IBM Research Report

Control of Large Scale Computing Systems

Yixin Diao, Joseph L. Hellerstein, Sujay Parekh
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Control of Large Scale Computing Systems

Yixin Diao, Joseph L. Hellerstein, and Sujay Parekh
IBM Thomas J. Watson Research Center
Hawthorne, New York 10532
Email: {diao, hellers, sujay}@us.ibm.com

Abstract— The rapidly increasing scale of computing systems means that it is vitally important to address the scaling challenges in the control of computing systems. We introduce a framework for describing the control problems for large scale computing systems that expand along two dimensions: the scale of the target system and the scale of the policy. Using this framework, we present control architectures that span a range from centralized schemes to distributed solutions. We further identify several research challenges related to issues such as target systems latencies and policy decomposition.

I. Introduction

In the last several years, there have been many examples of applying control theory to computing systems, including impact on commercial products [1] [2]. A common thread in this work has been to demonstrate the value of a control theoretic approach on a relatively small scale. With the rapid growth of the Internet, peer-to-peer networks, and pervasive computing, it is vitally important to address scaling considerations in the control of computing systems.

We illustrate the aspects of scaling through a discussion of a multitier eCommerce system. Figure 1 displays a three tier eCommerce system consisting of HTTP, application, and database servers. Requests arrive at the HTTP servers, some of which require processing by Application Servers, and a fraction of those also require processing by Database Servers. The eCommerce system can be scaled in several ways. It can be scaled in the vertical direction by adding more servers in a tier. It can also be scaled in the horizontal direction by adding more tiers, such as (a) edge servers at the entry tier to provide load balancing and admission control and (b) storage servers to provide disk access services for the database tier. Last, it can be scaled in terms of the ownership structure for multi-enterprise environments, such as computing utilities where the requestor is in a different company than the service provider and the service provider may subcontract to providers of application, database, and storage services.

Along with the scaling of computing systems, the focus of control solutions needs to be scaled from single system component to large scale systems that involve multiple interacting system components and objectives. This generally involves a transformation from owner specified policies to controller enforceable reference values, and coordination between multiple control systems. We refer to this as **scaling control**. Note that large scale control systems have been studied by the control theory community [3], [4], [5]. The goal of this paper is to initiate a discussion of the control problems and challenges
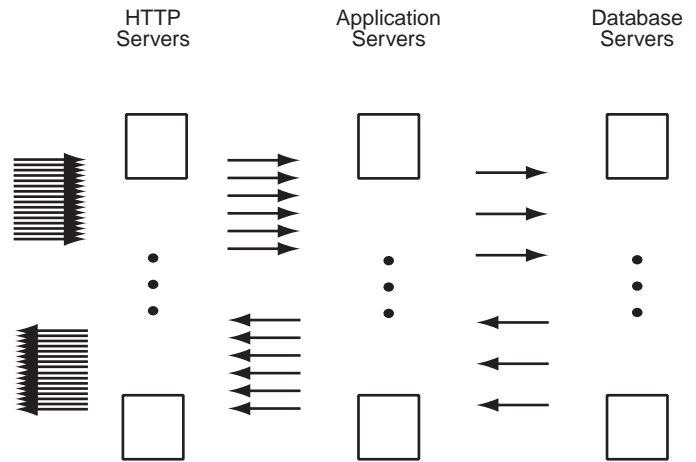

Fig. 1.   Multitier eCommerce system.

that arise in the context of computing systems. While a control-theoretic solution may not be appropriate in all cases, viewing the problem from a control perspective can provide valuable insights.

The remainder of this paper is organized as follows. Section I introduces a framework for characterizing the scale of control problems in computing systems. Section III presents several control architectures for large scale computing systems, and Section IV discusses important research challenges. Our conclusions are contained in Section V.

II. Control Problems

This section develops a framework for analyzing the scale of control problems in computing systems.

We begin with some terminology. Figure 2 depicts a simple control loop in which there is an **objective** (or reference input) that specifies the value to achieve for a **metric** (or performance measurement) produced by a target system. The **target system** is a set of hardware (e.g., computers, networks) and software (e.g., application servers, operating systems) that are being controlled. The target system has one or more **actuators** (e.g., concurrency levels, priorities) that affect its behavior, as indicated by the incoming arrow to the target system in Figure 2, and one or more sensors that expose metrics (e.g., response times, utilizations), as indicated by the outgoing arrow from the target system in Figure 2. The control objective is expressed in terms of the desired metric value (e.g., "response time should be 2 seconds"). The **controller** is
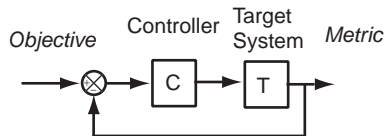
Fig. 2. Elements in a single feedback loop.

Fig. 3. Centralized control of multiple target systems using a single MIMO controller.

responsible for adjusting actuator settings on the target system so as to achieve the objective specified.

We believe that the control problems for large scale computing systems can be grouped along two dimensions: scaling of the target system resources, and scaling related to policies. In Table I, we have taken examples of control problems from the literature and characterized them along these two dimensions. Most of these references are from examples where feedback control theory is used in the solution.

The first dimension is the scale of the target system. This is characterized in terms of the number of actuators and metrics that need to be considered. In control terminology, target systems have inputs(I) and outputs(O) whose cardinality is denoted as single(S) or multiple(M). Common combinations are single-input single-output (SISO), multiple-input single-output (MISO), and multiple-input multiple-output (MIMO). The presence of multiple actuators and the use of multiple metrics create design challenges because of potential interactions and correlations between them.

SISO target systems are either quite simple (single policy) or represent embedded components of a larger system. MISO and MIMO systems are practical representations when there are multiple actuators being manipulated in a coordinated way and with an interrelated impact on system performance (e.g., utilities throttling [1]). In addition to the techniques and examples of such systems shown in Table I, general studies found in the control literature include an emphasis on computational aspects such as model reduction, sparse matrices, and parallel algorithms [3] [22].

In the context of scaling, we further subdivide the MIMO category into centralized (MIMO-C) and distributed (MIMO-D) target systems. In a MIMO-C system, all the resources are physically co-located (such as CPU and memory within a server [12]). In MIMO-D, there are multiple resources that are physically distributed, and this distributed structure can result in communication delays and hence dead times for actuators and sensors. Both MIMO-C and MIMO-D can benefit from decomposing the overall MIMO system into lower degree target systems that simplify control design. Additional challenges in MIMO-D over MIMO-C systems are to decompose the control design complying to resource distribution and to consider the effect of communication delays.

The second dimension in Table I is the scale of the **policy**. We view policy scale arising from two attributes: the number of objectives and the number of owners. Here, owners refers to the owners of the controlled resources – the target system(s). Each of these two attributes can be either singular (one) or many.
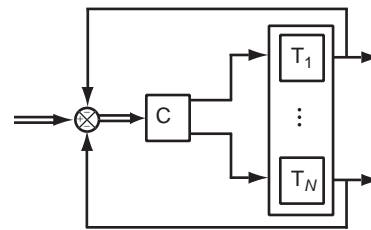
Multiple objectives arise naturally when there are different kinds of requests to be processed. An example of a single-owner, multi-objective system is the problem of providing differentiated service in a website [9], [10] or e-Commerce site [17], [23] with multiple classes of work. An eCommerce system such as Figure 1 may process browse requests and buy requests, but there may be different response time objectives for these two classes of requests. These systems are MIMO by necessity. For controllability, the target system must have multiple inputs (actuators), otherwise we cannot achieve multiple objectives. Moreover, the target system must have multiple outputs, otherwise we cannot measure the objectives we are trying to achieve. A multi-objective problem is more complex than a single-objective case since there may be trade-offs involved in satisfying the multiple objectives. However, having a single owner means that there is (implicitly or explicitly) a global objective that guides the decisions about tradeoffs.

Multiple owners arise as a result of specialization in the information economy, so that a particular end-user service may internally require resources from multiple other providers such as a credit card company, a shipping company, and web and database/storage providers. Multiple owners naturally lead to the existence of multiple objectives. Because of the differences in the business focus of each owner, it is very likely that they will have different objectives for their IT services as well. In such a situation, conflicts between objectives may not be easily resolved and it can be a significant challenge to translate the business-level objectives into IT-level objectives.

As we move to systems with multiple objectives and owners, the policy complexity increases. From Table I, it is evident that policy complexity goes hand in hand with target system complexity.

Before concluding this section, we observe that in the early literature on applying control theory to computing systems, the focus was on SISO target systems. Only in the last couple of years have MIMO target systems been addressed using control techniques.

III. CONTROL ARCHITECTURES

Since our focus is scaling, in this section we concentrate on the five non-SISO cells in Table I. We present three control architectures: centralized control, distributed control, and hierarchical control. For each one, we discuss some key properties that can help in choosing which architecture

Target System	Policy		
	1 Objective, 1 Owner	Many Objectives, 1 Owner	Many Objectives, Many Owners
SISO	Lotus Notes admission control [6] TCP RED congestion control [7] Dynamic Voltage scaling [8]		
MISO	DB2 utilities throttling [1]		
MIMO-C (Centralized)	DB2 self-tuning memory [2]	Web server QoS [9], [10], [11] Web server CPU/MEM [12]	
MIMO-D (Distributed)		EUCON [13], D-EUCON [14] Cluster performance [15] zOS WLM [16] Multi-tier performance [17]	Storage systems [18] Utility & Grid computing [19], [20] Networks [21] Peer-to-peer systems

TABLE I
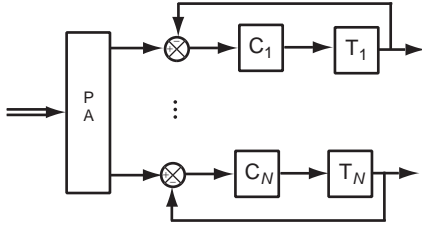
Some control problems in computing systems

Fig. 4. Distributed control of multiple target systems in which target systems are controlled independently. The Policy Authority (PA) is responsible for decomposing the overall objective into separate objectives for each target system.
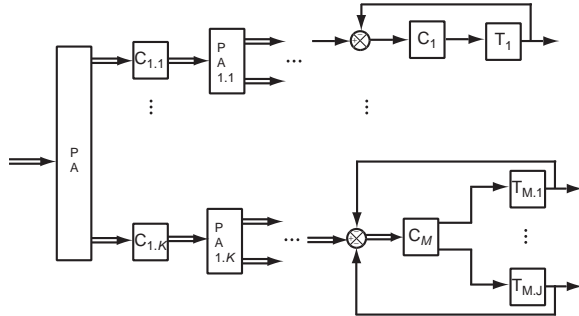

Fig. 5. Hierarchical control of target systems. There are multiple levels of Policy Authority that are used to decompose higher level objectives into lower level objectives.

is applied to a particular problem. Research challenges are detailed in Section IV.

One control architecture, depicted in Figure 3, is to treat the system like a centralized MIMO control problem. Thus, there is a single, centralized MIMO controller. Although it is also applicable for MIMO-D systems, such an approach makes the most sense when there are no communication delays between the controller and the target system (i.e., like a MIMO-C system). Examples of MIMO-D systems that satisfy this include cases where the target systems are connected by a high speed network (e.g., a cluster of compute servers) or where the target systems are composed of multiple resource layers within a server. The presence of a single centralized controller allows us to use well understood MIMO control techniques, as seen in [12], [13]. This approach fully models

interactions between the actuators and metrics, and the scaling is limited only by the capacity to manipulate large matrices and matrix singularity issues [3], [22].

In contrast to the centralized approach, Figure 4 depicts an architecture that uses distributed control in which there is a SISO controller for each target system. This makes sense when there is little interaction between the target systems and the overall system objective can be decomposed into independent sub-objectives for each target system. The **Policy Authority (PA)** in Figure 4 provides this decomposition of objectives. As an example of policy decomposition, consider the eCommerce system in Figure 1 with an end-to-end response time objective. In a distributed solution, the end-to-end response time objective can be decomposed into response time objectives for each tier, so that a separate controller for each tier can be used. Thus, if each of the tier objectives is met, the end-to-end objective is met. We note that there are disadvantages to this solution if resource bottlenecks change over time: some tiers may be unable to meet their objectives whereas other tiers could easily reduce their response times further. Thus, it may be necessary have an additional control mechanism that adjusts tier objectives in response to such changes. Another example of a distributed approach is seen in [14], where distributed control simplifies the controller design. The distributed architecture is also adequate for MIMO-D systems with communication delays.

Figure 5 depicts a solution that is a combination of the centralized and distributed approaches. This figure depicts a hierarchical structure in which higher level controllers (those more to the left) set the objectives for lower level controllers. The result is a tree of controllers in which the leaves of the tree (those more to the right) are either SISO controllers or centralized MIMO controllers as in Figure 3. Note that along with a hierarchy of controllers there is also a hierarchy of Policy Authorities that are responsible for decomposing the objectives provided to the next level of controller. An example of this is that the high level controllers admit the service requests to fulfil service differentiation objectives, but treat the server cluster as a single virtual server for simplicity of decision making; on the other hand, the low level controllers perform the load balancing to actually route service requests to different servers [15].
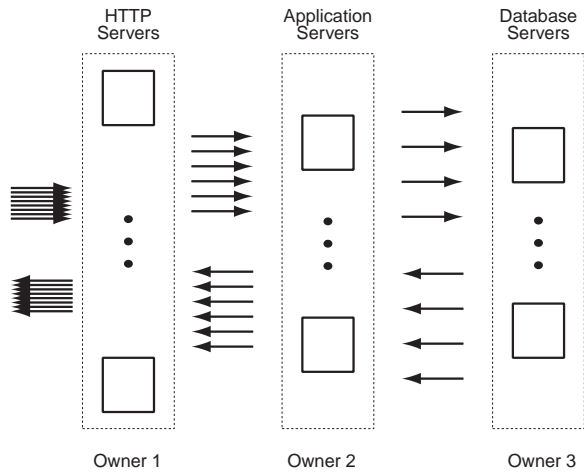
Fig. 6. Multiple owner eCommerce System in which ownership is structured by tier (as indicated by the dotted lines).
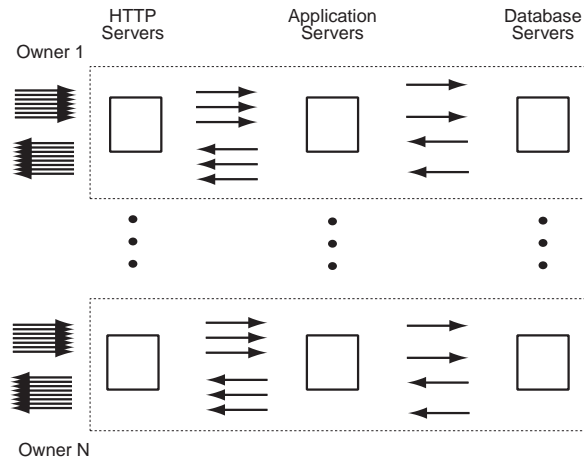

Fig. 7. Multiple owner eCommerce System in which ownership is structured by type of request (as indicated by the dotted lines).

The manner in which ownership is structured also impacts the choice of the control architecture. For example, Figure 6 displays an ownership structure for the eCommerce System in Figure 1 which is organized by tier. Achieving end-to-end response time objectives requires agreements among multiple owners, something that most likely results in a very complex structure of Policy Authorities. In contrast, Figure 7 displays an ownership structure in which responsibility is partitioned by request type. Owners work independently to ensure that objectives are met for the requests that they serve. Here, we can use the solution in Figure 4 by having (a) the target systems be separate eCommerce systems (one for each owner) and (b) the Policy Authority route requests to the appropriate eCommerce system.

Let us briefly discuss some examples considering all three aspects: system, policy and ownership. For a localized problem like QoS management in a single server, a MIMO-C structure is appropriate. This allows us to take into account the per-class control knobs as well as per-class policy objectives. The server

is generally owned by a single user, so ownership structure does not arise. When we employ vertical scaling (adding more servers to the tier), a more MIMO-D approach may be necessary, since it would be desirable for the replicated servers to operate fairly independently. Finally, moving to a multi-tier scenario, the ownership structure also influences the control architecture. Assume a per-tier ownership structure (Figure 6). In this case, we may employ a hierarchical approach (Figure 5). A higher-level Policy Authority computes a per-tier objective per class. These objectives may, in fact, have been determined a-priori between the tier owners, independent of the agreements with the actual end-users of the service. Within each tier, a distributed control computation may be used by combining per-node MIMO-C controllers to achieve the per-tier objectives.

IV. RESEARCH CHALLENGES

In addition to the control architecture, and intertwined with it, there are some fundamental challenges that arise when addressing larger-scale systems. In parallel to the problem structure of Table I, we discuss these challenges as arising from the scaling of policies or the target system resources. We also discuss challenges that are not unique to large-scale systems but become more severe due to scaling.

A. Policy Challenges

1) Decomposing the Policy Objectives: Policy objectives are generally defined at a system level. However, the design of feedback controllers often requires explicit reference values for each feedback loop. Therefore, there exists potential for an objective mismatch so that certain policy decomposition or objective transformation is required. For example, in the database memory management problem discussed in [2], the objective is to minimize the data access time (especially disk access time) by properly allocating memory to different buffer pools; however, if we want to design the controller, we need to know the desired data access time for each buffer pool. In a multi-tier web application, the objective is usually specified in terms of end-to-end response time; however, since the service is provided by each tier, if we want to have per-tier controllers, we need to decompose the end-to-end objective into per-tier objectives [17]. A similar example also exists when the request is serviced by different resources [16].

2) Handling Actuator Mismatch: In a MISO system, there are more actuators than sensors. This implies the existence of non-unique control solutions, but additional considerations are required for having a better actuator settings [1]. On the other hand, the actuator settings may be constrained since they all compete for the same resource, so that certain projection algorithms are required to meet the constraints [9].

3) Optimizing with Multiple Owners: Having multiple owners makes it difficult to achieve a globally optimal solution since owners may have different objectives. Sometimes, there are simple approaches that work well in practice without consideration for the details of the objectives of owners. An example is the analysis of TCP/IP networks in which the state

of the art is limited to mostly local solutions and the use of stability analysis [24]. In general, we assume that for multi-owner environments there are explicit contracts called service level agreements with penalties for violating objectives set in these agreements (e.g., [25]). Thus, owners must balance the cost of service level penalties with the cost of allocating more resources in order to avoid these penalties. In this way, owners operate independently to maximize their profits. However, it is inevitable that conflicts arise whereby one owner takes an action to increase his profits that causes the profits of another owner to decrease. One approach to handling conflicts between owners is to translate the objectives of each owner into common units such as "utilities", and then perform optimizations on these translated objectives with appropriate constraints to ensure fairness. However, doing so requires prior agreement among the owners, which may be difficult to achieve in practice. Another approach is to adopt ideas from Game Theory (e.g., [26]) to design policies that achieve equilibrium solutions in which no owner can benefit from a unilateral action such as reducing resource allocations.

4) Dealing with the Mismatch Between Policy Metrics and Sensor Metrics: It is often the case that policy metrics are expressed in terms of key performance indicators (KPIs) based on business processes. An example of this is a "buy" business process in which the KPI is the response time to complete a buy transaction. Unfortunately, many KPI metrics are either not produced target systems or are very expensive to collect. Thus, it there is often a requirement to translate from sensor metrics that are easily obtained into KPI metrics used for control objectives. For example, queue lengths are often easy to obtain, and, under steady-state assumptions, queue lengths can be translated into response times using Little's Result. While the problem of a mismatch between policy and sensor metrics is not specifically a scaling problem, it tends to arise more frequently as systems scale since there is increasing interest in connect IT with business processes.

B. Target System Challenges

1) Decomposing the Target Systems with Interactive Flow Effects: In distributed systems, work (in the form of requests) flows from one system to another. Actuators in one system affect the processing of work in that system, which can in turn impact the systems which are both upstream and downstream. These flow effects also introduce a source of dependency and delay. Further, the flows may depend on the nature of the request so may be unpredictable. The general approach of modeling system workloads as a disturbance generally ignores such effects, which can cause problems. Two strategies exist for addressing scaling issues. One is to rely on matrix-algebraic MIMO control techniques. Another strategy is to decompose the target system into smaller, (relatively) independent subsystems. The choice among the three control architectures, as discussed in Section III, depends on the considerations on the complexity of MIMO modeling, the strength of cross-resource interactions, and the trade-off between performance requirement and design simplicity.

2) Coping with Multiple Time Constants: In a system of heterogenous components, the different components operate at different timescales. In the discrete-time control framework, one approach to managing different time scales is to pick a large enough control interval that encompasses the slowest system. However, this may lead to an overall slow response, and it also ignores the dynamics of the target systems. Another approach is to use continuous-time techniques. However, the fundamentally discrete nature of computing systems can make the continuous time analysis difficult. Finally, time constants need to be taken into account when translating the business policies into control objectives. It does not make sense to define goals in terms of an averaging interval of 1sec if the system operates at scales of 10's of sec.

3) Adjusting for Delays Induced by Distributed Systems: If the target system is distributed, then there may be delays introduced in effecting control actions and obtaining metric values. It is well known in control theory that such delays can degrade control performance, possibly causing instabilities (e.g., [27]). If these delays are predictable, then they can be incorporated into the controller design. Handling unpredictable delays is an inherently difficult problem that involves considerations of control intervals and control architecture.

4) Exposing Sufficient Metrics and Actuators: In software systems, unlike most physical systems, it is possible in theory to expose almost any system metrics and place any controls. In practice, however, desired sensors and actuators are often not available due to a variety of factors. For example, software engineering practice (encapsulation, abstraction, etc) and intellectual property requirements generally means that software providers may not expose the internal state of their systems. In distributed systems the amount of information available "at a distance" is often limited so as to reduce communication overheads. Open source systems provide some mitigating factors since in theory the source code could be modified by control designers to expose internal state. The lack of good/complete metrics often leads to observability issues – complicating the system modeling task. For solutions that rely on online modeling especially, such observability issues require the introduction of safeguards and heuristics to prevent the controller from making incorrect decisions. While this problem is not specifically a scaling problem, it tends to arise more frequently as systems scale.

V. CONCLUSION

The rapidly increasing scale of computing systems means that it is vitally important to address scaling the control of computing systems.

To this end, we have introduced a framework for describing the scale of control problems in computing systems. Our framework has two dimensions: the scale of the target system and the scale of the policy. The former is expressed in terms of the number of inputs (actuators) and outputs (sensor metrics) in the target system being controlled. The latter is structured in terms of the number of objectives and owners. We observe that there is a trend towards multiple-input, multiple-output

target systems because of policies with multiple objectives. The advent of multiple owner target systems (e.g., eCommerce systems with multiple service providers) further complicates matters because of the potential for conflicting objectives.

We discuss how the framework can guide decisions of how to decompose a large-scale system into smaller more manageable ones. The decomposition strategies span a range from centralized schemes that work well when latencies are low between the controller and the target systems to distributed solutions that are effective if overall objectives can be decomposed into independent sub-objectives. In practice, there are cases of longer latencies in which it is difficult to decompose overall objectives. Hence, we propose a third solution that employs hierarchical control.

We present several research challenges that arise when scaling control systems. Among these are coping with delays and timescales induced by distributed systems, modeling and managing dependencies introduced by flow of work, challenges raised by multiple owners and some general control issues that are exacerbated in large-scale systems. These are formidable challenges that point to an exciting research agenda ahead, which will require an inter-disciplinary approach with advanced techniques from the control community and also the best distributed systems design and implementation techniques. Moreover, note that not all the problems are best formulated as control theory problems; this may require further multi-disciplinary studies with other systems management approaches such as queueing theory methods.

REFERENCES

[1] S. S. Parekh, K. R. Rose, J. L. Hellerstein, S. Lightstone, M. Huras, and V. Chang, "Managing the performance impact of administrative utilities," in *Proceedings of the 14th International Workshop on Distributed Systems: Operations & Management (DSOM 2003)*, (Heidelberg, Germany), pp. 130–142, Oct. 20–22 2003.

[2] Y. Diao, J. L. Hellerstein, A. Storm, M. Surendra, S. Lightstone, S. Parekh, and C. Garcia-Arellano, "Using MIMO linear control for load balancing in computing systems," in *Proceedings of the 2004 American Control Conference*, (Boston, MA, USA), pp. 2045–2050, June 30–July 2 2004.

[3] M. S. Mahmoud, M. F. Hassan, and M. G. Darwish, *Large Scale Control Systems*. Marcel Dekker, 1985.

[4] A. Mutambara, *Decentralized Estimation and Control for Multisensor Systems*. CRC Press, 1998.

[5] D. Godbole and S. S. Sastry, *Complex Systems: Adaptive Hierarchial Control*. Prentice Hall, 2001.

[6] N. Gandhi, S. Parekh, D. M. Tilbury, and J. L. Hellerstein, "Feedback control of a Lotus Notes server: Modeling and control design," in *Proceedings of the 2001 American Control Conference*, (Washington, DC, USA), June 25–27 2001.

[7] C. V. Hollot, V. Misra, D. Towsley, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, (Anchorage, AK, USA), IEEE, Apr. 22–26 2001.

[8] Q. Wu, P. Juang, M. Martonosi, and D. W. Clark, "Formal online methods for voltage/frequency control in multiple clock domain microprocessors.," in *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-XI)* (S. Mukherjee and K. S. McKinley, eds.), (Boston, MA, USA), pp. 248–259, ACM Press, Oct. 7–13 2004.

[9] C. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son, "A feedback control approach for guaranteeing relative delays in web servers," in *Proceedings of the 7th IEEE Real-Time Technology and Applications Symposium (RTAS 2001)*, (Taipei, Taiwan), May/June 2001.

[10] Y. Lu, T. F. Abdelzaher, C. Lu, L. Sha, and X. Liu, "Feedback control with queueing-theoretic prediction for relative delay guarantees in web servers," in *Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2003)*, (Washington DC, USA), May 27–30 2003.

[11] Y. Lu, T. F. Abdelzaher, and G. Tao, "Direct adaptive control of a web cache system," in *Proceedings of the 2003 American Control Conference*, (Denver, CO, USA), June 2003.

[12] Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury, "Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache web server," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2002)*, (Florence, Italy), pp. 219–234, Apr. 15–19 2002.

[13] C. Lu, X. Wang, and X. Koutsoukos, "End-to-end utilization control in distributed real-time systems," in *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS-24)*, (Tokyo, Japan), Mar. 2004.

[14] X. Wang, D. Jia, C. Lu, and X. Koutsoukos, "Decentralized utilization control in distributed real-time systems," in *Proceedings of the 26th IEEE Real-Time Systems Symposium (RTSS'05)*, (Miami, FL, USA), IEEE, Dec. 5–8 2005.

[15] R. Levy, J. Nagarajarao, G. Pacifici, M. Spreitzer, A. Tantawi, and A. Youssef, "Performance management for cluster based web services," in *Proceedings of the 8th IFIP/IEEE International Symposium on Integrated Network Management, (Colorado Springs, CO, USA)*, pp. 247–261.

[16] J. Aman, C. K. Eilert, D. Emmes, P. Yocom, and D. Dillenberger, "Adaptive algorithms for managing a distributed data processing workload," *IBM Systems Journal*, vol. 36, no. 2, 1997.

[17] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications," in *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, (Banff, Alberta, Canada), June 6–10 2005.

[18] M. Karlsson, C. Karamanolis, and X. Zhu, "Triage: Performance isolation and differentiation for storage systems," in *Proceedings of the Twelfth International Workshop on Quality of Service (IWQoS 2004)*, (Montreal, Canada), June 7–9 2004.

[19] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, M. Kalantar, S. Krishnakumar, D. Pazel, J. Pershing, , and B. Rochwerger, "Océano – SLA based management of a computing utility," in *IM 2001: Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management*, (Seattle, WA, USA), May 2001.

[20] J. Rolia, X. Zhu, and M. Arlitt, "Resource access management for a utility hosting enterprise applications," in *Proceedings of the 8th IFIP/IEEE International Symposium on Integrated Network Management, (Colorado Springs, CO, USA)*, pp. 549–562.

[21] P. Narvaez and K.-Y. Siu, "Optimal feedback control for ABR service in ATM," in *Proceedings of the 1997 International Conference on Network Protocols (ICNP '97)*, (Atlanta, GA, USA), pp. 32–41, Oct. 28–31 1997.

[22] G. Schmidt, ed., *Real Time Control of Large Scale Systems (Lecture Notes in Control & Information Sciences Vol 67)*. Springer, 1985.

[23] D. A. Menasce, D. Barbara, and R. Dodge, "Preserving QoS of e-commerce sites through self-tuning: A performance model approach," in *Proceedings of the 3rd ACM Conference on Electronic Commerce (EC'01)*, (Tampa, FL, USA), pp. 224–234, ACM, ACM Press, Oct. 14–17 2001.

[24] S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 525–536, 2003.

[25] J. L. Hellerstein, K. Katircioglu, and M. Surendra, "An on-line, business-oriented optimization of performance and availability for utility-based computing," *Journal on Selected Areas of Communications*, vol. 23, no. 10, 2005.

[26] R. B. Myerson, *Game Theory*. Harvard University Press, 1991.

[27] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*. Wiley-Interscience, 2004.