# Research Summary

**Leana Golubchik**

**July 29, 2000**

My research interests are in the area of computer systems modeling and performance evaluation. One aspect of my research has been development of models and efficient methodologies for evaluation of systems at design time. High level abstractions that allow for evaluation of design *choices* (rather than prediction of performance numbers of the final product) can provide insight into the system being designed, expose the tradeoffs involved, and lead to significantly better designs of systems. In general, design choices are better studied through *analytical* models — at design time it is desirable to obtain results quickly since there is a large number of possible design choices. Higher accuracy which may be obtained through detailed simulations or measurements is often not needed at design time or simply not possible or economical to achieve. Hence, I work on analytical performance evaluation methodologies that lead to (a) reasonably simple *abstractions*, needed for ease of applicability, and (b) *efficient* solution techniques, needed for rapid evaluation of design ideas.

Another aspect of my research has been quality-of-service-based design of large-scale continuous media storage systems and their subsequent performance evaluation. Many systems today are moving towards applications whose primary function is to transfer large volumes of data from an I/O subsystem through the network, possibly subject to some form of real-time constraints. Therefore, efficient storage system designs are fundamental to the viability of these applications, and I expect our work to be of use to a broad range of current and future systems and applications.

More recently, I have focused on performance aspects of scalable *upload* applications over wide-area networks. To the best of our knowledge there is no existing work, except ours, on making upload applications scalable and efficient. This corresponds to an important class of applications, which includes digital government, digital democracy, and Internet-based storage. My current work in this area is on design of an application layer infrastructure that will improve *scalability* and *security* of current and future Internet-based services.

# 1 Modeling and Performance Evaluation Methodology

The basic approaches to performance evaluation of computer systems include analytical modeling, simulation, and measurements. My work focuses on analytical modeling. An example model is a multiclass multiserver queueing system whose attributes include the system's workload in the form of $N$ job arrival processes each with its own arrival rate, the system's resources in the form of a queue and $K$ servers each with its own service rate, and a service scheduling discipline. In general, arrival and service rates may be a function of the current model state, and not all servers need be activated at all times (where activation of more servers improves the performance but may increase cost). Given such models, common performance metrics of interest include expected job response time, server utilization, throughput, and so on.

Performance evaluation of real systems is complex, suffers from scalability problems (such

as the so-called "state space explosion" problem), and usually requires advanced model solution techniques. Often, such techniques are based on exploitation of "special structure" in the models. With large and complex models, these special structures are usually too expensive to expose automatically (as that may involve searching through a combinatorial number of permutations). My work on analytical modeling focuses on discovery of special structure in models which results in order(s) of magnitude improvement in computational efficiency.

## 1.1 Threshold-based Systems

We have developed Markovian models and efficient solutions of systems that employ *threshold-based* techniques with *hysteresis* behavior to dynamically adjust the amount of resources available for job service [A4,D10,D20]. A threshold-based approach is useful in systems which incur significant server setup, usage, and removal costs. It is also a good approach to dynamic management of a pool of resources between multiple workload classes using the same system. The hysteresis is needed to guard against momentary fluctuations in workload. For instance, such techniques are used in Novell file servers in managing their buffer pool. However, performance prediction for threshold-based systems is often difficult as such systems exhibit anomalous response time behavior, as observed in our work as well as that of others.

There has been much interest in developing efficient performance evaluation techniques for such systems. However, past results have been obtained only for fairly restricted models which include limitations on one or more of the following: (a) number of servers in the system (often restricted to 2), (b) size of the waiting room, (c) heterogeneity of servers, (d) number of job classes (usually 1), and (e) absence of hysteresis. We have developed a divide-and-conquer type methodology (using stochastic complementation) which allows extremely efficient computation of performance metrics for a variety of threshold-based models with hysteresis without such restrictions, handling heterogeneous servers, bulk arrivals, non-instantaneous server activation, and multi-class workloads, all without restrictions on the number of servers or the size of the waiting room.

For a variety of single-class models [A4,D10] we derived either: (1) an *exact closed-form* expression, or (2) an *exact* algorithmic solution, or (3) *provable* (through sample path analysis) and tight performance bounds, which, for all practical purposes, is as good as having an exact solution (e.g., experiments in [D10] show less than 1.5% spread between upper and lower bounds with an order of magnitude improvement in computational cost over the exact solution). Moreover, allowing *non-instantaneous* resource activation in a model, while difficult to analyze accurately, is important because activation time is often the distinguishing characteristic of a resource management policy. No other work addresses this problem with more than 2 servers.

The efficiency is obtained by breaking the model into smaller pieces, solving each piece separately, and then using these solutions in constructing the exact solution to the original model. For cases where it was not possible to apply the divide-and-conquer approach directly, we developed modified models which were suited for the divide-and-conquer approach and produced *provable* and tight bounds on performance metrics of the original model [D10]. Due to the divide-and-conquer nature of our approach, the reduction in computation grows as the number of servers in the system grows.

Based on the success with single-class models, we have developed an efficient method for

*multi-class* models [D20]. In systems with multiple job classes it is desirable to find good approaches to sharing resources without statically partitioning them among the job classes. Threshold-based techniques constitute one category of such approaches. There have been no prior analytical results for the *multi-class* version of this model, which corresponds to an important characteristic of real systems.

Solving this model exactly is difficult because it is infinite in multiple dimensions and appears to lack sufficient structure. We developed [D20] an *approximate iterative* solution method. This method "breaks up" the original $N$-class model into $N$ single class models, which are "coupled" through a set of blocking probabilities that express the interaction between classes. Empirical comparison against simulation indicates that this method is fast and fairly accurate. Most of our test cases are within 5% of the simulation results with more than two orders of magnitude improvement in computation time. Although this is an approximate solution technique, our experiments indicate that good threshold settings result in much higher performance improvements (e.g., 50%) than the loss in accuracy due to the approximation. Hence, our solution technique is a promising approach to evaluating *multi-class* threshold-based systems.

## 1.2 Mixed-workload Multimedia Systems

This work focuses on modeling and analysis of storage servers that serve a variety of applications requesting video, image, audio, and text data with vastly different performance and quality-of-service (QoS) requirements as well as resource demand characteristics. For instance, the QoS requirement of continuous media (CM) workload (e.g., video) is to meet specific deadlines in data delivery with high probability (e.g., greater than .99), while the non-CM workload (e.g., images and text) requires fast response time with no errors. Hence, we model multimedia servers which share resources dynamically among the different types of workloads while satisfying their performance requirements.

We first designed a family of disk scheduling algorithms [A12,D16] in order to explore the possible design choices and tradeoffs of interest. Such algorithms (ours as well as those of other researchers) often use cycle-based scheduling of the CM workload which results in models that are non-Markovian (because each chunk of CM workload must be completed by the end of a specified time unit) and non-work-conserving (because early service carries penalties in the form of increased buffer space requirements). These algorithms mostly differ in: (1) how they "interleave" service of CM and non-CM workloads, (2) how they order the service of non-CM requests, and (3) how work-conserving they are with respect to CM workload.

We developed a family of analytical non-Markovian models which represent this class of scheduling algorithms and have tractable analytical solutions [A5]. Prior approaches to modeling such systems include M/G/1 queue with vacation models. Our model, which can be viewed as a polling-type system, more faithfully captures the behavior of scheduling algorithms, is extensible to a wider class of such algorithms, and allows computation of a larger number of important performance metrics, for both types of workloads.

In this work [A5], each scheduling algorithm being modeled is captured through construction of appropriate service time distributions for CM and non-CM workloads, where the CM workload distribution is then matched with an Erlang and the non-CM with a load-dependent exponential. (Parameters for these distributions can be obtained by benchmarking the disk

and extracting the seek function, transfer times, and so on.) Hence, all the events in the model are exponential, except for the deterministic time unit during which a chunk of the CM workload should be completed. Our solution of the models is based on an existing methodology, by de Souza e Silva, H.R. Gail, and R.R. Muntz[1], of using embedded Markov chains to analyze *non-Markovian* stochastic processes. It involves a combination of steady state and transient analysis.

# 2 QoS-oriented Design of Large-scale Storage Systems

The viability of large-scale multimedia applications, such as video-on-demand systems, digital libraries, tele-medicine, and distance learning, depends on the performance of storage systems and communication networks. In my work I focus on *storage* systems for continuous media (CM) applications. Common among these applications are their high I/O bandwidth and storage requirements, coupled with some form of QoS data delivery for admitted users. CM applications place high demands for performance, scalability, and reliability on storage servers, which in turn requires efficient data placement, scheduling, admission control, and provision for fault tolerance. Some early works focused on single logical disk designs which do not scale up. My work focuses on large-scale storage systems (with parallel or distributed I/O).

Although many basic principles of parallelism apply, some characteristics are unique to storage subsystems. Unlike, say CPU resources, all storage resources are not equivalent in the sense that their utility is determined by the data placed on them which is determined a priori to a large extent. This "partitioning" of resources (based on data placement) contributes to some of the difficulties in designing large-scale I/O systems. Inappropriate data placement can lead to load imbalance problems due to skewness in data access patterns (common to CM applications). Moreover, most storage devices can be viewed as "2-dimensional" (storage and bandwidth) resources, which contributes to hardness of related resource allocation problems. For instance, in [D18] we consider data placement on parallel disks with applications to CM servers, and show this problem to be NP-hard as well as study approximation algorithms with tight bounds.

## 2.1 Data Sharing

Due to the skewness in data access patterns, I/O bandwidth is often the critical resource for CM data access. One approach to reducing the I/O demand on the server is to "share" data between requests for the same object thereby increasing the number of simultaneous users. Prior to our work two approaches to data sharing were in use, namely batching and buffering. Our work [A1,D4] introduced a third novel technique of *stream merging* (termed "adaptive piggybacking") which adjusts display rates of requests *in progress* until their corresponding I/O streams can be "merged" into one. It accomplishes significant reduction in I/O bandwidth demand (e.g., $50-80\%$) while eliminating the cost of additional latency or additional buffer space (incurred by batching or buffering, respectively). This work has been extended by other groups, for example at IBM T.J. Watson[2].

Since batching, buffering, and merging are not mutually exclusive, we continued our work

[1] E. de Souza e Silva, H.R. Gail, and R.R. Muntz, "Efficient Solutions for a Class of Non-Markovian Models", in Proceedings of the 2nd International Workshop on the Numerical Solution of Markov Chains, 1994.

[2] C. Aggarwal, J. Wolf, and P. Yu, "On Optimal Piggyback Merging Policies for Video-On-Demand Systems", in Proceedings of the ACM SIGMETRICS Conference, 1996.

and studied appropriate combinations of these techniques. In all such techniques once the users are "grouped" for the purpose of data sharing they may leave the group due to the use of VCR functionality (e.g., fast-forward) and hence require additional resources. Thus, we introduced models [A10,D9] for determining the amount of resources required for supporting both normal playback and VCR functionality under predefined constraints on performance and QoS characteristics. These models allow us to maximize the benefits of data sharing techniques as well as make system sizing decisions (e.g., to determine the appropriate compromise between use of buffer and I/O bandwidth resources).

In another work on data sharing [A7], we construct both a theoretical framework and practical optimal algorithms for reducing the I/O bandwidth requirements by partitioning client requests into groups and simultaneously servicing each group with a single set of resources. Our framework can accommodate a large class of objective functions and a variety of data sharing techniques and applications, including those with more interactive environments.

## 2.2 Data Placement and Dynamic Resource Management

The scalability of a CM server's architecture refers to its ability to maintain performance characteristics under workload growth and degradation of system resources (losses in network and storage capacities). The choice of data placement techniques can have a significant effect on scalability and efficiency of a distributed CM server. In the past, a great deal of work has focused on data placement based on"wide" data striping as an approach to dealing with load imbalance problems. Another approach to this problem is replication. An appropriate compromise between the degree of striping and the degree of replication, which we term a hybrid system, is crucial to the design of scalable CM servers. We show [D19] that hybrid designs, in conjunction with dynamic replication techniques which adjust to *changes* in data access patterns, are less dependent on interconnection network constraints, provide higher reliability (even under conservative assumptions), and can be properly sized so as to result in cost-effective scalable systems.

We also studied dynamic replication policies [D23] under *changes* in data access patterns. One distinguishing characteristic of our work [D23] is that we consider schemes which do not rely on knowledge of object access frequencies but rather make the adjustments, in a threshold-based manner, based on the amount of resources currently available for servicing user requests for those objects. We believe that this is an important consideration, as determination of when a change in frequency requires a change in system state may not be a simple matter.

Our other work on dynamic resource management [A6,C1] focuses on exploitation of the multiresolution property of compressed CM streams in environments that face fluctuations in workload (e.g., due to VCR functionality) or lack of system resources — here we use the flexibility of scalable compression to adjust a stream's bit rate as needed. We design a data placement scheme and study its affects on disk bandwidth and buffer space utilization, user requests' start-up delay, and probability of request rejection due to system overload [A6]. We also propose techniques for re-scheduling of data retrieval, which dynamically adjust resolution of streams in progress to react to fluctuations in workload and system resource availability while satisfying QoS constraints [C1].

### 2.3 Fault Tolerance and Tertiary Storage

Our work on fault tolerance [D5] introduced the first non-RAID fault-tolerant design of a multidisk CM storage server. This research produced data placement and scheduling techniques which resulted in highly reliable but significantly more cost-effective servers with higher throughputs and lower latencies than more traditional, previously pursued RAID-based techniques. More recent work focused on reduction of degradation in system performance under failure and rapid, but "non-performance-degrading", recovery from failure to normal operation [A9]. Construction of a framework for performance and reliability evaluation of possible design choices is described in [A2].

Our work on tertiary storage systems focused on analysis of striping techniques in robotic tape storage libraries, which indicated that significant improvements in system response time can be achieved through the use of non-RAID striping techniques [D6], and QoS-based delivery of CM objects from tertiary storage [D13].

## 3 Scalable Infrastructure for Wide-area Uploads

Hotspots are a major obstacle to achieving scalability in the Internet; they are usually caused by either high demand for some data or high demand for a certain service. At the application layer, hotspot problems have traditionally been dealt with using some combination of increasing capacity, spreading the load over time and/or space, and changing the workload. Some examples of these are data replication (web caching, ftp mirroring), data replacement (multi-resolution images, video), service replication (DNS lookup, Network Time Protocol), and server push (news or software distribution).

These classes of solutions have been studied in the context of applications using the following types of communication: (a) one-to-many (data travels primarily from a server to multiple clients, e.g., web download, software distribution, video-on-demand); (b) many-to-many (data travels between multiple clients, through either a centralized or a distributed server, e.g., chat rooms, video conferencing); and (c) one-to-one (data travels between two clients, e.g., e-mail, e-talk). However, to the best of our knowledge there is no existing work, except ours [D21,D24,D25], on making applications using *many-to-one* communication scalable and efficient; existing solutions, such as web based uploads, simply use many independent one-to-one transfers. This corresponds to an important class of applications, whose examples include the various *upload* applications such as submission of income tax forms to IRS, conference paper submission, proposal submission through the NSF FastLane, homework and project submissions in distance education, Internet-based storage, and many more. The main focus of our work is scalable infrastructure design for wide-area upload applications.

Traditional solutions aimed at downloads are data replication (e.g., caching) and data replacement. Clearly, these techniques are not applicable to uploads since all the data is distinct. Recently [D21] we proposed *Bistro*, a framework for building scalable wide-area *upload* applications which employs the use of intermediaries, termed *bistro*s, for improving the efficiency and scalability of uploads. We observed that the existence of hotspots in many upload applications is due to approaching deadlines and long transfer times (although here we focus on uploads with deadlines, our framework can provide a scalable solution to other upload applications as well). We also observed that what is actually required by many upload applications is an assurance that specific data was submitted before a specific time, and that

the transfer of the data needs to be done in a timely fashion, but does *not* have to occur by that deadline (since the data is often not consumed by the server immediately upon receipt). Thus, our approach is to break the original deadline-driven upload problem into the following pieces: (a) a real-time *timestamp* subproblem, where we ensure that the data is timestamped and that the data cannot be subsequently tampered with; (b) a low latency *commit* subproblem, where the data goes "somewhere" (to an intermediary) and the user is assured that the data is safely and securely "on its way" to the server; and (c) a timely *data transfer* subproblem, which can be carefully planned (and coordinated with other uploads) and results in data delivery to the original destination. This means that we have taken a traditionally *synchronized client-push* solution and replaced it with a *non-synchronized* solution that uses some combination of *client-push* and *server-pull* approaches. Consequently, we eliminate the hotspots by spreading most of the demand on the server over time.

*Bistro*'s ability to share an infrastructure, such as an infrastructure of proxies, between a variety of wide-area applications has clear advantages over the more traditional solutions. In [D25] we conducted a performance study which demonstrated the potential performance gains of the *Bistro* framework as well as provided insight into the general upload problem. Moreover, *Bistro* does not rely on the existence of a private infrastructure; however it does not preclude it either. Since confidentiality of data as well as other security issues are especially important in upload applications and in our solution where we introduced untrusted (public) intermediaries (i.e., *bistro*s), we also developed [D24] a secure data transfer protocol within the *Bistro* framework, which not only ensures the privacy and integrity of the data but also takes scalability considerations into account.

# 4 Research Directions

One of my current interests is Internet-based applications. Within the *Bistro* framework, our current focus is on several open research problems which we stated in [D21], including design of data transfer protocols over wide-area networks with constraints on security, performance, fault tolerance, and QoS. We believe that the *Bistro* framework is extensible to other Internet-based applications which have a many-to-one data transfer component, such as e-commerce, online auctions, and many more. Since the scalability of many-to-one data transfer has not been addressed yet, solving the many-to-one problem will improve the scalability of all these applications.

We are also carrying over our efforts on QoS-oriented design from storage systems to delivery of CM data over the Internet. Our main interest is in application-layer end-to-end mechanisms for providing QoS in CM data delivery over wide-area networks.

Another area of interest is the potential use of analytical models in dynamic control of resource management policies. For instance, in the case of mixed workload multimedia systems we are experimenting with integration of analytical models described earlier into disk scheduling policies for runtime control. Similar directions can be taken with threshold-based systems, where we first need to solve the open problems we recently posed in [D22], including optimal setting of threshold values, policies which share a resource between a subset of workload classes, policies for server allocation at departure instances, and so on.