New Results on the Performance Effects of Autocorrelated Flows in Systems

Evgenia Smirni[1], Qi Zhang[1], Ningfang Mi[1], Alma Riska[2], and Giuliano Casale[1]

[1]College of William and Mary
Dept. of Computer Science
Williamsburg, VA
{esmirni,qizhang,ningfang,casale}@cs.wm.edu

[2]Seagate Research
Interfaces and Architecture
Pittsburgh, PA
alma.riska@seagate.com

Abstract

Temporal dependence within the workload of any computing or networking system has been widely recognized as a significant factor affecting performance. More specifically, burstiness, as a form of temporal dependency, is catastrophic for performance. We use the autocorrelation function in a workload flow to formalize burstiness and also to characterize temporal dependence within a flow. We present results from two application areas: load balancing in a homogeneous cluster environment and capacity planning in a multi-tiered e-commerce system. For the load balancing problem, we show that if autocorrelation exists in the arrival stream to the cluster, classic load balancing policies become ineffective and solutions that focus on "unbalancing" the load offer superior performance. For the case of multi-tiered systems, we show that if there is autocorrelation in the flows, we 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. For multi-tiered systems, this underutilization of resources falsely indicates that the system can sustain higher capacities. We present analysis of the above phenomena that aims at the development better scheduling policies under autocorrelated flows.

1. Motivation

Burstiness is expressed by the dependence structure of the request flows in the various system components. This dependence structure is described and quantified via the *autocorrelation function* (ACF). Autocorrelation is used as a statistical measure of the relationship between a random variable and itself [1]. 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 time separation between the occurrences 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, implying that the further in time the two occurrences of the random variable are, the higher the probability that they are independent of each other. The decay rate of the ACF distinguishes the time series as short-range dependent (SRD) or long-range dependent (LRD). The ACF essentially captures the "ordering" of random values in the time series. High positive ACF values imply strong temporal locality: a value of the random variable in the time series has a high probability to be followed by another a value of the random variable in the time series has a high probability to be followed by another variable of the same order of magnitude, while negative ACF values imply the opposite.

To illustrate the magnitude of the performance effects of autocorrelation in open systems, i.e., systems with an infinite waiting queue, we parameterize a simple queueing model of a single server. The arrival process is drawn from a Markov Modulated Poisson Process (MMPP). We selected a Markovian-Modulated Poisson Process (MMPP), a special case of the Markovian Arrival Process (MAP) [3], to model autocorrelated service times because it is analytically tractable. Its basic building block is a simple exponential but it can be easily parameterized to have dependence in its structure. The MMPP that is used here is parameterized

such that it results in three levels of dependence as illustrated in Figure 1(a): no ACF (i.e., arrivals are independent), short-range dependence (SRD), and long-range dependence (LRD).

The PDFs of these three arrival processes are identical (i.e., all their moments are the same), but what distinguishes them is the *order of sampling* from the PDFs, which introduces autocorrelation in the sample. The service process is a simple exponential distribution and the queueing discipline first-come-first-serve. Inter-arrival times are scaled so that we examine the system performance under different utilization levels. Figure 1(b)-(c) presents performance measures for the three different arrival processes as a function of server utilization. The effect of ACF on system performance is tremendous: the higher the ACF, the worse the system performance, which can worsen as much as 3 orders of magnitude when comparing to the case with no ACF arrivals. This figure shows the important performance effects of autocorrelation in queueing.

In this research, we have explored the effect of autocorrelated flows in *open systems* (i.e., systems with infinite queue capacities) and *closed systems* (i.e., systems with finite queue capacities). For the open system case, we have developed analytic techniques that allow for modeling the departure process of a queue with autocorrelated flows, thus allow for queue-by-queue analysis. These results will not be presented here, instead we refer the interested reader to [2, 10, 13]. Load balancing in a homogeneous cluster with infinite buffers is a case of an open system. In Section 2 we summarize the performance effects of autocorrelated flows for load balancing and the design of new policies. In Section 3 we present another application area: capacity planning in multi-tiered systems that effectively operate like closed systems, where we present experimental as well as modeling results. Finally, Section 4 outlines our future work.

2. Load Balancing

We present an application of the effect of autocorrelation to a classic problem: scheduling and load balancing in a cluster environment. Size-based policies have been shown to successfully balance load and improve performance in homogeneous cluster environments where a dispatcher assigns a job to a server strictly based on the job size (see [12] and references within). While the success of size-based policies is based on avoiding the unfavorable performance effects of having short jobs been stuck behind long jobs [12], we have shown that their effectiveness quickly deteriorates in the presence of job arrivals that are characterized by correlation in their dependence structure [11, 6].

We give an overview of these observations using trace-driven simulations. The service process is obtained from

traces of the 1998 World Soccer Cup Web site,[1] that have been used in several studies to evaluate the performance in load balancing policies in clustered web servers. Trace data were collected during 92 days, from 26 April 1998 to 26 July 1998. Here, we use part of the June 24th trace (10 million requests), that corresponds to nearly ten hours of operation and we extract the file size of each transfered request. Because the Web site contained only static pages, the size of the requested file is a good approximation of the request service time. In the trace used for our experiments, the average size of a requested file is 5059 bytes and its coefficient of variation (CV) is 7.56.

Unfortunately, we cannot use the arrival process of the World Cup trace data because it is not detailed enough: arrival timestamps of requests are provided in seconds, so *multiple* requests arrive within one second periods. To examine the effect of autocorrelation in the arrival process, we use instead a 2-state MAP, which, with appropriate parameterization, allows for changing *only* the ACF while maintaining the same PDF. The ACF of the three arrival processes that we use is illustrated in Figure 2(a). This allows for sensitivity analysis with respect to autocorrelation in the arrival stream. We compare the performance of the following policies: ADAPTLOAD, a size-based policy that does not require *a priori* knowledge of the service time distribution and has been shown to be effective under changing workload conditions [12], the *Join Shortest Weighted Queue* (JSWQ) policy, *Join Shortest Queue* (JSQ), and *Round Robbin* (RR).

ADAPTLOAD constructs the histogram of all requests and partitions it in equal areas, i.e., representing equal work for each server, while preserving the fact that requests of the same size fall within the same partition. ADAPTLOAD self-adjusts the area boundaries by predicting the incoming workload based on the histogram of the last K requests. In JSWQ the length of each queue in the system is weighed by the size of queued requests, therefore each incoming request is routed to least loaded server. With JSQ when a request arrives, it is assigned to a server with the smallest waiting queue. With the round-robin (RR) algorithm, requests are routed to servers in a rotated order.

We evaluate the effect of autocorrelated inter-arrival times on the performance of load balancing policies by analyzing the average response time (i.e., wait time plus service time), the average slowdown (i.e., the ratio of the actual response time of a request to its service time), and the mean system utilization. Figure 2 plots performance results for the four load balancing policies. The figure shows that correlation in the arrival process degrades overall system performance for all four policies. For example, the overall performance under independent arrivals (NOACF) is two orders of magnitude better than under SRD inter-arrivals,
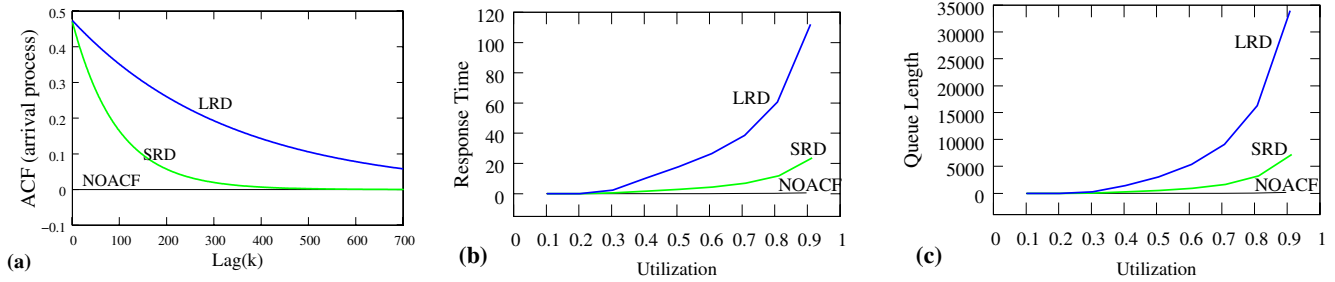

Figure 1. (a) ACF of the arrival process at the queue, (b) average response times (wait plus service times), and (c) average waiting queue length for different server utilization levels. Because of the scale used in the figure and because of the difference of the three curves, the performance measures with no ACF look flat. With no ACF for utilization equal to 0.9, queue length is equal to 152 as expected, but this number is dwarfed in comparison to the LRD and SRD numbers.
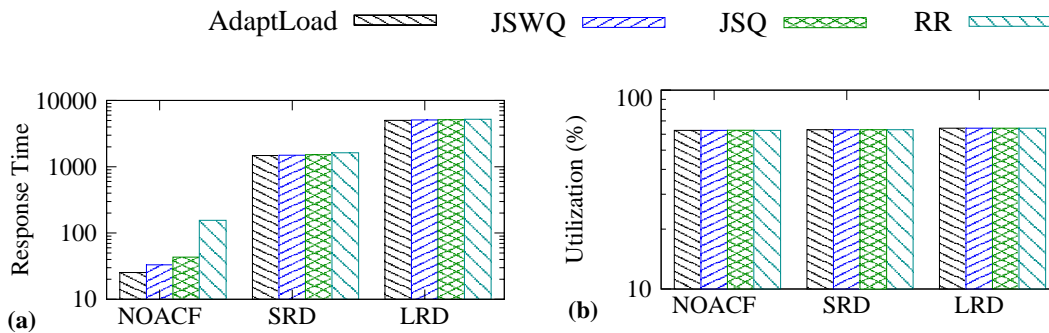

Figure 2. Average response time (a) and average utilization(b) of the four load balancing policies under different degrees of autocorrelation in the arrival stream.

and three orders of magnitude better than under LRD inter-arrivals, despite the fact that average system utilizations are exactly the same for all experiments, i.e., the average utilizations are about 62%, see Figure 2(c). Most importantly, the figure also shows that ADAPTLOAD outperforms other policies *only* under independent inter-arrivals, see Figure 2(b). Under autocorrelated arrival processes, ADAPTLOAD's performance is comparable to the three other policies, showing that separating requests according to their sizes is not sufficient to improve performance.

Based on the above observations, we use the same histogram information as ADAPTLOAD but re-set the partition boundaries by shifting to neighboring servers a percentage R of the work assigned to each server. This is based on the observation (see also Figure 1) that in order to achieve similar performance levels under autocorrelated arrivals, the system utilization must be lower than under independent arrivals. Naturally, performance improvements depend on the degree of load unbalancing that is introduced by the shifting percentage R. A good choice of R can result in significant performance improvements, but an unfortunate choice may
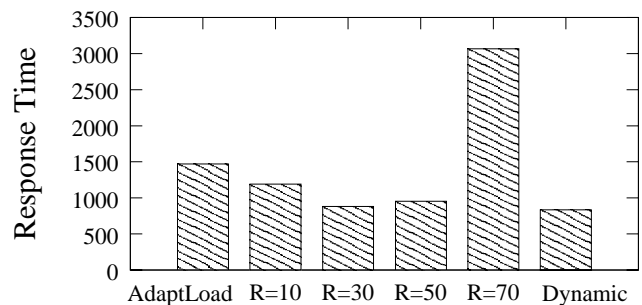

Figure 3. Average response time for the original ADAPTLOAD, the policy with static shifting ratio R, and a dynamic version that adjusts R in an on-line fashion.

also result in poor performance. We have developed an on-line version of this policy that monitors workload as well as the effectiveness of load balancing. Its performance is now independent of the choice of R. By observing past ar-
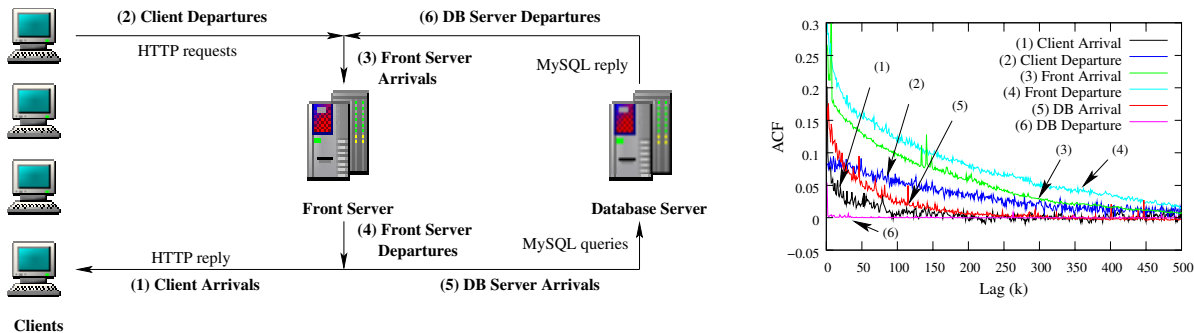
Figure 4. TPC-W experimental environment (left) and autocorrelation flows in various marked points of the system under the the default browsing mix with 384 emulated browsers (right).

rival and service characteristics as well as past performance, it self-adjusts its configuration parameters. Figure 3 shows the average response time of this on-line policy in comparison to various static versions with fixed shifting percentages R, showing that unbalancing load in a cluster server under autocorrelated arrivals results in significant performance gains. Key to this solution was the detection of autocorrelation in the arrival process. For more details, we refer the interested reader to [11, 6].

3. Capacity Planning of Multi-tiered Systems

In computer systems, in contrast to the open systems example illustrated above, there is always an upper bound on the total number of jobs or outstanding requests that exist in the system at all times, a bound that is dictated by the finite buffers in the system. This upper bound makes the system operate like a closed system, i.e., a system with a finite queue, and performance deterioration due to autocorrelated flows is more contained.

Multi-tiered systems, a prevalent architecture of today's web sites, is an example of closed systems because the hardware imposes a limit on the number of simultaneous connections. Capacity planning and workload characterization in such systems aims at identifying bottlenecks and the conditions which trigger them and aid the development of resource management policies to improve performance or provide service level provisioning. In our preliminary work [5], we observed that burstiness in the service process of *any* of the tiers (queues) may result in very high user response times even if the bottleneck resource in a system is not highly utilized, while measured throughput and utilizations of *all* other resources are also modest. When burstiness is not considered, this underutilization may falsely indicate that the system can sustain higher capacities.

In collaboration with researchers at Seagate Research we have built an e-commerce server according to the TPC-W [8] e-commerce benchmark to identify the presence of

autocorrelation in different tiers of the system. A high-level overview of the experimental set-up is illustrated in Figure 4, which also shows the flow of requests. The workload generation modules are done using the TPC-W specifications that do not introduce any correlation in the flows. Autocorrelation is nonetheless observed in various flows in the system as shown in the right graph of Figure 4. According to our analysis, the origin of these autocorrelated flows is the service process in the front server (details can be found in [5]).

3.1. Theory: Autocorrelation in Systems

In closed systems, because of the limited buffers, we expect that the performance effects of autocorrelation are more contained in comparison to open systems. To better understand the observed behavior of our TPC-W experiments, we use the simplest closed queuing system (see Figure 5) that resembles the topology of a two-tiered application. Autocorrelation in the arrival or service processes directly implies that the system is not product-form [4], therefore one can only use simulation for its analysis. We assume that a fixed number of jobs circulate in the queuing network, known as the *multiprogramming level* (MPL). We assume that server 2 is the bottleneck device and that server 1 is twice as fast as server 2. We have done two experiments. In the first experiment (baseline case), the service processes in server 1 and 2 have no ACF. In the second experiment, only the service process of the bottleneck server 2 has ACF. Parameterization was done in such a way that the stochastic processes in the two experiments have the same PDF (therefore means and higher moments are identical) – only the autocorrelation function is different for server 2. We measured the autocorrelation of the flows in the system and naturally no autocorrelation was detected in the first experiment, while ACF propagated from server 2 in the *entire* flow in the second experiment.

This propagation has a tremendous effect on performance. Figure 5(b)-(d) compares performance measures of
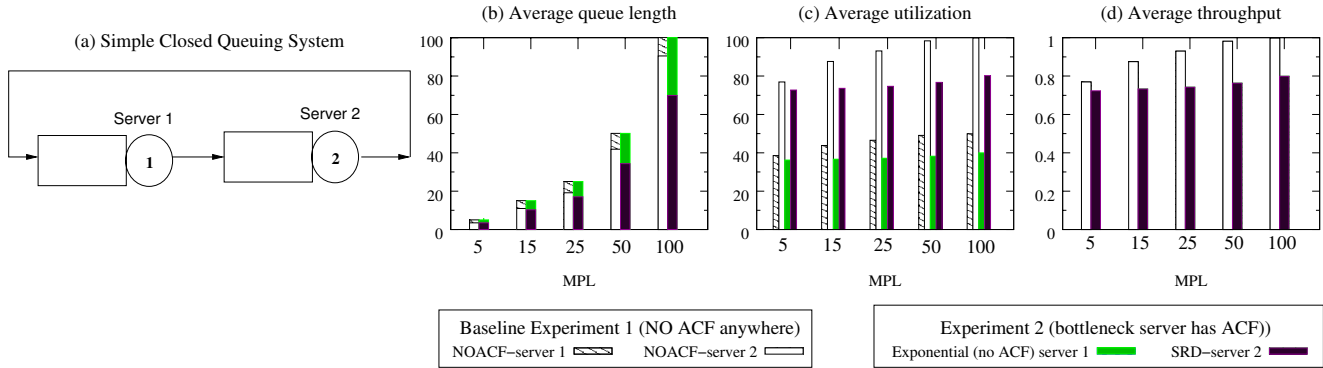
4

Figure 5. A closed system with 2 queues and performance measures: (b) mean queue length, (c) mean utilization, and (d) mean system throughput. Performance measures are presented for two systems: one with no ACF in the service processes for server 1 and server 2 (baseline case, where classic queueing theory analytic models apply) and one where only the bottleneck queue (server 2) has autocorrelated service times (existing analytic models do not apply).

the two experiments. Round-trip times are not shown due to lack of space, but as expected, in the experiment with ACF they are consistently higher than the first experiment. Counter-intuitively for experiment 2, the expected utilization of each queue decreases, including the expected utilization at the bottleneck device, which is also reflected in the mean queue length of all servers. Furthermore, with experiment the system throughput remained consistently lower than experiment 1. *Existing analytic models would predict the performance shown in experiment 1, but would be far off from the performance of experiment 2.*

3.2. New Analytic Models: Approximate Lumpability

We have developed a new analytic model that can capture the behavior of the two queue closed system shown in Figure 5 for the case where the second queue (bottleneck server) has an autocorrelated service process. To aid the development of a model, we used an Markov Modulated Poisson Process of order 2, MMPP(2), to model the service process in the second server. The process of the first server is a simple exponential.

The use of MMPP(2) allows for defining two "macrostates" in the system, that capture the temporal dependency of the MMPP(2) service process. An approximate solution for the Markov chain of the closed network may be obtained by first analyzing the performance of each macrostate in isolation, and then weighting the results according to the fraction of time spent in each macrostate. This approach is based on classic *decomposition* arguments, and provides accurate results whenever the behavior of the system in a macrostate is largely independent of the dy-

namics inside the other macrostates. This is not the case though for the MMPP(2) case: temporal dependency in the macrostate exists due to autocorrelation, therefore decomposition does not work well.

We obtained accurate estimates of the mean throughput under heavy-load conditions using approximate lumpability. This allows to exploit the special structure of the MMPP(2) process to refine the solution obtain with decomposition. In particular, we derived a single-step approximation on the *mean* utilization of the MMPP(2) server by manipulating the core equations of Takahashi's iterative exact method [7]. A description of our method is given in Algorithm 1. Our technique requires a single-iteration, and requires to compute the macrostate probability terms by global balance. We approximate these terms by first removing the related states in the Markov chain, then heuristically solving the resulting symmetric Markov chain, and finally computing the result by a single global-balance equation for each macrostate. The symmetric Markov chain is solved by first approximating the innermost states of the chain with the closed-form solution of a network where only the bottleneck stations are considered in each macrostate, and then solving the reduced chain by global balance. We limited in experimentation the maximum size of the reduced chain to 50 states, so that a global balance solution can be obtained in negligible time. The results of our technique compared to decomposition are shown in Figure 6. As we see, our method is able to capture the evolution of throughput for a heavy-loaded system, while decomposition is unable to follow the throughput growth as the network enters in heavy-load conditions (i.e., increased MPLs).

5

Algorithm 1 Single-Step Analysis of Mean Utilization by Approximate Lumpability.

1: $U_p^k \overset{\text{def}}{=}$ joint probability that $\mathcal{Q}_p$ is *busy* and the MMPP(2) process is in state $k \in \{0, 1\}$

2: $\widetilde{U}_p^k \overset{\text{def}}{=}$ unscaled joint probability that $\mathcal{Q}_p$ is *busy* and the MMPP(2) process is in state k

3: $\widetilde{\pi}_p^k \overset{\text{def}}{=}$ unscaled joint probability that $\mathcal{Q}_m$ is *empty* and the MMPP(2) process is in state k

4: Initialize U_p^0 and U_p^1 using decomposition as in [9]

5: $\widetilde{U}_p^0 \leftarrow \left(\dfrac{P_{1,0} U_p^1}{P_{0,1} U_p^0 + P_{1,0} U_p^1} \right) U_p^0$

6: $\widetilde{U}_p^1 \leftarrow \left(\dfrac{P_{0,1} U_p^0}{P_{0,1} U_p^0 + P_{1,0} U_p^1} \right) U_p^1$

7: Compute $\widetilde{\pi}_p^k$ by global balance for $k \in \{0, 1\}$

8: Compute the normalizing constant $G \leftarrow \widetilde{U}_m^0 + \widetilde{U}_m^1 + \widetilde{\pi}_p^0 + \widetilde{\pi}_p^1$

9: Normalize $U_p^k \leftarrow \widetilde{U}_p^k / G$, $k \in \{0, 1\}$

10: Return by Little's Law the mean throughput $X = (U_p^0 + U_p^1)/S_p$.

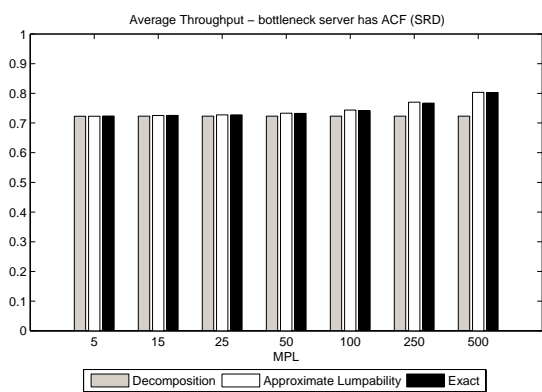**Figure 6. Comparison of throughput approximation of decomposition and approximate lumpability with respect to the exact steady-state solution.**

4. Future Work

We have identified autocorrelation as an important stochastic characteristic in flows for both closed and open systems. Our future work spans several dimensions. On the theoretic side, we will focus on generalizing the approximate lumpability algorithm for networks composed of more than 2 queues. On the application side, we will continue working on new policies in multi-tiered systems especially focusing on the development of new admission control poli-cies that depend on on-line detection of the degree of autocorrelation in flows. We have also started exploring the development of policies at the storage system level (where strong autocorrelation flows have been observed by our industrial partners) to improve storage system reliability, as well as scheduling policies at the disk level that are oblivious to all other specific knowledge of the workload that are sometimes impossible to know a priori (e.g., at the driver level execution times of reads and writes are rarely known before their execution).

References

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

[2] A. Heindl, Q. Zhang, and E. Smirni. ETAQA truncation models for the MAP/MAP/1 departure process. In *Proc. 1st Int. Conf. on Quantitative Evaluation of Systems*, 2004.

[3] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. SIAM, Philadelphia PA, 1999. ASA-SIAM Series on Statistics and Applied Probability.

[4] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc, New York, 1984.

[5] N. Mi, Q. Zhang, A. Riska, E. Riedel, and E. Smirni. Performance impacts of autocorrelation in multi-tiered systems. In *(submitted for publication)*, Nov. 2006.

[6] N. Mi, Q. Zhang, A. Riska, and E. Smirni. Load balancing for performance differentiation in dual-priority clustered servers. In *Proceedings of the 3rd International Conference on Quantitative Evaluation of SysTems (QEST'06)*, Riverside, CA, Sept. 2006.

[7] Y. Takahashi. A lumping method for numerical calculations of stationary distributions of Markov chains. Research Report B-18, Department of Information Sciences, Tokyo Institute of Technology, 1975.

[8] Transaction processing performance council. http://www.tpc.org/.

[9] J. Zahorjan, E. D. Lazowska, and R. L. Garner. A decomposition approach to modelling high service time variability. *Performance Evaluation*, 3:35–54, 1983.

[10] Q. Zhang, A. Heindl, and E. Smirni. Characterizing the BMAP/MAP/1 departure process via the ETAQA truncation. *Stochastic Models*, 21(2-3):821–846, 2005.

[11] Q. Zhang, N. Mi, A. Riska, and 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.

[12] Q. Zhang, A. Riska, W. Sun, E. Smirni, and G. Ciardo. Workload-aware load balancing for clustered web servers. *IEEE Transactions on Parallel and Distributed Systems*, 16(3):219–233, March 2005.

[13] Q. Zhang, E. Smirni, A. Stathopoulos, and A. Heindl. MAP approximations of the departure process of the BMAP/MAP/1 queue. *(submitted for publication)*, nov 2006.