Feedback controllers in the cloud

2 authors:

Tharindu Patikirikorala
Swinburne University of Technology
18 PUBLICATIONS **197** CITATIONS
SEE PROFILE

Alan W. Colman
Swinburne University of Technology
123 PUBLICATIONS **987** CITATIONS
SEE PROFILE

Feedback controllers in the cloud

Tharindu Patikirikorala
Swinburne University of Technology
Victoria, Australia
tpatikirikorala@swin.edu.au

Alan Colman
Swinburne University of Technology
Victoria, Australia
acolman@swin.edu.au

Abstract— **Autonomic management of quality of service attributes by dynamic resource allocation is one of the requirements in cloud computing environments. In order to provide these requirements in a flexible way existing cloud providers expose static rule /threshold/heuristic based decision implementation frameworks to their consumers. These rule /threshold/heuristic based methods are relatively easy to design and develop. However, they suffer from lack of well-founded design process to decide important design parameters (e.g.: thresholds, the number of instances to add/remove, calm time) and difficulty of dynamically adjusting to different conditions. In addition, incorrect/non-adaptable rule settings could cause long term instabilities leading to service outages. The feedback control has been shown to be useful for performance management and resource allocation in many complex software systems. In this work, we investigate the advantages and limitations of applying feedback controllers in cloud platforms. In addition, we illustrate the suitability of standard feedback controllers depending on the consumer requirements/applications. Finally, we propose a novel platform as a service architecture to design, develop, integrate and runtime manage feedback controllers for the cloud consumer applications.**

Keywords- Cloud, decision making, QoS management, feedback control, control theory

I. INTRODUCTION

Cloud computing has become a popular utility computing model during past few years because of its potential to offload some of the important management concerns from small or large scale software service providers. Almost zero startup cost on infrastructure, less effort on system management, avoiding resource over/under provisioning depending on the demand, and pay-as-you-go billing models are some of these attractive advantages and features delivered by existing cloud providers [1-3]. In addition, runtime management of Quality of Service (QoS) attributes, such as response time and throughput while efficient/cost-effective resource allocation can be of great value to cloud consumers. However, existing cloud environments provide few facilities to management of QoS performance and resources to achieve the consumer objectives. For instance, Amazon [4], RightScale [5] and Azure [6] expose non-adaptive rule/threshold/heuristic based instance scaling facilities through their management APIs. These approaches are useful because they are easy to understand and design. However, the rule/ threshold /heuristic based policies lack formal well-founded design process [7-8]. For example parameters such as the number of instances to be added/removed and time to wait before the next allocation decision (so called calm time) have to be set usually based on trial and error [7]. If these parameter settings are not selected carefully, long term instabilities cloud cause in the system leading to service outages [7-8]. In addition, these design parameter settings are static with no online adjustment schemes, hence depending on the different operating conditions these parameter setting could be either more aggressive or less aggressive causing oscillatory behaviors in the system (as experimentally shown in [8]). In practice, cloud providers tend to offer service guarantees for QoS attributes such as availability and security, rather than performance related measures such as response time and throughput [9]. This is because unpredictable workload conditions, heterogeneous requirements of consumers and application behavior/dynamics make it difficult to provide response time/throughput guarantees using generalized resource allocation schemes [9].

As critical business functions are deployed on the cloud, it is important that the cloud computing platforms to provide adequate facilities to achieve QoS management objectives required by the consumers. By delegating application specific QoS management to cloud consumers the cloud providers can focus on infrastructure objectives while giving consumers the ability to meet their specific requirements [8, 10]. When this decoupling is achieved cloud consumer can implement effective/efficient runtime decision making to achieve required objectives addressing the limitations of existing (rule/threshold based) methodologies. The feedback control is one decision making technique utilized in last decade to achieve QoS management objectives of complex software systems (e.g.: [7, 11-13]).

In this work we investigate the advantages and limitations of applying feedback control in cloud computing platforms. In addition, we point out the effectiveness and suitability of various feedback control schemes in control literature for different consumer/application requirements. Finally, we propose a new architecture to design, develop, integrate and runtime manage off-the-shelf standard feedback controllers for QoS management to achieve cloud consumers objectives.

The rest of the paper is organized as follows: Section II provides the overview of classical feedback control systems and potential advantage/limitations of their application to cloud systems. The suitability of different standard feedback control algorithms are discussed in section III. Finally, a

proposed architectural solution to design, develop, integrate and runtime manage off-the-shelf standard feedback controllers in to cloud environment will be presented.

II. TYPE FEEDBACK CONTROL BASICS, ADVANTAGES AND LIMITATIONS

Many server-based applications in general and cloud applications in particular have the following characteristics/requirements.

- Workload settings could be on-off, predictable or unpredictable bursts [14]. Hence, the decision making system should be able to adapt to these changing workload conditions.
- Software systems show nonlinear characteristics. Some of the characteristics cannot be modeled effectively. The control system should have the capabilities to make decisions under these unmolded and nonlinear dynamics. Moreover, low-level (source code level) details of these systems may not be available, hence the decision making system should not require detailed knowledge of implementation or any modification to the system implementation/source-code.
- Many server-based systems have a single application environment that has to cater to multiple client classes. More particularly Software as a Service (SaaS) systems in the cloud are typically multi-tenanted. This requires that the decision making system should have the capability cope with the interactions/resource competitions between multiple tenants and achieve the QoS management and performance differentiation [15].

Many of the above requirements are ignored in the design of rule/ threshold /heuristic based methodologies to achieve the simplicity. However, when these requirements are incorporated rules/polices will be complex. The conflicting rules may arise when the complexity of the rules increase. Due to lack of systematic/formal design process it is hard to validate these rules for consistency. The limitations of rule-based policies are discussed in [8, 16] in detail.

In contrast, feedback control may provide effective methodologies to achieve the above requirements. Feedback control techniques have been already applied and proved useful in web server systems [11, 17], cache and storage systems [12, 18], data centers/server clusters [19], QoS performance and resources management in shared resource environments with multiple client classes (e.g.: [12, 15, 17, 19-22]). From these existing approaches it is clear that feedback control is able to handle unpredictable workload conditions, nonlinear characteristics and provide effective control under disturbance and un-modeled dynamics. At the same time it can fulfill the QoS management requirements of multi-tenant systems and integrate ad-hoc policies in to decision making. Furthermore, there is systematic design process to develop an effective controller given the high-level objectives. There are well proven tools and methodologies to calculate the controller tuning parameters and validate the controller. In addition, there are many standard controller algorithms available from control literature with formal stability proofs.

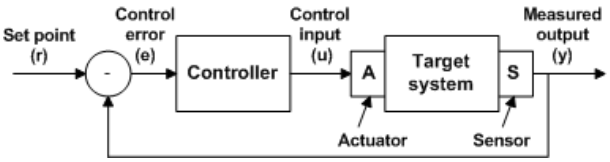

Figure 1.Block diagram of a feedback control system

Figure 1 shows a block diagram of a classical feedback control system. The system controlled by the controller is referred to as target software system. The target software systems provides a set of performance matrices for properties of interest (e.g.: response time, CPU) referred to as outputs. Sensors monitor the outputs for the target system, while control input (e.g.: resource allocation) can be adjusted through actuators to change the behavior of the system. The controller is the decision making unit of the control system.

The main objective of the controller is to maintain the output of the system sufficiently close to the set point, by adjusting the control input. The set point gives the option for the control system designer to specify the goal/desired value of output. Depending on the objects control systems could be designed in two different ways. If the requirement is single objectives (single set point) the control system can be designed as Single-Input Single-Output. Multi-Input Multi-Output control systems, on the other hand, can be used to meet multiple objectives.

The control system design generally consists of two main steps. Firstly, a formal relationship between the control input and the output has to be constructed. In control theory this relationship referred to as behavioral *model* of the system. The system identification (SID) is the tool used to construct this model. Linear Time Invariant (LTI) [23] black-box models are used to capture the relationship between the input and output of the system, which eliminates the need for detailed knowledge about the system implementation. The model of the system is then utilized in the second step, which includes controller design, analysis and testing. Then designed controller is integrated in to the system externally without any modifications to the software system.

Control theory has been used in many other engineering disciplines in a systematic way to control real/production systems. In software engineering, however, control of real/production software systems using formal feedback control is not widely adapted [7, 24-25]. There are a number of challenges and limitations in applying control theory in software systems [7, 26-27]. As categorized and listed below, these challenges arise from the nature of software system in general and the cloud platforms in particular, as

well as practical difficulties because of the lack of background/knowledge applying these rigorous techniques.

Software systems:

- Lack of first principle models makes it difficult to decide what outputs to sense and which actuators to control for specific QoS objectives,
- Inherent nonlinear behavior,
- Discrete/discontinuous inputs,
- Operating regions including that caused by arbitrary behavior and environmental conditions.
- Rapid changes and the volatility in software systems/components unlike physical plants (e.g.: new features, bug fixes)

Cloud platforms:

- Consumer versus global QoS,
- Highly discrete control inputs (e.g.: instance count, instance size),
- Time lag for starting Virtual Machines (VM) (actuation delays),
- Noise and delay in sensor data,
- Limited to what sensors and actuator the cloud provider makes available,
- From available different sensors and actuators which should be selected to construct the control system,
- Limitations on sampling interval ,
- Granularity of sensors and actuator to control multi-tenant systems.

Practicalities:

- Lack of knowledge and skills,
- Lack of commercially-off-the-shelf (COTS) control components and development environments.

Issues, due to actuator delays and sensor delays can be tackled up to some extent by existing techniques in control literature. However, many of these limitations have to be solved by the cloud provider to provide the decoupling demanded by the consumers. For instance, to achieve heterogeneous requirements on multi-tenanted consumer applications it is vital to provide facilities to divide the provisioned servers and route/load balance the requests among them.

The challenges pointed out in the above discussion need to be addressed if the potential benefits of feedback control approaches are to be realized.

In this work we propose a solution for lack of COTS control components and development environments to apply feedback control in cloud environments (see section IV). As argued in [26], these standard control algorithms could be used as COTS components/products if the systems are build with facilities to apply them. The existing cloud providers provide adequate support/facilities via the management APIs to apply feedback controllers however, there exists no tool support or platform currently to design, develop and integrate feedback controllers in to the systems deployed in cloud computing environments. In addition, there is no COTS control libraries that provides the implementations of standard feedback controllers which could be integrated in to software systems effectively. The design support must abstract away the rigorous mathematical foundation from the software developers, while COTS control libraries will reduce design, development and testing effort by providing well tested standard COTS feedback controllers.

III. STANDARD FEEDBACK CONTROLLERS FOR CONSUMER REQUIREMENTS

In this section, we briefly describe the different types of well-established feedback control schemes and discuss their suitability for meeting the needs of cloud consumers.

Fixed gain controllers: the fixed gain controllers are basic type of controllers, where the controller tuning parameters are set by analyzing the dynamic model constructed over the SID experiment and high-level requirements from the users. The Proportional Integral Derivative (PID) controllers are the widely adapted fixed gain controller due to their robustness against modeling errors, disturbance rejection capabilities and simplicity [28] . There are rigorous mathematical techniques (e.g.: Root Locus, Routh–Hurwitz method) to select the tuning parameters of the controllers in order to meet control objectives and desired performance [28-29]. However, after selecting the tuning parameters, they remains *fixed* during the time controller is in operation (i.e. cannot be changed at runtime). Consequently, controllers are called *fixed gain controllers*. The fixed gain controllers may be useful for the consumers which has operations that do not have highly divergent conditions. For example, in the cases where workload conditions change in predictable fashion within some nominal range, or may be sites which provide static contents. However, in the cases where operating conditions are highly varying, still the required objectives can be achieved with short-term performance degradations avoiding instabilities. The fixed gain controllers are already used in [8, 10] to achieve scaling objectives under varying workload conditions.

Model predictive control (MPC)[30]: Fixed gain controllers as described above are reactive controllers, in that the future behavior of the system is not considered in current time instance when making control decisions. In contrast, MPC uses the dynamic model and predict future behavior of the system to come up with the decisions to optimize the future behavior. In addition, MPC is attractive for multi-objective scenarios and to derive close to optimal decisions in the presences of complex policies and constraints. Thus, this is proactive or self-optimizing techniques. Recent cloud computing surveys [2] have pointed out the demand for proactive decision making strategies compared to reactive methods. In this regard MPC is a useful feedback controller for cloud environments.

Table 1. Property comparison of control schemes √-Yes x - No

Property	Fixed	MPC	Adaptive	Reconfiguring
Provide systematic/formal design tools	√	√	√	√
Less design complexity	√	√	x	x
Adapts at runtime to different conditions	x	x	√	√
Ability to handle fast varying condition	x	x	x	√
Makes proactive decisions	x	√	x	x
Provides MIMO control	√	√	√	√

Adaptive control[11, 31]: Adaptive control address some of the limitations of fixed gain controllers, by adjusting the controller tuning parameters online. It has the online estimation techniques (usually recursive least squares method [31]), which construct the dynamic model of the system in each sampling instance. Then given the high-level objectives of the user and using the estimated model adaptive controller tunes it parameters online without any human interventions. The self-tuning PID controllers [11], gain-scheduling and self-tuning regulators [31] are designed following this methodology. Applications of adaptive control in software systems could be found in [12-13, 15, 22, 32-34]. Such types of controllers can operate and achieve consumer objectives in a wide range of conditions. For instance, under on-off, predictable and slowly varying-unpredictable workload conditions. In addition, when the systems are operating in different conditions such as serving static content and dynamic contents. However, this flexibility is only suitable when these operating conditions are slowly varying. This is because the online model estimation process may fail to capture dynamics of the system sufficiently when the system/environment conditions are changing quickly.

Reconfiguring control: In standard adaptive control the controller parameters are derived online but the control algorithm (law) remains the same. The reconfiguring controllers are form of adaptive controller but the controller (algorithms) can be changed at runtime depending on the different conditions. There are different adaptive reconfiguring control techniques proposed for software systems [24, 35], however they lack stability proofs. Multi-Model-Switching and Tuning (MMST)[36] adaptive control is one of the reconfiguring control techniques proposed in control literature with formal stability proofs. This technique may be useful in software systems that face well known operating conditions such as on-off workloads, predictable bursts (e.g. : tax sites active in the period of tax claims, e-commerce sites during special offers) and unpredictable bursts/conditions. This scheme is also useful to combine the aforementioned feedback controllers depending on the requirements.

Table 2 provides a comparison of properties of the above control schemes. Depending on the consumer requirements, system behavior and operating characteristics different control schemes are most suitable.

In the next section, we propose an architecture for existing cloud provider platforms to aid design, develop, integrate and runtime manage aforementioned feedback control schemes to achieve cloud consumers goals and objectives. The proposed solution is called as Feedback Controllers As A Service (FCAAS) platform.

IV. FCAAS PLATFORM

As shown in Figure 2, the FCAAS platform resides on top of cloud provider infrastructure providing design and implementation and management support of feedback controllers for the cloud consumers. This can be regarded as a platform as a service where design development support is provided. The PAAS platform architecture consists of *system identification (SID) toolbox* and *controller toolbox*. The SID toolbox provides modeling support, including facilities to design SID experiments, control signals and data gathering. Then the data gathered from SID experiments could be further analyzed by the SID toolbox to construct sufficiently accurate models to describe the behavior/dynamics of the system.

The controller toolbox provides standard COTS control algorithms or facilities to extend the existing algorithms depending on the consumer requirements. Given the model from the SID toolbox, the control toolbox will provide standard methodologies to analyze and derive tuning parameters of the controllers with respect to the consumer objectives. For instance, root locus and pole placement design are some of the well known formal methodologies to derive these parameters which would be provided by the proposed toolbox. After selecting the necessary parameters the respective *controller template* will be populated. The controller template abstract away application specific details from the standard control algorithm. For instance, the sensors, actuators and ad-hoc policies are different from consumer to consumer. Thus, the controller template integrates these concerns/properties to the standard controller providing separation of concerns. Final step of the design process is to compile the designed controller using the *controller compiler* (Figure 2) to create the installation of the controller. The installation could be used to install in desired machine (inside cloud provider or externally). The controller compiler and the controller template has to be designed considering cloud provider specific details/implementations. The FCAAS *remote management*

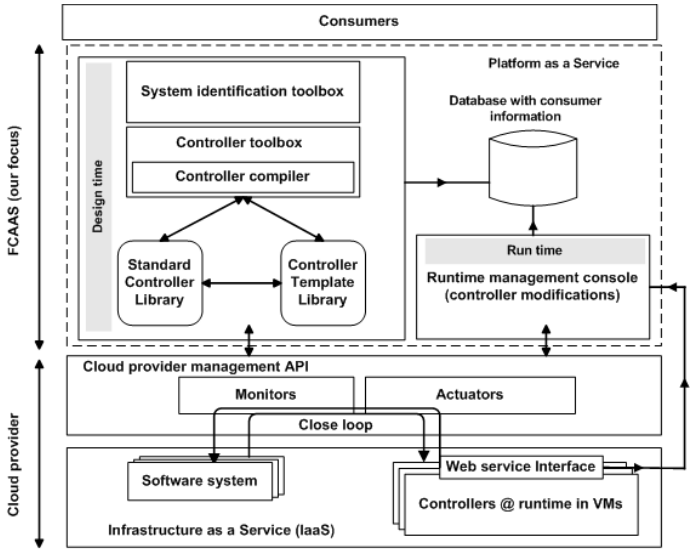

Figure 2. Proposed Platform as a service architecture (FCAAS)

console provides a web interface where the consumers can login, check the status and make modifications to their controllers at runtime. These design process could be done and tested in the staging environment provided by many exiting cloud platforms (e.g.: [4-6]) without affecting the deployed systems, which is an added advantage of designing controllers for cloud consumers.

The implemented controller could be installed in cloud provider platform which is recommend (to avoid sensor and actuation delays) or external infrastructure. After the installation the controller provides a web service based management interface to enable runtime management of the controller. This interface could be utilized to make necessary changes to the controller or retune the controller parameters using auto-tuning algorithms [31] in control literature. The same facilities will be available from the remote management console.

The closed feedback loop is the basic concept in control system. In the proposed architecture closed loop is achieved by utilizing the management APIs exposed by cloud providers without any modifications to the provider environment. For instance, Amazon query APIs [37] and Windows Azure performance counters and management APIs [14] are such management facilities provided by existing cloud providers. Use of these management APIs provide the opportunity to decouple QoS management functionalities as discussed in section I. In addition, the consumer applications do not need any modification to design the closed loop control systems.

Currently we have completed standard COTS controller library, standard algorithms for SID and basic template library implementations, including

Single-input single-output (SISO) controllers

a. PID controller - can be used as 6 different types of controllers (algorithm [29])
b. Model predictive controllers (algorithm [30])
c. Model predictive controller with laguerre networks (algorithm [30])
d. Self-tuning regulators(algorithm [31])
e. Self-tuning PID controller (algorithm [11])

Multi-input Multi-output (MIMO) controllers

f. PID controller (algorithm [11])
g. Model predictive controller with laguerre networks (algorithm [30])

MMST schemes supported (algorithms from [38-40])

h. Type 1: All adaptive models
i. Type 2: All Fixed models
j. Type 3: One Adaptive model and one Fixed model
k. Type 4:Two Adaptive model and n-2 Fixed models

As future work we intend to implement the controller compiler for existing open source IaaS framework [41-42] as proof of concept. One of the main limitations of feedback control system design is the requirement of a control system design expert (control engineer). Such expertise is rarely seen in software engineering companies. To address this limitation we hope to design a step by step user-friendly web interface for SID and controller design. In addition, expert system to encapsulate the expertise of control engineer to aid the design process of the controller and runtime management.

V. CONCLUSION

The autonomic management of quality of service attributes and resources is one of the requirements of the cloud computing platforms. Existing platforms provide rule/threshold based schemes to address these concerns. However, these have been shown to be inefficient and ineffective. In this work we argue that the standard feedback

controllers could help to achieve the quality of service and resources management objectives demanded by cloud consumers. We also point out the advantages, limitations of applying feedback controllers in existing could platforms and suitability of different standard feedback controllers depending on the system characteristics and consumer needs. Finally, we propose a generic architecture to implement PaaS environment for existing cloud providers, which could be utilized by consumers to integrate standard COTS feedback controllers to runtime manage QoS attributes and resource allocation.

REFERENCES

[1] A. Weiss, "Computing in the clouds," *netWorker,* vol. 11, pp. 16-25, 2007.

[2] M. Armbrust, *et al.*, "Above the Clouds: A Berkeley View of Cloud Computing," 2008.

[3] R. Buyya, "Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing as the 5th Utility," presented at the Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009.

[4] Amazon EC2 , http://aws.amazon.com/ec2/.

[5] Rightscale , http://www.rightscale.com.

[6] Windows Azure, http://www.microsoft.com/windowsazure/.

[7] X. Zhu, *et al.*, "What does control theory bring to systems research?," *SIGOPS Oper. Syst. Rev.,* vol. 43, pp. 62-69, 2009.

[8] X. Dutreilh, *et al.*, "From Data Center Resource Allocation to Control Theory and Back," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, 2010, pp. 410-417.

[9] W. Iqbal, *et al.*, "SLA-Driven Adaptive Resource Management for Web Applications on a Heterogeneous Compute Cloud," presented at the Proceedings of the 1st International Conference on Cloud Computing, Beijing, China, 2009.

[10] H. C. Lim, *et al.*, "Automated control in cloud computing: challenges and opportunities," presented at the Proceedings of the 1st workshop on Automated control for datacenters and clouds, Barcelona, Spain, 2009.

[11] J. L. Hellerstein, *et al.*, *Feedback Control of Computing Systems*: John Wiley \& Sons, 2004.

[12] M. Karlsson, *et al.*, "Triage: Performance differentiation for storage systems using adaptive control," *Trans. Storage,* vol. 1, pp. 457-480, 2005.

[13] M. Karlsson and C. Karamanolis, "Non-intrusive Performance Management for Computer Services," in *Middleware 2006.* vol. 4290, M. van Steen and M. Henning, Eds., ed: Springer Berlin / Heidelberg, 2006, pp. 22-41.

[14] T. S. Grzegorz Gogolowicz. (2010, Dynamically Scaling Windows Azure - Sample Application (http://code.msdn.microsoft.com/Project/Download/FileDownload.aspx?ProjectName=azurescale&DownloadId=10136).

[15] M. Karlsson, *et al.*, "An Adaptive Optimal Controller for Non-Intrusive Performance Differentiation in Computing Services," Hewlett Packard LaboratoriesFebruar1 18 2005.

[16] J. O. Kephart and W. E. Walsh, "An Artificial Intelligence Perspective on Autonomic Computing Policies," in *POLICY*, ed, 2004, pp. 3-12.

[17] C. Lu, *et al.*, "Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers," *IEEE Trans. Parallel Distrib. Syst.,* vol. 17, pp. 1014-1027, 2006.

[18] Y. Lu, *et al.*, "An Adaptive Control Framework for QoS Guarantees and its Application to Differentiated Caching Services," presented at the Tenth IEEE International Workshop on Quality of Service, 2002.

[19] D. Kusic and N. Kandasamy, "Risk-Aware Limited Lookahead Control for Dynamic Resource Provisioning in Enterprise Computing Systems," in *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*, 2006, pp. 74-83.

[20] R. Zhang, *et al.*, "ControlWare: A Middleware Architecture for Feedback Control of Software Performance," presented at the Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02), 2002.

[21] T. Kwok and A. Mohindra, "Resource Calculations with Constraints, and Placement of Tenants and Instances for Multi-tenant SaaS Applications," presented at the Proceedings of the 6th International Conference on Service-Oriented Computing, Sydney, Australia, 2008.

[22] P. Padala, *et al.*, "Automated control of multiple virtualized resources," presented at the Proceedings of the 4th ACM European conference on Computer systems, Nuremberg, Germany, 2009.

[23] L. Ljung, *System identification: theory for the user*: Prentice-Hall, Inc., 1997.

[24] Kokar M. M, *et al.*, "Control Theory-Based Foundations of Self-Controlling Software," *IEEE Intelligent Systems,* vol. 14, pp. 37-45, 1999.

[25] M. Shaw, "Beyond objects: a software design paradigm based on process control," *SIGSOFT Softw. Eng. Notes,* vol. 20, pp. 27-38, 1995.

[26] C. Karamanolis, *et al.*, "Designing controllable computer systems," presented at the Proceedings of the 10th conference on Hot Topics in Operating Systems - Volume 10, Santa Fe, NM, 2005.

[27] J. L. Hellerstein, "Self-Managing Systems: A Control Theory Foundation," presented at the Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, 2004.

[28] R. C. Dorf and R. H. Bishop, *Modern Control Systems*: Prentice-Hall, Inc., 2000.

[29] K. Ogata, *Modern Control Engineering*: Prentice Hall PTR, 2001.

[30] L. Wang, *Model Predictive Control System Design and Implementation Using MATLAB*: Springer Publishing Company, Incorporated, 2009.

[31] K. J. Åstrom and B. Wittenmark., *Adaptive Control, 2nd ed. Electrical Engineering: Control Engineering*: Addison-Wesley Publishing Company, 1995.

[32] X. Liu, *et al.*, "Adaptive entitlement control of resource containers on shared servers," in *Integrated Network Management*, ed: IEEE, 2005, pp. 163-176.

[33] K. Wu, *et al.*, "The Applicability of Adaptive Control Theory to QoS Design: Limitations and Solutions," presented at the Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 15 - Volume 16, 2005.

[34] Y. Lu, *et al.* (2002, *An Adaptive Control Framework for QoS Guarantees and its Application to Differentiated Caching Services.*

[35] B. Solomon, *et al.*, "A real-time adaptive control of autonomic computing environments," presented at the Proceedings of the 2007 conference of the center for advanced studies on Collaborative research, Richmond Hill, Ontario, Canada, 2007.

[36] K. S. Narendra and O. A. Driollet, "Adaptive Control using Multiple Models, Switching, and Tuning," presented at the Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000., 2000.

[37] Amazon. (2009, Amazon Auto Scaling: Developer Guide , http://awsdocs.s3.amazonaws.com/AutoScaling/latest/as-dg.pdf.

[38] K. S. Narendra, *et al.*, "Adaptation and learning using multiple models, switching, and tuning," *Control Systems Magazine, IEEE,* vol. 15, pp. 37-51, 1995.

[39] K.S.Narendra and C. Xiang, "Adaptive control of discrete-time systems using multiple models," presented at the Automatic Control, IEEE Transactions, 2000.

[40] K. S. Narendra, *et al.*, "Adaptive control using multiple models, switching, and tuning," *International journal of adaptive control and signal processing* pp. pp.1-16, 2003.

[41] Eucalyptus. *http://open.eucalyptus.com*.

[42] D. Nurmi, *et al.*, "The Eucalyptus Open-Source Cloud-Computing System," presented at the Cluster Computing and the Grid, IEEE International Symposium on, 2009.