Modeling Performance-Driven Workload Characterization of Web Search Systems

C.S. Badue[1], R. Baeza-Yates[2], B. Ribeiro-Neto[3], A. Ziviani[4], N. Ziviani[5]

[1,3,5]Department of Computer Science
Federal University of Minas Gerais
Belo Horizonte, Brazil
{claudine,berthier,nivio}@dcc.ufmg.br

[4]National Laboratory for
Scientific Computing (LNCC)
Petrópolis, Brazil
ziviani@lncc.br

[3]Google Engineering Belo Horizonte
Belo Horizonte, Brazil
berthier@google.com

[2]Yahoo! Research
Barcelona, Spain & Santiago, Chile
ricardo.baeza@upf.edu
rbaeza@dcc.uchile.cl

ABSTRACT

In this paper we model workloads for a web search system from the performance point of view. We analyze both workload intensity and service demand parameters expressed in the context of web search systems as the distribution of the interarrival times of queries and the per-query execution time, respectively. Our results are derived from experiments in an information retrieval testbed fed with real-world experimental data. Our findings unveil a certain number of unexpected and interesting features. We verify in practice that there is a high variability in both interarrival times of queries reaching a search engine and service times of queries processed in parallel by a cluster of index servers. We also show that this highly variable behavior can be accurately captured by hyperexponential distributions. These results shed light on the usual assumption taken by previous analytical models for web search systems found in the literature that interarrival times and service times are exponentially distributed. We find evidence that the intensity and service demand workloads of a typical web search system present long-range dependence characteristics, leading to self-similar behavior. This finding is important because, in the presence of long-range dependence and self-similarity, exponential-based models tend to underestimate response times as self-similarity leads to increased queueing delays, resulting in significant performance degradation. Based on our findings, we also discuss possible steps toward a generative model for synthetic workloads.

1. INTRODUCTION

Performance evaluation of web search systems become increasingly important given the key role such systems play in Internet usage nowadays. The performance of any system heavily depends on the characteristics of the load it is subject to. Workload characterization is an important component of performance evaluation and prediction in general, being particularly critical for proper capacity planning of web search systems. Nevertheless, previous work on workload characterization for web search systems mainly focuses on the characterization of user search behavior, reporting parameters such as the number of queries within a session, mean number of terms per query, or number of resulting pages a user views [31, 9, 29, 6, 16, 19]. However, a clear understanding of both interarrival times of queries and query execution times is needed for performance evaluation purposes [27, 21].

The workload parameters for performance evaluation can be classified into two categories: workload intensity parameters and workload service demand parameters [22]. The workload intensity parameters provide a measure of the load placed on a system, indicated by the number of units of work that contend for system resources. In the case of web search engines this measure corresponds to the number of incoming queries per second. The workload service demand parameters specify the total amount of service time required by each basic component of the workload on each resource of the system. In a web search system, the service time for a user query is equivalent to the amount of time needed to retrieve the most relevant documents for this query.

When a user query reaches a search engine, the query processing is split into two consecutive major phases [8]. The first phase consists of retrieving documents from the storage devices that contain each query term, executing a conjunction of the sets of these documents, and finally ranking the selected documents using some relevance metric. The second phase consists of taking the top ranked answers of the first phase, typically 10, and generating snippets, title, and

URL information for each of them. For this, the search engine needs to examine the full texts of the top ranked documents. Both phases play important roles in the performance of modern search engines. Nevertheless, as the size of a search engine increases, the processing cost of the second phase remains basically constant, whereas the processing cost of the first phase increases—larger lists of documents have to be read from the disks and processed. Therefore, the performance of the first phase is crucial for maintaining the scalability of modern search engines that deal with an ever-increasing amount of Web documents.

Based on these considerations, we focus on the first phase of the query processing task, i.e., the retrieval of the most relevant documents for a given user query. In our architecture, the whole collection of documents is partitioned among a set of index servers organized in a computational cluster, such that each server stores its own local subcollection. Upon the arrival of an user query, each index server performs the retrieving task for the query only on its partition of the document collection. A broker is responsible for merging the partial ranked answers from the servers to produce the final ranked answer. In the case of web search systems, the workload intensity parameters are represented by the distribution of interarrival times of queries at the broker, and the workload service demand parameters are expressed by the distribution of service times of queries processed in parallel by the cluster of index servers.

In this paper, we characterize the workload intensity and service demand parameters. Our results are derived from experiments in an information retrieval testbed fed with a log of a real-world search engine. The main experimental results on workload characterization are as follows. First, they shed light on an assumption taken as valid by previous analytical models for web search systems [11] that interarrival times and service times are governed by exponential distributions. In contrast, we verify in practice that there is a high variability in both interarrival times of queries reaching a search engine and service times of queries processed in parallel by a cluster of index servers. This high variability in service times of queries is mainly due to the presence of a certain level of imbalance in per-query execution times among homogeneous index servers participating in the parallel processing [3]. We show that this highly variable behavior of the intensity and service demand workloads is very well captured by hyperexponential distributions, thus allowing the definition of some steps toward a generative model for synthetic workloads of web search systems. Second, our results confirm for web search systems the widespread evidence that interarrival processes and service processes in internet-related systems exhibit high variability [12, 1].

This paper is organized as follows. In Section 2, we present an architecture for web search systems. In Section 3, we present a characterization of the test collection used in the experiments. In Section 4, we characterize the workload intensity parameters and provide best fits for them. In Section 5, we characterize the workload service demand parameters and provide best fits for them as well. In Section 6, we discuss the applicability of our results on the definition of a generative model for typical synthetic workloads of web search systems. In Section 7, we present our conclusions and directions for future work.

2. ARCHITECTURE

In this section we present an architecture for web search systems and a description of our cluster testbed.

2.1 Cluster of Index Servers

Search engines typically adopt a computational cluster as a platform for query processing [8, 28]. Usually, this cluster is composed of a single broker and p index servers. The whole collection of documents is partitioned among the index servers, such that each server stores its own local subcollection. Figure 1 illustrates this architecture for a typical search engine. The broker accepts queries from a user and forwards them to the index servers, triggering the parallel query processing. Each index server searches its own local subcollection and produces a partial ranked answer. These partial ranked answers are collected by the broker and combined through an in-memory merging operation. The final list of ranked documents is then sent back to the user.

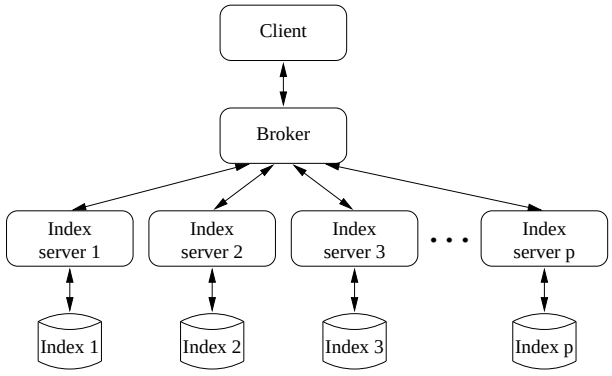

Figure 1: Architecture of a typical search engine.

2.2 Index Organization

An inverted index is adopted as the indexing structure for each subcollection [7, 35]. Inverted files are useful because they can be searched based mostly on the set of distinct words in all documents of the collection. They are simple data structures that perform well when the pattern to be searched for is formed by conjunctions and disjunctions of words [7, 35].

The structure of our inverted indexes is composed of a *vocabulary* and a *set of inverted lists*. The vocabulary is the set of all unique terms (words) in the document subcollection. Each term in the vocabulary is associated with an inverted list that contains an entry for each document in which the term occurs. Each entry is composed of a document identifier and the within-document frequency $f_{t,d}$ representing the number of occurrences of term t within the document d. The inverted lists are sorted by decreasing within-document frequencies.

The documents of the whole collection are uniformly distributed among index servers. Let n be the size of the whole collection and p the number of index servers in the cluster. The size of any local inverted file is $O(n/p)$. This type of

index organization, hereafter referred to as a *local index organization* [25, 4], is currently the de facto standard in all major search engines.

2.3 Parallel Query Processing

We use the standard vector space model [30] to rank the selected documents. In this model, queries and documents are represented as weighted vectors in a t-dimensional space, where t is the number of terms in the vocabulary of the collection. Each pair term-document is weighted by the frequency $f_{t,d}$ of term t in document d, the term frequency *tf* and the inverse document frequency (*idf*) of the term t among the documents in the whole collection. The rank of a document with regard to a user query is computed as the cosine of the angle between the query and document vectors. Using the *idf* weight implies global knowledge about the whole collection to be available at the index servers. This could be accomplished if servers exchange their local *idf* factors after the local index generation phase. Each index server may then derive the global *idf* factor from the set of local *idf* factors [25].

In our experiments, a client machine submits queries to the broker. This broker then broadcasts each query to all index servers. Once each index server receives a query, it retrieves the full inverted lists relative to the query terms, intersects these lists to produce the set of documents that contains all query terms (i.e., the conjunction of the query terms), computes a relevance score for each document, and sorts them by decreasing score—this results in a partial ranked answer to be sent by each index server to the broker. Query terms are processed by decreasing *idf*, i.e., by increasing order of the number n_t of documents in the whole collection containing the term t, thus leading to a significantly more efficient conjunction of their inverted lists. As soon as the ranking is computed, the top ranked documents selected at each index server are transferred to the broker machine. The broker is then responsible for combining the partial ranked answers received from the index servers through an in-memory merging operation. The final list of top ranked documents is then sent back to the client machine.

The broker is not a bottleneck in this architecture of local document partitioning [5]. Our experimental results show that the average execution time per query at the broker is quite small, on the order of 10^{-5} seconds in our cluster with 8 index servers. There are two fundamental reasons for this. First, broker's operation is fully carried out using main memory. Second, all the tasks the broker executes are simple tasks that do not take much CPU time. It should be noted that the broker does not have to make ranking computations and does not have to execute algebraic operations, other than comparing document identifiers.

2.4 Cluster Testbed

For the experiments, we use a cluster of 8 identical index servers. In our setup, each index server is a Pentium IV with a 2.4 gigahertz processor, 1 gigabytes of main memory and a ATA IDE disk of 120 gigabytes. The broker is an ATHLON XP with a 2.2 gigahertz processor and 1 gigabytes of main memory. The client machine, responsible for managing the stream of user queries, is an AMD-K6-2 with a 500 megahertz processor and 256 megabytes of main memory. All of them run the Debian Linux operating system version 2.6.

3. TEST COLLECTION

The test collection, referred to as WBR03, is composed of 10 million Web pages collected by the TodoBR[1] [32] search engine from the Brazilian Web in 2003. The inverted index for the whole collection occupies roughly 12 gigabytes, and the collection vocabulary is composed of $3,541,678$ terms. The query set used in our tests is composed of 100 thousand queries extracted from a partial log of queries submitted to the TodoBR search engine in September 2003. There is a total of $27,396$ unique queries and $17,126$ unique terms in the query set.

The distribution of terms in queries follows a Zipf distribution, as shown in Figure 2. The plot shows the normalized frequency of terms in queries. The x-axis shows the resulting rank of each term when these are sorted in decreasing order of occurrence in queries. Therefore, the frequency in queries that are expected for the x most frequent term is given by

$$f(x) = O(x^{-b}), \qquad b > 0. \tag{1}$$

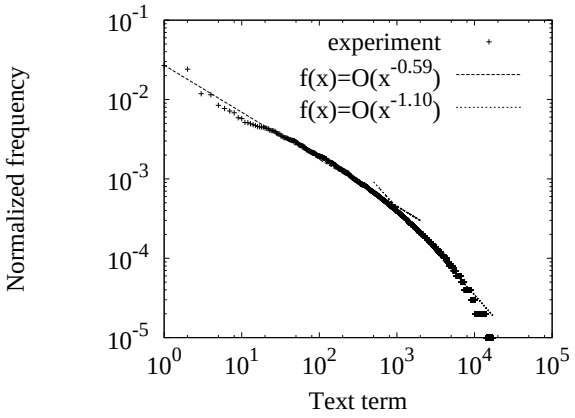

Figure 2: Frequency of terms in queries.

We identify two different regions in the distribution of terms in queries as shown in Figure 2. The value of parameter b is 0.59 for the first region with $x \leq 1000$, and 1.10 for the second region with $x > 1000$.

We analyze the distribution of the sizes of queries in our query log, as shown in Figure 3. The plot shows the probability mass function(PMF)[2] of the sizes of queries in our query log. The size of a query is given by the number of terms that appears in this query. The best fit we found

[1]TodoBR is a trademark of Akwan Information Technologies, which was acquired by Google in July 2005.

[2]For discrete random variables, such as the size of queries, we use a probability mass function (PMF). For continuous random variables described later, such as the service time of queries, we use a probability density function (PDF).

for the PMF of the sizes of queries was a decay exponential function of query size given by $O(e^{-0.88x})$, $x \geq 2$. The average size of queries in our log is 2.09 terms.

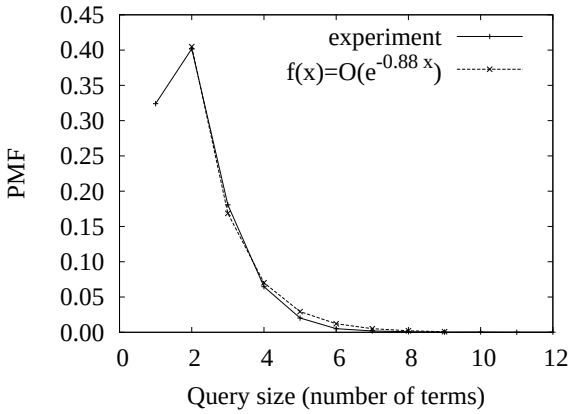

Figure 3: PMF of the sizes of queries.

We further evaluate the relationship between terms and the average size of queries in our query log, as shown in Figure 4. The x-axis shows the resulting rank of each term when these are sorted in decreasing order of occurrence in queries. We observe that as the frequency of a term in the query log decreases, the average size of queries that contain such term increases.

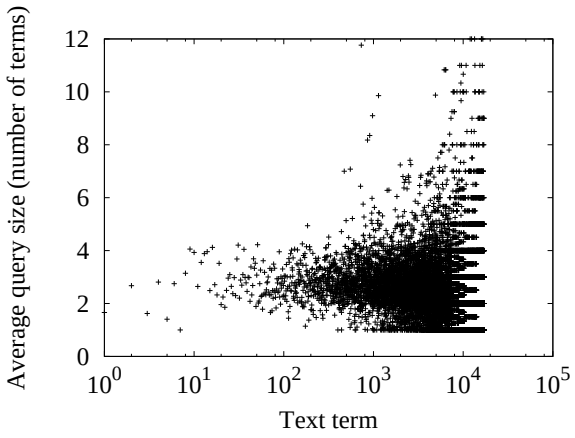

Figure 4: Relationship between terms and average query size.

It is also important to investigate the relationship between the frequency of terms in queries and the frequency of terms in documents. Figure 5 shows the normalized frequency of terms in the document collection as a function of the normalized frequency of terms in the query collection. Comparing the normalized frequency of terms in documents and in queries, we observe that—even if dealing with rare query terms—it is likely that query terms are mentioned in a large number of documents. This is important because this indicates that such a query set consistently generates a significant query processing load in our system.

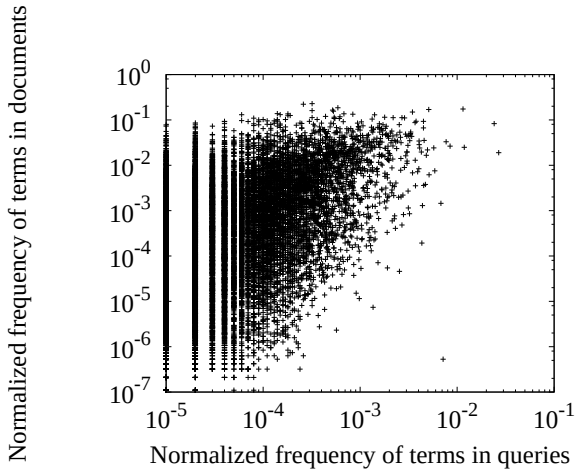

Figure 5: Relationship between the frequency of terms in queries and in documents.

4. CHARACTERIZING THE WORKLOAD INTENSITY PARAMETERS

In this section, we evaluate the workload intensity parameters indicated by the distribution of interarrival times of queries that reach the cluster of index servers. If queries arrive at times $t_1, t_2, \ldots, t_i$, the random variables $\tau_i = t_i - t_{i-1}$ are called the interarrival times of queries [15]. We first compute the coefficient of variation of the interarrival times observed in our test collection. The coefficient of variation of a random variable is defined as the ratio of its standard deviation by its mean. It can be thought of as a measure of the deviation of the distribution found in the empirical data from the exponential distribution, whose coefficient of variation is equal to one. In our test collection, the coefficient of variation of interarrival times of queries is equal to 2.64, suggesting that interarrival times of queries might be well fitted by an hyperexponential distribution, which is characterized by a coefficient of variation larger than one [33].

The hyperexponential distribution is an special case of the phase-type (PH) distribution [26] that results from the combination of a certain number of exponential distributions. A PH distribution is fully represented by a vector τ and a matrix $\mathbf{T}$, related as in the following. The cumulative distribution function (CDF) of a PH distribution is given by:

$$F(x) = 1 - \tau e^{\mathbf{T}x}\mathbf{e}, \tag{2}$$

where $\mathbf{e}$ is a column vector of ones with the appropriate dimension. The probability density function (PDF) of a PH distribution is equal to:

$$f(x) = \tau e^{\mathbf{T}x}(-\mathbf{T} \cdot \mathbf{e}), \tag{3}$$

and the n^{th} moment is given by:

$$m_n = (-1)^n \cdot n! \tau \mathbf{T}^{-n} \mathbf{e}. \quad (4)$$

Observe that in the PDF and CDF definitions of a PH distribution, the matrix $\mathbf{T}$ appears as an exponential coefficient. As such, PH distributions are called matrix-exponential distributions. The term $e^{\mathbf{T}x}$ is defined as

$$e^{\mathbf{T}x} = \sum_{n \geq 0} \frac{1}{n!} (\mathbf{T}x)^n. \quad (5)$$

The cases of PH distributions that are used in practice typically have the majority of the elements in τ and $\mathbf{T}$ equal to zero. For an hyperexponential distribution, $2n$ parameters are needed to define this distribution, where n is the number of combined exponential distributions that compose the hyperexponential. Indeed, for an hyperexponential distribution—a special case of PH distributions—, the definition of τ and $\mathbf{T}$, assuming $n = 3$, is given by:

$$\tau = [p_1, p_2, p_3], \text{ and} \quad (6)$$

$$\mathbf{T} = \begin{bmatrix} -\lambda_1 & 0 & 0 \\ 0 & -\lambda_2 & 0 \\ 0 & 0 & -\lambda_3 \end{bmatrix}. \quad (7)$$

By applying these definitions to Equation 3, we can derive the PDF of an hyperexponential distribution H_n with n phases as

$$f(x) = f_{H_n} = \sum_{i=1}^{n} p_i \lambda_i e^{-\lambda_i x}, \quad (8)$$

i.e., the weighted combination of n exponentials, each characterized by λ_i, where p_i represents the relative weight of exponential i in this combination ($\sum_i p_i = 1$). We use the EMpht software package [23] to estimate the parameters of hyperexponential distributions in this paper. EMpht implements the estimation maximization (EM) algorithm presented in [2] to do so.

Figure 6 shows the empirical distribution of interarrival times of queries, which is fitted into an exponential distribution and into an hyperexponential distribution H_3 composed of 3 exponential phases. The complexity and precision of the fitting procedure into an hyperexponential distribution increases with the number of exponential phases that composes this hyperexponential distribution. We fitted our empirical data into three distinct hyperexponential distributions H_2, H_3 and H_4 composed of 2, 3 and 4 exponential phases, respectively. To evaluate the accuracy of the fitting, we use the Kolmogorov-Smirnov goodness-of-fit test [13] that verifies that the maximum ordinate difference between the distribution of interarrival times and each of the fitted hyperexponential distributions H_2, H_3 and H_4 is 2.29%, 1.37% and 1.37%, respectively. We observe that the fit H_3 is more accurate than the fit H_2. Nevertheless, the more complex fit H_4 does not gain in precision over the fit H_3. Therefore, we select H_3 as the best fit for our data because it achieves the best balance between accuracy and complexity in number of exponential phases. The parameters τ_a and $\mathbf{T}_a$ of the best fit H_3 for the distribution of interarrival times of queries are:

$$\tau_a = [0.18, 0.80, 0.02], \text{ and}$$

$$\mathbf{T}_a = \begin{bmatrix} -1.24 & 0 & 0 \\ 0 & -0.90 & 0 \\ 0 & 0 & -0.04 \end{bmatrix}.$$

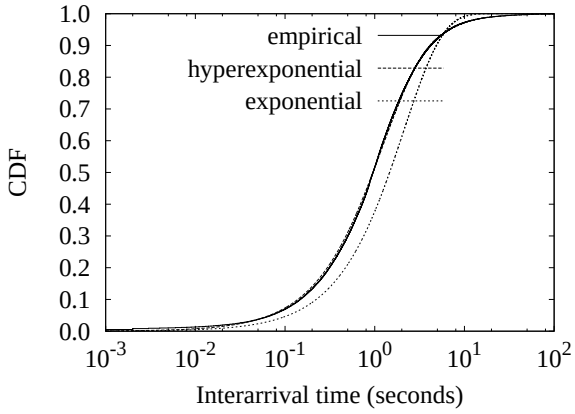

Figure 6: Distribution of interarrival times.

We observe that the hyperexponential distribution captures the highly variable behavior and does a good job in matching the distribution of interarrival times. Conversely, we observe that the exponential fitting is poor and is not capable of capturing the high variability present in interarrival times. This finding contradicts the basic assumption of exponentially distributed interarrival times adopted by previous queueing models for web search systems [11]. The Kolmogorov-Smirnov goodness-of-fit test verifies that the maximum ordinate difference between the distribution of interarrival times and the fitted exponential distribution is 15.40%, while this difference for the best fit hyperexponential distribution H_3 is only 1.37%.

The high variability found in interarrival times of queries in the test collection may have significant impact on the performance of a web search system. The previous assumptions of exponentially distributed interarrival times of queries suppose memoryless behavior. Nevertheless, we found evidence that the workload intensity of a typical web search system presents long-range dependence (LRD) [18], leading to self-similar behavior. The predominant way to quantify LRD

is through the Hurst parameter in the range $0.5 < H < 1$. The value 0.5 is the boundary between long and short range dependence. The effect of low range dependence is higher for H close to one. In our experimental data, interarrival times of queries presented $H = 0.88$. A LRD workload implies bursty behavior in comparison with loads of a workload characterized by a Poisson process, i.e., with exponentially distributed interarrival times. For a LRD workload, this bursty behavior usually presents a certain level of periodicity. Also, LRD characterizes processes that keep statistically significant correlations across large time scales, while a process is said to be self-similar if its behavior is roughly the same across different spatial or time scales. We evaluate the distribution of interarrival times for the first $10,000$ queries of our query log, and verify that its shape is similar to that of all $100,000$ queries of our log. Indeed, this similarity on distribution shapes across different scales is an indication of the self-similar behavior.

Our findings thus suggest that typical workload intensity from a real-world web search system also presents LRD and self-similarity characteristics as it has already been shown for LAN [20] and WAN [24] traffic, as well as for WWW traffic [12]. This finding is important because, in the presence of LRD and self-similarity, exponential-based models tend to underestimate response times as self-similarity leads to increased queueing delays, resulting in significant performance degradation [14].

5. CHARACTERIZING THE WORKLOAD SERVICE DEMAND PARAMETERS

We now characterize the workload service demand parameters expressed by the distribution of service times of queries processed in parallel by the cluster of index servers. In our architecture, characterized by a local partitioning of the document collection, the per-query service time is determined by the maximum execution time among index servers participating in the parallel processing of this particular query.

Our experimental results show that the coefficient of variation of the service times of queries in our cluster with 8 index servers is equal to 1.39. This high variability in service times of queries is mainly due to the presence of a certain level of imbalance in per-query execution times among homogeneous index servers participating in the parallel processing. The primary source of the per-query imbalance is the heterogeneous use of disk caching among the homogeneous index servers, as pointed out in [3]. Thus, the coefficient of variation indicates the hyperexponential distribution as an appropriate kind of distribution for fitting the service times of queries.

Figure 7 shows the distribution of service times of queries fitted into an hyperexponential distribution H_3 composed of 3 exponential distributions. Similarly to the case of interarrival times, we observe that the fitting provided by the hyperexponential distribution with 3 phases is very accurate, besides achieving the best balance between accuracy and complexity when compared to those with 2 and 4 phases. The Kolmogorov-Smirnov goodness-of-fit test verifies that the maximum ordinate difference between the distribution of service times and each of the fitted hyperexponential distributions H_2, H_3 and H_4 is 3.21%, 2.29% and 2.29%, respectively. In contrast, the goodness-of-fit test verifies that the difference between the distribution of service times and the fitted exponential distribution is higher and equal to 20.43%. The parameters τ_s and $\mathbf{T}_s$ of this best fit H_3 distribution for service times in our experiments are:

$$\tau_s = [0.20, 0.14, 0.65], \text{ and}$$

$$\mathbf{T}_s = \begin{bmatrix} -568.48 & 0 & 0 \\ 0 & -4359.17 & 0 \\ 0 & 0 & -49.68 \end{bmatrix}.$$

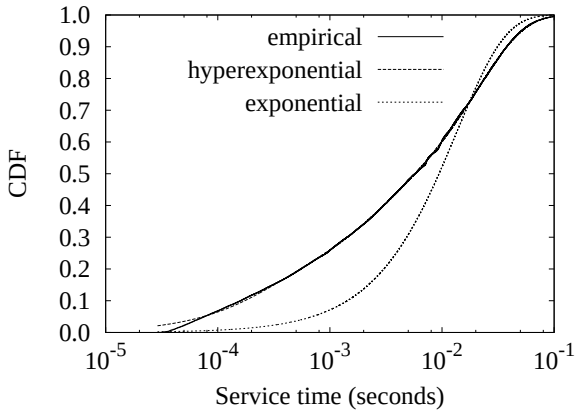

Figure 7: Distribution of per-query service times.

Likewise the distribution of interarrival times, the distribution of per-query service times also presents LRD and self-similarity characteristics. In the case of the considered distribution of per-query service times, the Hurst parameter is $H = 0.74$. This finding again suggests that a exponentially-based model for per-query service times may underestimate the performance of the system, leading to unrealistic results, as the prediction without taking into account the high variability in service times might underestimate queueing delay and, as a consequence, response time of the system as a whole. Thus, an hiperexponential distribution is more appropriate here.

6. TOWARD A GENERATIVE MODEL FOR SEARCH SYSTEM WORKLOADS

In this section we discuss the applicability of our findings in terms of workload characterization to generate synthetic workloads of query traces. This model is not meant to be unique, but an illustration on how useful our results may be to the generation of such synthetic workloads. Basically, we discuss choices on which variables can be used to define a *generative model* for a synthetic workload, in a similar way as Veloso et al. [34] do when characterizing workloads for live streaming media.

The use of a generative model for a synthetic query trace based on our workload characterization may be as follows.

To generate a synthetic query log, we must determine *when* these queries arrive, *how many* terms each query contains, and *how frequent* each query term is employed by users. To determine *when* a query arrives, one may use the hyperexponential distribution H_3 that characterizes the interarrival times of queries as shown in Figure 6 with the parameters τ_a and $\mathbf{T}_a$ of Section 4. To determine *how many* terms should be associated with each query, one may use the decay exponential function of Figure 3. Finally, to determine *how frequent* users employ each term in queries, one may apply the Zipf distribution of Figure 2. We assume here that the probability of a term occurring in a query is proportional to that term's frequency in the query log, similar to the usual assumption taken by previous query models in [17, 10].

Note that this simple model assumes that the number of terms associated with a query is independent of the term. Nevertheless, we observe in our query log that rare terms are associated with queries of varied sizes, while popular terms usually appears in short queries, as shown in Figure 4. Also, this model disregards the usage correlation of words. An improved algorithm might be to choose the first term using Figure 2, and then choose the next terms using a joint-distribution of words in queries that have that first term. This is still an approximation, but considering that most queries have one or two terms, the error would not be that large. The main drawback of this improvement is the need to keep the joint-distribution of all the words in the query log.

7. CONCLUSIONS

In this paper we characterize performance-driven workloads for a web search system by analyzing workload intensity and service demand parameters based on experiments using real world data. Our findings lead to a certain number of unexpected and interesting features. We verify in practice that there is a high variability in both interarrival times of queries reaching a search engine and service times of queries processed in parallel by a cluster of index servers. We also show that this highly variable behavior can be accurately captured by hyperexponential distributions. These results shed light on the usual assumption taken by previous analytical models for web search systems that interarrival times and service times are exponentially distributed.

Further, we find evidence that the intensity and service demand workloads of a typical web search system present long-range dependence (LRD) characteristics, leading to self-similar behavior. This finding is important because, in the presence of LRD and self-similarity, exponential-based models tend to underestimate response times as self-similarity leads to increased queueing delays, resulting in significant performance degradation. As future work, we intend to investigate analytical models for performance prediction and capacity planning of web search systems, while taking into account the high variability and LRD behavior found in the intensity and service demand workloads.

8. REFERENCES

[1] M. Arlitt and T. Jin. Workload characterization of the 1998 world cup web site. Technical report, HP Labs, Hewlett Packard, Palo Alto, CA, September 1999.

[2] S. Asmussen, O. Nerman, and M. Olsson. Fitting phase-type distributions via the EM algorithm. *Scandinavian Journal of Statistics*, 23:419–441, 1996.

[3] C. Badue, R. Baeza-Yates, B. Ribeiro-Neto, A. Ziviani, and N. Ziviani. Analyzing imbalance among homogeneous index servers in a web search system. *Information Processing and Management*, Submitted to.

[4] C. S. Badue, R. Baeza-Yates, B. Ribeiro-Neto, and N. Ziviani. Distributed query processing using partitioned inverted files. In *Proceedings of the Eighth String Processing and Information Retrieval Symposium (SPIRE'2001)*, pages 10–20, Laguna de San Rafael, Chile, 2001. IEEE Computer Society.

[5] C. S. Badue, R. Barbosa, P. Golgher, B. Ribeiro-Neto, and N. Ziviani. Basic issues on the processing of web queries. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Reseach and Development in Information Retrieval (SIGIR'2005)*, pages 577–578, Salvador, Bahia, Brazil, 2005.

[6] R. Baeza-Yates, C. Hurtado, M. Mendoza, and G. Dupret. Modeling user search behavior. In *3rd Latin American Web Congress (LAWEB'2005)*, pages 242–251. IEEE CS Press, October 2005.

[7] R. Baeza-Yates and B. Ribeiro-Neto, editors. *Modern Information Retrieval*. ACM Press New York, Addison Wesley, 1999.

[8] L. A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, 23(2):22–28, 2003.

[9] S. Beitzel, E. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'2004)*, pages 321–328, New York, NY, USA, 2004. ACM Press.

[10] F. Cacheda, V. Plachouras, and I. Ounis. A case study of distributed information retrieval architectures to index one terabyte of text. *Information Processing and Management*, 41(5):1141–1161, 2004.

[11] A. Chowdhury and G. Pass. Operational requirements for scalable search systems. In *Proceedings of the Twelfth international Conference on Information and Knowledge Management (CIKM'2003)*, pages 435–442, New Orleans, LA, USA, 2003.

[12] 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, December 1997.

[13] R. D'Agostino and M. Stephens. *Goodness-Of-Fit Techniques*. Marcel Dekker Inc., 1986.

[14] A. Erramilli, O. Narayan, and W. Willinger. Experimental queueing analysis with long-range dependence packet traffic. *IEEE/ACM Transactions on Networking*, 4(2):209–223, April 1996.

[15] R. Jain. *The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, Inc., 1991.

[16] B. Jansen and A. Spink. An analysis of web searching by european alltheweb.com users. *Information Processing and Management*, 41(2):361–381, 2005.

[17] B. S. Jeong and E. Omiecinski. Inverted file partitioning schemes in multiple disk systems. *IEEE Transactions on Parallel and Distributed Systems*, 6(2):142–153, 1995.

[18] T. Karagiannis, M. Molle, and M. Faloutsos. Long-range dependence: Ten years of Internet traffic modeling. *IEEE Internet Computing*, 8(5):57–64, September 2004.

[19] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *Fourteenth International World Wide Web Conference (WWW'2005)*, pages 391–400, Chiba, Japan, 2005. ACM Press.

[20] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic. *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[21] H. Li, D. Groep, and L. Wolters. Workload characteristics of a multi-cluster supercomputer. In *10th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'2004)*, Lecture Notes on Computer Science, New York, NY, USA, 2004.

[22] D. Menascé, V. Almeida, and L. Dowdy. *Performance by Design: Computer Capacity Planning by Example*. Prentice Hall, 2004.

[23] M. Olsson. *The EMpht-Programme*. Department of Mathematics, Chalmers University of Technology, and Gotebörg University, June 1998.

[24] V. Paxson and S. Floyd. Wide area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[25] B. A. Ribeiro-Neto and R. A. Barbosa. Query performance for tightly coupled distributed digital libraries. In *Proceedings of the Third ACM Conference on Digital Libraries (JCDL'1998)*, pages 182–190, 1998.

[26] A. Riska, V. Diev, and E. Smirni. An EM-based technique for approximating long-tailed data sets with PH distributions. *Performance Evaluation Journal*, 55(1-2):147–164, 2004.

[27] A. Riska, M. Squillante, S. Yu, Z. Liu, and L. Zhang. Matrix-analytic analysis of a MAP/PH/1 queue fitted to web server data. In G. Latouche and P. Taylor, editors, *Matrix-Analytic Methods; Theory and Applications*, pages 333–356. World Scientific, 2002.

[28] K. M. Risvik, Y. Aasheim, and M. Lidal. Multi-tier architecture for web search engines. In *Proceedings of the First Latin American Web Congress (LAWEB'2003)*, pages 132–143, Santiago, Chile, 2003.

[29] D. Rose and D. Levinson. Understanding user goals in web search. In *Thirteenth International World Wide Web Conference (WWW'2004)*, pages 13–19, New York, NY, USA, 2004. ACM Press.

[30] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[31] C. Silverstein, M. Henzinger, H. Marais, and M. Moriez. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[32] TodoBR. http://www.todobr.com.br, 2003.

[33] K. Trivedi. *Probability and Statistics with Reliability, Queueing, and Computer Science Applications*. John Wiley and Sons, 2001.

[34] E. Veloso, V. Almeida, W. Meira, A. Bestravos, and S. Jin. A hierarchical characterization of a live streaming media workload. *IEEE/ACM Transactions on Networking*, 14(1):133–146, February 2006.

[35] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, Inc., second edition, 1999.