ARTICLE IN PRESS

Available online at www.sciencedirect.com

ScienceDirect

Performance Evaluation ▮ (▮▮▮▮) ▮▮▮–▮▮▮

PERFORMANCE EVALUATION

An International Journal

www.elsevier.com/locate/peva

Performance impacts of autocorrelated flows in multi-tiered systems

Ningfang Mi[a,*], Qi Zhang[b], Alma Riska[c], Evgenia Smirni[a], Erik Riedel[c]

[a] College of William and Mary, Williamsburg, VA, USA

[b] Microsoft Corporation, Redmond, WA, USA

[c] Seagate Research, Pittsburgh, PA, USA

Abstract

This paper presents an analysis of the performance effects of burstiness in multi-tiered systems. We introduce a compact characterization of burstiness based on autocorrelation that can be used in capacity planning, performance prediction, and admission control. We show that if autocorrelation exists either in the arrival *or* the service process of any of the tiers in a multi-tiered system, then autocorrelation propagates to *all* tiers of the system. We also observe the surprising result that in spite of the fact that the bottleneck resource in the system is far from saturation and that the measured throughput and utilizations of other resources are also modest, user response times are very high. When autocorrelation is not considered, this underutilization of resources falsely indicates that the system can sustain higher capacities.

We examine the behavior of a small queuing system that helps us understand this counter-intuitive behavior and quantify the performance degradation that originates from autocorrelated flows. We present a case study in an experimental multi-tiered Internet server and devise a model to capture the observed behavior. Our evaluation indicates that the model is in excellent agreement with experimental results and captures the propagation of autocorrelation in the multi-tiered system and resulting performance trends. Finally, we analyze an admission control algorithm that takes autocorrelation into account and improves performance by reducing the long tail of the response time distribution.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Multi-tiered systems; Autocorrelation; Capacity planning; Workload characterization; Queuing networks

1. Introduction

We focus on the general problem of capacity planning and performance prediction of multi-tiered systems. Workload characterization studies of such systems usually examine the stochastic characteristics of arrivals to the system and wait/service times at various tiers aiming at bottleneck identification, diagnosing the conditions under which bottlenecks are triggered, and assisting the development of resource management policies to improve performance or provide service level provisioning [5,8,14,19].

In this paper, we examine how burstiness in the arrival or service process of *any* of the tiers affects end-to-end performance. More specifically, our focus is on systems where there is always an *upper bound* on the number of jobs

* Corresponding address: College of William and Mary, Computer Science, Williamsburg, VA 23185, USA.
E-mail addresses: ningfang@cs.wm.edu (N. Mi), qizha@microsoft.com (Q. Zhang), alma.riska@seagate.com (A. Riska), esmirni@cs.wm.edu (E. Smirni), erik.riedel@seagate.com (E. Riedel).

0166-5316/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.peva.2007.06.016

that are in the system at all times. A system with finite buffers and/or admission control, which is indeed the case in a multi-tiered system as buffer sizes or the maximum number of simultaneous connections is restricted by software and/or hardware, behaves in essence like a closed system, i.e., a system with a closed loop structure [21].

This paper complements prior workload characterization studies of multi-tiered systems [3] by providing a formalization of the concept of burstiness, which is expressed by the *dependence structure* of the request flows across the various tiers of the closed loop structure. This dependence structure is described and quantified via the *autocorrelation function* (ACF).

In an open system, i.e., a system with infinite buffers, the presence of autocorrelation in the arrival or service process of a queue is expected to affect the performance of downward queues only, i.e., queues that feed from the departure process of the queue with autocorrelation. Instead, in multi-tiered systems with a closed loop structure, if autocorrelation exists in the service process of *any* of the tiers, then it propagates across the *entire* loop in the closed system and is present in the arrival stream of tiers (queues) that *precede* that tier (queue), unexpectedly affecting their performance as well as end-to-end performance.

Comparing the performance effects of the presence of autocorrelated flows in multi-tiered systems with the performance of the same system with independent flows, we show that end-to-end performance significantly deteriorates while bottleneck devices are less utilized, falsely indicating that the system is able to sustain higher load. Furthermore, we show that in contrast to systems where no burstiness is observed, the tails of the overall response time distributions do not necessarily reflect the time spent at the bottleneck tier, but instead are shaped by the response time tail at the tier that is the source of autocorrelation, irrespective of its utilization level. To the best of our knowledge, this is the first time that autocorrelated flows are identified as an important stochastic characteristic in multi-tiered systems with a closed-loop structure.

We also present some discussion on how to use knowledge of autocorrelation in flows for the development of resource management algorithms that consider dynamic system behavior [12]. We show that dependence in flows within the system is critical for effective admission control or capacity planning. If dependence is ignored, then resource provisioning that is based solely on the number of simultaneous user requests or on bottleneck analysis makes incorrect decisions.

This paper is organized as follows. In Section 2 we motivate this work by presenting experimental evidence of the presence of autocorrelation in multi-tiered Internet and storage systems. We then use a simple queuing network to quantify the performance effects of autocorrelation in systems (see Section 3). In Section 4, we present an experimental study with the TPC-W benchmark that shows how autocorrelation propagates across all the tiers in a multi-tiered system and a simple queuing model that captures the benchmark's behavior. In Section 5, we discuss implications of autocorrelation for system design. Section 6 gives an overview of related work. Finally, Section 7 summarizes our contributions and outlines future work.

2. Finding autocorrelation

Autocorrelation is used as a statistical measure of the relationship between a random variable and itself [4]. Consider a stationary time series of random variables $\{X_n\}$, where $n = 0, \ldots, \infty$, in discrete time. The autocorrelation function (ACF) $\rho_X(k)$ shows the value of the correlation coefficient for different time lags $k > 0$:

$$\rho_X(k) = \rho_{X_t, X_{t+k}} = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\delta^2},$$

where μ is the mean and δ^2 is the common variance of $\{X_n\}$. The argument k is called the lag and denotes the number of observations that separate X_t and X_{t+k}. The values of $\rho_X(k)$ may range from -1 to 1. If $\rho_X(k) = 0$, then there is no autocorrelation at lag k. In most cases ACF approaches zero as k increases. The ACF essentially captures the "ordering" of random values in the time series.

Intuitively, if there is no autocorrelation (i.e., ACF is zero which implies that there is independence in the stochastic process), then a random variable is generated as follows: first a random number is drawn between 0 and 1 and then this random number is mapped into the distribution space via the inverse distribution function to obtain the random value. This way of sampling does not create any temporal locality, i.e., given the current drawn value, *any* value of the distribution space is equally likely to occur. In distributions that have correlation there is a temporal bias in

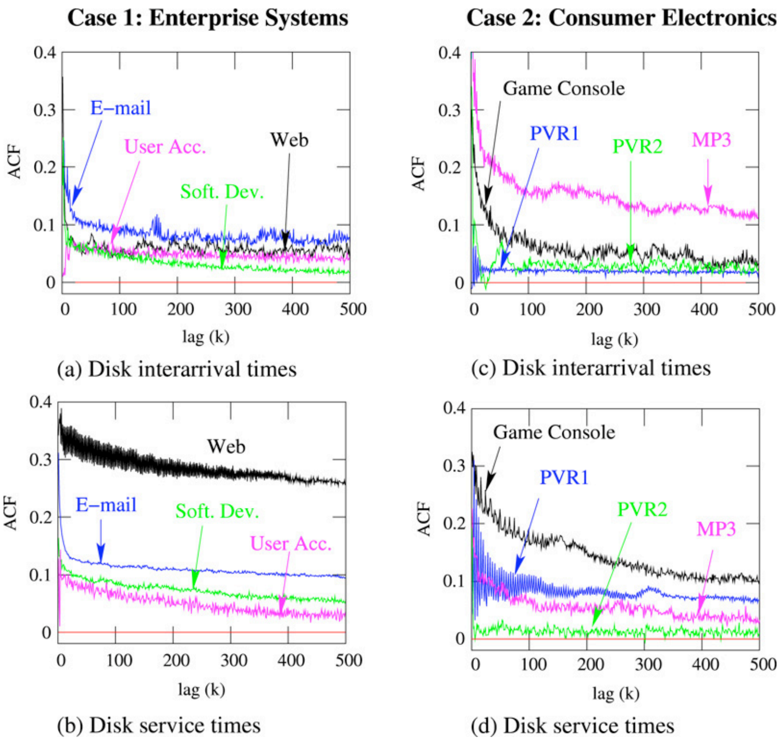


Fig. 1. ACF of inter-arrival and service times for disk level traces measured in enterprise systems and consumer electronics devices.

this sampling, e.g., random variables are sampled within a certain range for some time before moving into another range. This creates temporal locality but overall all values of the distribution space are sampled as dictated by the distribution function. *High* positive ACF values imply that in the time series a value of the random variable has a high probability to be followed by another variable of the same order of magnitude, while negative ACF values imply the opposite.

We now examine the presence of autocorrelation in real systems. Case 1 in Fig. 1 shows the ACF for a collection of enterprise storage systems, i.e., a Web server, an E-mail server, a Software Development server, and a User Accounts server. These are open systems with finite buffers. User arrivals feed into the server (first tier) where they are modified by caches and other processing before being passed to the disk (second tier). Measurements of the arrival process and the service process are taken at the disk (second tier). Fig. 1(a) shows the ACF of the arrival process at the disk, which differs markedly from the ACF of the service process at the disk in Fig. 1(b). These are measured, "live" systems that we did not control, but in later sections of the paper we will analyze a similar structure using the front server and database server in a TPC-W setup that we are able to measure in more detail.

Case 2 in Fig. 1 shows the ACF for a collection of traces from consumer electronics devices, i.e., a Personal Video Recorder (PVR) in two different application scenarios, an MP3 player, and a game console. These are classical closed-loop systems such as the one we consider in Section 3. Since there is only a single user, the first tier server (application processing) passes requests directly to the disk (second tier) which feeds back to the request arrival process. Fig. 1(c) shows the ACF of the arrival times at the disk, which also differs markedly from the ACF of the service process at the disk in Fig. 1(d) in some cases.

In both sets of measurements, inter-arrival times and service times are correlated, with some cases of pronounced long-range dependence, i.e., ACF lines that decay slowly to zero, most pronounced in the service times of the Web server in Fig. 1(b) and the game console in Fig. 1(d). These measurements show that autocorrelation exists at the disk tier for different workloads in large and small systems. Server processing – in particular the cache hierarchy and algorithms along the I/O path – determines how arrivals and service demands are shaped at the disk.

Having established the existence of autocorrelation in "live" workloads, we proceed to analyze these effects in more detail using queuing analysis.

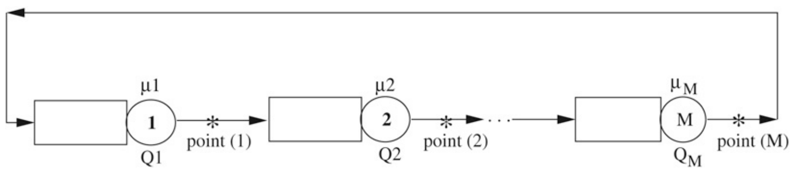


Fig. 2. A closed system with M queues.

3. Autocorrelation in closed systems

We use the simplest closed queuing system (see Fig. 2) that resembles the topology of a multi-tiered application. The purpose of this analysis is twofold: (1) to observe how autocorrelation propagates through all tiers in the system and (2) to observe how autocorrelation affects system performance.

Autocorrelation in the arrival or service processes directly implies that the system is not product-form [16], therefore one can only use simulation for its analysis. We present performance results for closed systems and these results readily apply to open systems with admission control [21].

We stress that no analytic methodology exists for modeling closed queuing networks with autocorrelated service processes. The general observations in this section result from a large number of simulation experiments.

3.1. A 2-tier system

Consider the simplest case of the system depicted in Fig. 2, i.e., a closed queuing network with two queues only, Q_1 and Q_2, with mean service rates μ_1 and μ_2, respectively.[1] We assume that a fixed number of jobs circulate in the queuing network. This number is also known as the *multiprogramming level* (MPL).

Let Q_2 be the bottleneck device and let Q_1 be twice as fast as Q_2.[2] The source of any autocorrelation in the flows of this network is the service process of at least one of the queues. We use a 2-state Markovian-Modulated Poisson Process (MMPP), a special case of the Markovian Arrival Process (MAP) [15], to model autocorrelated service times because it is analytically tractable. Its basic building block is a simple exponential but it can be easily parameterized so that it shows correlation. Other stochastic processes have been shown in the literature to capture burstiness and dependence, but their parameterization is not easy for our purpose here. MMPP allows us to build two distinct sets of samples that share the same PDF but allow for different ordering in each set. Therefore, by appropriate parameterization, one could create an MMPP that gives an independent process (NOACF) and an MMPP with dependence (ACF), while keeping all moments identical, essentially maintaining the same PDF. We evaluate two scenarios.

Scenario 1: The service times of Q_1 are exponentially distributed with mean rate $\mu_1 = 2$, while the service times of Q_2 are drawn from a 2-state MMPP with mean service rate $\mu_2 = 1$ and squared coefficient of variation SCV $= 20$. The service process of the bottleneck device Q_2 has autocorrelation.

Scenario 2: The service time of Q_1 is autocorrelated and is drawn from a 2-state MMPP with $\mu_1 = 2$, SCV $= 20$. The service process of Q_2 is exponentially distributed with mean service rate $\mu_2 = 1$. Now, Q_1, with ACF in its service process, is not the bottleneck.

In order to quantify the effect of autocorrelation on system performance, we also conduct experiments with different MMPPs for *Scenario 1* and *Scenario* 2 such that we maintain the same mean, SCV, and higher moments in the service process but with no autocorrelation, i.e., ACF equal to 0 in all lags. These experiments are labeled as NOACF. Appendix A gives the parameterization of the MMPP processes used here. Table 1 summarizes the two scenarios.

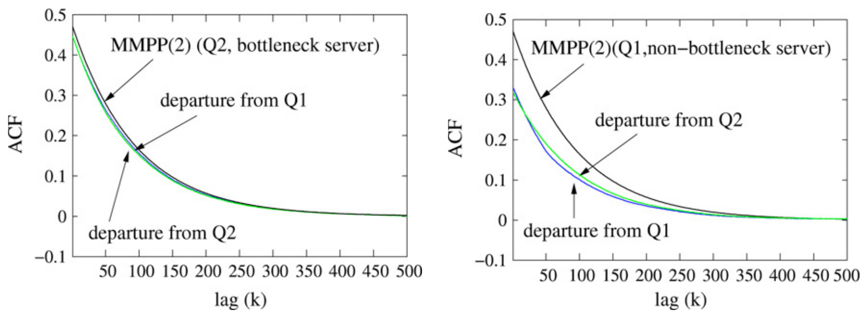
All simulations are done in a ten million sample space and results are reported with 98% confidence intervals.

[1] Here, we present a simple example of 2 queues only that allows us to better understand the system behavior.

[2] Experiments with varying relative speed of the two devices yield qualitatively the same results as those reported here due to lack of space.

Table 1
Summary of the two scenarios

	Q_1	Q_2 (bottleneck)
Scenario 1	Exponential	MMPP (ACF or NOACF)
Scenario 2	MMPP (ACF or NOACF)	Exponential



(a) Scenario 1: /Exponential/1 $\rightarrow$ /MMPP(2)/1. (b) Scenario 2: /MMPP(2)/1 $\rightarrow$ /Exponential/1.

Fig. 3. The ACF of departures from Q_1 (arrivals to Q_2), departures from Q_2 (arrivals to Q_1) for both scenarios. The ACF of the service process that generates autocorrelated flows in the system is also illustrated.

3.2. Autocorrelation propagation

First, we present how autocorrelation propagates through the queuing network by measuring the ACF of the departure process of Q_1 (i.e., at point (1) in Fig. 2) and the ACF of the departure process from Q_2 (i.e., at point (2) in Fig. 2).

Fig. 3 illustrates the autocorrelation propagation for *Scenario* 1 and *Scenario* 2, respectively, for MPL= 25. The ACF of the 2-state MMPP that generates the service times at Q_2 and Q_1 for *Scenario* 1 and *Scenario* 2, is also shown in the figure. Experiments with different MMPP(2) processes that maintain the same moments but different autocorrelation were also done but are not reported here due to lack of space. The interested reader is directed to [30]. In *Scenario* 1 (see Fig. 3(a)), ACF propagates through all tiers of the closed system with almost identical strength as the one at the service process of Q_2, the bottleneck device which injected autocorrelation into the system. In *Scenario* 2 (see Fig. 3(b)), autocorrelation propagates through the tiers, but with reduced strength compared to the autocorrelation of the service process that injected autocorrelation into the system (i.e., Q_1).

The above behavior is explained by the general queuing theoretic observation that the departure process of a busy queue resembles its service process rather than its arrival process. Instead, for a lightly loaded queue, its departure process resembles its arrival process. Hence, for *Scenario* 1, the departure process from the heavily loaded queue Q_2 resembles the service process of Q_2. The departure process of the lightly loaded queue Q_1 resembles its arrival process (i.e., the departure process of Q_2). Therefore, autocorrelation propagates with the same strength across all tiers.

In *Scenario* 2, any autocorrelation in the flows into Q_2 is reduced (see Fig. 3(b)) as departures from this queue are spaced further apart thanks to the slower exponential service times of Q_2. Experiments with variable MPL levels show the same qualitative behavior for both of the above scenarios. Our first observation is summarized as follows.

Observation 1. *Autocorrelation becomes present at all queues in a closed queuing network as long as it exists in the service process of one queue.*

3.3. Performance effects

We now turn to the effects of autocorrelation on system performance. We evaluate the mean response time (i.e., wait time plus service time), the mean queue length, and the mean utilization in each queue. We also report on the mean round trip time (i.e., sum of all response times) that captures end-to-end system performance. In an effort to quantify

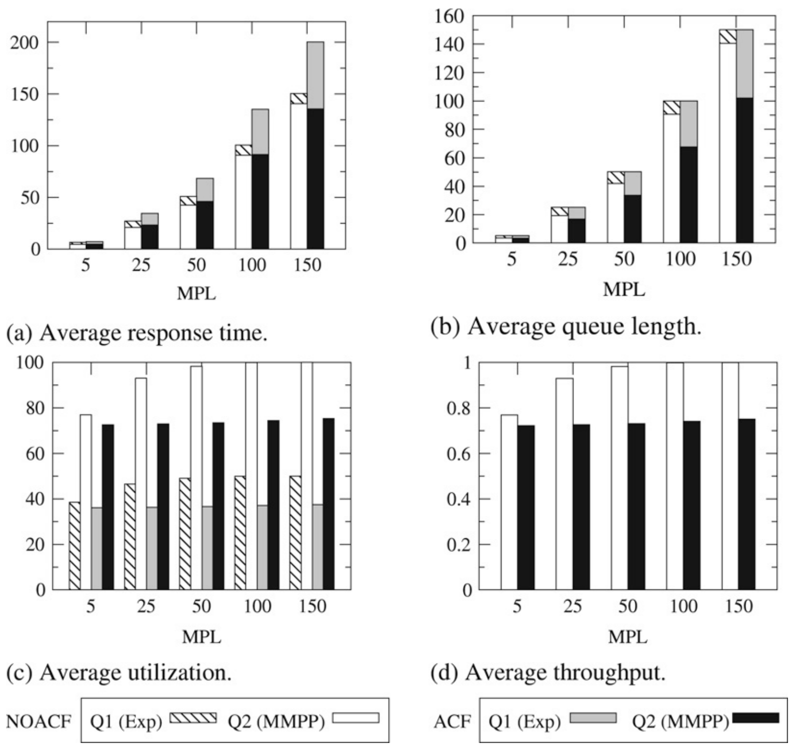


Fig. 4. Performance measures: (a) mean round trip time, (b) mean queue length, (c) mean utilization, and (d) mean throughput at each queue for *Scenario* 1. In all experiments the service time in Q_1 (non-bottleneck queue) is exponentially distributed. NOACF indicates that Q_2 has independent service times. ACF indicates that the Q_2 has autocorrelated service times.

the effect of ACF on system performance, we also conduct the same experiments as those described in *Scenario* 1 and *Scenario* 2 but using the NOACF MMPP processes.

Fig. 4 shows performance under *Scenario* 1 and Fig. 5 shows performance under *Scenario* 2. Autocorrelated flows in the closed system degrade overall system performance – compare round trip bars labeled NOACF with bars labeled ACF in both Figs. 4(a) and 5(a). Each bar also depicts how round trip time is distributed between the two queues as a function of MPL. With higher MPLs in the ACF case, the ratio of the average time spent in Q_1 to the average time spent in the bottleneck queue Q_2 significantly increases compared to the NOACF experiment, e.g., for MPL= 150, it increases by 10.5 times in *Scenario* 2. This is also reflected in Figs. 4(b) and 5(b) that plot average queue lengths.

Although performance of the non-bottleneck queue degrades in both scenarios, the reasons are different. In *Scenario* 1, performance degrades due to autocorrelated arrivals to the non-bottleneck queue Q_1 – recall that there is no autocorrelation in its service process. In *Scenario* 2, autocorrelation in both arrival and service processes of the non-bottleneck Q_1 queue degrades performance. More customers accumulate there and cause mean queue length of the bottleneck to decrease, as requests spend now more time in Q_1. This redistribution of requests in the system “balances” the load, as it is also reflected in the per queue utilizations. Also, in systems with ACF, the utilization of the non-bottleneck queue Q_1 decreases while its average queue length increases compared to the respective NOACF experiment. Looking closely into the per queue length distributions, we see the existence of heavy tails due to the bursty flows, which contribute to high average queue lengths. For systems with correlation, per queue utilizations decrease (see Figs. 4(c) and 5(c)). This is due to the fixed MPL level, that is effectively an upper bound on the number of jobs that circulate in the system at all times and to job redistribution in the system because of autocorrelation.

Figs. 4(d) and 5(d) show the system throughput for the two scenarios as a function of MPL. Consistently with the utilization behavior, we see that the system with no autocorrelation reaches its maximum throughput at MPL= 100, which is equal to the service rate of the bottleneck queue. Throughput remains flat after this point, as expected according to simple asymptotic bound analysis arguments [16]. For the experiments with ACF, we see that throughput increases very slowly as MPL increases, consistent with the very slow growth of utilization at the bottleneck queue.

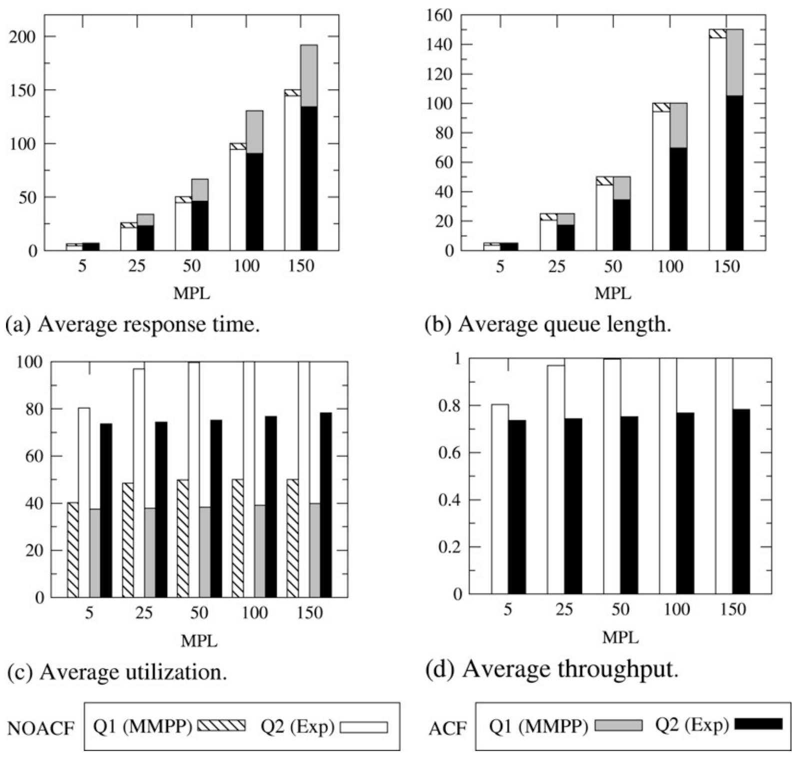


Fig. 5. Performance measures: (a) mean round trip time, (b) mean queue length, (c) mean utilization, and (d) mean throughput at each queue for *Scenario* 2. In all experiments the service time in Q_2 (bottleneck queue) is exponentially distributed. NOACF indicates that Q_1 has independent service times, ACF indicates that Q_1 has autocorrelated service times.

Indeed, maximum throughput and the corresponding 99.5% utilization are reached with much higher MPL= 3000 for the ACF experiments. Appendix B gives the system utilization, as well as the throughput, under MPLs up to 3000. Therefore, throughput and utilization, metrics that are easily obtainable from measurements and are prevalently used to gauge system capacity, give a distorted view of the user-perceived performance. Our observations are summarized as follows.

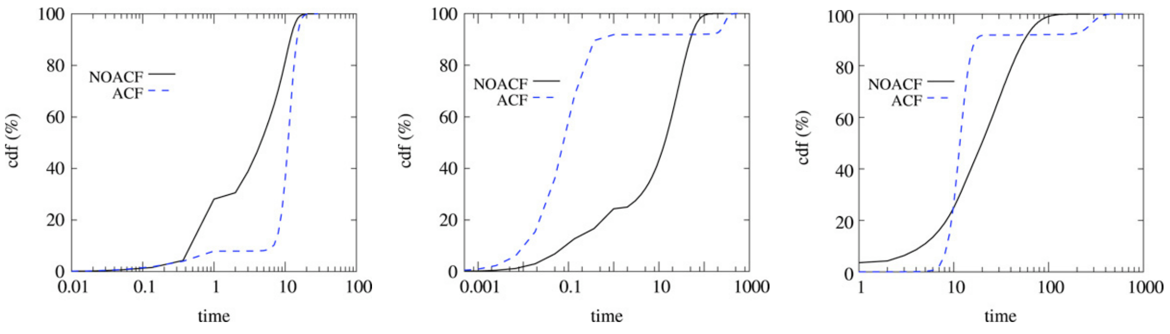
Observation 2. *Autocorrelated flows in a closed system degrade overall system performance, i.e., increase mean round trip time and decrease mean throughput. They also decrease the expected utilization of each queue, including the expected utilization at the bottleneck device.*

Observation 3. *Autocorrelated flows in a closed system balance the load among all queues, i.e., decrease mean queue length and mean response time of the bottleneck queue and increase those of the non-bottleneck queue. Counter-intuitively, despite this balancing, overall performance measures (i.e., round trip time) become worse.*

These observations have an important effect on capacity planning. If autocorrelated flows exist in the system, then reduced utilization levels at a queue do not mean that the system can sustain more load.

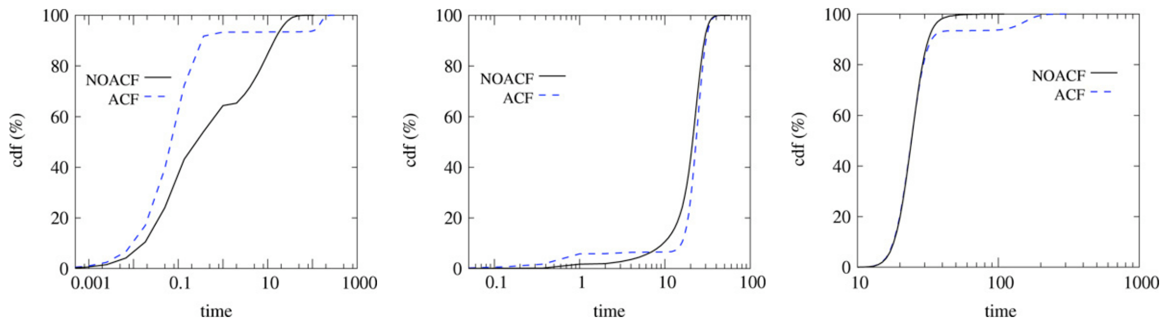
To better understand where each job spends most of its time waiting, we plot in Figs. 6 and 7 the CDFs of response times (per queue and round trip) for the ACF and NOACF experiments. For a substantial range of response times the system with ACF is better in comparison to the experiment with no autocorrelation (see the cross-over points in CDFs in Figs. 6(c) and 7(c) as well as cross-over points in CDFs of per-queue response times). However, response time tails at queues with ACF in their service process (Figs. 6(b) and 7(a)) dominate tails of round trip times and significantly bias mean response times in contrast to the NOACF experiments when the bottleneck device determines the tails of round trip times.

Observation 4. *In a closed system, the service process with autocorrelated structure (be it in the bottleneck queue or not) is the source of tails in the end-to-end response times and dominates average performance measures.*



(a) Queue 1 response time, MPL = 25. (b) Queue 2 response time, MPL = 25. (c) Round trip time, MPL = 25.

Fig. 6. CDFs of (a) response time at Q_1, (b) response time at Q_2, and (c) round trip time for *Scenario* 1 with MPL= 25. Q_2 remains the bottleneck queue and Q_1 is exponentially distributed. The bottleneck queue Q_2 has autocorrelated service times in the experiment labeled ACF, and has independent service times in the experiment labeled NOACF.



(a) Queue 1 response time, MPL = 25. (b) Queue 2 response time, MPL = 25. (c) Round trip time, MPL = 25.

Fig. 7. CDFs of (a) response time at Q_1, (b) response time at Q_2, and (c) round trip time for *Scenario* 2 with MPL= 25. Q_2 remains the bottleneck queue and its service process is exponentially distributed. The non-bottleneck queue Q_1 has autocorrelated service times in the experiment labeled ACF, and has independent service times in the experiment labeled NOACF.

The immediate implication of the above observation is that capacity planning or admission control at a queue with autocorrelation that aims at reducing response time tails, should apply to the queue that is the source of ACF in the system to obtain significant performance improvements.

4. Experimental case study: TPC-W

In this section, we present a case study based on the TPC-W benchmark and report on the existence of autocorrelated flows in a multi-tiered system that is built according to the TPC-W specifications. TPC-W is a widely used e-commerce benchmark that simulates the behavior of a Business-to-Consumer (B2C) site [10]. A high-level overview of the experimental set-up is illustrated in Fig. 8 and specifics of the software/hardware used are given in Table 2. Fig. 8 also illustrates the flow of requests from the clients to the front server (which hosts the web and application servers) and the back-end database server. We concentrate on the service demands of dynamic requests. We opt to put both the web server and the application server on the same machine because the web server simply forwards dynamic requests to the application server. Our experiments show that images, i.e., static content attached to each dynamically generated page, are served directly by the web server without the involvement of the application server. All images are in the front server's memory and their service times are negligible when compared to that of dynamic requests, thus service at the front server is dominated by the work of the application server.

According to TPC-W specifications, the number of customers or emulated browsers (EBs) is kept constant throughout the experiment. For each EB, TPC-W statistically defines the user session length, the user think time, and the queries that are generated by the session. Think times are exponentially distributed with mean 7 s. Four Pentium 4 machines are used to simulate the EBs. If there are n EBs in the system, each machine simulates $n/4$

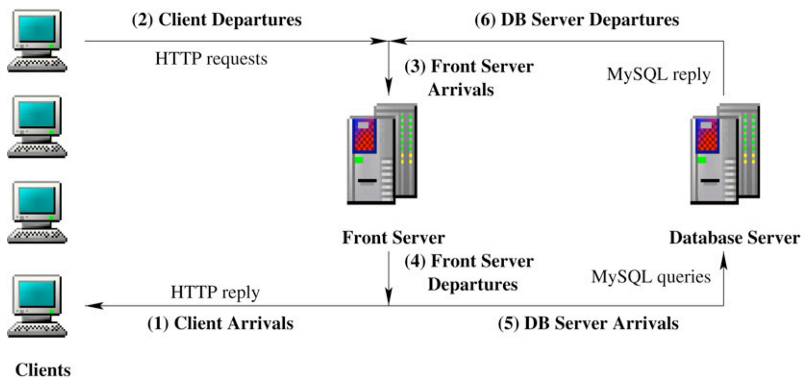


Fig. 8. TPC-W experimental environment.

Table 2
Hardware components of the TPC-W system

	Processor	Memory	OS
Clients (emulated browsers)	Pentium 4/2 GHz	256 MB	Linux Redhat 9.0
Web server-Apache2.0/Tomcat4.0	Pentium III/1.3 GHz	2 GB	Linux Redhat 9.0
Database serve-MySQL4.1	Intel Xeon/1.5 GHz	1 GB	Linux Redhat 9.0

EBs. We collect measurements at the client machines, the front-end server, and the back-end database server. Data is collected at several points as illustrated in Fig. 8. Specifically, we record

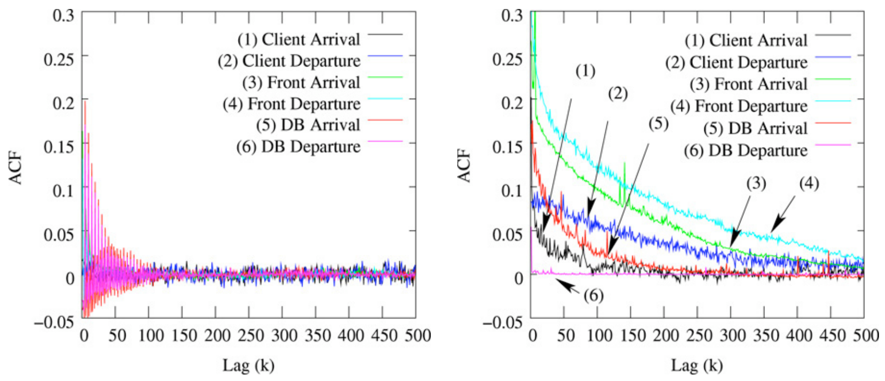
- all responses sent from the front-end server to the client at point (1) labeled “client arrivals”, collected at the workload generation modules;
- all requests sent from the clients to the front-end server at point (2) labeled “client departures”, collected at the workload generation modules;
- all requests received by the front server (i.e., both client requests and database responses) at point (3) labeled “front server arrivals”, collected at the workload generation modules and MySQL logs;
- all responses sent from the front-end server (i.e., to both the clients and database queries) at point (4) labeled “front server departures”, collected at the workload generation modules and MySQL logs;
- all queries sent from the front server to the database server at point (5) labeled “DB server arrivals”, collected at the MySQL logs;
- all query results sent from the database to the front server at point (6) labeled “DB server departures”, collected at the MySQL logs.

TPC-W defines several customer interactions. Here, we report on the “browsing mix”, one of the default TPC-W workload mixes.

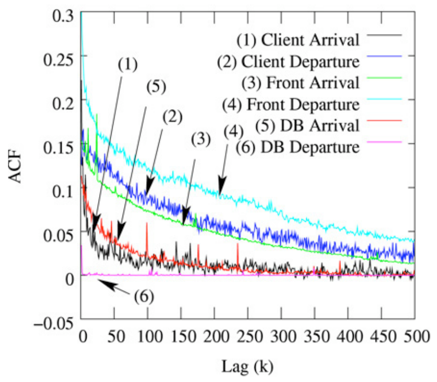
4.1. Experimental measurements

Fig. 9 shows the measured ACF at the various points indicated in Fig. 8 for three browsing mix experiments and a database of 10,000 items for different number of concurrent EBs in the system (i.e., different workload intensities). Fig. 10 plots the average queue lengths, average response times, and average CPU utilizations, at the clients, front-end, and database servers.

With only 128 EBs, the system is lightly loaded, the front-end utilization is 20% and the database server utilization is 38%. The ACF in Fig. 9(a) is very close to zero in almost all the measurement points. The ACF at point (5), i.e., at the DB arrival process, is oscillating at low lags (from -0.05 to 0.2) and quickly decreases to nearly zero. The ACF of the database departure process, i.e., at point (6), follows the ACF of arrivals, consistent with the discussion in Section 3.2, where we showed that under low load in a queuing system the arrival process rather than the service process determines the shape of the departure process. Looking carefully into the traces, we notice that the source

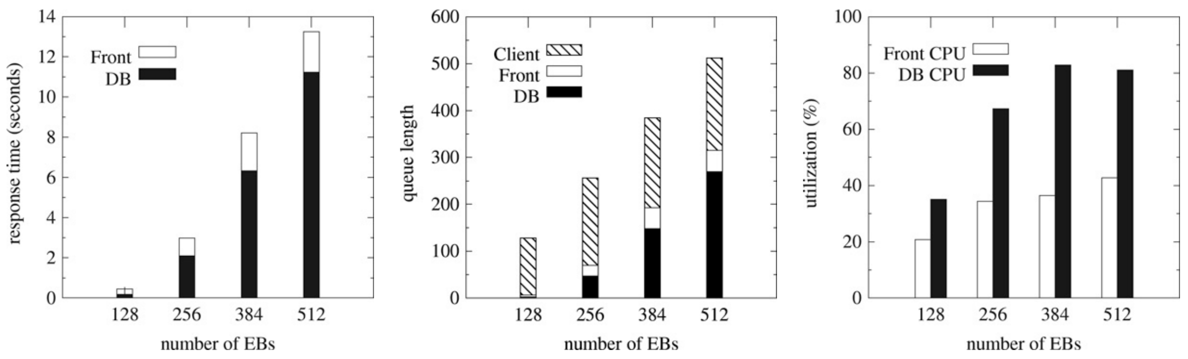


(a) 128 EBs. (b) 384 EBs.



(c) 512 EBs.

Fig. 9. ACF at various points in the system. Experiments are done using the browsing mix, a database with 10,000 items, and (a) 128 EBs, (b) 384 EBs, and (c) 512 EBs.



(a) Average response time. (b) Average queue length. (c) Average utilization.

Fig. 10. Average performance measures with the browsing mix.

of the correlated arrivals to the database comes from the JDBC drivers connecting Tomcat Java servlets and MySQL database server as one long query usually follows several small queries there.

As we increase the number of EBs in the system to 384, the system load increases to 38% utilization at the front-end and to 82% at the bottleneck DB server. This is a case of heavy load, where oscillating ACF values are not observed anymore. Fig. 9(b) shows that autocorrelation is higher now in almost all measurement points, with the exception of points (1) and (6) which represent client arrivals and database departures, respectively. Experiments with 512 EBs capture very similar, although much stronger, trends on ACF propagation as depicted in Fig. 9(c). Inspecting the ACFs

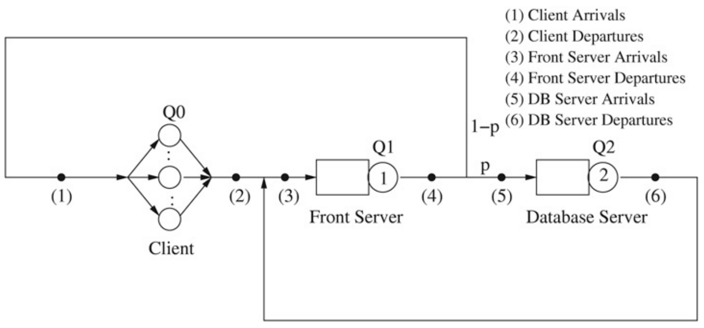


Fig. 11. A queuing model of TPC-W.

at points (3) and (4) in Fig. 9(b)–(c), we infer that the service process at the front-end server is correlated because ACF at its departure point is much higher than ACF in its arrivals. The autocorrelation measured at point (4) is the strongest among all measurement points, and becomes even stronger as load increases, see Fig. 9(c). These ACF values suggest that there is no correlation in the DB service process as measurements at point (6) show a flat-to-zero ACF line – the service process in the DB "takes away" the correlation in the flow of arrivals.

Average performance values are presented in Fig. 10. Despite the fact that queue lengths and response times increase fast as a function of EBs (see Fig. 10(a) and (b)), utilization levels increase very slowly, consistently with the results presented in Section 3.3.

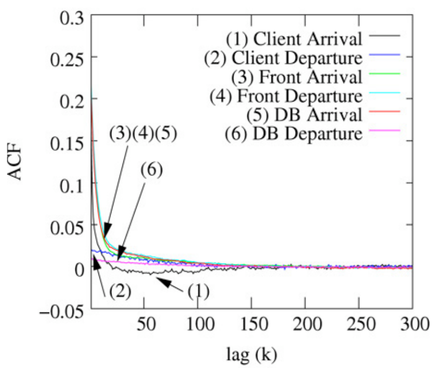
We have conducted several experiments using different TPC-W workload mixes (i.e., shopping and ordering) and larger database sizes (i.e., 100,000 and 1,000,000 items). These experiments are not presented here due to lack of space but can be summarized as follows. The amount of ACF that propagates through the system and measured at various points is different for the three TPC-W workload mixes. This is expected as each workload has different service demands. We stress that autocorrelated flows are *not* always observed. In some experiments there is very little or no autocorrelation. For the cases that ACF is observed, we attribute its presence to autocorrelated service processes in the front and/or database servers because the workload generation at the EBs guarantees that there is no autocorrelation in the arrival process coming from the clients. In the following section, we present a simple model that captures the performance trends observed here and that confirms our conjecture about the existence of autocorrelation in the service process.

4.2. TPC-W model

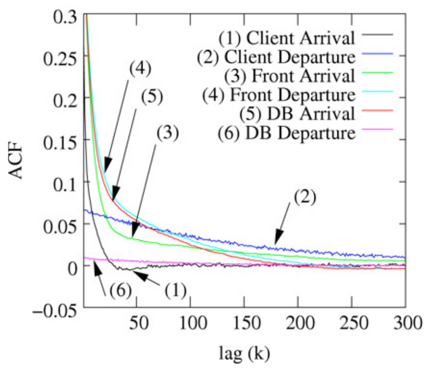
Here, we present a model of a closed queuing network that captures the behavior observed in the TPC-W testbed of Section 4.1 and confirms our speculations. The model is illustrated in Fig. 11. Queues Q_1 and Q_2 correspond to the front-end server and the back-end database server, respectively. Because the TPC-W benchmark is session-based, we use a queue with as many servers as the system's MPL to emulate client activity (Q_0). The collected TPC-W trace data shows that each dynamic request at the application server generates several database requests. We capture this behavior by adding a feedback loop: with probability p a completed request from Q_1 is forwarded to queue Q_2 and with probability $1 - p$ it goes back to the client, in Q_0. We also define the same six measurement points as in the real system of Fig. 8.

This model is solved using simulation and is parameterized using the measurements from our TPC-W testbed under light load, (i.e., when there is virtually no queuing). Measuring the service process in each of the system tiers is not straightforward but in a lightly loaded system that guarantees nearly zero wait times, response times give a good approximation of service times.

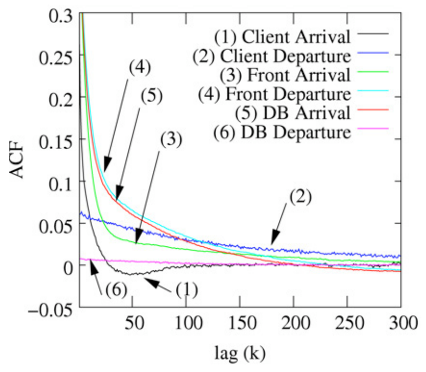
- The *think time* in each server of Q_0 is exponentially distributed with mean 7 s, as specified by TPC-W.
- We use a 2-state MMPP to generate service times in the front-end server with rate $\mu_1 = 582.70$ requests per second and $\text{SCV}_1 = 20$ (as measured in our TPC-W testbed). This MMPP has autocorrelation which is equal to 0.47 at lag 1 and decays to nearly zero at lag 300. Note that we do not perform a rigorous fitting to capture the exact shape of autocorrelation in the service process, we simply match the first two moments of the measurement data and adjust the MMPP parameters in order to induce autocorrelation.



(a) MPL = 128.



(b) MPL = 384.



(c) MPL = 512.

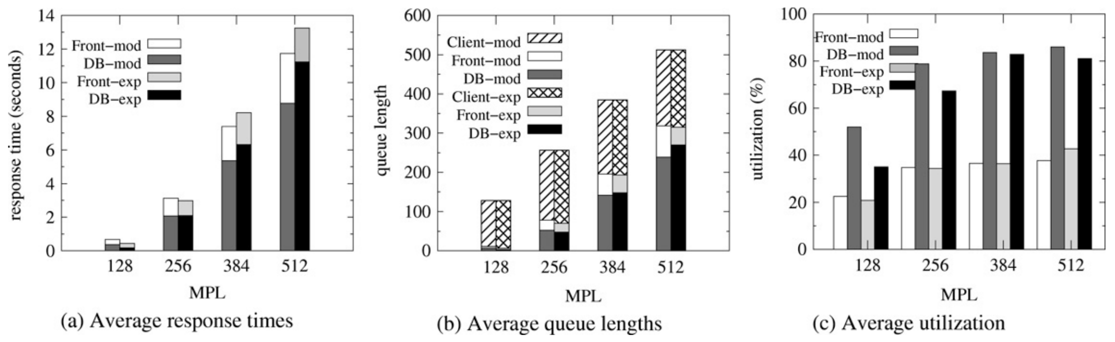
Fig. 12. Autocorrelation propagation in our queuing model parameterized using the measurements of Section 4.1 with MPL equal to (a) 128, (b) 384, and (c) 512.

- Service times at the database server are generated using a 2-stage hyperexponential with $\mu_2 = 224.34$ requests per second and $\text{SCV}_2 = 100$ (as measured in our TPC-W testbed).
- The probability p is set to 0.876, which is again obtained by our measurements.

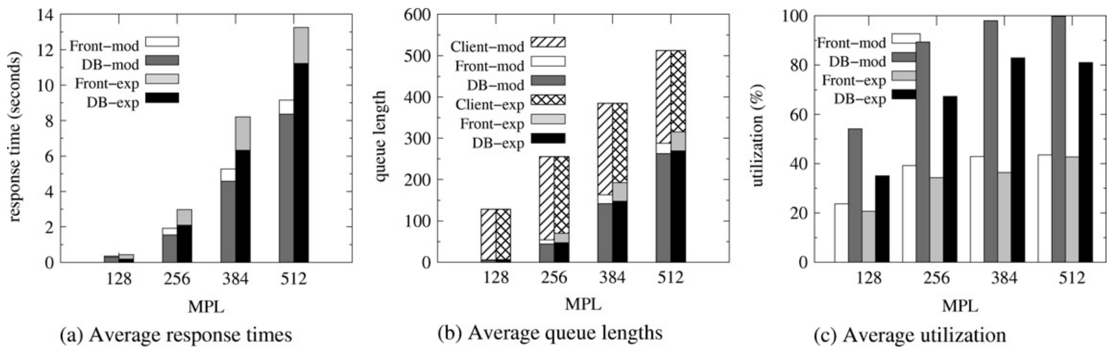
Fig. 12(a)–(c) show the ACF propagation with MPL set to 128, 384, and 512, respectively. The queuing model captures well the autocorrelation trends observed in the TPC-W experiments (compared with Fig. 9). Consistently with experimental results, the departure intervals from the front-end server at (4) have the highest autocorrelation. The slowest decaying ACF is at point (2), i.e., the departures from the clients. Note that the independent service process at the database server results in independent departures at (6), which also minimally affects arrivals at the clients at (1). Fig. 12 verifies our speculation about the existence of autocorrelation in the service process of the front-end server. Autocorrelation only at the service process of Q_1 causes the entire system to operate under almost independent flows when the load is low (Fig. 12(a)) because the workload generation at the clients (which is driven by the exponential distribution) dominates the departure and arrival processes at all the queues. When the load is high (Fig. 12(b) and (c)), the autocorrelated service process at Q_1 (i.e., front-end server) dominates the departure process at Q_1 and as showed in Section 3.2 propagates in the entire closed system.

Fig. 13-(I) presents average performance measures obtained from the proposed model and, to facilitate comparison, also the corresponding measurements from our TPC-W testbed. Model and measurement results are in excellent agreement, despite the fact that the simple fitting used here qualitatively captures autocorrelation structure of the service process at the front-end server. The agreement between model and measurements diminishes quickly if the same model assumes uncorrelated processes throughout the closed system, as shown in Fig. 13-(II).[3] Omitting

[3] The service process at the front-end server is again an MMPP with the same mean, SCV, and higher moments as the MMPP that models the correlated service process, but has no correlation, i.e., ACF equals 0 for all lags.



(I) ACF model (successful match).



(II) No ACF model (unsuccessful match).

Fig. 13. Model prediction and experimental performance measures, where the service processes at the front-end server in the proposed model are (I) autocorrelated and (II) uncorrelated.

autocorrelation in the model results in underestimated response times and queue lengths as well as overestimated per-queue utilization for all tiers in general. The gap between measured and modeled performance metrics is particularly high for the tier that is measured to have correlated service process (i.e., the front-end server in our TPC-W system).

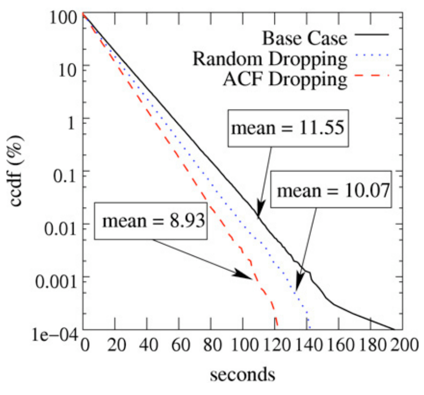
5. Taking advantage of ACF

After having established the importance of autocorrelated flows for the performance of multi-tiered systems, we now turn to how to use this information for effective system design. In general, capturing burstiness in the flows of complex systems can be used to implicitly model caching, context switching overhead, contention for memory or locks, while keeping the model surprisingly simple. Here, we present a case study that illustrates how ACF can guide admission control. Naturally, a myriad of policies exist for admission control. Presenting an ideal admission control policy is outside the scope of this work. Instead, we focus on how to use knowledge of autocorrelated flows to improve policy development.

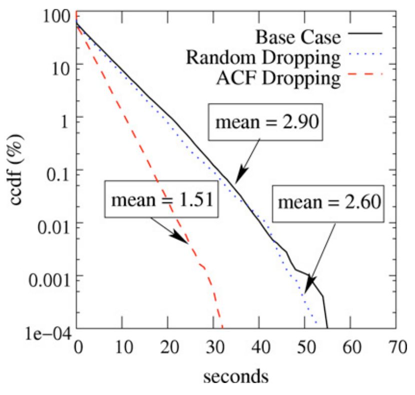
In Section 3 we show that the server with ACF in its service process is the one that most contributes to the response time tails. Based on this observation, we devise a simple admission control strategy that rejects the jobs which are highly probable to contribute to the long tail of round trip times. Identification of these jobs is based on the temporal locality of autocorrelated flows.

Assuming that we know a priori which is the tier that is the source of autocorrelation, we deploy admission control at that tier.[4] Admission control is triggered when the queue length at the ACF tier reaches a pre-defined threshold Q_T of MPL. Upon each job completion, the current queue length is checked to see whether it exceeds threshold Q_T. If this is the case, then the request at the head of the waiting queue is dropped (i.e., directed back to Q_0, the client

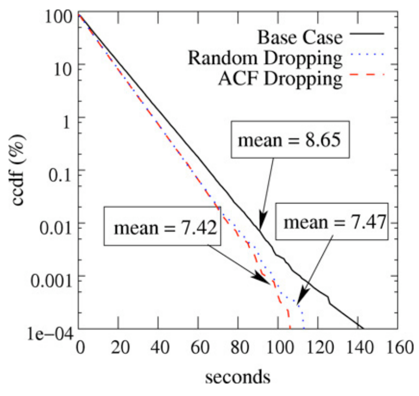
[4] Even if the autocorrelation function of the flow in the tier is not known a priori, it is possible to calculate it on-line using a modified version of Welford's one-pass algorithm to calculate the mean and variation of a sample [28].



(a) Round trip time, ACF in front server.



(b) Front server response time, ACF in front server.



(c) DB server response time, ACF in front server.

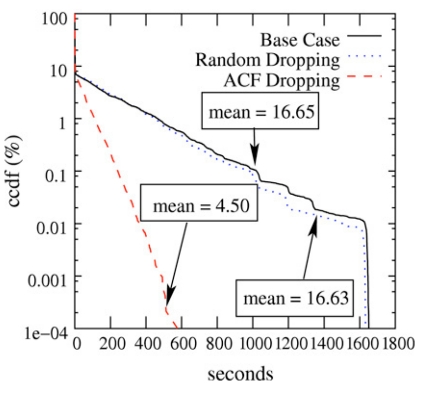
Fig. 14. CCDFs of (a) round trip time, (b) response time of front server, (c) response time of database server using the model of Section 4.2 where the front server has ACF in its service process. In all experiments MPL is equal to 512.

queue), with probability weighed by the ACF value of the stream at lag(1), provided that ACF has a positive value. Then, the next waiting request is also dropped with a probability weighed by the ACF value at lag(2). The dropping of waiting requests stops when the queue length reaches Q_T or a job is admitted for service.[5]

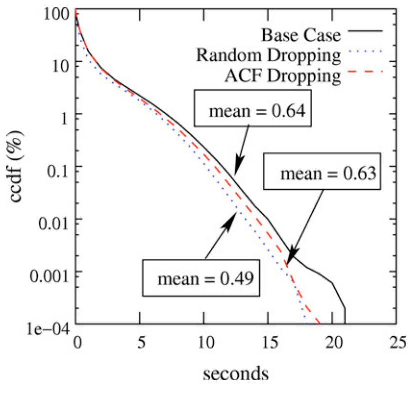
We use the model in Section 4 to evaluate this admission control policy. The base line for the evaluation is the case with no admission control. For comparison, we also evaluate a policy with random dropping at the same tier. Similar to the ACF-guided policy, the random policy drops always from the head of the waiting queue with probability equal to the overall dropping rate of the ACF-guided admission control policy. This way, we maintain equal dropping rates in both admission control policies.

We first evaluate exactly the same setting as in Section 4.2, i.e., the front server has ACF in its service process that starts at 0.47 for lag 1 and decays to nearly zero beyond lag 300. MPL is set to 512. Q_T is defined as 60% of MPL because the front tier is not the bottleneck and it is expected to be less loaded than the DB tier. Consistently with experiments presented in Section 4, we assume that the service process at the DB tier is not correlated and that the DB is the bottleneck. The dropping rate for the ACF-guided policy is 8.2% and average round trip times become 8.93 s. The round trip times under the no-dropping and random dropping scenarios are 11.55 and 10.07 s, respectively. To focus on tail performance, Fig. 14 illustrates the complementary cumulative distribution function (CCDF) of round trip times and of response times at the front server and the database server. The figure shows that ACF-guided policy improves the tail of the front server response times and respectively round trip times, given that the tails of response times at the DB server of the two admission control policies are almost identical.

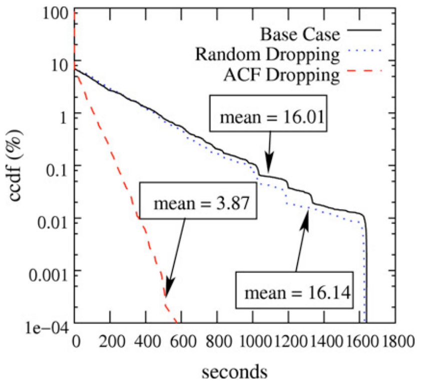
[5] We experimented with different probability weights that gave us qualitatively similar performance. Here, we present results with weight equal to 1, i.e., we use directly the ACF function as the dropping probability.



(a) Round trip time, ACF in DB server.



(b) Front server response time, ACF in DB server.



(c) DB server response time, ACF in DB server.

Fig. 15. CCDFs of (a) round trip time, (b) response time of front server, (c) response time of database server when the database server has ACF in its service process. In all experiments MPL is equal to 512.

We now use the model of Section 4.2 but assume that the bottleneck tier (i.e., the database server) has ACF in its service times. This is motivated by the disk ACF service times in Fig. 1. Again MPL is 512 but Q_T is now set to 90% of the MPL, because the DB is the bottleneck and we expect most of the jobs to be stuck there. The ACF-guided admission control drops only 5.8% of the total requests and achieves an average round trip of 4.50 s. Round trip times with no-dropping and random dropping are 16.65 and 16.63 s, respectively. Both round trip times and database server response times significantly improve with the ACF-guided policy (see the CCDFs in Fig. 15). With random dropping, improvements are very small. Both experiments, although preliminary, confirm that selective dropping as guided by ACF can dramatically improve performance. By selectively dropping those requests that contribute most to ACF, the queue lengths in the queue with autocorrelation significantly reduce, the ACF flows in the entire system weaken, and performance in every server improves.

Similar analysis can be done for capacity planning studies. There, the focus should be on first identifying and then bolstering the server that is the source of autocorrelation, which is not necessarily the system bottleneck.

6. Related work

Burstiness as expressed by self-similarity has been identified as a salient characteristic of traffic in communication networks [6,7,17,22] that critically impacts their capacity and performance [3,17]. Burstiness as a form of self-similarity has also been shown to exist in computer systems including CPU utilization levels in a cluster of workstations [29], inter-arrival times at a large memory system with nonblocking caches [24], and file system activity [11]. The above works concur that burstiness results in unpredictability of system performance and argue for feedback-control frameworks to dynamically adapt resource allocation to changing service demands [5,25].

Workload characterization studies that focus on bottleneck identification of multi-tiered systems [10] aim at guiding the development of admission control strategies that improve peak throughput [8,14], prioritized scheduling at the database server to meet different service level agreements [19], and scheduling policies that minimize consistency overheads in clustered environments that support query caching and database replication [2]. None of the above characterization studies of multi-tiered systems has identified autocorrelation as an important system characteristic for performance.

The effect of autocorrelation in open systems has been examined in [9] where it was shown via simulation that long-range dependence in the arrival process of a single queue results in sharp performance degradation. Similar results are reported in [1] where the performance effects of short-range dependence versus long-range dependence in the arrival streams are examined. In [31], it is shown that the performance benefits of size-based load balancing policies in homogeneous clusters quickly vanish if the arrival process in the system is autocorrelated and that load unbalancing policies offer superior performance in clusters that admit dependent arrivals. In [13], the authors provide some discussion on the perils of modeling TCP/IP-based networks as open system. In [18], Li and Hwang point out that the input traffic stream in high speed networks is highly correlated and use discrete spectral analysis for modeling the input correlation functions.

Traditionally, models of multi-tiered systems focus on modeling the bottleneck tier [23,27] or modeling all tiers using a single queue [14]. A closed-system model of a multi-tiered system that is based on Mean Value Analysis (MVA) and does consider all tiers has been proposed in [26]. Aggregation of models of individual resource demands and interaction overheads for each tier in a multi-tiered system is also proposed to predict system throughput and response times [25].

To the best of our knowledge, this is the first time that autocorrelation is proposed as a compact characterization of burstiness in multi-tiered systems. Our work further demonstrates that stochastic processes that capture autocorrelation can be used in surprisingly simple models that can effectively capture performance trends of burstiness in complex systems.

7. Conclusions and future work

We presented a performance evaluation study that shows the presence of autocorrelated flows in a multi-tiered system with a closed-loop structure and their performance effects. Via queuing models we have shown that identifying autocorrelated flows in such a system is critical for capacity planning. If autocorrelated flows are ignored, then throughput and utilization of specific devices – metrics often used in capacity planning and admission control – may give a distorted view of system load. Using a TPC-W benchmark system, we have demonstrated that autocorrelated flows propagate across the entire system and can originate from the stochastic behavior in the service processes of only one of the tiers. We have shown that stochastic processes that capture autocorrelation may be used to effectively model even complex systems via simple queuing models.

To the best of our knowledge, this is the first time that autocorrelated flows are identified as an important stochastic characteristic in multi-tiered systems with a closed-loop structure, i.e., systems that accept arrivals from the outside world but have some limitation in their incoming buffer size. Future work on the theoretical side will be to develop analytic models of closed systems that consider autocorrelation. Such models could be used to identify the conditions in a closed system that trigger autocorrelation, as well as the conditions that augment or reduce autocorrelation. On the systems side, we are working on new resource management techniques that take autocorrelation into account. We also intend to widen the set of workloads considered and understand in more detail how autocorrelation arises in the service process.

Acknowledgments

We would like to thank Emilia Rosti, Larry W. Dowdy, Giuliano Casale, and Gianfranco Ciardo for insightful discussions and useful feedbacks in this work. This work was partially supported by the National Science Foundation under grant ITR-0428330 and by Seagate Research.

Appendix A

The analysis and evaluation of systems that operate under correlated arrival or service processes is facilitated by various analytic models that capture correlation, see [4] for an overview. Matrix-analytic methods [15] provide a tractable framework for evaluation of queuing systems with correlated processes [20].

In MAP notation, a 2-state MMPP process is represented by two 2×2 matrices, i.e., $\mathbf{D}_0$ and $\mathbf{D}_1$. The matrix $\mathbf{D}_0$ captures the variability in the process and $\mathbf{D}_1$ captures the dependence structure of the process. The matrix $\mathbf{D}_1$ for an MMPP process has non-zero elements only in its main diagonal. If matrix $\mathbf{D}_1$ has only one non-zero element in its main diagonal, then the MMPP process is independent and reduces to a phase-type (PH) renewal process.

A.1. MMPPs used in Section 3

Eqs. (1) and (2) describe 2-state MMPPs with mean service time of 1 and squared coefficient of variation (SCV) equal to 20 used in Scenario 1 of Section 3. The MMPP of Eq. (1) represent a correlated stochastic process while Eq. (2) represents an independent stochastic process.

$$D_0^{(S)} = \begin{bmatrix} -12.01027354 & 0.01027353645 \\ 0.000852559306 & -0.08800697603 \end{bmatrix}, \qquad D_1^{(S)} = \begin{bmatrix} 12 & 0 \\ 0 & 0.08715441672 \end{bmatrix}. \tag{1}$$

$$D_0^{(S)} = \begin{bmatrix} -13.06140351 & 1.061403509 \\ 0.09649122807 & -0.09649122807 \end{bmatrix}, \qquad D_1^{(S)} = \begin{bmatrix} 12 & 0 \\ 0 & 0 \end{bmatrix}. \tag{2}$$

Eqs. (3) and (4) describe the MMPPs with mean $\mu = 2$ and SCV $= 20$ used in Scenario 2 of Section 3. The MMPP of Eq. (3) represents a correlated stochastic process while the MMPP of Eq. (4) represents an independent stochastic process.

$$D_0^{(S)} = \begin{bmatrix} -12.00861079 & 0.008610792193 \\ 0.001584619872 & -0.1613127422 \end{bmatrix}, \qquad D_1^{(S)} = \begin{bmatrix} 12 & 0 \\ 0 & 0.1597281223 \end{bmatrix}. \tag{3}$$

$$D_0^{(S)} = \begin{bmatrix} -12.87719298 & 0.8771929825 \\ 0.1754385965 & -0.1754385965 \end{bmatrix}, \qquad D_1^{(S)} = \begin{bmatrix} 12 & 0 \\ 0 & 0 \end{bmatrix}. \tag{4}$$

A.2. MMPPs used in Section 4.2

In Section 4.2, we use a 2-state MMPP to describe the correlated service process (see Eq. (5)) at the front-end server and a PH-type renewal process to describe the independent but highly variable service process at the back-end database server (see Eq. (6)).

$$D_0^{(S)} = \begin{bmatrix} -2001.004655 & 1.004654668 \\ 0.3846421784 & -40.45703417 \end{bmatrix}, \qquad D_1^{(S)} = \begin{bmatrix} 2000 & 0 \\ 0 & 40.07239199 \end{bmatrix}. \tag{5}$$

$$D_0^{(S)} = \begin{bmatrix} -1000 & 0 \\ 0 & -3.46114928 \end{bmatrix}, \qquad D_1^{(S)} = \begin{bmatrix} 987.9914443 & 12.00855569 \\ 3.419585876 & 0.04156340391 \end{bmatrix}. \tag{6}$$

Appendix B

Fig. 16 presents per-queue utilization and system throughput as a function of MPL for a two-queue closed system. The left two plots in Fig. 16 correspond to *Scenario* 1 and the right two plots correspond to *Scenario* 2, described in Section 3.1. If the traffic flows between the two queues in the closed system are not correlated, the bottleneck queue, i.e., Q_2, quickly becomes 100% utilized for MPL as high as 50. Consequently, the utilization of the non-bottleneck queue, Q_1, reaches 50% when MPL = 50. The utilizations of Q_1 and Q_2 remain flat for MPLs higher than 50, in both scenarios. In contrast, if the traffic flows between the two queues in the closed system are correlated then utilization of both queues increases slowly as MPL increases (i.e., much slower than in the case of uncorrelated flows). Specifically, the utilization of the bottleneck queue Q_2 reaches 96.1% and 99.5% for *Scenario* 1 and *Scenario 2*, respectively, only for MPL as high as 3000. Consistently with the utilization behaviors, the system with no ACF reaches its maximum throughput when MPL = 50 while the throughput grows much slower when the system is correlated.

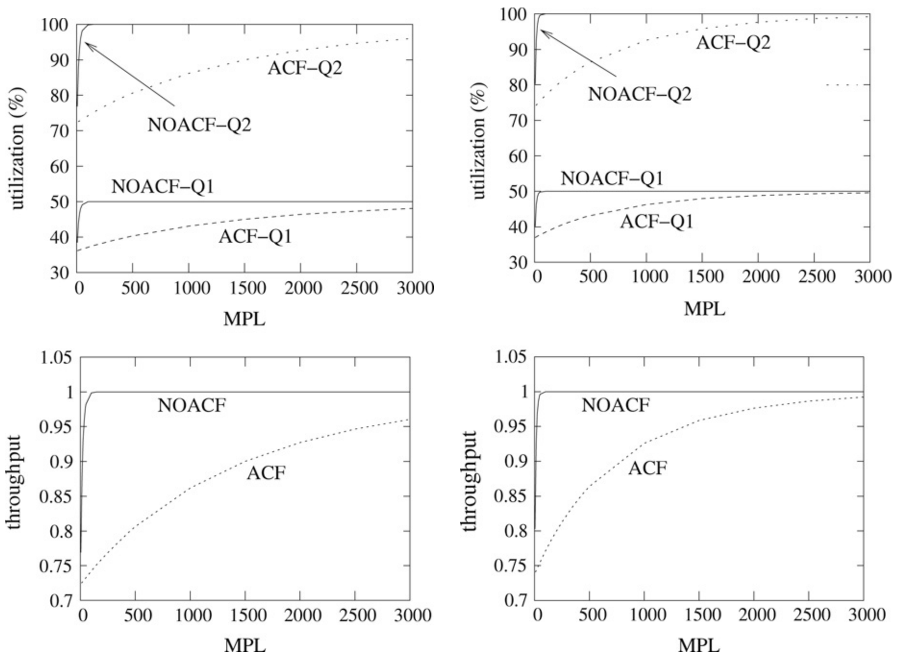


Fig. 16. The mean system utilization at each queue and the system throughput for (a) *Scenario 1* and (b) *Scenario* 2.

References

[1] A.M. Adas, A. Mukherjee, On resource management and QoS guarantees for long range dependent traffic, in: INFOCOM, 1995, pp. 779–787.
[2] C. Amza, A. Cox, W. Zwaenepoel, A comparative evaluation of transparent scaling techniques for dynamic content servers, in: Proceedings of ICDE, 2005, pp. 230–241.
[3] G. Banga, P. Druschel, Measuring the capacity of a Web server, in: Proceedings of USITS, Monterey, CA, Dec. 1997.
[4] J. Beran, Statistics for Long-Memory Processes, Chapman & Hall, New York, 1994.
[5] I. Cohen, J.S. Chase, M. Goldszmidt, T. Kelly, J. Symons, Correlating instrumentation data to system states: A building block for automated diagnosis and control, in: OSDI, 2004, pp. 231–244.
[6] M.E. Crovella, A. Bestavros, Self-similarity in World Wide Web traffic: Evidence and possible causes, in: SIGMETRICS, ACM Press, 1996, pp. 160–169.
[7] R.L. Cruz, Service burstiness and dynamic burstiness measures: A framework, Journal of High Speed Networks 1 (2) (1992).
[8] S. Elnikety, E. Nahum, J. Tracey, W. Zwaenepoel, A method for transparent admission control and request scheduling in e-commerce web sites, in: Proceedings of the 13th International Conference on World Wide Web, ACM Press, 2004, pp. 276–286.
[9] A. Erramilli, O. Narayan, W. Willinger, Experimental queueing analysis with long-range dependent packet traffic, IEEE/ACM Transactions on Networking 4 (2) (1996) 209–223.
[10] D. Garcia, J. Garcia, TPC-W E-commerce benchmark evaluation, IEEE Computer (2003) 42–48.
[11] S.D. Gribble, G.S. Manku, D. Roselli, E.A. Brewer, T.J. Gibson, E.L. Miller, Self-similarity in file systems, in: Proceedings of the ACM SIGMETRICS, ACM Press, 1998, pp. 141–150.
[12] W. Jin, J.S. Chase, J. Kaur, Interposed proportional sharing for a storage service utility, in: ACM SIGMETRICS/Performance, 2004, pp. 37–48.
[13] Y. Joo, V. Ribeiro, A. Feldmann, A.C. Gilbert, W. Willinger, TCP/IP traffic dynamics and network performance: A lesson in workload modeling, flow control, and trace-driven simulations, SIGCOMM Computer Communication Review 31 (2) (2001) 25–37.
[14] A. Kamra, V. Misra, E. Nahum, Yaksha: A self-tuning controller for managing the performance of 3-tiered web sites, in: Proceedings of International Workshop on Quality of Service, IWQoS, 2004, pp. 47–58.
[15] G. Latouche, V. Ramaswami, Introduction to Matrix Analytic Methods in Stochastic Modeling, in: ASA-SIAM Series on Statistics and Applied Probability, SIAM, Philadelphia PA, 1999.
[16] E.D. Lazowska, J. Zahorjan, G.S. Graham, K.C. Sevcik, Computer System Analysis Using Queueing Network Models, Prentice-Hall, Inc, New York, 1984.
[17] W.E. Leland, M.S. Taqqu, W. Willinger, D.V. Wilson, On the self-similar nature of Ethernet traffic, IEEE/ACM Transactions on Networking 2 (1994) 1–15.
[18] S.-Q. Li, C.-L. Hwang, Queue response to input correlation functions: Discrete spectral analysis, IEEE/ACM Transactions on Networking 1 (5) (1993) 522–533.

[19] D. McWherter, B. Schroeder, N. Ailamaki, M. Harchol-Balter, Priority mechanisms for OLTP and transactional web applications, in: 20th International Conference on Data Engineering, ICDE 2004, Boston, MA, April 2004.

[20] B.F. Nielsen, Modelling long-range dependent and heavy-tailed phenomena by matrix analytic methods, in: G. Latouche, P. Taylor (Eds.), Advances in Algorithmic Methods for Stochastic Models, Notable Publications, 2000, pp. 265–278.

[21] R.O. Onvural, H.G. Perros, Equivalencies between open and closed queueing networks with finite buffers, Performance Evaluation 9 (1989) 263–269.

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

[23] S. Ranjan, J. Rolia, H. Fu, E. Knightly, QoS-driven server migration for internet data centers, in: Proceedings of the Tenth IEEE International Workshop on Quality of Service, IWQoS 2002, Miami Beach, FL, May 2002.

[24] D.J. Sorin, J.L. Lemon, D.L. Eager, M.K. Vernon, Analytic evaluation of shared-memory architectures, IEEE Transactions on Parallel and Distributed Systems 14 (2) (2003).

[25] C. Stewart, K. Shen, Performance modeling and system management for multi-component online services, in: Proceedings of USENIX NSDI, Boston, MA, May 2005.

[26] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, A. Tantawi, An analytical model for multi-tier internet services and its applications, in: Proceedings of the ACM SIGMETRICS Conference, Banff, Canada, June 2005, pp. 291–302.

[27] D. Villela, P. Pradhan, D. Rubenstein, Provisioning servers in the application tier for E-commerce systems, in: Proceedings of the Twelfth IEEE International Workshop on Quality of Service, IWQoS 2004, Montreal, Canada, June 2004.

[28] B.P. Welford, Note on a method for calculating corrected sums of squares and products, Technometrics 4 (1962) 419–420.

[29] R. Wolski, N.T. Spring, J. Hayes, Predicting the CPU availability of time-shared Unix systems on the computational grid, Cluster Computing 3 (4) (2000) 293–301.

[30] Q. Zhang, The effect of workload dependence in systems: Experimental evaluation, analytic models, and policy development, Ph.D. Thesis, College of William and Mary, 2006.

[31] Q. Zhang, N. Mi, A. Riska, E. Smirni, Load unbalancing to improve performance under autocorrelated traffic, in: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, ICDCS'06, Lisboa, Portugal, July 2006, p. 20.

Ningfang Mi received her B.S. degree in Computer Science from Nanjing University, China, in 2000, and her M.S. degree in Computer Science from the University of Texas at Dallas, in 2004. She is currently a Ph.D. candidate in the Department of Computer Science, College of William and Mary, Williamsburg, VA (ningfang@cs.wm.edu). Her research interests include resource allocation policies, performance analysis of multi-tiered systems, workload characterization, and analytic modeling. She is a student member of the IEEE.

Qi Zhang currently is a software engineer in the Windows Server Performance team at Microsoft. She received her Ph.D. degree in Computer Science from College of William and Mary, Williamsburg, VA, USA, in December 2006. She got the B.S. degree in computer science from Huazhong University of Science and Technology, Hubei, China, in 1998, and the M.S. degree in computer science from the University of Science and Technology of China, Anhui, China, in 2001, respectively. Her research interests include performance evaluation, scheduling and load balancing policies, workload characterization and queuing modeling of multi-tiered systems, and departure processes. Qi Zhang is a member of ACM and IEEE.

Alma Riska received her Ph.D. in Computer Science from the College of William and Mary, in Williamsburg, VA, in 2002. Currently, she is a Research Staff Member at Seagate Research in Pittsburgh, Pennsylvania. Her research interests are on performance and reliability modeling of computer systems, in general, and storage systems, in particular. The emphasis of her work is on applying analytic techniques and detailed workload characterization in designing more reliable and better performing storage systems that can adapt their operating into the dynamically changing operational environment. She is a member of IEEE and ACM.

Evgenia Smirni is the Wilson and Martha Claiborne Stephens Associate Professor at the College of William and Mary, Department of Computer Science, Williamsburg, VA (esmirni@cs.wm.edu). She received her Diploma in Computer Engineering and Informatics from the University of Patras, Greece, in 1987, and her M.S. and Ph.D. in Computer Science from Vanderbilt University in 1993 and 1995, respectively. From August 1995 to June 1997 she had a postdoctoral research associate position at the University of Illinois at Urbana-Champaign. Her research interests include analytic modeling, stochastic models, Markov chains, matrix analytic methods, resource allocation policies, Internet systems, workload characterization, and modeling of distributed systems and applications. She has served as program co-chair of QEST'05 and of ACM SIGMETRICS/Performance'06. She is a member of ACM, IEEE, and the Technical Chamber of Greece.

Erik Riedel received his doctorate from Carnegie Mellon University in 1999 for work on Active Disks as an extension to Network-Attached Secure Disks. Erik currently leads the Interfaces & Architecture Department at Seagate Research in Pittsburgh, PA. His group focusses on novel storage devices and systems with increased intelligence to optimize performance, improve security, improve reliability, automate management, and enable smarter organization of data. The group's work targets Seagate products in enterprise, personal, networked, consumer and mobile storage.

Before joining Seagate, Erik was a researcher in the storage program at Hewlett-Packard Labs in Palo Alto, CA working on networked storage, distributed storage and security. He has authored and co-authored several granted patents and a number of pending patent applications, as well as numerous technical publications on a range of storage-related topics.