A General Methodology for Characterizing Access Patterns and Analyzing Web Server Performance

Arun K. Iyengar, Edward A. MacNair, Mark S. Squillante, Li Zhang
IBM Research Division
Thomas J. Watson Research Center
Yorktown Heights, NY 10598
{aruni,macnair,mss,zhangl}@watson.ibm.com

Abstract

We develop a general methodology for characterizing Web server access patterns based on a spectral analysis of finite collections of observed data from real systems. Our approach is used together with the access logs from the IBM Web site for the 1996 Olympic Games to demonstrate some of its advantages over previous methods and to analyze certain aspects of large-scale Web server performance.

1. Introduction

The World Wide Web is growing at an extremely rapid pace. It continues to play an ever increasing and important role in our daily life. A critical issue for continued and successful growth concerns the performance of Web servers, which must provide reliable, scalable and efficient access to Internet services.

A significant body of work in the research literature has focused on various aspects of Web server performance; e.g., see [7, 8, 10] and the references therein. Crovella and Bestavros have found evidence that Web traffic is self-similar [6]. Arlitt and Williamson identified ten workload invariants by analyzing logs from six different Web servers [2]. Barford and Crovella have developed a tool that generates workloads based on empirical measurements of server file size distribution, request size distribution, relative file popularity, embedded file references, temporal locality of reference, and idle periods of individual users [3].

A detailed mathematical analysis of Web server access patterns is fundamental to gaining additional key insights and realizing the additional improvements in Web server caching, prefetching and overall performance that is required for providing reliable, scalable and efficient access to Internet services. This is especially the case for large-scale, heavy-traffic Web server environments, such as the the IBM Web site for the 1996 Olympic Games. SPECweb96 [15] and WebStone [14] are two commonly used Web benchmarks. WebStone tries to measure the maximum request rate that a Web server can sustain by issuing as many requests as possible from synchronous clients. WebStone merely stress tests the system, and it is not capable of characterizing the burstiness, trends and interdependencies encountered in the access patterns of real Web sites such as those found at the Olympic Games. Although SPECWeb96 introduces delays between client requests, interrequest times are not obtained by analyzing and characterizing real Web site data. Hence, one objective of this paper is to develop a general methodology for characterizing and analyzing Web server access patterns. This then provides the basis for a class of benchmarks, called *Flintstone*, that is more accurate than WebStone or SPECweb96 for the large-scale Web server environments of interest. Our methodology can also be applied to different Web access logs to construct realistic benchmarks and workloads that are representative of these different environments. In addition to providing more effective benchmarks for evaluating Web server performance, these methods can be exploited to gain a better understanding of Web server access patterns for the development of more effective caching and prefetching algorithms.

Our methodology for representing the access patterns of Web server requests is based on constructing a set of general stochastic processes that captures the behavior found in real systems. In particular, we view collections of observed data from a Web server as realizations of a set of underlying time-varying stochastic processes, and we exploit spectral analysis to obtain the key properties of this set of time-series processes from the collections of data. This includes decomposing the spectral analysis into piecewise time-series processes, and the use of statistical methods to isolate and determine the trends, interdependencies and noise of each time-series process. By combining this set of processes in

a general manner, we obtain a methodology for characterizing Web server access patterns that is more general and flexible than TES processes [12]. Moreover, we believe that heavy-tailed distributions can be accommodated in the set of time-series processes of our methodology. This is in addition to addressing the trends, interdependencies and noise of nonstationary processes.

Another objective of this paper is to apply our Flintstone methodology to the access logs from the IBM Web site for the 1996 Olympic Games to further examine certain Web server performance issues. Our mathematical representation of the access patterns makes it possible to scale the Web server traffic in a general and flexible manner so that Web server performance can be examined at all traffic intensities of interest, which cannot be done correctly without such a detailed characterization of the trends, interdependencies and noise in the access patterns. A comparison between Flintstone and approaches that have been previously used in the research literature demonstrates some of the important advantages of our methodology, with respect to characterizing the traffic patterns, scaling these patterns to consider different server loads, and the impact of such issues on certain aspects of Web server performance.

The remainder of this paper is organized as follows. We first present our methodology for characterizing Web server access patterns. Section 3 then describes our Web server system model and analysis. A small representative sample of the results of applying our methodology to the Olympic access logs together with the corresponding Web server performance measures are presented in Section 4. Our concluding remarks are then provided in Section 5. Throughout this paper we use $\mathbb{N}$ and $\mathbb{R}^+$ to denote the set of nonnegative integers and positive real numbers, respectively.

2. Web Server Access Model and Analysis

A general methodology for accurately representing the access patterns of Web server environments must capture the trends, interdependencies and random noise of the request patterns for the system of interest. This methodology must further capture the dynamic changes in these trends, interdependencies and noise that often occur in Web server environments. Our approach achieves these objectives by constructing a set of time-series processes based on a spectral analysis of finite collections of observed data from real Web server systems. By decomposing the spectral analysis into piecewise time-series processes and combining these processes in a general manner, we obtain a general methodology for characterizing Web server access patterns.

Consider collections of time-varying request observations from the Web server environment of interest such that each observation is indexed by its time parameter t. We partition this time series of requests into *M phases* of access pattern behavior, each of which is a time-series process $Z_m = \{Z_{m,t}\,;\, t \in \mathbb{N}\}$ where $Z_{m,t}$ is a random variable corresponding to the t^{th} request of the m^{th} time-series phase, $1 \leq m \leq M$. Within this framework, $Z_{m,t}$ can be interpreted as the time epoch of the t^{th} request, or the t^{th} interrequest time, or the number of requests at (discrete) time t, all with respect to the time-series process Z_m. The Web server request patterns follow a particular time-series process Z_i for some period of time, at which point there is a transition to another time-series process Z_j (where we allow $i = j$). When the requests are following the time-series process Z_m, we say that the Web server access model is in *state* m. The duration of time that the system spends in each of these states is governed by a set of stochastic processes $L_{i,j} = \{L_{i,j,n}\,;\, n \in \mathbb{N}\}$ in the manner described below.

Consider the case where the Web server access model makes a transition from state i to state j at time s for the n^{th} time. To clarify the exposition, and without loss of generality, we assume that $Z_{j,t}$ represents the time between the t^{th} and $t-1^{\text{st}}$ request of the time-series process Z_j (where $Z_{j,0}$ has a proper initial value). Let $N_j(s) \in \mathbb{N}$ denote the number of requests that the system has taken from Z_j at time s, and define $S_{j,\ell} \equiv \sum_{h=1}^{\ell} Z_{j,h}$. The Web server request patterns then follow the time-series process Z_j for the period of time given by $L_{i,j,n} + R_{i,j,n}$, where $L_{i,j,n}$ is the base lifetime of the n^{th} visit to state j from state i and $R_{i,j,n}$ is the residual life of this visit defined by $R_{i,j,n} \equiv 1_{\{L_{i,j,n} > S_{j,N_j(s+L_{i,j,n})} - S_{j,N_j(s)}\}} \cdot (S_{j,N_j(s+L_{i,j,n})+1} - S_{j,N_j(s)} - L_{i,j,n})$. In particular, the Web server requests from time s to time $s + L_{i,j,n} + R_{i,j,n}$ occur at the time epochs $\{s + Z_{j,t+1}, s + Z_{j,t+1} + Z_{j,t+2}, \ldots, s + L_{i,j,n} + R_{i,j,n}\}$ where $t = N_j(s)$. The remaining elements of our Web server access model are the initial time-series probability vector $\underline{\alpha} \equiv (\alpha_1, \ldots, \alpha_M)$, where the access model starts in state m with probability α_m, and the state transition probability matrix $\mathcal{P} \equiv [p_{i,j}]_{\{1 \leq i,j \leq M\}}$, where the access model makes a transition to state j upon leaving state i with probability $p_{i,j}$.

Our methodology is based on the use of general stochastic processes for $L_{i,j}$ and the use of general ARIMA time-series processes for Z_m. One can observe in practice (e.g., see Section 4) a time series that is stationary for some period of time, and then it exhibits a shift in mean, variance, etc to a new stationary period. Note that our approach can be applied to any (sufficiently large) collection of observed data to capture this behavior, and thus we can divide any given finite set of observations into phases and use the spectral analysis below to characterize each piecewise time series as an ARIMA model. By decomposing the spectral analysis into piecewise ARIMA, or *PARIMA*, processes and combining these processes in the manner described above, we obtain a general methodology for accurately characterizing Web server access patterns. Moreover, this approach makes

it possible to use smaller order ARIMA processes to represent each of these piecewise time series.

Our Web server access model can consist of a separate set of stochastic processes for each type of Web server request, or a single set of stochastic processes for all requests, or any of the possibilities in between these extremes. A single set of stochastic processes is considered in what follows, from which the other cases can be handled in a straightforward manner as a mixture of such sets of stochastic processes. Our approach is also easily extended to Markov chains $\mathcal{P}$ of order greater than 1. By using the appropriately defined time-series process, our general methodology supports any level of access granularity that is desired, or that is necessitated by the granularity of the set of observation data available. In particular, for very fine-grained access data where each Web server request has a unique arrival time, a time-series process Z_m can be used where $Z_{m,t}$ is a continuous random variable representing the time between the t^{th} and $t-1^{\text{st}}$ request. When only coarse-grained access data is available, then it is often more appropriate to use a time-series process Z_m where the interrequest times $Z_{m,t}$ are discrete random variables. More generally, each arrival epoch in both of these cases can represent a batch of Web server requests such that $Z_m = \{(Z_{m,t}, K_{m,t}) ; t \in \mathbb{N}\}$, where $Z_{m,t}$ is as defined above and $K_{m,t} \in \mathbb{N} - \{0\}$ denotes the request batch size. When very coarse-grained access data is available for certain large-scale, heavy-traffic environments, such as the time series data considered in Section 4, then a time-series process Z_m can be used where $Z_{m,t} \in \mathbb{N}$ represents the number of requests at time t.

In the remainder of this section, we focus on our use of the general class of ARIMA models for the time-series processes Z_m. ARIMA models are essentially based on decomposing the time series into three components, namely the autoregressive (AR), the integration (I) and the moving average (MA) linear filters. We describe each of these components of the ARIMA models in turn, and then briefly discuss the fitting of data to these models. To clarify the presentation, we will drop our use of the subscript m with the understanding that our discussion applies for each of the M time-series processes Z_m. We refer the interested reader to [17, 11] for additional technical details.

Stationarity. Given a nonstationary time series $\{Z_t ; t \in \mathbb{N}\}$, one needs to remove the trends to obtain a stationary process. There are many ways to detect and remove trends in a time series. One common method for removing polynomial-order trends is by differencing (the integration filter), where the differencing operator $\Delta \equiv 1 - B$ is defined in terms of the backward shift operator B such that $BZ_t = Z_{t-1}$, and thus $\Delta Z_t = Z_t - Z_{t-1}$. For example, if the process Z_t is growing linearly with time, the first order difference, $Y_t = \Delta Z_t = Z_t - Z_{t-1}$, will be stationary. When Z_t is growing quadratically, then Y_t grows linearly, and thus we have to difference Y_t to obtain a stationary series. This consists of essentially taking the second order difference of Z_t, which yields $Y_t - Y_{t-1} = Z_t - 2Z_{t-1} + Z_{t-2} = \Delta^2 Z_t$.

The rank test [13] and the Unit Root Test [5] provide effective methods for determining the appropriate order d of differencing required to obtain a stationary process from the original time series. In the remainder of this section, we assume $Z_t = \Delta^d Z'_t$ is a stationary series with mean 0. Since the process that starts at time 0 is not necessarily stationary, we consider the stationary version of the process by assuming a proper initial value for Z_0 (or equivalently by assuming the process starts at $-\infty$).

The AR Model. A stationary time series $\{Z_t ; t \in \mathbb{N}\}$ is said to be governed by an order p *autoregressive process* AR(p) if the current value of the series, Z_t, depends linearly on the values of the previous p observations, $Z_{t-1}, \ldots Z_{t-p}$, and a random noise variable ϵ_t. Mathematically, this process can be written as $Z_t = \phi_1 Z_{t-1} + \ldots + \phi_p Z_{t-p} + \epsilon_t$ where $\{\epsilon_t\}$ is an i.i.d. sequence, each of whose elements has distribution $\mathrm{N}(0, \sigma_\epsilon^2)$. Note that $\{\epsilon_t\}$ is called *white noise*.

We define the autocovariance function with lag k to be $\lambda_k = \mathsf{Cov}(Z_t, Z_{t-k}) = \mathsf{E}[Z_t Z_{t-k}] - \mathsf{E}[Z_t]\ \mathsf{E}[Z_{t-k}]$. The *autocorrelation function* (acf) with lag k is then defined as $\rho_k = \lambda_k / \lambda_0$, which characterizes the key dependencies within the series. The coefficients $\phi_{k,\ell}$ of the equation

$$\begin{bmatrix} 1 & \rho_1 & \ldots & \rho_{k-1} \\ \rho_1 & 1 & \ldots & \rho_{k-2} \\ \vdots & \vdots & \ldots & \vdots \\ \rho_{k-1} & \rho_{k-2} & \ldots & 1 \end{bmatrix} \begin{bmatrix} \phi_{k,1} \\ \phi_{k,2} \\ \vdots \\ \phi_{k,k} \end{bmatrix} = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_k \end{bmatrix} \quad (1)$$

are called the *partial autocorrelation function* (pacf) with lag k, and equation (1) is called the Yule-Walker equation. If the autocorrelation functions ρ_k are known, we can calculate the partial autocorrelation functions $\phi_{k,\ell}$ via (1).

We then can show for the AR(p) process that the acf drops to 0 exponentially fast, and that the pacf drops to 0 for lags greater than p. In particular, the order p of the AR process is determined by the lag at which the pacf becomes negligible.

The MA Model. A stationary time series $\{Z_t ; t \in \mathbb{N}\}$ is said to be governed by an order q *moving average process* MA(q) if the current value of the series, Z_t, depends linearly on the values of the previous q white noise variables, $\epsilon_{t-1}, \ldots \epsilon_{t-p}$. Mathematically, this process can be written as $Z_t = \epsilon_t - \theta_1 \epsilon_{t-1} - \ldots - \theta_q \epsilon_{t-q}$ where $\{\epsilon_t\}$ is white noise with mean 0 and variance σ_ϵ^2.

We can calculate the autocovariance function as

$$\mathsf{E}[Z_t Z_{t-r}] = \begin{cases} \sigma_\epsilon^2(\theta_r + \theta_{r+1}\theta_1 + \ldots + \theta_q \theta_{q+r}),\ 0 \le r \le q \\ 0, \quad r > q. \end{cases}$$

The autocorrelation function is then given by

$$\rho_r = \begin{cases} \frac{(\theta_r+\theta_{r+1}\theta_1+\ldots+\theta_q\theta_{q+r})}{(1+\theta_1^2+\ldots+\theta_q^2)}, & 0 \le r \le q \\ 0, & r > q. \end{cases}$$

Observe that the acf drops to 0 after lag q. The pacf is complicated for the general MA(q) model. However, we can show that it is infinitely long and dies out rather quickly. In general, the lag at which the acf drops to a negligible level tells us about the order of the MA process.

The ARMA Model. We next combine the AR and MA processes to obtain the more general class of ARMA models. A stationary time series is said to be an ARMA(p, q) process if it can be represented as $Z_t - \phi_1 Z_{t-1} - \ldots - \phi_p Z_{t-p} = \epsilon_t - \theta_1\epsilon_{t-1} - \ldots - \theta_q\epsilon_{t-q}$, where p and q are the order of the AR and MA processes, respectively.

For the general ARMA(p, q) models, the acf and pacf are quite complicated, and they can be difficult to obtain in closed form. Characteristically, however, we can tell that both the acf and pacf converge to 0 exponentially fast. Hence, the lag beyond which the acf and pacf become negligible provides information about the MA and AR orders.

Using the backward shift operator we can obtain a simple form for the ARMA(p, q) process. Define $\Phi(B) = (1-\phi_1 B-\ldots-\phi_p B^p)$ and $\Theta(B) = (1-\theta_1 B-\ldots-\theta_q B^q)$, from which we have $\Phi(B)Z_t = \Theta(B)\epsilon_t$. Multiplying both sides of these definitions by $\Theta^{-1}(B)$ and $\Phi^{-1}(B)$, respectively, we then can represent the ARMA(p, q) process as the following infinite order AR and MA processes, which shows the relationship between the AR and MA models.

$\Theta^{-1}(B)\Phi(B)Z_t = \epsilon_t$	AR representation;
$Z_t = \Phi^{-1}(B)\Theta(B)\epsilon_t$	MA representation.

The ARIMA Model. Upon incorporating the differencing filter into the ARMA model, we obtain the general class of ARIMA models. A time series is an ARIMA(p, d, q) process if $\Delta^d Z_t$ is an ARMA(p, q) process, i.e., if $(1 - \phi_1 B - \ldots - \phi_p B^p)\Delta^d Z_t = (1 - \theta_1 B - \ldots - \theta_q B^q)\epsilon_t$.

Model Estimation and Diagnostics. Assuming a stationary series, we consider constructing an ARMA model to fit data. We first need to estimate the acf and pacf, and identify the AR and MA orders. It has been shown that given a time series $Z_1, \ldots, Z_n$, then a consistent estimator for ρ_k is given by $\hat{\rho}_k = c_k/c_0$ where $c_k = n^{-1}\sum_{t=1}^{n-k} Z_t Z_{t+k}$. Once we have an estimate of ρ_k, we can then solve equation (1) to obtain an estimate $\hat{\phi}_{i,j}$ for $\phi_{i,j}$.

In addition to the methods described earlier, Bartlett [4] has proposed an approach to determine the MA order by estimating the standard error of ρ_k. Akaike [1] introduced the AIC which can be used to determine the AR order by studying the residual variance of the AR(k) model.

Having determined the order of the ARMA model, we can use the standard least-square method to estimate the parameters of the model. We can also use the maximum likelihood method to obtain an even closer estimate. Once we have estimates for the model parameters, we can then calculate the estimates for the residuals ϵ_t. If the model is correct, the residuals should be white noise. One approach to test if the residuals are truly white noise is to look at the acf of the residuals. The sample autocorrelations should be close to zero for all lags. Another approach is to use the *Portmanteau test* in [13] by jointly considering all the autocorrelations.

If the residuals are approximately random we can conclude that the model fits the data well. If the residuals are not random we will need to modify the model, e.g., increase the AR and MA orders, and continue the analysis following the methods described above.

3. Web Server System Model and Analysis

Our Web server system model is based in part on the IBM Web site for the 1996 Olympic Games, which is representative of a particular Web server environment under heavy traffic conditions. The system consisted of a 19-node SP2 machine where all incoming requests were routed to specific SP nodes by the Network Dispatcher (ND) [7, 9] in a weighted round-robin fashion, with the weight for each node being a function of its current load. This scalable approach balances the Web server load across the set of SP nodes by sending more traffic to less loaded nodes. It appears likely that ND has the effect of smoothing out the request arrival process, although we do not have enough information at this time to quantify this effect. All of the requested documents are sent directly to the clients without going back through ND [7, 9].

We focus on an individual node of the Web server system consisting of P identical processors to which requests arrive from an exogenous source that is governed by a stochastic process $Y(t)$. This approach is based in part on the Olympic Web site data available to us, although it is important to note that our methodology can be used to address other system models by appropriately modifying the stochastic process $Y(t)$. An FCFS infinite-capacity queue is used to hold all requests that need to be processed. The Web server requests are divided into two types, namely those that access pages which are created dynamically and those that access static files such as gif and jpg images. When a request for static data arrives, the server returns the contents of the corresponding file. In contrast, dynamic requests are for HTML pages that require the server to execute a program to generate the dynamic pages. Dynamic data generally change

more frequently than static data; e.g., a dynamic page may contain some information that is maintained in an evolving database such as the current score of a game.

A significant percentage of the pages at the 1996 Olympic Web site were created dynamically, including all but one of the HTML pages. The Web server performance was limited by the processing power of servers and not by the network. Embedded image files comprised the majority of requests. The proportion of requests for dynamic pages was around 20%. The time to serve a dynamic page request is often significantly more than that required to satisfy a static request. In particular, the average processor time to satisfy requests for static files was on the order of $5ms$, whereas the average processor time to serve a dynamic page request was on the order of $500ms$. We consider a single-class workload model based in part on the data available to us, and thus the service demands of the Web server requests are modeled as i.i.d. $\mathbb{R}^+$-valued random variables having PDF $\mathbf{B}(\cdot)$. We use an order m_B PH-type PDF for $\mathbf{B}(\cdot)$, arranged according to the classical Coxian distribution, with mean $\mu^{-1} = 104ms$ and coefficient of variation $cv_B = 0.6$ to match system measurements.

The Web server request arrival process $Y(t)$ is formulated according to the general methodology presented in Section 2 from log files that were kept for every node of the Olympic Web server site. The per-node log file records the origin of each request (i.e., the IP address of the client machine), the time of the request (in seconds), the page requested, and the size of the page. With a few exceptions, the log file contains multiple requests at each second. Given the coarse granularity of the access request times in these logs and the large-scale, heavy-traffic Web server environment they represent, we use the set of time-series processes $\{Z_1, Z_2, \ldots, Z_M\}$ where $Z_m = \{Z_{m,t} \in \mathbb{N}\,;\, t \in \mathbb{N}\}$ represents the number of requests at the t^{th} step of the m^{th} time-series phase, $1 \leq m \leq M$. Simulation is used to obtain performance measures for this instance of our Web server system model due to this complex arrival process.

For comparison in Section 4, we also consider the use of an order m_A PH-type PDF with mean λ^{-1} and cv_A (arranged according to the classical Coxian distribution) to approximate the arrival process $Y(t)$ by matching its coefficient of variation. In this case the system model is equivalent to a PH/PH/P queue, for which we derive a matrix-analytic solution. We sketch the solution here and refer the interested reader to [16] for the technical details.

When $\lambda < P\mu$, the stationary distribution $\boldsymbol{\pi}$ for the PH/PH/P queue is uniquely determined by solving $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$ together with $\boldsymbol{\pi}\mathbf{e} = 1$, where $\mathbf{Q}$ is the generator matrix for the queue, having the structure

$$\mathbf{Q} = \begin{bmatrix} B_{00} & B_{01} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ B_{10} & B_{11} & A_0 & \mathbf{0} & \mathbf{0} & \cdots \\ \mathbf{0} & A_2 & A_1 & A_0 & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & A_2 & A_1 & A_0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

and $\mathbf{e}$ is a column vector of all ones. It then can be established that the components of $\boldsymbol{\pi}$ are given by

$$(\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_P) \begin{bmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} + RA_2 \end{bmatrix} = \mathbf{0}$$

and $\boldsymbol{\pi}_{P+n} = \boldsymbol{\pi}_P R^n$, $n \in \mathbb{N}$, where R is the minimal non-negative matrix that satisfies $R^2A_2 + RA_1 + A_0 = \mathbf{0}$ and $(\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_P)$ is normalized by $(\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{P-1})\mathbf{e} + \boldsymbol{\pi}_P(\mathbf{I} - R)^{-1}\mathbf{e} = 1$.

Given the stationary probability vector $\boldsymbol{\pi}$, we can calculate the mean number of server requests in the system as $\overline{N} = \sum_{n=1}^{P-1} n\boldsymbol{\pi}_n\mathbf{e} + P\boldsymbol{\pi}_P(\mathbf{I} - R)^{-1}\mathbf{e} + \boldsymbol{\pi}_P R(\mathbf{I} - R)^{-2}\mathbf{e}$. The mean response time of a request can then be obtained from Little's law, which yields $\overline{T} = \lambda^{-1}\overline{N}$.

4. Results

In this section we apply the mathematical methods of Section 2 to the Web server system of Section 3. A large number of simulation experiments and detailed analysis was performed. We now provide a small representative sample of the results from our analysis.

Figure 1 shows the number of requests received by one of the servers over a period of one hour. Since the request arrival process is nonstationary over the course of the entire hour, the graph in the figure was divided into five phases during which the arrival process was stationary. Each of these phases was analyzed within the context of the general Web server access model presented in Section 2.

In Table 1 we provide an analysis of each of the five phases corresponding to the traffic data in Figure 1, as well as of the entire hour as a whole. We note that an ARMA(1,2) process characterizes each phase, as well as the whole process, reasonably well. Although there appears to be some trend in the fourth phase, the length is relatively small and it still can be accommodated by the ARMA(1,2) process. The ARMA(1,2) parameters for the various phases do not differ by very much. This suggests that the parameters characterize the *nature* of the time-series process within the particular hour.

One way to scale the ARMA(1,2) processes to arbitrary request rates is to keep the AR and MA parameters constant. Specifically, we set $Z_i = \phi_1 Z_{i-1} + \epsilon_i - \theta_1\epsilon_{i-1} - \theta_2\epsilon_{i-2}$, $\epsilon_i \sim \mathrm{N}(0, \sigma_\epsilon^2)$ and $Batch_i = Z_i + mean$. We also set ϕ_1,

Time Range	0-1848	1857-2003	2004-2615	2651-3232	3233-3598	0-3598
Mean Request Rate	6.823688	5.775510	8.457516	7.670103	6.778689	7.111142
Mean Interrequest	0.146548	0.173145	0.118238	0.130376	0.147521	0.140624
cv_A	0.517780	0.480576	0.631715	0.804402	0.423921	0.997007
ARIMA Order	(1,0,2)	(0,0,0)	(1,0,2)	(1,0,2)	(0,0,1)	(1,0,2)
AR, Lag 1	0.76112	0	0.65424	0.95310	0	0.88650
MA, Lag 1	0.68836	0	0.51111	0.85280	0.09608	0.75446
MA, Lag 2	-0.08419	0	-0.13756	-0.09956	0	-0.08203
σ	2.4319	2.3176	2.9160	2.7230	2.2871	2.6110

Table 1. Piecewise Analysis of the Traffic Data in Figure 1.

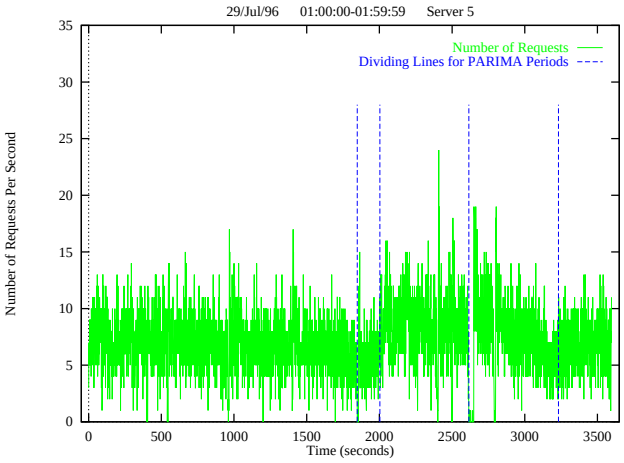

Figure 1. Traffic data for one of the Olympic Web server nodes over the course of an hour.

θ_1 and θ_2 equal to their estimate from the real trace, i.e., $\phi_1 = 0.88$, $\theta_1 = 0.75$ and $\theta_2 = -0.08$. The variance (σ_ϵ^2) can be scaled according to any positive increasing function f as follows

$$\sigma_\epsilon(new) = f\left(\frac{mean(new)}{mean(estimate)}\right)\sigma_\epsilon(estimate)$$

where $mean(estimate) = 7.1$ and $\sigma_\epsilon^2(estimate) = 6.8$. When $f(x) = \sqrt{x}$, this scales the variance (σ_ϵ^2) linearly with the mean

$$\sigma_\epsilon^2(new) = \frac{mean(new)}{mean(estimate)}\sigma_\epsilon^2(estimate). \quad (2)$$

We will perform our experiments based on this scaling function.

The set of PARIMA and scaled PARIMA processes obtained from the above analysis (and their variants) provides the basis for a class of benchmarks, called *Flintstone*, that is intended to be representative of large-scale, heavy-traffic Web server environments such as the IBM Web site for the Olympic Games. We performed several experiments to generate scaled processes from our Flintstone methodology and compare them with the real trace. These experiments suggest that our scaling approach is quite accurate. As a simple example, Figure 2 shows a scaled process from Flintstone with a mean request rate of 9.52 requests per second.

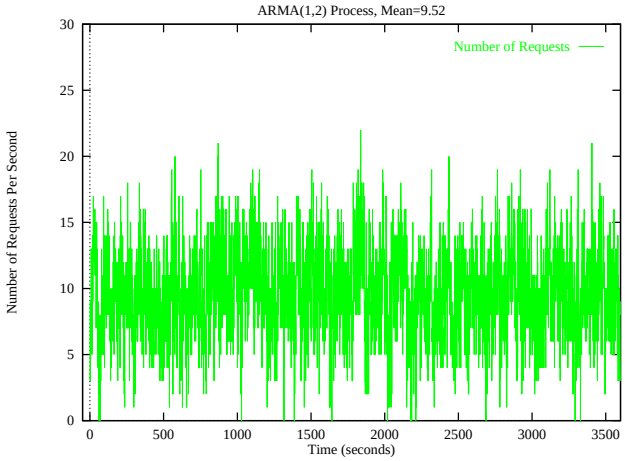

Figure 2. Scaled ARMA(1,2) process.

An alternative method, which we call M2M, has been used in other studies (e.g., [10]) to analyze and scale Web server access distributions. Specifically, the Web server request distributions were characterized by the interrequest time mean and cv_A as calculated from the Web access logs. In order to scale an interrequest time distribution to an arbitrary request rate, there is sufficient information under M2M to only scale the mean while keeping cv_A constant. This approach uses an i.i.d. sequence of interrequest times that matches the mean and cv_A, from which the batch size process can be obtained by counting the number of requests within each second. It can be rigorously shown under general conditions that the batch sizes obtained in this manner are i.i.d. for different time intervals.

It is well known that the first two moments are not suf-

ficient to completely characterize a general stochastic process, and this is particularly the case for bursty processes. As a specific (simple) example, observe that in Table 1 the value of cv_A for each phase is much smaller than the cv_A value for the entire hour. This is because the process is nonstationary and there are some phases for which the mean of the process shifts. The value of cv_A is very sensitive to such shifts, as illustrated by the results in Table 1. Because it is based solely on the first two moments, the M2M approach is not very robust with respect to characterizing a general stochastic arrival process, especially a nonstationary process (i.e., one with non-negligible trends). Our Flintstone methodology, however, can be used to accurately characterize the access patterns of the real Web server system. It is important to also note that it was not an objective of [10] to develop a methodology for characterizing Web server access patterns and that M2M was used as an approximation.

Figure 3 shows a scaled batch process with a mean request rate of 9.52 requests per second generated using the M2M method described above. Upon comparing the plots in Figures 2 and 3, we observe that the request process generated with the M2M method is less bursty than the corresponding process generated by the scaled ARMA(1,2) process from Flintstone. The M2M method does not have sufficient information nor sufficient flexibility to appropriately scale the request distribution so that it is representative of real Web server access patterns under different load conditions. In particular, it does not characterize the dependencies between requests, and this results in the less bursty request process shown in Figure 3. By constructing a detailed mathematical characterization of the access patterns of the real system, our Flintstone methodology can provide an accurate Web server request process for the full spectrum of load conditions.

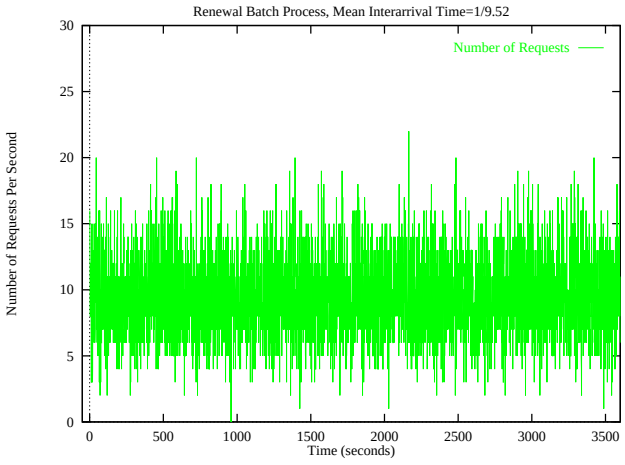

Figure 3. Scaled M2M process.

To demonstrate how the Flintstone and M2M methodologies differ in terms of the Web server performance obtained under their respective workloads, we used the simulation and analysis of Section 3 to obtain mean response time measures for the single-processor ($P = 1$) and 4-processor ($P = 4$) instances of both workloads. A portion of our results are given in Figures 4 and 5 as a function of the traffic intensity $\lambda/(P\mu) < 1$ following the notation of Section 3. Note that there are two sets of results for the M2M approach, one in which we scaled the interrequest process as described above (i.e., scale the mean while keeping the cv_A fixed to that of the largest stationary portion of the traffic data), and the other where we scaled the interrequest process to match the mean and cv_A generated by our simulations under the Flintstone methodology.

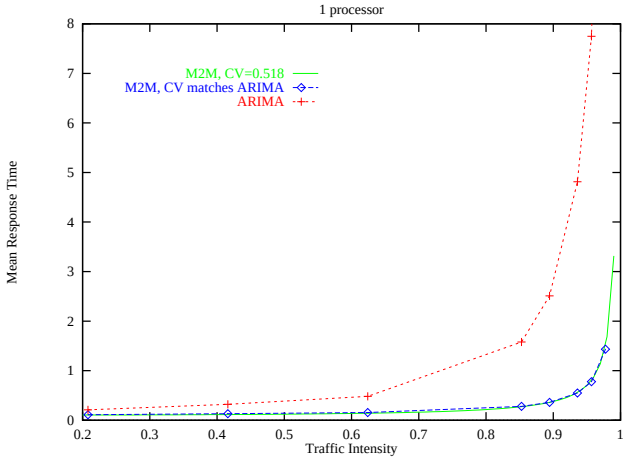

Figure 4. Comparison of M2M and Flintstone.

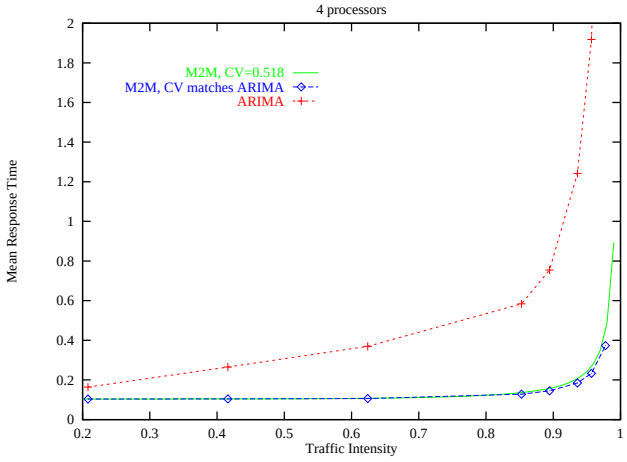

Figure 5. Comparison of M2M and Flintstone.

Comparing the mean response times in Figures 4 and

5 under the two methodologies, we find that the Web access patterns from Flintstone always result in larger mean response times, and this performance gap increases as the traffic rises. In particular, we observe that the Flintstone methodology yields significantly larger mean response times once the traffic intensity exceeds 0.6. This is because the PARIMA approach of Flintstone captures the burstier request process of the real system, whereas the M2M approach does not. Such burstiness has a greater effect on system performance as the system traffic increases. We also observe that the value of cv_A decreases as we scale up the request rate using equation (2). This is because the mean is increasing at a faster rate than the variance. The range of cv_A values is from 0.9934 when $\lambda = 2$ to 0.1830 when $\lambda = 37.6$. As the value of cv_A decreases, the curve for M2M using a cv_A value matching that of the Flintstone process drops slightly below the other M2M curve. This is because the system yields a smaller response time under a less variable arrival process. Note that scaling up the request rate faster than the function used in equation (2) results in a larger performance gap between the Flintstone and M2M curves, whereas a slower scaling function has the opposite effect.

5. Conclusions and Ongoing Work

In this paper we presented a general methodology for characterizing and analyzing Web server access patterns based on a spectral analysis of piecewise time-series processes which are combined in a general manner. Our mathematical representation of these access patterns makes it possible to scale the Web server traffic in a general and flexible manner so that Web server performance can be examined at all traffic intensities of interest. Our methodology was applied to the access logs from the IBM Web site for the 1996 Olympic Games. This provides the basis for the Flintstone class of benchmarks, which provides considerable advantages over previous methods with respect to characterizing the traffic patterns, scaling these patterns to consider different server loads, and the impact of such issues on Web server performance.

We are currently pursuing several areas related to the research presented in this paper. This includes: examining the spectrum of scaling functions found in practice; exploring the use of heavy-tailed distributions in the set of PARIMA processes of our methodology; exploiting the methods developed in this paper for predicting the request sequence needed by on-line caching and prefetching algorithms; and exploring the analysis of queueing systems under the arrival processes considered in this paper and approximations thereof. We are also starting to apply our methodology to the access logs from the IBM Web site for the 1998 Olympic Games, as well as working to automate our approach for constructing benchmarks from collections of real system data. Finally, we hope to explore the possibility of combining our Flintstone methodology with the SURGE methods in [3].

References

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC-19:716–722, 1974.

[2] M. F. Arlitt and C. L. Williamson. Web server workload characterization: The search for invariants. In *Proceedings of the ACM Sigmetrics Conference*, pages 126–137, 1996.

[3] P. Barford and M. E. Crovella. Generating representative Web workloads for network and server performance evaluation. In *Proceedings of the ACM Sigmetrics Conference*, 1998.

[4] M. Bartlett. *An Introduction to Stochastic Processes with Special Reference to Methods and Applications.* Cambridge University Press, Cambridge, second edition, 1966.

[5] G. Box and G. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, Inc., revised edition, 1976.

[6] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. In *Proceedings of the ACM Sigmetrics Conference*, pages 160–169, 1996.

[7] D. Dias, W. Kish, R. Mukherjee, and R. Tewari. A Scalable and Highly Available Web Server. In *Proceedings of the 1996 IEEE Computer Conference (COMPCON)*, 1996.

[8] J. Hu, I. Pyarali, and D. Schmidt. Measuring the Impact of Event Dispatching and Concurrency Models on Web Server Performance Over High-speed Networks. In *Proceedings of GLOBECOM '97*, 1997.

[9] G. Hunt, G. Goldszmidt, R. King, and R. Mukherjee. Network Dispatcher: A Connection Router for Scalable Internet Services. In *Proceedings of the 7th International World Wide Web Conference*, 1998.

[10] A. Iyengar, E. MacNair, and T. Nguyen. An Analysis of Web Server Performance. In *Proceedings of GLOBECOM '97*, 1997.

[11] A. K. Iyengar, E. A. M. Nair, M. S. Squillante, and L. Zhang. A general methodology for characterizing access patterns and analyzing web server performance. Technical Report RC 21063, IBM Research Division, 1997.

[12] D. L. Jagerman and B. Melamed. The transition and autocorrelation structure of TES processes part I: General theory. *Stochastic Models*, 8(2):193–219, 1992.

[13] M. Kendall and J. Ord. *Time Series.* Oxford University Press, 1990.

[14] Silicon Graphics, Inc. World Wide Web Server Benchmarking. http://www.sgi.com/Products/WebFORCE/WebStone/.

[15] System Performance Evaluation Cooperative. SPECweb96 Benchmark. http://www.specbench.org/osg/web96/.

[16] M. S. Squillante. A matrix-analytic approach to a general class of G/G/c queues. Technical report, IBM Research Division, 1998.

[17] L. Zhang and M. S. Squillante. A time-series analysis of large-scale web server performance. Technical Report RC 21062, IBM Research Division, 1997.