, each time we use $1,000$ observations in one intra-session stream to predict the next 10. Our experimental results are given in Table 2, in which the values written in the parentheses of the heading line are the parameters used in models.

In Table 2, the second column records the mean values for all observations in the stream, which are used for comparison with absolute errors to reveal their degrees of significance. Obviously, ARMA, ARIMA and ARFIMA models are very inaccurate in data prediction, which again strongly rules out the existence of self-similarity and LRD. Other models generally produce less errors, which are still comparable with the mean value of the observations. Hence, we conclude from our study that time-series models are not suitable for search engine workloads generation.

6.1.3 Statistical Modeling of Single Streams

It is proved in previous sections that the time-series streams are IID. We take the following steps in our modeling process:

Identify Distribution Family: As the first step, we need to decide what distributions are considered as possible candidates for our modeling process. After studying previous suggestive studies and plotting the CDF (*cumulative distribution function*) of our target streams, four distributions are selected in the subsequent analysis: 1) *hyper-exponential* (which is good at modeling high-variance behavior), 2) *normal* (which can model low variance), 3) *pareto* (suggested to model bursty / heavy-tailed behaviors), and 4) *two-parameter weibull* (flexible to model wide diversity of streams). Although this list is far from complete, the listed distributions are representative in modeling different cases.

Estimate Distribution Parameters: For each type of distribution, a predefined set of parameters needs to be determined. We use the *Maximum Likelihood Estimation* (MLE) method to calculate the parameters for a specific distribution, based on a collection of sampled observations. MLE is robust because every sample observation is used in estimating parameters, which is insensitive to the outliers. To achieve our goal, an open source software, *mle*[11], is used, which supports a rich variety of statistical distribution models. In our experiments, we deliberately sample out inter-session intervals within the standard hour (described in Section 6.1.4) for each client category (robot and user). Meanwhile, we carefully select several typical sessions corresponding to each category and mingle them together, in which the request sequence is preserved according to the time stamps. The purpose is to eliminate the possible bias caused by a specific client. We select 1,000 sessions, which look normal in their statistical behaviors. From each session, 50 successive intra-session intervals are selected and combined, creating a mingled stream of 50,000 observations. We use this stream to model intra-session intervals.

Goodness-of-Fit Test: After obtaining the candidate distributional models, we need to assess which one is the most fit with the original dataset. Kolmogorov-Smirnov (K-S) test, which uses the empirical cumulative distribution function as a tool for testing, is adopted to test the *goodness-of-fit*. To be more specific, K-S test returns the maximum distance between the cumulative distribution function of a model and the empirical data. The smaller is the distance, the better is the goodness-of-fit. Limited by the paper length, CDF plots are not included in the paper. Rather, we give the K-S comparison results in Table 3.

The K-S test results again confirm our expectation that Pareto model does not fit well because there is no LRD or

[11]http://faculty.washington.edu/∼djholman/mle/

Stream	Sample Mean	AR(16)	AR(4)	BM(16)	MA(16)	MEAN	LAST	ARMA (16,16)	ARIMA (16,2,16)	ARFIMA (8,.5,8)
Inter (R)	25	35	33	33	35	33	63	4925	3614	$3.5e+07$
Inter (H)	1.8	1.9	2.0	1.6	1.8	1.9	2.5	1.9	1.8	$4.9e+06$
Intra (R1)	5.3	5.4	3.8	6.7	5.4	3.7	5.4	62	5.5	6342
Intra (R2)	6.3	7.6	6.9	5.3	7.5	6.9	7.5	603	6.2	$8.8e+06$
Intra (R3)	1.6	1.0	1.1	3.1	1.0	1.4	5.5	1340	4.5	825840
Intra (R4)	27	17	16	18	17	16	21	233892	4075	$1.7e+08$
Intra (R5)	24	18	18	14	18	18	16	199126	19000	$2.3e+08$
Intra (H1)	41	95	93	122	116	80	128	193	6598	$1.1e+10$
Intra (H2)	42	23	20	22	39	21	13	507672	3437	$8.3e+08$
Intra (H3)	8.4	78	78	79	77	79	91	7181	87	$2.3e+09$
Intra (H4)	5.7	10	9.9	11	10	10	21	112888	6059	$2.5e+09$
Intra (H5)	12	13	13	11	13	13	13	1274	12	$1.2e+12$

Table 2: Prediction errors using RPS toolkit's time-series models.

Stream	Normal	Pareto	Weibull	Hyper-Exponential
Inter (R)	0.248	0.613	0.140	0.049
Inter (H)	0.334	0.607	0.137	0.099
Intra (R)	0.280	0.662	0.164	0.134
Intra (H)	0.268	0.653	0.143	0.108

Table 3: Goodness-of-fit test results by K-S.

heavy-tail behavior in our streams. It is found that with no exception the hyper-exponential distribution is best in all cases, which in return suggests that a high variance is present in all streams. The final parameters returned by *mle* software are listed in Table 4.

Stream	p	λ_1	λ_2
Inter (R)	0.501	0.410	0.029
Inter (H)	0.876	1.128	0.104
Intra (R)	0.087	0.005	0.028
Intra (H)	0.671	0.013	0.149

Table 4: Hyper-exponential distribution parameters for each stream.

6.1.4 *Effects of Time and Date*

Although the test results in Section 6.1.1 show no strong correlation in the streams we analyzed, the exact *date* and *time* do have an effect on the request intervals. It is detected that the traffic of Mondays is much heavier than Sundays; the traffic in mornings is much heavier than midnights. These impressions indicate that a time-dependent pattern exists which remains unseen in previous tests. The pattern will be revealed when requests are aggregated with larger scales (daily, hourly).

We summarize request access frequencies in day-scale buckets and plot them in Figure 5. It is obvious from Figure 5(a) that periodic patterns exist for user inter-session intervals. Meanwhile, from Figure 5(b), the average number of requests within a session does not change too much over time, especially for users. It is suggested that a client's behavior is basically not influenced by time factor. Thus, it is inferred that what is affected by the time factor is the inter-session interval stream, in other words, the number of sessions at a time.

Figure 6 plots the ACF values for hour-scale session frequency distributions. Unlike the ACF plots shown previously, from the plotted curve we can find obvious correlation in the frequencies, which statistically confirms the existence of periodical frequency patterns.

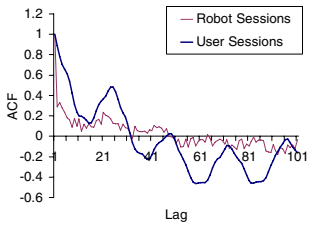
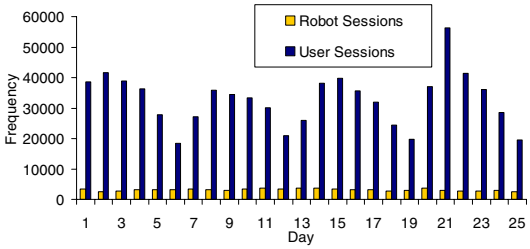


Figure 6: ACF plot for hour-scale frequency distribution.

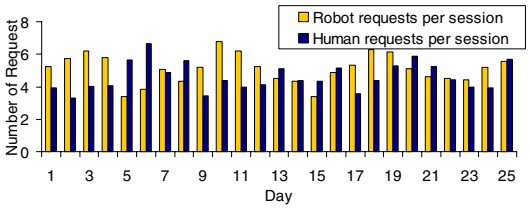
The design goal of SearchGen is to synthesize highly representative search engine access traces. To generate a long-running trace representing a workload for a given time period, SearchGen needs to accommodate date and time effects in the synthetic trace. Observing that date and time mainly affects inter-session intervals, we introduce a *time impact function* into the inter-session interval model. To be specific, our generated inter-session interval is expressed as $T_{inter} = f_t \cdot t_{inter}$, where t_{inter} is generated by the statistical model obtained in Section 6.1.3 and f_t is the impact function, which takes the hour within a week as the input and returns the scaling adjustment. Considering the periodical and discrete nature of the distribution (the pattern proximately repeats weekly), for each hour t of a week we record the overall number of sessions (N_t) and compare it with a *standard hour* ($N_{standard}$), which is specified as the hour from Monday 8 am to 9 am in our implementation. This standard hour is also the hour we choose as the sampling period in modeling inter-session interval t_{inter}. So the impact function can be determined by $f_t = \frac{N_{standard}}{N_t}$. For studies in which the effects of time are not important, the value of f_t can be simply set to be 1 to simplify the model.

6.2 Request Types

Here we analyze request type patterns for users and robots. For a system like CiteSeer, typically, a user would first visit the homepage, type in some query keywords, and browse the result list. If he finds interesting hits, he may click on the link to view the details of a document, and hence decide if he wants to go further to download it. If he realizes



(a) Day-scale session frequency distribution.



(b) Day-scale request per session distribution.

Figure 5: Effects of time study.

that the returned list does not contain relevant documents, he may refine his query and try again or simply turn off the browser to end this session. The aim of our study on client visiting path patterns is to reveal the inherent relationship between requests. This information is not only essential for a synthetic workload, it can also be used to by personalized recommender services and prefetch techniques.

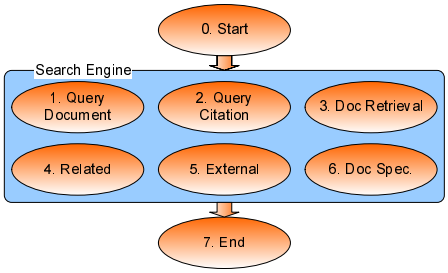


Figure 7: Request types in CiteSeer.

In our study, we statistically summarize log records and use probabilistic models to predict the *next-step* of clients. In [18], the performance of first and second-order Markov models, mixture of first-order Markov models with predefined number of clusters, and a Bayesian first order model with a prior distribution are studied. Considering the facts that most search engine sessions have short session lengths and we have a limited number of service types (Figure 7), a second order Markov model can work well to produce highly accurate request types within a session.

In Section 5, the application model is simplified into six services, which in return contribute the six unique states in the Markov model. In addition, in order to better summarize a client visiting path, two special states are introduced: (1) Start state, which represents the initialization of a new session; and (2) End state, which means the termination of a session. With the assistance of these two new states, we can use the probabilistic model to predict the possibility of the first request type and control the length of a session. Hence, the complete model includes eight distinct states, which are illustrated in Figure 7. In the Markov model, almost all state transitions are bidirectional, which is suggested by the log entries. This is partly because in CiteSeer we can generally find links to all possible services in a page. Two exceptions are the Start and End states, whose related edges are unidirectional. We collect data for robots and users respectively and calculate the probabilities of transitions between different states. Due to the limitation of pages, the detail of the transition probability matrices are not included in the paper. In the model, except for the first request in a session, which is decided solely based on the probability from the Start state, all other request types are determined by the trace of the previous two requests.

6.3 Request Parameters

The actual meanings of parameters for various request types are different. Based on the simplified application model presented in Section 5, there are six services under study. Among the six services, query document and query citation take the queried terms as the parameter; document retrieval, related, and document specification specify target document identifier in their parameters; external URLs service contains the target URL as the parameter which does not need to be forwarded to the search engine and thus is neglected. So the services are grouped into two categories:

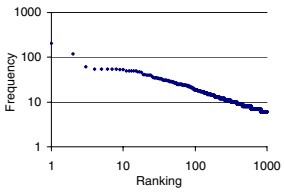
Query-Centric Service, which specifies query terms in the requests.

Document-Centric Service, which specifies document identifiers in the requests.

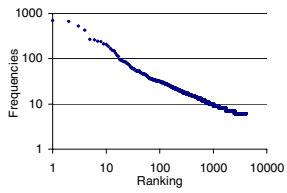
If the actual requested items do not make a difference to the underlying system, there is no need to include the parameters into the workload because they do not influence the performance. However, further analysis into the workflow of search engines drives us to believe it is also required to generate meaningful request parameters. For example, the sizes of inverted lists for different query terms differ dramatically, which influence the response time of the search engine. On the other hand, each stored document has its own data volume and physical location, which can also change the request latency significantly. Request parameter is the factor that cannot be omitted by the workload generator. Basically, we look into the above mentioned two categories and study the popularity distribution for query terms and requested documents respectively. The observed distributions are adopted in SerachGen to provide parameters.

Previous studies [13] suggest that Zipf distributions are common for web accessed items. A specific Zipf distribution follows the form of $f_i = K/i^{\alpha}$, in which f_i stands for the frequency of the i_{th} popular observation. As we look through the logs, we find that there exist a set of *hot* query terms and documents in the system. The requests to these terms and documents are very frequent. Correspondingly, the large portion of the term corpus and document corpus only contributes a small portion of the entire traffic. we rank terms and documents according to their request frequencies and plot the distributions in Figure 8, where Y-axis shows the log-scaled request frequency, while X-axis shows the log-scaled ranking of the request, sorted by their frequencies. The distributions shown in Figure 8 are close to straight lines, suggesting the existence of the Zipf distribution.

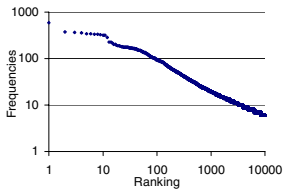
The parameters of the observed Zipf distributions are given



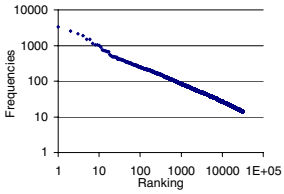
(a) Robot query term distribution.



(b) Robot retrieved document distribution.



(c) User query term distribution.



(d) User retrieved document distribution.

Figure 8: Log-log scale request frequency distribution.

in Table 5, from which it is suggested that users tend to provide more focused requests while robots issue diverse ones.

Stream	α	K
Robot queried terms	0.473	157.40
Robot document identifiers	0.658	941.89
User queried terms	0.603	1288.25
User document identifiers	0.538	3451.44

Table 5: Zipf distribution parameters for requested item frequencies.

7. OUTPUT STREAM SYNTHESIZATION

7.1 Implementation

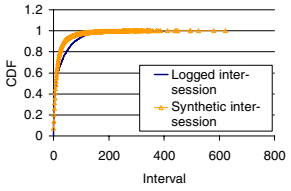
We implemented SearchGen based on our analysis and models obtained earlier. Basically, SearchGen takes a *start time* (in terms of the day of a week and the hour of a day) to create two processes corresponding to users and robots, respectively, using appropriate models and settings. Once the generating processes are initialized, each of them will use the inter-session interval model as well as the time impact function to determine how long it takes to initialize a new thread, which represents a new session. The new thread, once created, chooses from the Markov model its start request. After that, each thread uses the intra-session interval model to predict the thinking time of the client. The Markov model will supply the thread with the selected next step. Based on the supplied request type, corresponding request parameter is extracted from the system dictionary (i.e., query terms and document identifiers collected in CiteSeer), following Zipf distributions. The process repeats until the *end* state is reached. The thread is terminated, meaning the end of an active session. SearchGen is implemented in Java.

7.2 Synthetic Fit Test

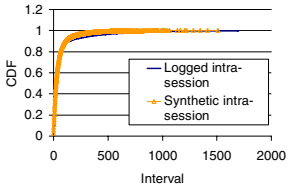
We picked logs of one month to train model parameters which are presented in Section 6. To evaluate the closeness of our prediction, log pieces of the following month are used in the evaluation process. Generally, the actual data is compared with our synthetic trace. As discussed in previous sections, the characteristics we choose in the synthetic workload are those with impact on server performance. If the traces in comparison turn out to be similar, it is shown the trace generator works well in modeling real life search engine traffics. We choose the adjacent month because we are sure there is no obvious system modification in CiteSeer during that period. Therefore, the traffic pattern will not be affected by changes in system settings.

7.2.1 Time Intervals

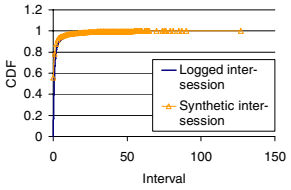
In this experiment, SearchGen is asked to produce 100,000 intervals for each studied time interval stream. The CDF of the logged and synthetic traces are plotted in Figure 9, in which the closeness between two curves indicates the accuracy of the synthetic trace. It can be found that the two curves in each plot are very close to each other, with the largest distance no more than 0.15 for all plots, indicating the synthetic stream simulates well in the time interval distributions. It is noted that the most obvious difference in Figure 9 happens in generating user intra-session intervals, which is caused by the high variance in user think times. To improve accuracy, good sampling and clustering strategies can be applied.



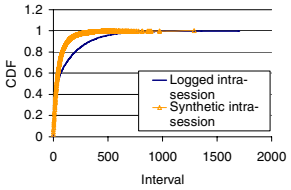
(a) Robot inter-session.



(b) Robot intra-session.



(c) User inter-session.

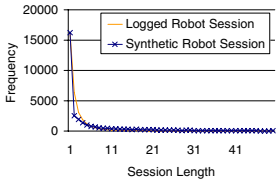


(d) User intra-session.

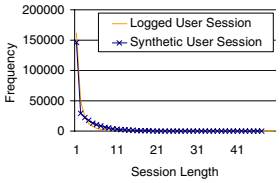
Figure 9: CDF Plots of time interval distributions. X-axis is the time interval, and Y-axis is the CDF.

7.2.2 Request Types

Another important factor, the request type, is evaluated by looking into the distribution of session lengths [18], which is a good indicator of the goodness in predicting web visiting paths. In this experiment, we collect 500,000 sessions from each generated trace as the experiment target. The results are shown in Figure 10. It is shown that generally the synthetic session lengths are close to logged ones.



(a) Robots.

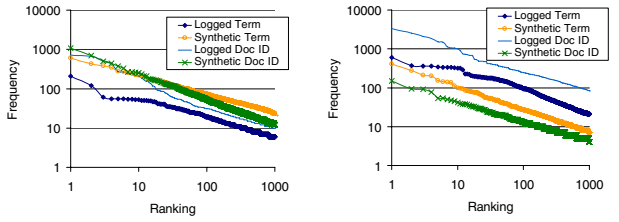


(b) Users.

Figure 10: Session length frequency comparison.

7.2.3 Request Parameters

To validate whether the synthetic workload actually produces Zipf-like request distribution, 50,000 requests are collected from each synthetic trace and plotted in Figure 11, in a log-log scale. To facilitate comparison, we also plot logged parameter frequency distributions. Although the test collection size varies, we can find the corresponding plotted points form two lines that are roughly parallel to each other, suggesting the synthetic trace follows summarized distributions.



(a) Robots. (b) Users.

Figure 11: Log-log scale request parameter frequency distribution comparison.

8. CONCLUSION

Synthesizing representative workloads for particular application domains can be challenging. In this paper, we analyze log traces of a popular scientific literature digital library and search engine - CiteSeer. After investigating the web logs, we realize previous web server traffic models are not suitable for search engines. We define the unique characteristics of search engine traffic based on a decomposition of the entire trace. For each aspect of the characteristics, appropriate methodologies are employed to generate individual streams. The proposed models are synthesized into an output workload.

Synthetic workloads are extremely useful for system performance analysis. Our generated workload can be used by a relevant benchmark to model server-side traffic as well. Additionally, we hypothesize that the methodologies described in this paper can be applied in modeling workloads for other similar speciality systems. SearchGen will be published as a system service of the next generation CiteSeer - CiteSeer$^{\mathcal{X}}$.

Future work will investigate fine-grained semantics of requests. We are interested in how we can use our workload analysis to improve CiteSeer service qualities by implementing workload-based traffic predication and server tuning.

9. ACKNOWLEDGEMENTS

We gratefully acknowledge partial support from Microsoft Research and NSF.

10. REFERENCES

[1] G. Abdulla. *Analysis and modeling of world wide web traffic*. PhD thesis, 1998. Chair-Edward A. Fox.

[2] C. S. Badue, R. Barbosa, P. Golgher, B. Ribeiro-Neto, and N. Ziviani. Distributed processing of conjunctive queries. In *HDIR '05, SIGIR 2005*, 2005.

[3] P. Barford and M. E. Crovella. Generating representative Web workloads for network and server performance evaluation. In *SIGMETRICS '98*, pages 151–160, July 1998.

[4] L. Bent, M. Rabinovich, G. M. Voelker, and Z. Xiao. Characterization of a large web site population with implications for content delivery. In *WWW '04*, pages 522–533, 2004.

[5] J. Beran. *Statistics for Long-Memory Processes*. Chapman & Hall, New York, NY, 1994.

[6] B. Berendt, B. Mobasher, M. Spiliopoulou, and J. Wiltshire. Measuring the accuracy of sessionizers for web usage analysis. In *Proceedings of the Web Mining Workshop at the 1st SIAM International Conference on Data Mining*, Chicago, Illinois, April 2001.

[7] G. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

[8] S. Chaudhuri, P. Ganesan, and V. R. Narasayya. Primitives for workload summarization and implications for SQL. In *VLDB*, pages 730–741, 2003.

[9] M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, Dec. 1997.

[10] P. Dinda and D. O'Hallaron. An extensible toolkit for resource prediction in distributed systems, 1999.

[11] C. L. Giles, K. Bollacker, and S. Lawrence. CiteSeer: An automatic citation indexing system. In *The Third ACM Conference on Digital Libraries*, pages 89–98, Pittsburgh, PA, 1998.

[12] M. E. Gómez and V. Santonja. Analysis of self-similarity in I/O workload using structural modeling. In *MASCOTS*, page 234, 1999.

[13] T. Kelly and J. Mogul. Aliasing on the World Wide Web: Prevalence and Performance Implications. In *WWW'02*, Honolulu, Hawaii, May 2002.

[14] D. E. Knuth. *The Art of Computer Programming*. Four volumes. Addison-Wesley, 1969.

[15] H. Li, I. G. Councill, W.-C. Lee, and C. L. Giles. CiteSeerX: an architecture and web service design for an academic document search engine. In *WWW*, pages 883–884, 2006.

[16] Y. Lu, T. Abdelzaher, C. Lu, and G. Tao. An adaptive control framework for QoS guarantees and its application to differentiated caching services, 2002.

[17] E. Manavoglu, D. Pavlov, and C. L. Giles. Probabilistic user behavior models. In *ICDM '03*, page 203, Washington, DC, USA, 2003.

[18] R. Sen and M. Hansen. Predicting a web user's next access based on log data. *Journal of Computational and Graphical Statistics*, 12:143–155(13), 2003.

[19] R. Simmonds, C. L. Williamson, R. Bradford, M. F. Arlitt, and B. Unger. Web server benchmarking using parallel WAN emulation. In *SIGMETRICS'02*, pages 286–287, 2002.

[20] A. Streit. *Self-Tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids*. PhD thesis, Faculty of Computer Science, Electrical Engineering and Mathematics, University Paderborn, 2003.

[21] P. Tan and V. Kumar. Discovery of web robot sessions based on their navigational patterns. *Data Mining and Knowledge Discovery*, 6:9–35, 2002.

[22] N. Tran and D. A. Reed. ARIMA time series modeling and forecasting for adaptive I/O prefetching. In *Proceedings of the 15th international conference on Supercomputing*, pages 473–485, June 2001.

[23] Y. Wang, M. J. Rutherford, A. Carzaniga, and A. L. Wolf. Weevil: a tool to automate experimentation with distributed systems. Technical Report CU-CS-980-04, Department of Computer Science, University of Colorado, Oct. 2004.

[24] J. Zhang, A. Sivasubramaniam, H. Franke, N. Gautam, Y. Zhang, and S. Nagar. Synthesizing representative I/O workloads for TPC-H. In *HPCA*, pages 142–151, 2004.

[25] S. Zhang, I. Cohen, M. Goldszmidt, J. Symons, and A. Fox. Ensembles of models for automated diagnosis of system performance problems. In *DSN '05: Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 644–653, 2005.